python

# Table of Contents

# PyGuide

# مقدمة

## لماذا هذا الكتاب ؟

السبب الرئيسى هو أنه كتاب **مجانى** ومتكامل ولااعتقد انك بعد قرائته ستواجه صعوبات مع Python

## ماذا تتوقع بعد قراءتك لهذا الكتاب؟

بعد قرائتك للكتاب ستكون قد علمت الكثير حول بايثون وكيفية توظيف اللغة بصورة جيدة وجدية، ستصبح متأقلم مع تطبيقات قواعد البيانات والشبكات -ربما تفكر فى كتاب خادم ويب خاص بيك؟- والتطبيقات الرسومية والكثير والكثير من طرق معالجة البيانات وغيرها الكثير.. ستكتشف هذا بنفسك!

## المتطلبات

- جهاز عليه احد نظم GNU/Linux او Windows (يفضل XP or better )
- معرفة جيدة بكيفية إستخدام Text Editor رفة باشياء مثل
- معرفة Permissions, Groups, Users .. فى حال عدم المعرفة يفضل الإتجاه إلى LinuxAC ستجد العديد من المواضيع لتساعدك

## الإصدارات القادمة والتحديثات

- ستجد اخر الإصدارات والتحديثات على Programming-Fr34ks.NET

## المساهمين

- Ahmed Youssef

## حقوق النسخ



الاقتراحات والانتقادات يرجى إرسالها على guru.python[at]gmail.com

# *Chapter 1 (Programming Concepts)*

مامعنى كلمة **Programming** ؟

هى بتعنى القدرة على التخاطب مع الكمبيوتر وتنفيذ الهدف (حل المشكلة) على ارض الواقع .. الكمبيوتر لايفهم اى شئ سوا 0 و 1 وصعب على البشر تعلمها إن لم يكن مستحيلا فنلجأ لخيارات بديلة وهى إستخدام لغات البرمجة

مامعنى **Programming Language** ؟

بكل بساطة هى وسيلة للتخاطب مع الكمبيوتر .. ولكننا قلنا إن الكمبيوتر لايفهم اى شئ سوا ال 0 وال 1 ومستحيل على الإنسان تعلمها! .. إذا الحل هو إستخدام لغات وسيطة .. على سبيل المثال واحد عربى وواحد فرنسى والعربى مش بيفهم فرنسى ولا الفرنسى بيفهم عربى .. فالحل هو إنهم يتكلمو إنجليزى مثلا... او يجيبو مترجم بين الإتنين مش كدا ؟
فهنا الحل إننا هنحصل على مترجم يترجم افكارنا للغة الكمبيوتر 0 و 1 ويقوم المترجم بنفس الدور بتحويل رد فعل الكمبيوتر الى لغتنا المفهومة :)
وهنا دور ال Programming Language انت هتتعلم اللغة وكيفية التعامل معاها عشان تقدر تفهم المترجم "المقدم من اللغة" ماتريده وهو يفهمه للكمبيوتر بدوره

ماهو ال **Source Code** ؟

بكل اختصار هو حلك لمسألة رياضيات وتفكيرك وإستنتاجاتك لما تكتبها فى ورقة ولكن هنا هو حلك لبرنامج مطلوب منك على ملف Text

مامعنى ال **Debugging ؟**

على فرض إنك بتحل مسألة رياضيات و إكتشفت خطأ فى طريقة حلك .. فإنت بتتبع المشكلة اللى حصلت وتشوف إزاى تصححها وهو دا معنى ال Debugging اى تصحيح الأخطاء :)

## ***Compiled vs Interpreted***

كتير منا إشتغلو على نظم Windows وكان ديما بيشوف ملفات إمتدادها .exe فإيه معنى ال exe ؟ معناها Executable او قابل للتنفيذ ..
فى لغات برمجة مثل ال C و Pascal بيتوافر الناتج النهائى بتاع برنامجك على صورة ملف exe وهو عبارة عن تعليماتك اللى إديتها للمترجم عشان يفهمها للكمبيوتر ولكن فى صورتها النهائية (الكلام اللى قاله المترجم لل كمبيوتر) فمستحيل على الإنسان إنه يقرا الملف دا وهنا معنى ال compiled فهى ملف ال exe يشمل التعليمات اللتى كتبتها ولكن بلغة الكمبيوتر وهو وحده القادر على فهمها

وإذا نظرنا من جانب آخر إلى لغات مثل Python, Perl هنجد إن الملف بيكون إمتداده .py او .pl ولكنك تقدر تفتحه فى اى Text Editor وتقراه -لفهمه لازم تكون عارف اللغة- والملف دا هو ال Source Code بتاعك نفسه بدون اى تحويلات ولا شئ ولكن لتنفيذه بنستدعى ال المفسر (Interpreter) فى كل مرة بحيث إنه يقرا ال Source Code ويبلغه للكمبيوتر ويتم التنفيذ

من مميزات ال Compiled Languages مثل ال C هى السرعة
ومن القصور هو إنك لازم تعمل Compile لل Source Code بتاعك على النظام اللذى تريد ان تنفذ البرنامج عليه فبرنامج مكتوب على Windows محتاج يتعمله recompile على ال Linux وهكذا ..

من مميزات ال Interpreted Languages هى انك ال Source Code بتاعك القياسى لايحتاج لعمل Recompile على مختلف النظم وال archS
ومن القصور البطء

**<u>ملحوظة:</u>**

لما بنتكلم على كلمة البطء فى ال Interpreted Languages بيكون المقصود البطء بالنسبة لل Compiled Language وليس البطء للمستخدم لأنك مش هتلاحظ الفرق لأن البطء فى شئ لايكاد يذكر

## *Chapter 2 ( Introduction to Python)*

Python هى لغة برمجة عامة لمعظم المجالات ان لم تكن جميعها، وهى High Level Programming Language اى انها قريبة جدا من لغة الإنسان "الإنجليزية" بدأت فى عام 1989 على يد Guido Van Rossum وهو عالم هولندى

Python تتميز ب

1- سهولة التعلم
2- وضوح الكود وسهولة صيانته
3- ال Portability المحمولية -لأنها بتعمل على اكتر من Platform-
4- Python تعطيك قوة ال Scripting Languages وبكل تأكيد اخفاء مشاكل ادارة الذاكرة وتوابعها عنك
5- Open Source: فيقوم على تطويرها الألاف من المطورين
6- Python تقدم تكامل مع ال .NET وال Java من خلال IronPython, Jython
7- بتدعم اكثر من paradigm ك Functional Programming, OOP
8- Python لغة ممتعة!

جدير بالذكر ان Python حاليا هى de facto فى عالم الأوبن سورس حيث تفوقت على Perl من حيث الشعبيةوهى لغة العام حسب تقرير Tiobe

ملفات Python بيكون امتداها .py او .pyc او .pyo

.py => ملف بايثون
.pyc => ملف بايثون مترجم
.pyo => ملف كائن لبايثون

Downloading/Installing Python

ادخل على /http://python.org/download

لمستخدمى UNIX/UNIX-Like: ف Python غالبا مرفقة مع توزيعتك.. فى حال لأ قم بتحميل ال Source واعمل Build

```
./configure
make
make install
```

اى خطأ قم بمراجعة ملف ال README او INSTALL

لمستخدمى Windows: قم بتحميل ملف ال .msi

تشغيل بايثون

تقدر تشغل ال المفسر (Interpreter) كجلسه تفاعلية بمعنى انك تمرر ليه statement معينه وهو ينفذها
لمستخدمى Linux: افتح ال Terminal او ال Console واكتب Python
لمستخدمى Windows :

```
~Start -> Run -> Cmd
~cd /Python_PATH/
python.exe
```

او قم بدعم ال PYTHON_PATH فى ال Environment Variables -متغيرات البيئة- فى ويندوز كالتالى

set path=%path%;C:\Python25
( • ح ô ƒ ¶
او التالى

1) *Right Click on My Computer -> Properties*
2) *Advanced Tab*
3) *Environment Variables*
4) *in Variables for (UserName): Click on PATH - > Edit*
5) *; Add C:\Python25*

لاتنسى الفاصلة المنقوطة

اكتب اى statement جمله مثلا 1+2 او "Hello, World" واضغط Enter

# *Chapter 3 (Basics)*

هنتناول فى الفصل دا مفاهيم اساسية زى ال variables وال Loops وال Conditions وهى جزء حيوى من كل اللغات

اولا ماهو ال variable (متغير)؟
هو قيمة متغيرة فى برنامج وبيهمك انك تتابعها وتتابع اى تغير يتم فيها.
مثلا حساب بنكى فيه فلوس "قيمة" قابلة للتغيير فلازم نعبر عنها بمتغير وليكن "money" ونشوف قيمته خلال برنامجنا سواء بالزيادة او النقص

افتح ال IDLE .. هشتغل بنظام تفاعلى فى الأكواد الصغيرة..

```
>>>balance=90000 #an integer
```

نطبع القيمة كالتالى

```
>>> print balance
90000
```

نضيف ليه قيمة ولتكن 100 (زيادة)

```
>>> balance = balance + 100
>>> print balance
90100
```

ننقص منه اى قيمة وليكن 890 (نقص)

```
>>> balance = balance - 890
>>> balance
89210
```

على فرض اننا عندنا variable بإسم name بيعبر عن قيمة معينة لل name دا
ملحوظة ال variable ماهو الا alias لقيمة متغيرة فى برنامجك

```
>>> name="ahmed" #a string
>>> print name
ahmed
```

وهكذا تقدر تعمل متغير يعبر عن العمر مثلا age

```
>>> age=50
>>> print age
50
```

## List

طب جميل موضوع المتغيرات هيدفعنا نتكلم عن ال Lists او القوائم هى عبارة عن Enhanced Array لمبرمجى ال C
اقرب مثال هو طلاب الفصل او زملاء العمل هل تعتقد ان اذا عندنا 30 طالب اننا نعمل شئ مشابه للتالى

```
student_1=”Ahmed”
student_2=”Wael”
student_3=”Ayman”
student_4=”Tina”
.....
student_30=”Youssef”
```

هل تتوقع اننا ننشئ 30 متغير ل 30 طالب بالصورة دى ؟ ممكن بس يفضل تكتب استقالتك بعدها :D

بالطبع لأ.. وهنا تيجى اهمية ال List وهى بإختصار ال Grouping لل Data Types المشتركة بمعنى ان كل دول students صح ؟

```
students=[“Ahmed”, “Wael”, “Ayman”, “Tina”]
>>> type(students)
<type 'list'>
```

لاحظ ان اول عنصر فى ال list ال index بتاعه هو 0 والتانى هو 1 والتالت هو 2

القاعدة العامة

```
idx=n-1
```

مثلا عندنا list كالتالى

```
>>>friends=["St0rM", "Squall", "Tina", "Ayman"]
```

لاحظ موضوع ال indexing لأنه بسيط وهو بنستخدمه للوصول لعنصر فى ترتيب معين فى اى sequence

```
>>> friends[0] # 1st
'St0rM'
>>> friends[2] # 3rd
'Tina'
```

ال List بتقدملنا العديد من الوظائف او التسهيلات فى التعامل على فرض ان عندنا List كالتالى

```
>>> students=["Ahmed", "Ayman", "Tina", "Wael"]
>>> students
```

```
['Ahmed', 'Ayman', 'Tina', 'Wael']
```

.append(element)

لإضافة عنصر بنستخدم ال append ميثود كالتالى

```
>>> students.append("Gina")
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina']
```

-- تقدر تضيف عنصر كالتالى

```
>>> students += ['Marian']
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina', 'Marian']
```

فى الواقع تقدر تضيف عدة عناصر

```
>>> students += ["Omar", "Waleed"]
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina', 'Marian', 'Omar', 'Waleed']
```

او بإستخدام extend

.extend(iterable)

بتقوم بدمج ال iterable على ال list الحالية كالتالى مثلا

```
>>> students.extend(["Omar", "Waleed"])
>>> students
['Ahmed', 'Ayman', 'Tina', 'Wael', 'Gina', 'Marian', 'Omar', 'Waleed']
```

.remove(value)

بيتم حذف اول ظهور لل value فى ال list مباشرة كما فى المثال

```
>>> students.remove("Ayman")
>>> students
['Ahmed', 'Tina', 'Gina', 'Marian', 'Omar', 'Waleed']
```

للحذف بنقوم بتحديد ال index (الترتيب) الخاص بالعنصر ليتم حذفه كالتالى مثلا

```
>>> del students[3]
>>> students
['Ahmed', 'Ayman', 'Tina', 'Gina', 'Marian', 'Omar', 'Waleed']
```

.insert(idx, item)

لإضافة عنصر معين فى ترتيب معين بنستخدم ال insert ميثود كالتالى مثلا هيتم إضافة Ayman فى الترتيب ال

3 (العنصر ال 4)

```
>>> students.insert(3, "Ayman")
>>> students
['Ahmed', 'Tina', 'Gina', 'Ayman', 'Marian', 'Omar', 'Waleed']
```

.pop(idx=-1)

بتقوم بحذف + اعادة عنصر فى ال list بإستخدام ال idx وفى حال عدم توفيره هيتم التطبيق على آخر عنصر بال list

```
>>> popped=students.pop()
>>> popped
'Waleed'
>>> popped=students.pop(2)
>>> popped
'Tina'
>>> students
['Ahmed', 'Ayman', 'Wael', 'Omar']
```

.reverse()

بيتم عكس الترتيب من الآخر للأول كالتالى

```
>>> students
['Ahmed', 'Ayman', 'Wael', 'Omar']
>>> students.reverse()
>>> students
['Omar', 'Wael', 'Ayman', 'Ahmed']
```

.sort()

بتقوم بترتيب العناصر

```
>>> students.sort()
>>> students
['Ahmed', 'Ayman', 'Omar', 'Wael']
```

.index(value)

الحصول على الترتيب الخاص باول ظهور للعنصر

```
>>> students.index("Ayman")
1
```

.count(value)

للحصول على عدد مرات ظهور عنصر معين فى ال list

```
>>> students.append("Ayman")
>>> students
['Ahmed', 'Ayman', 'Omar', 'Wael', 'Ayman']
>>> students.count("Ayman")
2
```

## List Comprehension

اذا اخدت كورس رياضيات من قبل فربما تكون واجهت ال List Comprehension

$$S = \{\, 2 \cdot x \mid x \in \mathbb{N},\ x \leq 10 \,\}$$

هنا تنقسم ال LC الى جزئين جزء ماقبل ال بايب -انبوبة- | وجزء مابعدها
ماقبلها يسمى function يتم تطبيقها على كل x
ومابعدها يسمى قائمة المدخلات وبعض الشروط

```
>>> [number*3 for number in range(20)]
[0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57]
```

فيما سبق قمنا بتطبيق ال LC فى بايثون حيث قمنا بتحديد ال function (وهى ضرب كل عدد فى 3)
لكل عدد ينتمى الى الفترة من صفر ل 20

```
Prelude> [number*3 | number <- [1..20] ]
[3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,48,51,54,57,60]
```

نفس الشئ بالنسبة لكود haskell اذا حيث تقوم بتطبيق العملية بحذافيرها ماقبل ومابعد البايب

وربما تقوم بتحديد شرط ما (هنا لكل الأعداد الزوجية)

```
>>> [number*3 for number in range(20) if number%2==0]
[0, 6, 12, 18, 24, 30, 36, 42, 48, 54]
```

كود haskell

```
Prelude> [number*3 | number <- [1..20], even number]
[6,12,18,24,30,36,42,48,54,60]
```

طبعا تقدر تستخدم الطرق العادية للفلترة ولكن دى on the fly solution

## Tuples

ال Tuple هى Container -حاوية- اخرى زى ال List ولكن الفرق هى إنها غير قابلة للتغيير..

التعريف بيتضع العناصر بين (  )

```
>>> t=(1, 2, 3, 4, 5)
>>> t
(1, 2, 3, 4, 5)
```

او تقدر تعرفها كالتالى مثلا

```
>>> t=1, 2, 3, 4, 5
>>> t
(1, 2, 3, 4, 5)
```

len(tup)

هتقوم بإعادة عدد عناصر ال tup  (نفس السلوك مع اى كونتينر)
احنا قلنا انها غير قابلة للتغيير يعنى لو حاولنا نعدل اى عنصر المفروض يحصل مشكلة

```
>>> t=1, 2, 3, 4, 5
>>>t[0]=9 #Try to set the first element to 9
```

هنا حاولنا نخلى العنصر اللى فى الترتيب الأول قيمته تساوى 9
رد بايثون هيكون مشابه للتالى

```
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    t[0]=9 #Try to set the first element to 9
TypeError: 'tuple' object does not support item assignment
```

## Strings

بايثون بتقدملك نوع من البيانات ليعبر عن ال String ولكن اولا ماهو ال String ؟
هو بكل بساطة مجموعة من الحروف

لغات مثل سى لاتعترف بال String ولكن تعتبره مصفوفة (او قائمة) من الحروف

انشاء string

تقدر تعرفه كالتالى

```
>>> astring="Hello, World!"
```

هنا تم انشاء متغير بإسم astring وتم تخزين القيمة Hello, World داخله
لاحظ لإنشاء اى سترينج استخدم علامات التنصيص " " او ' ' او """ """ هنشرح الفرق خلال الكتاب
زى ماأليست بتقدملنا خدمات تسهل علينا التعامل فكذلك ال strings
اولا اسم الداتاتايب هو str (اختصار string)

```
>>> type(astring)
<type 'str'>
```

len(string)

تقوم بحساب عدد الحروف الموجودة بال string

```
>>> len("Hello, World")
12
```

.capitalize()

تقوم بإعادة كائن جديد تحول فيه اول حرف إلى حرف uppercase

```
>>> "hello".capitalize()
'Hello'
```

.count(sub)

حساب عدد مرات تكرار مقطع معين

```
>>> "Hello".count('l')
2
```

.lower()

اعادة كائن بيه كل الحروف lowercase

```
>>> "HELlO".lower()
'hello'
```

.upper()

اعادة كائن بيه كل الحروف uppercase

```
>>> "helLo".upper()
'HELLO'
```

.swapcase()

بتقوم بإعادة كائن بعكس حالة الأحرف

```
>>> "hElLo".swapcase()
'HeLlO'
```

.title()

بتعيد كائن على صورة title -عنوان-

```
>>> "hello, world".title()
'Hello, World'
```

.startswith(sub)

بتختبر هل السترينج يبدأ بمقطع معين او لأ

```
>>> "hello".startswith("he")
True
```

.endswith(sub)

بتختبر هل السترينج ينتهى بمقطع معين او لأ

```
>>> "hello".endswith('lo')
True
```

سؤال: إيه اللى يخلينى استخدم tuple مكان list ؟ مع إنها بتقدملى امكانيات اقل؟
لأنك غالبا مش هتستخدمها ك list ولكن هتحتاج ميزة عدم التعديل عليها

.find(sub)

بتقوم بإعادة ال index الخاص بأول ظهور لمقطع معين

```
>>> s="Hello, World"
>>> s.find('W')
7
```

.strip([chars])

بتقوم بحذف ال chars من الستيرنج وفى حال عدم تحديدها هيعتبر انها المسافات

.lstrip([chars])

مثل الأولى ولكن من على اليسار فقط

rstrip([chars])

مثل الأولى ولكن من اليمين فقط

.isalpha()

بتختبر هل الستيرنج مكون من حروف الأبجدية او لأ

.isalnum()

بتختبر هل الستيرنج مكون من حروف من الأبجدية او ارقام او لأ

.isdigit()

بتختبر هل الحروف المكونة للستيرنج عبارة عن ارقام او لأ

.isupper()

بتختبر هل الستيرنج فى حالة uppercase او لأ

.islower()

بتختبر اذا كان الستيرنج فى حالة lowercase أو لأ

.isspace()

بتختبر هل الستيرنج دا مسافة

.istitle()

بتختبر هل الستيرنج دا title أو لأ

ويوجد العديد بكل تأكيد تقدر تطلع على باقى الميثودز فى الوثائق الخاصة بالبيثون

ملحوظة هامة ال string مش mutable!
لتستخدم mutable strings تابع فصل ال UserString

## UserString

المشكلة:

```
somestring="hola"
print somestring, ", ", id(somestring)

somestring += " guyZ"
print somestring, ", ", id(somestring)
```

الناتج

```
hola ,  3084138464
```

```
hola guyZ ,  3084161896
```

لاحظ اختلاف ال id -المعرف- الخاصة بالsomestring بعد اضافة مقطع ليه ، والسبب ان تم انشاء كائن جديد من ال str class وتم دمج القيمة القديمة + المقطع الجديد

الحل UserString Module

كما نعلم ان فى python ال Strings are immutable زى العديد من اللغات زى Java/C# .. etc
بمعنى إن اى تعديل هيتم على String مثل ال Concatenation -الدمج-بيحذف ال Variable -متغير- ويعمل Variable جديد كالتالى

```
>>> s="I'm a string"
>>> s
"I'm a string"
>>> id(s) # get the location
13511240
>>> s +=", and you ?"
>>> s
"I'm a string, and you ?"
>>> id(s) #get the location now!
13522272
```

لاحظ بعد ال Concatenation مع ال string الجديد إتغير ال location بتاعه .. لكن اللى حصل بالفعل هو إن اتعمل شئ مشابه للتالى

```
>>> s ="I'm a string"
>>> del s
>>> s ="I'm a string, and you?"
```

طب جميل ولكن على فرض إنك تريد تعمل mutable string او فى بعض اللغات إسمه StringBuffer/StringBuilder ؟

الحل هو إنك تستخدم class من UserString Module ودى Module -وحدة- موجودة عندك فى PythonPath/Lib/UserString.py

استدعى ال Module كالتالى

```
>>> import UserString as us
```

استخدمت as us لتعبر عن alias لل UserString module

ال Module بتتكون من 2 Classes هما UserString,MutableString
ال UserString هو ال base class و immutable تقدر تستخدمه بصورة مشابه لل str type
ال MutableString هو sub class من ال UserString و mutable

ملاحظة : ال MutableString معمول فيه override -اعادة تعريف- ل hash function لأنه طالما Mutable يبقة

unhashable ! -لاتقلق فى حال عدم فهمك للجملة السابقة-

```
def __hash__(self):
   raise TypeError, "unhashable type (it is mutable)"
```

ال MutableString فيه method بإسم immutable ودى بتدى return ل immutable string !

```
>>> m_string=us.MutableString("Hello, ")
```

بنعمل Object بإسم m_string

تحديد ال location

```
>>> id(m_string)
13511136
```

نعمل ref -مرجع- ليه

```
>>> ref_m_string=m_string
```

تحديد ال id ل ref هنلقيه نفس ال id الخاص ب m_string

```
>>> id(ref_m_string)
13511136
```

عمل دمج

```
>>> m_string += "CRUEL WORLD!"
```

نحدد ال id تانى هنلقيه مازال هو هو

```
>>> id(m_string)
13511136
```

ال id الخاص بال ref مازال زى ماهو

```
>>> ref_m_string
'Hello, CRUEL WORLD!'
>>> id(ref_m_string)
13511136
```

عمل Object ولكن immutable يعنى ال id بتاعه هيتغير فى اى اى تغيير هيتم عليه

```
>>> im_string=m_string.immutable() # Return immutable string!
>>> im_string
```

```
'Hello, CRUEL WORLD!'
```

تحديد ال id

```
>>> id(im_string)
13532856
```

عمل ref ليه وتحديد ال id

```
>>> ref_im_string=im_string
>>> id(ref_im_string)
13532856
```

عملية الدمج

```
>>> im_string +=" blah blah blah"
```

تحديد ال id هنلاقيه إنه إتغير

```
>>> id(im_string)
13553416
```

ال ref مازال زى ماهو مش تم عليه اى تغيير لأنه بيشير لمكان Object تم الإستغناء عنه واصبح اسم
im_string بيشير ل 'Hello, CRUEL WORLD! blah blah blah'

```
>>> ref_im_string
'Hello, CRUEL WORLD!'
>>> id(ref_im_string)
13532856
>>> im_string
'Hello, CRUEL WORLD! blah blah blah'
```

يفضل اعادة قراءة الفصل بعد قراءة جزئية الكائنات
ابحث عن موديلز مشابهه ل UserString

## Dictionaries

انشائه

```
>>> d={}
```

اسم -نوع البيانات- هو dict

```
>>> type(d)
<type 'dict'>
```

وبيخزن فيه البيانات على صورة Key, Values زى القاموس بالظبط (من هنا جت التسمية)

```
>>> d['Name']='Ahmed'
>>> d['Age']=19
>>> d['Sex']='m'
```

هنا خزنا فى القاموس الخاص بنا 3 keys وهما 'Name' , "Age', 'Sex'

.keys()

للحصول على ال Keys او الكلمات الدليلية فى صورة ليست

```
>>> d.keys()
['Age', 'Name', 'Sex']
```

.values()

للحصول على ال Values فى صورة ليست

.get(key)

بتعيد لك ال value الخاصة ب key معين

```
>>> d.get('Name')
```

```
'Ahmed'
```

فى حال عدم وجوده هيتم إعادة None

```
>>> print d.get('Team')
None
```

Indexing

تقدر تحصل على القيمة الخاصة ب key معين من خلال ال Indexing -الفهرسة-كالتالى مثلا

```
>>> d['Name']
'Ahmed'
```

او تقدر تضيف Key -مفتاح- جديد مثلا

```
>>> d['Lang']='Ruby'
```

او تقدر تعدل على قيمة موجودة

```
>>> d['Name']='Youssef'
```

.pop(key)

بتقوم بحذف ال Key وال Value الخاصة به من القاموس وبتعيدلك ال value

.update(d)

بتقوم بعمل تحديث للقاموس ببيانات قاموس d

```
d.update({'Lang':'Python', 'Singer':'Delta'})
>>> d
{'Lang': 'Python', 'Age': 19, 'Singer': 'Delta', 'Name': 'Youssef'}
```

.has_key(key)

بتختبر وجود key معين فى القاموس

```
>>> d.has_key('Lang')
True
```

.items()

بتعمل list مكونة من tuples بتشمل ال key, value فى صورة زوج

```
>>> print d.items()
[('Lang', 'Ruby'), ('Singer', 'Delta'), ('Name', 'Youssef'), ('Country', 'EG'), ('Age', 19), ('Sex',
'm')]
```

.iteritems()

بتستخدم غالبا فى ال Iterations (سنتعرض ليها لاحقا)

```
for key, val in d.iteritems():
        print key, " => ", val

Lang  =>  Ruby
Singer  =>  Delta
Name  =>  Youssef
Country  =>  EG
Age  =>  19
Sex  =>  m
```

مثال اخر

```
>>> i=d.iteritems()
>>> i.next()
('Lang', 'Ruby')
>>> i.next()
('Singer', 'Delta')
```

## Conditions

حياتنا مبنية على الإحتمالات والبرمجة مش خارج نطاقها.. مثلا اذا الخدمات اللى بتطلب تسجيل الدخول منك مبنية على احتمال "هل انت او لأ"

الصورة العامة

```
if condition as True then
        if_suite
```

مثلا "مع فارق التشفير وقواعد البيانات فى الإستخدام"

```
>>> if name=="ahmed" and password=="123456":
        print "Welcome ahmed"

Welcome ahmed
```

لاحظ هنا اختبرنا هل الإسم قيمته مساوية "==" لاحمد **و** الباسورد قيمته "123456" فإذا الناتج True يتنفذ البلوك اللى بعدها وهو print Welcome ahmed
فى حال ال ناتج الأساسى للشرط مش True يعنى False مش هيتنفذ حاجة.
تمام.. طب اذا حبينا نعالج موضوع ان الشرط يكون False ؟ بمعنى اننا لو غيرنا مثلا الباسورد لأى قيمة مخالفة ل 123456 ال condition قيمته هتكون False وفى الحالة دى مش هيحصل شئ.. ولكن نريد ان نعرف اذا ال condition كان False مثلا؟

تابع المثال التالى وهو إستخدام if, else
الصورة العامة

```
if condition as True then
        if_suite
else then
```

```
        else_suite
```

لاحظ المثال التالى

```
>>> name="ayman"
>>> password=147859
>>> if name=="ahmed" and password=="123456":
        print "Welcome ahmed" #if_suite
else:
        print "Welcome, Who R U?" #else_suite

Welcome, Who R U?
```

حسنا فى حال وجود عدة احتمالات قد تكون سليمة او لأ هنستخدم if, else if, else

```
if condition as True then
        if_suite
else_if condition as True then
        else if suite
else_if condition as True then
        else_if suite
else then
        else_suite
```

تابع المثال التالى

```
>>> if name=="ahmed" and password=="123456":
        print "Welcome ahmed" #if_suite
elif name=="tina" and password=="36987456":
        print "Welcome tina" #elif_suite
elif name=="ayman" and password==147859:
        print "Welcome, Ayman" #elif_suite
else:
        print "Who R U?"


Welcome, Ayman
```

الأول تم أختبار البلوك دا

```
if name=="ahmed" and password=="123456":
```

ولكنه مش True فيتعمل Escape للى بعده وهو

```
elif name=="tina" and password=="36987456":
```

ولكنه مش True بردو فيتعمل Escape للى بعده وهو

```
elif name=="ayman" and password==147859:
```

والبلوك دا قيمته True فيتنفذ البلوك تبعه وهو

```
        print "Welcome, Ayman" #elif_suite
```

ملحوظة:

```
نستخدم == لإختبار عملية التساوي
نستخدم = لعملية الإسناد
```

## Loops

مثل ال conditions معظم حياتنا مبنية على التكرار.. مثل كل يوم تصحى الصبح وتاخد دش وتفطر وتنزل شغلك وهكذا لحد الأجازة مثلا تنام للمغرب :D

طالما انت فى دراسة او شغل "مش فى اجازة"
اصحى الساعة 6
خد دش
افطر
انزل شغلك

الصيغة العامة

```
while condition as True do
        while_suite
```

تعالى نجرب ابسط لوب ممكن

```
>>> i=0
>>> while i<10:
        print "i: ", i
        i += 1


i:  0
i:  1
i:  2
i:  3
i:  4
i:  5
i:  6
i:  7
i:  8
i:  9
```

لاحظ ان الكود بيتنفذ كالتالى
الشرط هو ان i تكون اقل من 10 وطالما الشرط دا حقيقى (صحيح) هيتنفذ البلوك التابع لل loop -الحلقة او الدوارة-

```
        print "i: ", i
        i += 1
```

السبب فى إننا بنزود ال I اننا نخلى ال I توصل ل 10 بحيث ان ال loop تقف وإلا الشرط هيكون True للأبد.

```
Foreach element in container do
        for_suite
```

نطبقها على Containers -الحاويات- مثل ال string او list او tuple مثلا

```
>>> string="Hello, World!"
>>> for char in string:
        print char


H
e
l
l
o
,

W
o
r
l
d
!
```

معناها كالتالى : لكل (حرف) فى ال string اطبع (الحرف) دا

على فرض ان عندنا list كالتالى بإسم students

```
>>> students=["Ahmed", "Tina", "St0rM", "Salma"]
>>> for student in students:
        print student


Ahmed
Tina
St0rM
Salma
```

المعنى هنا: لكل Element -عنصر- او student موجود فى ال Container او ال students list اطبع ال عنصر دا

ملحوظة:

لطباعة العنصر الموجود فى ال Container بدون اضافة سطر جديد ضيف كومة لل print كالتالى

```
>>> string="Hello, World!"
>>> for  char in string:
        print char,

H e l l o ,  W o r l d !
```

raw_input

زى ماشفنا print وعرفنا انها مختصة بالطباعة هنشوف المسؤل عن الإدخال وهنا raw_input/input

الصيغة العامة

```
raw_input(prompt)
```

حيث ان ال prompt هى الرسالة اللى هتظهر للمستخدم
مثال

```
>>> name=raw_input("Enter your name: ")
Enter your name: ahmed
>>> print "Hola, ", name
Hola,  ahmed
```

س: ايه الفرق بين raw_input, input ؟
استخدامك ل input بيساوى بالظبط التالى

```
eval(raw_input(prompt))
```

مش فى احسن من التجربة العملية تابع المثال التالى

```
>>> val=raw_input("Enter: ")
Enter: 2+13+541
>>> print val
2+13+541
```

لكن مع استخدامنا ل input هيتعمل eval -اختصار ل evaluate- للمدخلات ك Python Expression كالتالى

```
>>> val=input("Enter: ")
Enter: 2+13+541
>>> print val
556
```

تم تحقيق ال 2+13+541 من خلال python واعادة الناتج ليك

*eval(expression)*

```
>>> eval("1+2")
3
```

eval بتاخد expression وتحاول اعادة الناتج ليك اذا كان ليه معنى

## Don't break up with me

بنستخدم break للخروج من حلقة عند استيفاء شرط معين (مثلا تمت قراءة كل البيانات من ملف فلاداعى لمحاولة القراءة او تم قراءة 100 رقم فردى فلا داعى للإستمرار)
لدينا كلمة مثل "hellopython" نريد ان نعرف موقع حرف ال t فيها فالفكرة ان نقوم بعمل حلقة على الأحرف ونختبر ماذا كان t او لا.. واذا كان t نقم بتخزين قيم المركز الحالى ونخرج من الحلقة "لعدم احتياجنا لها بعد الآن"

```
word="hellopython"

whereist=0
count = 0
while count < len(word):

    if word[count]=='t':
        whereist=count
        break
        #no need to keep going on
    count += 1
```

الناتج

```
striky@striky-desktop:~/workspace/pytut/src$ python tstbreak.py
Now count is  1
Now count is  2
Now count is  3
Now count is  4
Now count is  5
Now count is  6
Now count is  7
t was found at word[7]
```

## Our dance will continue

بتستخدم continue للهروب من الحلقة الحالية (ربما لعدم استيفاء عنصر الشروط المطلوبة للعمل عليه) واستكمالها على العنصر الذى يليه مثلا

```
tstvars=['123mx', 'hello', 'acc', '9']

for var in tstvars:
   if var[0].isdigit():
      continue #no work will be done on this item, maybe the next?
   else:
       print var, " => ", "is valid."
```

هنا نختبر كل عنصر من عناصر tstvars ماإذا كان يصلح ان يكون اسم متغير فى بايثون
فنقوم بعمل حلقة على العناصر

```
for var in tstvars:
```

ونختبر ماإذا كان يبدأ برقم (احد الشروط عدم  بدا تسمية المتغيرات فى بايثون برقم) فإذا كان رقم نهرب من الحلقة الحالية ونستكمل على العنصر التالى فى القائمة tstvar

```
   if var[0].isdigit():
       continue #no work will be done on this item, maybe the next?
```

تدريب:
اكتب برنامج لتسجيل الدخول بالبينات التالية

*user_name="Ahmed"*
*user_pass="123456"*

ل 3 محاولات وفى حالة الفشل تطلع رسالة ب
Account Suspended!
*أستخدم raw_input للحصول على الداتا من المستخدم

# *Chapter 4(Functions & Procedures)*

ال Functions واستخدامها اتسمى عليها paradigm كامل Functional Programming او Procedural Programming وكان/مازال شائع جدا للآن
لغات مثل السى وباسكال بتعتمد على ال Functions كحجر اساس

ايه هى ال Function او ال Procedure ؟
هى بلوك (قسم) من الأكواد اتكتب لإمكانية استخدامه اكثر من مرة

مثلا نريد ان نطبع رسالة كالتالى

Python rocks!

لأكتر من مرة فى برنامجنا لسبب ما.. هل يعقل انى اكتب

```
print Python rocks!
```

فى كل مرة ؟ ولنفرض انى كتبتها 20 مرة هل يعقل انى اذا حبيت اعدل التعبير بدل Python rocks ل Perl rocks انى اعدل فيه 20 مرة؟
من هنا جت اهمية ال Functions وهى تستخدم لتوفير وقت ومجهود وتطبيق مبدأ DRY

## *DRY: Don't Repeat Yourself*

كل اللى عليك هو انك تحط الرسالة فى function
حيث بنبدأ التعريف بالكلمة المفتاحية def وبعدها اسم الدالة rocks وبعدها قائمة المعاملات
الشرح الآن من خلال ال Python Shell بطريقة تفاعلية وبعد كدا فى جزئية انشاء الوحدات بنتعرض لعمل السكربت وكيفية استدعاءه

```
>>> def rocks():
        print "Python rocks!"
```

وتستدعيها كل ماتحب

```
>>> rocks()
Python rocks!
>>> rocks()
Python rocks!
>>> rocks()
Python rocks!
```

لاحظ انك اذا حبيت تعدل كلمة Python ل Perl مثلا مش هتحتاج تعدل فى اى شئ غير فى ال rocks function فقط تحولها للتالى

```
>>> def rocks():
        print "Perl rocks!"
```

لاحظت الفرق ؟توفير وقت ومجهود ومش كررت نفسك فى مليون سطر
طب تمام لكن فيه مشكلة لحد الوقتى وهى اننا اضطرينا نعدل جوا ال function ونستبدل كلمة مكان كلمة وهكذا فنريد ان نخليها ابسط فى الإستخدام بحيث انها تطبع اللى إحنا عايزنها تطبعه (ال دالة فيها متغير معين )

فالحل هو اننا نعيد تعريفها كالتالى مثلا

```
>>> def rocks(thing):
        print thing, "rocks!"
```

كدا هيتم طباعة ال argument اللى هيتمرر + كلمة rocks! كالتالى

```
>>> rocks("Python")
Python rocks!
>>> rocks("Perl")
Perl rocks!
```

جميل نفس الفكرة نريد ان نعمل function تجمع 2+3 وتطبع الناتج لينا

```
>>> def add():
        print 2+3
```

استخدمها كالتالى

```
>>> add()
5
```

لكن هل لاحظت شئ ؟ انها مقيدة بمعنى انى مش قادر استخدم غير 2+3 فقط طب افرض انا عايز احدد ارقام من عندى ايه الحل ؟
هممم نفس فكرة المثال اللى قبله انك تمرر الأرقام اللى تعجبك ك arguments لل add function ودا هيتم الأول
1- انك تعيد تعريف ال function كالتالى مثلا

```
>>> def add(first, second):
        print first + second
```

2- تستخدمها

```
>>> add(2, 3)
5
>>> add(3, 7)
10
```

ارسال ال arguments للدالة

```
def printArgs(first, second, third):
   print "First: ", first
   print "Second: ", second
    print "Third: ", third
```

لاحظ لإستخدام الدالة هنستدعيها كالتالى

```
printArgs("Hello", "Cruel", "World")
```

والناتج

```
First:  Hello
Second:  Cruel
Third:  World
```

ولكن على فرض انى عايز احدد قيم اساسية او حتى ادخل ال arguments -المعاملات- بطريقة عشوائية!؟
بكل بساطة تقدر تستدعيها كدا

```
printArgs(third="World", second="Curel", first="Hello")
```

بحيث انك تحدد قيمة كل عنصر بإستخدام الإسناد (لأنهم متغيرات واضحة للدالة ;)
ولتحديد قيم افتراضية ؟ مثلا عايز ادخل قيمتين او قيمة واحدة او حتى مش ادخل اى قيمة!!؟؟
تقدر تحدد دا من خلال تعريف الدالة نفسها مثلا كالتالى

```
def printArgs(first="Hello", second="Cruel", third="World"):
   print "First: ", first
   print "Second: ", second
   print "Third: ", third
   print "-"*20

printArgs("Bye") #Changes the first..
printArgs(second="") #only the second is set to “”
printArgs()
```

والناتج

```
First:  Bye
Second:  Cruel
Third:  World
--------------------
First:  Hello
Second:
Third:  World
--------------------
First:  Hello
Second:  Cruel
Third:  World
--------------------
```

وهكذا.. دا مفيد فى موضوع ال overloading -التحميل الزائد-لل function (بحيث ان يتم تنفيذ اكتر من وظيفه لنفس ال function إعتمادا على ال arguments المرسلة) تابع range
لحد الآن ال Functions غير مفيدة لأنها مش بتعيد قيمة يعنى مش تقدر تستفيد منها فى برنامجك انك تعمل كالتالى مثلا

```
>>> val=add(2, 3)
5
```

```
>>> print val
None
```

*ايه دا ؟ انا كنت متوقع ان val هتكون قيمتها 5!
دا هيتم فى حالة واحدة ان الطرف اليمين من ال expression تكون قيمته 5 لكن add(2, 3)) قيمته None لأنه بيعمل print لمجموع الرقمين لكن مش بيعمل بيهم return

*ايه None ؟

```
None=null=nil=Nothing
```

جميل طب ازاى اخلى قيمة ال Function تساوى مجموع الرقمين ؟
بسيطة اعمل return بالمجموع!
كالتالى مثلا

```
>>> def add(first, second):
        return first+second

>>> val=add(2, 3)
>>> print val
5
```

هنا return عبرت عن قيمة ال Function

كتير للأسف مش يعرف الفرق بين ال Functions وال Procedures
على كل حال اعتبرها كالتالى
ال Procedure هو اى Function مش ليها return

--لمبرمجى السى/++ والجافا اى Function ال return بتاعها void يبقة اسمها Procedure

بايثون بتقدم ليك العديد من ال Functions الجاهزة مثل

raw_input(prompt)

للحصول على الداتا من المستخدم

input(prompt)

زى ماقلنا هى بتستدعى التالى

eval(raw_input(prompt))

eval(expression)

بتحقق قيمة ال expression

abs(number)

بتعيد ليك ال Absolute value -القيمة المطلقة- وهى العدد بدون اشارة

```
>>> abs(10)
10
>>> abs(-10)
```

```
10
```

*تدريب:
اكتب Function بإسم getABS وبتاخد argument واحدة بإسم number ومشابهه ل abs

max(iterable)
هى function بتاخد container فى الغالب -اى شئ ممكن يطبق عليه foreach- وتعيد اكبر قيمة فيه كالتالى مثلا

```
>>> max([3, 4, 5, 6])
6
>>> max("Ahmed")
'm'
```

min(iterable)

هى العكس من max وهى بتعيد اصغر قيمة

```
>>> min("Ahmed")
'A'
>>> min([3, 4, 5, 6])
3
```

ملحوظة: اصغر قيمة للأحرف بتم بناء على قيمتها فى ASCII وتقدر تحصل عليها من خلال ord
وللحصول على الحرف من خلال القيمة بنستخدم chr

ord(char)

```
>>> ord("A")
65
>>> ord("a")
97
>>> chr(65)
'A'
>>> chr(97)
'a'
```

sum(seq)

بتستخدم للحصول على مجموع sequence ما كالتالى

```
>>> sum([1, 2, 3, 4, 5])
15
```

*تدريب اكتب function مشابهه ل sum وبإسم getSum وبتاخد sequence ك argument

oct(num)

بتعيد القيمة من النظام الثمانى لل num

```
>>> oct(15)
'017'
```

hex(num)

بتعيد القيمة من النظام الست عشرى لل num

```
>>> hex(15)
'0xf'
```

len(object)

فى الواقع len استخدامها بيختلف حسب نوع ال argument اللى هتتمرر ليها يعنى مثلا إذا كان string هيتم اعادة عدد الحروف وإذا كانت list هيتم إعادة عدد العناصر المكونة ليها وهكذا

```
>>> len("Ahmed")
5
>>> len([1, 2, 3, 4, 5, 6])
6
```

هنتعلم قريب ازاى نحدد الطريقة اللى هتتعامل بيها len مع ال objects بتاعتنا :D

round(f_num, digits)

هى function بتعمل تقريب ل f_num بعدد معين من الأرقام بيساوى digits كالتالى مثلا

```
>>> round(2678.367789)
2678.0
>>> round(2678.367789, 4)
2678.3678
```

copyright()

```
>>> copyright()
Copyright (c) 2001-2008 Python Software Foundation.
All Rights Reserved.

Copyright (c) 2000 BeOpen.com.
All Rights Reserved.

Copyright (c) 1995-2001 Corporation for National Research Initiatives.
All Rights Reserved.

Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
All Rights Reserved.
```

بتعرضلك ل copyright الخاص ببيثون

credits()

بتعرضلك ال credits

```
>>> credits()
    Thanks to CWI, CNRI, BeOpen.com, Zope Corporation and a cast of thousands
    for supporting Python development.  See www.python.org for more information.
```

range(end)

بتعيد ليك list من 0 لحد end بزيادة قيمتها 1

```
>>> range(10) #0 to 10
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

range(start, end)

بتعيد ليك List من start لحد end بزيادة قيمتها 1

```
>>> range(1, 5) #1 to 5
[1, 2, 3, 4]
```

range(start, end, step)

بتعيد ليك list من start لحد end بزيادة مقدراها step

```
>>> range(1, 20, 3) #1 to 20 with step=3
[1, 4, 7, 10, 13, 16, 19]
```

*تدريب اكتب function تقوم بعرض جدول ال ASCII

## To *args or not to *args

عنوان عجيب
مثال على دالة sum

```
def mysum(alist):
    total=0
    for i in alist:
        total += i

    return total
```

لإستخدامها هنمرر list من الأرقام الي الدالة ك argument كالتالى

```
print mysum([1, 2, 3, 4, 5])
```

مثال بإستخدام *args

```
def newsum(*args):
   total=0
   for i in args:
      total +=i
    return total
```

لإستخدامها: هنمرر الأرقام ك arguments للدالة كالتالى

```
print newsum(1, 2, 3, 4, 5)
```

انشاء دالة مشابة ل printf

```
def printf(fmt, *args):
   print fmt%args

printf("Name: %s, Age: %d", "Ahmed", 20)
#converted to  print "Name %s, Age: %d"%("Ahmed", 20)
```

لاحظ ان هيتم التحويل *args إلى دالة print لإستبدال ال %s, %d  (place holders) بالقيم المدخلة
*تدريب "اكتب sprintf بإستخدام بايثون

## To **kwargs or not to **kwargs

بتمرر مفاتيح keys وقيمها للدالة مثال

```
def newprintf(fmt, *args, **kwargs):
   print "fmt: ", fmt
   print "*args: ", args
   print "**kwargs: ", kwargs
   #do something usefu
   if kwargs.has_key("verbose"):
      print "Verbose..."
```

```
newprintf("This is some FMT", 1, 2, 3, 4, 5, 6, verbose=True, cleanup=True, use_ssl=False
```

الناتج

```
fmt:  This is some FMT
*args:  (1, 2, 3, 4, 5, 6)
**kwargs:  {'use_ssl': False, 'cleanup': True, 'verbose': True}
Verbose...
```

## Going global

انك توثق الدالة وبتعمل اية وبتاخد معاملات ايه شئ ممتاز فى اى برنامج لان اهمية التوثيق من اهمية الكود

```
def newprintf(fmt, *args, **kwargs):
    """Simple function to learn out howto use *, ** trick in functions."""
    print "fmt: ", fmt
    print "*args: ", args
    print "**kwargs: ", kwargs
    #do something usefu
    if kwargs.has_key("verbose"):
        print "Verbose..."
```

لاحظ ان اول سطر بعد التعريف بيعبر عن ال doc او وثيقة الدالة (ملف المساعدة بتاعها:d )

تقدر تستدعى ال .__doc__ من الدالة ليعرضلك جزئية المساعدة الخاصة بيها او تستدعى دالة help عليها .__doc__

```
>>> print newprintf.__doc__ #STR

Simple function to learn out howto use *, ** trick in functions.
Help on function newprintf in module __main__:
```

help

```
>>> help(newprintf)

newprintf(fmt, *args, **kwargs)
    Simple function to learn out howto use *, ** trick in functions.
```

على فرض ان عندنا متغيرات عامة فى الملف الخاص بنا ومتاجين نستخدمها فى دالة معينة هنعمل اية ؟
استخدم global كالتالى

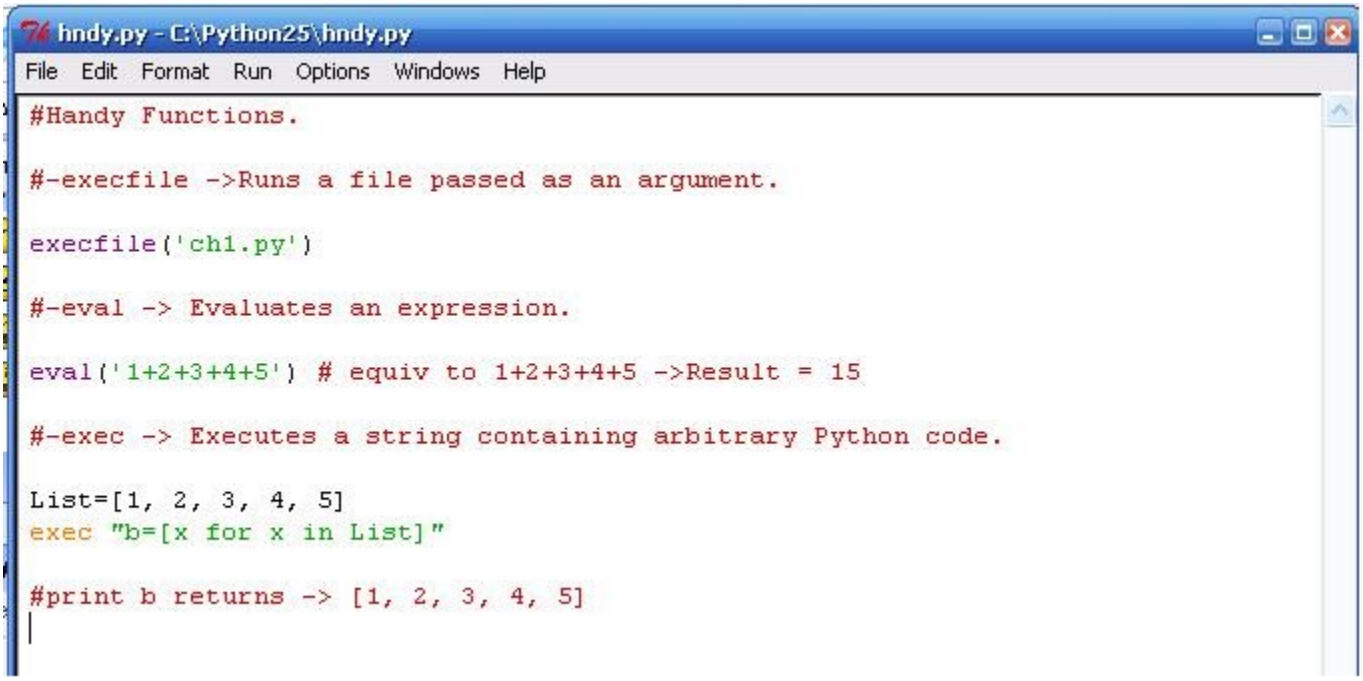
```
__DEBUG=True

def isdebug():
   global __DEBUG
   return __DEBUG

print "Debug? ", isdebug()

#output:
Debug?  True
```

كل المطلوب انك تعرف الدالة اللى هتستدعى فيها متغيرك العام بإنه global بإستخدام global ويليها اسم المتغير

او تقدر تستخدم globals() كقاموس للمتغيرات العامة كالتالى مثلا

```
__DEBUG=False

def isdebug():
   x, y, z=range(3)
   print locals()
   return globals()["__DEBUG"]

print isdebug()

#output:
#{'y': 1, 'x': 0, 'z': 2}
#False
```

لاحظ استخدام locals() هنا بتعيد قاموس ايضا بيعبر عن المتغيرات المحلية فى سياقها مثل x,y,z محليين فى السياق الموجودة فيه وهو الدالة isdebug




execfile(filepath)

تقوم بتنفيذ ملف بايثون

exec expr

تقوم بتنفيذ تعبير بايثون

## Lambda/Anonymous Functions

قراءتك لهذه الجزئية اختيارية

بايثون بتتيحلك استخدام الدوال المجهولة ودا باستخدام lambda مستعارة من لغة lisp

```
def getName(name):
   return name

anonyFunc=lambda name: name

print anonyFunc("Mido")
print getName("Mido")

#Output:
#Mido
#Mido
```

مابعد lambda هو ال args ومايليها هو ال return

```
def getSum(*args):
   return sum(args)

anonySum=lambda *args: sum(args)

print getSum(1, 2, 3, 4, 5)
print anonySum(1, 2, 3, 4, 5)

#Output:
15
15
```

map

بتطبق function معينة على مجموعة من العناصر

```
>>> print map(lambda w: w.upper(), ["ahmed", "mostafa", "omar"])
['AHMED', 'MOSTAFA', 'OMAR']
```

هنا هيتعمل ريترن -اعادة- بنسخة من العناصر بعد التعديل (التحويل للحروف الكبيرة)

استخدام lambda مش واضح بالنسبة ليك مش مشكلة استبدلها كالتالى

```
def toupper(w):
   return w.upper()

users=["ahmed", "mostafa", "omar"]
print map(toupper, users)

#output:
#['AHMED', 'MOSTAFA', 'OMAR']
```

صحيح ايه ال w اللى كانت فى lambda وموجود فى toupper ك parameter دى ؟
ال w دى بتعبر عن كل عنصر فى ال sequence هتطبق عليه الدالة

filter

بنستخدمها لتصفية sequence معينة هنشوف مثال

```
numbers=range(20)
print filter(lambda i: i&1, numbers) #odds.
print filter(lambda i: not i&1, numbers) #evens

#output:
#[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

لاحظ هنا فى المثال عندنا مجموعة ارقام من 0 ل 19

هنستخدم قاعدة n & 1 لإختبار هل الرقم فردى او زوجى (او اى طريقة تعجبك) زى ماشايفين فى المثال بإستخدام lambda وبالفعل بيتم التصفية بناءا على القاعدة اللى حاطينها فى ال function

المثال بدون استخدام lambda

```
def isodd(i):
   if i&1:
      return True
   return False

def iseven(i):
   return not isodd(i)

print filter(isodd, numbers) #odds
print filter(iseven, numbers)#evens

#output:
#[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

اماكن استخدام lambda ؟ يفضل تستخدمها مع ال properties كبديل لل getters (تابع الفصل القادم او فى دالة ب سطر واحد)

# Chapter 5 (OOP)

الموضوع هيتحول لصورة ممتعة اكتر بمراحل عن الصفحات اللى فاتت
الدنيا حولينا زى ماهى كلها احتمالات ولوبس فهى كلها OOP X OOP

اولا يعنى ايه OOP ؟
هى اختصار ل Object Oriented Programming وهى اسلوب مختلف فى البرمجة عن اسلوب ال Procedural Programming زى اللى فى السى مثلا وهو بكل بساطة = سهولة وكفاءة اكثر بمراحل من ال Procedural Programming وبكل تأكيد اكتر تنظيم!

اذا بصيت حواليك هتلقة الدنيا كلها عبارة عن Objects عربية انسان عصفورة طيارة قطة كل دول Objects

هندرس على فرض انك بتحدد تصميم لإنسان بتيجى بورقة وقلم كدا وتكتب

ال fields هى (كل مايستخدم فى وصف الإنسان)
اسم
سن
نوع
لون
طول
وزن
---
---

ال methods هى (كل مايؤديه الإنسان)
يتحرك
ياكل
يشرب
ينام
---
---
ال properties هى عبارة عن Encapsulation (تغليف) لل fields لمجرد حماية الصفات الخاصة بالإنسان
فهتكون كالتالى
اسم
سن
نوع
لون
طول
وزن
--
--
حول التغليف

| |
|---|
| اذا هناك مريض ويأخذ كبسولة -تعالجه من البرد- فهو يتعامل مع الكبسولة ولايتعامل مع المواد اللتى تغلفها الكبسولة ، تحتاج لصلاحيات وصول وفهم للتعامل ربما فى المختبر؟ |

هنتعرض ليها بالتفصيل ان شاء الله فى مثالنا

الصورة المبدأية للصف class الخاص بنا هتكون كالتالى

```
class Human(object):

    def __init__(self, name, color, sex, height, weight):
        self.name=name
        self.color=color
        self.sex=sex
        self.height=height
        self.weight=weight

    def eat(self):
        #code to eat.
        pass

    def drink(self):
        #code to drink.
        pass

    def sleep(self):
        #code to sleep.
        pass

    def play(self):
        #code to play.
        pass
```

*لكتابة اي صف class بنبدأ التعريف بكلمة class
*اسم الصف class يكون بادئ Uppercase
*الصف class بتاعك لازم يورث من Object (هنتكلم عن الوراثة) ولكن يكفيك ان اى انسان ماهو إلا object واى class فى الدنيا ماهو إلا object فيبقة اكيد هيورث صفات ال Object.

بدأنا ب ال Constructor -مشيد او بناء- وهنا فى بايثون بيكون عبارة عن function بإسم __init__
*لاحظ اى method تبدأمعاملاتها ب self .. تقدر تستبدل self بأى كلمة تناسبك ولكن مجتمع بايثون مرتبط ب self فإلتزم بالقواعد.

نبدأ نجهز ال fields بتاعت ال class زى name, sex, color,.. etc
بإننا نعمل field بكل إسم self.name, self.sex, self.color

ونسند له القيم اللى تم تمرريها من ال constructor كالتالى

```
    def __init__(self, name, color, sex, height, weight):
        self.name=name
        self.color=color
        self.sex=sex
        self.height=height
        self.weight=weight
```

جميل تعالى نختبر ال class الخاص بنا

```
ahmed=Human("Ahmed", "White", "M", 178,70)
```

1- اننا ننشئ object من ال Human class ونمرر الداتا اللى هنوصف بيها ال object دا
2- جملة print بسيطة لعرض ال fields

```
print ahmed.name
print ahmed.sex
print ahmed.height
print ahmed.weight
print ahmed.color
```

جميل الكود الخاص بنا 10/10 ولكن فيه مشاكل!

```
ahmed.name=17777777
print ahmed.name
```

تخيل حد يبقة اسمه عبارة عن ارقام !!! كدا الكود الخاص بنا فيه مشكلتين
1- انه مكشوف
2- انه غير آمن اى حد يقدر يحط اى قيم على مزاجه ودا اسمه تهريج
فقدامنا حل جميل جدا وهو اننا نستخدم اسلوب ال get/set ودا اسلوب شهير جدا لحماية الكود
ولكن ازاى نحمى الكود وهو مكشوف ؟ جميل جدا كدا انت بقيت ماشى معايا صح لازم نمنع الأكسس للمستخدم على ال fields بتاعتنا بس ازاى؟!
بسيطة اسبق اسم كل field ب 2 underscores كالتالى مثلا

```
    def __init__(self, name, color, sex, height, weight):
        self.__name=name
        self.__color=color
        self.__sex=sex
        self.__height=height
        self.__weight=weight
```

كدا ال fields بقت private مش فى حد يقدر يتعامل معاها غير ال class نفسه فى عملياته الداخلية لكن المستخدم الخارجى لأ
ضيف 2 methods لكل field واحدة get والتانية set كالتالى

```
    #Getters
    def get_name(self):
        return self.__name

    def get_color(self):
        return self.__color

    def get_sex(self):
        return self.__sex
```

```
    def get_height(self):
        return self.__height

    def get_weight(self):
        return self.__weight

    #Setters
    def set_name(self, new_name):
        self.__name=new_name

    def set_color(self, new_color):
        self.__color=new_color

    def set_height(self, new_height):
        self.__height=new_height

    def set_weight(self, new_weight):
        self.__weight=new_weight

    def set_sex(self, new_sex):
        self.__sex=new_sex
```

1- تعالى نختبر موضوع حماية ال fields كالتالى مثلا

```
>>> print ahmed.__name

AttributeError: 'Human' object has no attribute '__name'
```

هتلقة error كالتالى

2- تعالى نجرب نتعامل مع ال fields من خلال ال getters/setters كالتالى مثلا

```
ahmed=Human("Ahmed", "White", "M", 178,70)
print ahmed.get_name()
print ahmed.get_color()

ahmed.set_name("Youssef")
print ahmed.get_name()

ahmed.set_name(141241) #Wut?
print ahmed.get_name()

#Output:
Ahmed
White
Youssef
141241
```

جميل جدا ولكن بردو set_name مش عملت حاجة سمحت ان name ياخد قيمة رقم!

طب وإيه المشكلة ؟ عدل الكود كالتالى مثلا..

```
def set_name(self, new_name):
    if isinstance(new_name, str):
        self.__name=new_name
    else:
        print "new_name ain't a string!" #and do nothin
```

هنا بنختبر هل ال new_name عبارة عن كائن من str او لأ
بإستخدام isinstance

او تقدر تعدل اسلوب ال icheck ليكون  كالتالى

```
    if type(new_name)==str:
        self.__name=new_name
```

"يعنى هل هو string او لأ"
بإستخدام ال type function بدل isinstance

اذا كان string يبقة عادى نعدل الإسم اذا لأ تطلع رسالة new_name ain't a string ومش يتم اى تعديل!
تقدر تضيف ال rules اللى تناسبك مثلا الطول بتاعها وهل يشمل ارقام او لأ وهكذا... مع باقى ال fields
وهى دى عملية ال encapsulation! وهى بإختصار حماية ال data members او fields او ال attributes
الخاصة بال class بإستخدام getters/setters

جميل الإسلوب دا صح مش كدا ؟
لكن بايثون بتقدملك اسلوب ابسط لهندسة ال class بتاعك واستخدامه وهو إستخدام ال Properties
فاكر لما قلنا انها عبارة عن encapsulation لل fields ؟ تمام ال properties بتعمل class لل
getter/setter مباشرة

```
ahmed.name=new_name
```

بدلا من

```
ahmed.set_name(new_name)
```

```
ahmed.name
```

بدلا من

```
ahmed.get_name()
```

الله! ايه دا انت رجعت تانى لموال ال fields مش كنا قلنا اننا لازم نخليها private ونبعدها عن المستخدم؟!
تمام انا قلت كدا بس شكلك مش مركز لأننا بنتكلم عن ال properties :)

هيكون فى property مثلا بإسم name تقوم بشغل ال get_name و ال set_name كالتالى مثلا

```
name=property(fget=get_name, fset=set_name, doc="Gets/Sets the name.")
```

fget بتعبر عن الميثود المسئولة عن ال get وهنا هتكون get_name وهى بتستدعى فى حال

```
object.name
```

fset بتعبر عن الميثود المسئولة عن ال set وهنا هتكون set_name وهتستدعى فى حال

```
object.name=new_name
```

doc بتعبر عن وصف ال property

```
ahmed=Human("Ahmed", "White", "M", 178,70)
print ahmed.get_name()
print ahmed.name
ahmed.name="Youssef"
print ahmed.name
ahmed.name=979878 #uses the get_name rules!
```

طب كدا فى حاجة ايه الهدف من التكرار فى ؟ ليه يكون عندى name, get_name, set_name اسمح للمستخدم انه يستخدمهم ؟
تمام بكل بساطة اعمل ال get_name, set_name ك private method واتعامل بيها داخل الصف وخلى ال name ك property ظاهرة للمستخدم ويتعامل معاها بدل مايتعامل مع 2 methods

ازاى اخليهم private ؟ بكل بساطة اسبق اساميهم ب 2 underscores

ندخل فى ال Magic Methods

بداية هى كل method مبتدئة ومنتهية ب __ مثلا __init__
الفائدة: هى بتيح ليك تعريف سلوك التعامل مع ال Builtin Functions زى len مثلا!
بتتيح ليك دعم ال Operator Overloading هنشوف كل الكلام دا بالتفصيل

1- __init__
هى ميثود مسؤلة عن تجهيز ال fields فى حال انشاء ال object
--منتشرة بإسم Constructor

```
>>> class Human(object):
        def __init__(self, name):
                #Initialize the fields.
```

```
        self.name=name
        self.hands=2
```

يتم استدعائها فى حال الإنشاء ل object

```
>>> h1=Human("sami")
>>> h1.name
'sami'
>>> h1.hands
2
```

فى تعليق اضافى لما ندخل فى ال Inheritance

2- __getitem__(self, key), __setitem__(self, key, val)

على فرض ان عندنا صف بإسم MyDict

```
>>> class MyDict(object):
    def __init__(self, d={}):
        self.__d=d
    def __getitem__(self, key):
        if key in self.__d.keys():
            return self.__d[key]
    def __setitem__(self, key, val):
        self.__d[key]=val
```

احنا مثلا لانريد نتعامل مباشرة مع ال self.__d ولكن نريد ان نتعامل معاه من خلال الصف او عملية ال indexing

```
>>> md=MyDict({'Name':'Ahmed', 'Sex':'m'})
```

نستخدم __getitem__

```
>>> md['Name']  #Call __getitem__('Name')
'Ahmed'
```

نستخدم __setitem__

```
>>> md['Name']='Youssef'
>>> md['Name']
'Youssef'
```

3- .__len__(self)

بيها بنحدد سلوك الصف الخاص بنا مع الدالة الشهيرة len

```
>>> class Lener(object):

        def __init__(self, s, alist):
                self._s=s
                self._list=alist
        def __len__(self):
                return len(self._s)
```

هنا بتعريفنا لل __len__ قمنا بتحديد السلوك فى حال استخدام len مع اى كائن مع الصف دا وهنا هنخليها تعيد عدد حروف ال سترينج self._s

```
>>> l=Lener("Ahmed Youssef", ["Tina", "Salma"])
>>> len(l)  #Calls __len__
13
```

اذا اعدنا تعريفها لتكون كالتالى مثلا

```
        def __len__(self):
                return len(self._list)
```

فعند استدعاء len على اى كائن من النوع Lener هيتم اعادة عدد عناصر ال self._list

```
>>> l=Lener("Ahmed Youssef", ["Tina", "Salma"])
>>> len(l)
2
```

4- .__iter__(self)

بتحدد سلوك الصف من خلال لتعريفك ل generator وال iterations وتحديدا التعامل مع for loop

```
>>> class Tech(object):
        def __init__(self, langs, nums):
                self._langs=langs
                self._nums=nums
        def __iter__(self):
                for lang in self._langs: yield lang


>>> t=Tech(['Python', 'Ruby', 'Rebol'], [1500, 1414, 12515])
>>> for lang in t:
        print lang
```

للتوضيح اكثر

```
>>> i=iter(t)
>>> i.next()
'Python'
>>> i.next()
'Ruby'
>>> i.next()
'Rebol'
```

*ملحوظة: لازم تتعمل على contatiner

## Operator Overloading

1+4 دى إستخدام ال+Operator  وهو إن يجمع عددين
2*1 إستخدام ال Operator * هنا إنه يضرب عددين
1-2 إستخدام ال – Operator هنا إنه يطرح عددين ولكن !

هل ينفع يكون ل Operator اكثر من إستخدام ؟
اها مثلا + Operator بيستخدم فى عمل دمج بين ال Strings

```
>>> s1='Hello, '
>>> s2='World!'
>>> s=s1+s2
>>> s
'Hello, World!'
```

يعنى إستخدمنا ال + Operator فى وظيفة اخرى غير الجمع وهى الدمج دى بإختصار هى ال Overloading Operators .. يعنى يكون ل Operator اكثر من إستخدام.

فى Special Methods او بتسمى احيانا بال Magical Methods هى اللى بتوفرلنا موضوع ال Operator Overloading دا + بعض الأشياء الأخرى
__add__ للجمع
__sub__ للطرح
__mul__ للضرب وهكذا

فلنفترض إن عندى class وليكن Worker مثلا

```
class Worker(object):

    def __init__(self, name, work_hours):
```

```
        self.name=name
        self.work_hours=work_hours
```

وانت عايز تعمل زيادة لساعات العمل work_hours او نقصان او مضاعفة ؟!
فى عدة حلول زى إنك تعمل Methods 3 كالتالى مثلا

```
    def increment_workinghours(self, hours):
        self.work_hours += hours
       return self.work_hours

    def decrement_workinghours(self, hours):
        self.work_hours -= hours
       return self.work_hours

    def mul_workinghours(self, hours):
        self.work_hours *= hours
       return self.work_hours
```

حل آخر : هو إنك تعمل Overload لل Operators ال + و – و * كالتالى

```
    def __add__(self, hours):
        self.work_hours += hours
       return self.work_hours
    def __sub__(self, hours):
        self.work_hours -= hours
       return self.work_hours

    def __mul__(self, hours):
        self.work_hours *=hours
        return self.work_hours
```

هيكون صورة الصف كالتالى

```
class Worker(object):

    def __init__(self, name, work_hours):
        self.name=name
        self.work_hours=work_hours

    def increment_workinghours(self, hours):
        self.work_hours += hours
        return self.work_hours

    def decrement_workinghours(self, hours):
        self.work_hours -= hours
```

```
        return self.work_hours

    def mul_workinghours(self, hours):
        self.work_hours *= hours
        return self.work_hours

    def __add__(self, hours):
        self.work_hours += hours
        return self.work_hours

    def __sub__(self, hours):
        self.work_hours -= hours
        return self.work_hours

    def __mul__(self, hours):
        self.work_hours *=hours
        return self.work_hours
```

اعمل Object من ال Class وليكن w

```
>>> w=Worker('EVAN', 4)
>>> w.increment_workinghours(3)
7
>>> w.decrement_workinghours(2)
5
>>> w.mul_workinghours(2)
10
```

انا شايف إن الإسلوب دا ممل جدا مع إنه احيانا بيكون اءمن بعض الشئ ولكنه ممل!

اعمل Object تانى وليكن w1

```

>>> w1=Worker('ANN', 5)
>>> w1+2
7
>>> w1-4
3
>>> w1*5
15
```

جدول بكل المعاملات + ال Magic Methods الخاصة بيهم لتعريفهم

```
+ __add__, __radd__
```

- __sub__, __rsub__
* __mul__, __rmul__
/ __div__, __rdiv__, __truediv__ (for Python 2.2),
__rtruediv__ (for Python 2.2)
// __floordiv__, __rfloordiv__ (for Python version 2.2)
% __mod__, __rmod__
** __pow__, __rpow__
<< __lshift__, __rlshift__
>> __rshift__, __rrshift__
& __and__, __rand__
^ __xor__, __rxor__
| __or__, __ror__
+= __iadd__
-= __isub__
*= __imul__
/= __idiv__, __itruediv__ (for Python version 2.2)
//= __ifloordiv__ (for Python version 2.2)
%= __imod__
**= __ipow__
<<= __ilshift__
>>= __irshift__
&= __iand__
^= __ixor__
|= __ior__
== __eq__
!+, <> __ne__
> __gt__
< __lt__
>= __ge__
<= __le__

# *Chapter 6 (Inheritance)*

العلم كله مبنى على الوراثة والإكمال من حيث انتهى الآخرون

تخيل ان عندنا صف Human كالتالى

```
class Human(object):

    def __init__(self, name, sex):
        self._name=name
        self._sex=sex

    def _set_name(self, name):
        self._name=name

    def _set_sex(self, sex):
        self._sex=sex

    name=property(fget=lambda self:self._name, fset=_set_name)
    sex=property(fget=lambda self:self._sex, fset=_set_sex)
```

وعندنا صف Employer كالتالى

```
class Employer(object):

    def __init__(self, name, sex, salary):
        self._name=name
        self._sex=sex
        self._salary=salary

    def _set_name(self, name):
        self._name=name

    def _set_sex(self, sex):
        self._sex=sex

    def _set_salary(self, salary):
        self._salary=salary


    name=property(fget=lambda self:self._name, fset=_set_name)
    sex=property(fget=lambda self:self._sex, fset=_set_sex)
    salary=property(fget=lambda self: self._salary, fset=_set_salary)
```

اكيد لاحظت ان ال Employer هو Human ولكن مش فيه زيادة عن ال Human غير ال salary attribute و ال salary setter وال salary property

يعنى نقدر نقول ال
Employer ***is-a*** human
فكل اللى عليك انك تحسن الكود بحيث ان ال Employer يورث كل الصفات + الميثودز الموجودة بال Human ويضيف عليه المميزات الخاصة بيه زى ال salary كالتالى مثلا

```
class Employer(Human):
    def __init__(self, name, sex, salary):
        Human.__init__(name, sex)
        self._salary=salary

    def _set_salary(self, salary):
        self._salary=salary

    salary=property(fget=lambda self: self._salary, fset=_set_salary)
```

تعالى نتكلم بمثال اوضح وقريب من العالم الحقيقى بعض الشئ وعلى المثال السابق
إنسان وموظف ومدير
الموظف ماهو إلا إنسان والمدير ماهو إلا انسان مش كدا ؟ تعالى الأول نعرف ال Human class الخاص بنا

```
class Human(object):

    def __init__(self, name, color, sex):

        #Data members..
        self.__name=name
        assert sex in Gender.Options #Male or Female only.
        self.__sex=sex
        self.__color=color

    #getters/setters...
    def getName(self):
        return self.__name


    def setName(self, value):
        self.__name = value


    def getSex(self):
        return self.__sex


    def setSex(self, value):
        self.__sex = value
```

```
    def getColor(self):
        return self.__color


    def setColor(self, value):
        self.__color = value

    #properties...
    name = property(getName, setName, None, "Gets/Sets name.")

    sex = property(getSex, setSex, None, "Gets/Sets sex.")

    color = property(getColor, setColor, None, "Gets/Sets color.")

    __str__=lambda self: "<Human object: %s >"%self.__name

    __unicode__=__str__

    #methods..
    def eat(self):
        #Eating
        pass

    def drink(self):
        #Drinking
        pass

    def sleep(self):
        #Sleeping
        pass
```

هنا عرفنا صنف جديد بيعبر عن الإنسان وليه متغيرات داخلية زى الإسم واللون والنوع وبعض الميثودز لمعالجتهم وميثودز اخرى مثل eat, drink, sleep

لاحظ ان النوع لازم يكون موجود فى Gender.Options المعرفة كالتالى

```
class Gender(object):
    Male, Female="Male", "Female" #0, 1 whatever!
    Options=(Male,Female)
```

الإسلوب دا يقدر يفيدك لما تيجى عايز تعمل type لنوع صامت او جامد مثلا النوع او الألوان وهكذا
لاحظ ال __str__ دى magic method بيتم استدعائها عند استدعاء print او حتى ال casting بإستخدام str()
لاحظ اننا خلينا __unicode___ نفس المعنى من str (ويفضل انك تستخدم ال unicode على طول الأبلكيشن بتاعك)

تقدر تعرفهم اكيد كالتالى

```
def __str__(self):
        ....

def __unicode__(self):
        return self.__str__()
```

نيجى للموظف

```
class Employer(Human):

   def __init__(self, name, color, sex, salary, firm):
     #Construct the human with (name, color, sex)
      Human.__init__(self, name, color, sex)
      #Superize it :)
      #super(Employer, self).__init__(name, color, sex)

      self.__salary=salary

      self.__firm=firm

   def getFirm(self):
      return self.__firm


   def setFirm(self, value):
      self.__firm = value


   firm = property(getFirm, setFirm, None, "Gets/Sets the firm.")

   def getSalary(self):
      return self.__salary


   def setSalary(self, value):
      self.__salary = value


   salary = property(getSalary, setSalary, None, "Gets/Sets salary.")

   __str__=lambda self: "<Employer object:(%s, %d) >"%(self.getName(), self.salary)
#super(Employer, self)

   def eat(self):
      print "This is my break (eat or having fun hummm?"
```

```
    def sleep(self):
        print "Hi, it's me sleeping!"
```

هنا بننشئ صف جديد مشتق من ال Human وبنجهزه بالبيانات الخاصة بال Human (اسم ولون ونوع) والبيانات الخاصة بالموظف (المرتب والشركة) ومتغيرات تعالج الحالة

- اعدنا تعريف بعض الميثودز(الطرق) الخاصة بال Human بمعنى انهم اصبحت overriden

نيجى للمدير ومالمدير الا موظف ولكن له صلاحيات اعلى زى مثلا اعطاء علاوة او فصل موظف وهكذا

```
class Manager(Employer):

    def __init__(self, name, color, sex, salary, firm):
        #Employer.__init__(self, name, color, sex, salary, firm)
        super(Manager, self).__init__(name, color, sex, salary, firm)
        #print self.__dict__
        #{'_Human__color': 'white', '_Employer__salary': 200000, '_Human__name': 'Wael',
'_Employer__firm': 'Sun', '_Human__sex': 'Male'}

    def raiseSalaryOf(self, emp, theraise):
        assert (isinstance(emp,Employer) and emp.firm==self.firm)
        emp.salary += theraise
```

لاحظ انك لو طلبت .__dict__ لأى صف هيعرضلك قاموس dictionary فيه القيم اللى وراثها من المتغيرات الداخلية والمقطع الأول منها بيعبر عن منين موروثة

تعالى ننفذ عالمنا الصغير دا كتطبيق سريع

```
if __name__=="__main__":

    ahmed=Employer("ahmed", "white", Gender.Male, 50000, "Google")
    omar =Employer("omar", "black", Gender.Male, 40000, "Sun")
    tina =Employer("christina", "white", Gender.Female, 50000, "Google")
    emps=(ahmed, omar, tina)
    wael=Manager("Wael", "white", Gender.Male, 200000, "Sun")
    wael.raiseSalaryOf(omar, 9000)



    print issubclass(Employer, Human)
    print isinstance(ahmed, Human) #Ahmed is a Human..
    print Human.__bases__ #inherits object.
    print Manager.__bases__ #Inherits Employer..
```

لاحظ ان issubclass(C, B) بتسأل هل الصف C مشتق من B او لأ
لاحظ isinstance(object, Type) بتسأل هل ال object دا تم انشاءه من الصف Type او لأ
لاحظ ان __bases__ بتعبر عن الأب (او الأباء لو ورثت من اكتر من صف)
احنا استخدمنا assert فى كذا جزئية لكن ايه هى assert ?
Assert بتفيد فى عمل تصحيح سريع بحيث انك تضمن عدم التنفيذ فى حال وجود خطأ

امتة استخدم ASSERT ? استخدامها مقترن بحالة التطوير والمحل مثلا مش هتعمل كود فيه assert statement للمستخدم النهائى ، لكن ممكن تحطها فى مكتبة هيستخدمها مطور (هيراعى المتطلبات مظبوط) غير كدا استخدم ال Exceptions, Errors

**الوراثة المتعددة**

```
class Wolf(object):
    def __init__(self):
        self.__bite=True


    can_bite=lambda self:self.__bite

    def fullmoon(self):
        print "Wooooooooooooooooooooooo"

    def bite(self, h):
        print "I'm cursed"

class Werewolf(Human, Wolf):

    def __init__(self, name, color, sex):
        Human.__init__(self, name, color, sex)
        Wolf.__init__(self)

وصف بإسم مستذئب بيورث الإنسان والذئب wolf تابع مثالنا السابق هنا عندنا صف بإسم
    yasser=Werewolf("yasser", "black", Gender.Male)
    print yasser.__dict__
    print "Can bite? ", yasser.can_bite()
    yasser.bite(ahmed)
    print Werewolf.__bases__
الناتج
```

```
{'_Human__color': 'black', '_Wolf__bite': True, '_Human__name': 'yasser', '_Human__sex': 'Male'}
Can bite?  True
I'm cursed
(<class '__main__.Human'>, <class '__main__.Wolf'>)
__bases__ الأباء للمستذئب زى ماشفت فى ناتج
(<class '__main__.Human'>, <class '__main__.Wolf'>)
```

# *Chapter 7 (Exceptions, Errors)*

مش فى شئ كامل وطالما كتبت كود توقع انك هتلقة فيه مشكلات!  مثلا انقطاع الإتصال مع الداتابيز او مشكلات فى النتورك او مكتبيات غير متوافرة او كود خاطئ SyntaxError او بيانات دخلها المستخدم بصورة خاطئة او خطئ رياضي او او

اولا ايه هى ال exceptions ؟ هى شئ حصل يتسبب فى تغيير مسار برنامجك "اللى انت خططه"
المفتاح للموضوع دا 4 حاجات (مش شرط يكون خطأ) ولكن شئ غير محسوب
1- try

2- except
3- finally
4- raise

try مش فيها شئ عجيب كل المطلوب منك هو انك تضيف الجزء اللى هتشك ان ممكن يحصل فيه exception

except: بتصطاد فيها ال exception وتعالجه

finally : الكود المرتبط بيها هيتم تنفيذ دائما حتى لو مش عالجت ال exception بتكون غالبا لإغلاق الريسورسز المفتوحة (مثلا فايل او كونكشن معين)
raise: هى المسئولة عن اطلاق ال exceptions دى

تعالى نشرحها بمثال شهير جدا
هنحاول نقسم عدد على صفر

```
>>> 1/0
```

رد بايثون

```
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    1/0
ZeroDivisionError: integer division or modulo by zero
>>>
```

لاحظ السطر ZeroDivisionError: integer division or modulo by zero
بيتكلم فيه عن حدوث Error للقسمة على صفر + وصف ال Error
طب جميل جدا انا كدا عرفت ان ممكن يتقسم على صفر طب انا عايز اصطاد ان حد حاول يعمل كدا
1- ضع الكود بتاعك فى try بلوك
2- هندل -تعامل مع- ال exception فى except بلوك

```
try:
        print 1/0
except ZeroDivisionError, e:
        print e.message
#output: integer division or modulo by zero
```

مثال آخر ادخال قيم غير سليمة او منطقية

```
>>> def sayHi(name):
        if not isinstance(name, str):
                raise ValueError("name ain't string.")
        else:
                print "Hi, %s"%name
```

هنا عرفنا function بإسم sayHi بتاخد بارمتر واحد بإسم name (منطقيا لازم يكون str) وإلا مثلا مش هنقول hi لرقم!؟ فى حالة ان المستخدم هيبمرر رقم او اى شئ غير str هنعمل raise لإيرور بإسم ValueError وبرسالة name ain't string (معرف مسبقا)

```
>>> sayHi(9)

Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    sayHi(9)
  File "<pyshell#26>", line 3, in sayHi
    raise ValueError("name ain't string.")
ValueError: name ain't string.
```

تعالى نعمل مثال لإنشاء exception خاص بيك
نريد ان نكتب فنكشن معينة تقرا رقم من المستخدم وتعمل بيه return لكن لو المستخدم دخل قيمة مخالفة يتم عمل raise ل Exception بإسم IntOnly

1- انشئ الصف الجديد واشتقه من نوع Exception او Error مناسب

```
>>> class IntOnly(ValueError):
                def __init__(self, msg):
                        self.message=msg
                def __str__(self):
                        return repr(self.message)
```

2- اكتب الفنكشن واعمل raise لل IntOnly فى حال عدم التوافق

```
>>> def readInt():
        inp=raw_input("Enter a num: ")
        try:
                i=int(inp) #cast to integer.
                return i

        except Exception:
                raise IntOnly("Integers only are allowed.")
```

3- اكتب كود برنامجك

```
>>> try:
        j=readInt()
except IntOnly, e:
        print e.message
```

هنا بنستدعى readInt ونسند قيمتها ل j  فى ال try block
فى حالة حدوث ايرور(خطأ) من النوع IntOnly هنعمل منه كائن (اللى هو e ) ونشوف e.message
ودا الأسلوب المفضل
تقدر تتعامل مع كذا اكسبشن بإستخدام except حسب ماتحب طالما الكود بتاعك فيه مشاكل

مثال على finally

```
#!/usr/bin/python


f = None #out of try block. As finally doesn't have try' context

try:
   f = file('somefile, 'r')
   lines = f.readlines()
   for line in lines:
      print line, #avoid \n\n !
except IOError, e:
   print 'IOErrorError'
finally: #cleaning up
   if f:
      f.close()
```

كدا وصلنا للصيغة العامة وهى

```
try:
        suite
except EX1, ex1: suite

except EX2, ex2: suite

except EX3, ex3: suite

finally: suite
```

يوجد بعض الصفوف المجهزة للتعامل مع الإستثناءات/الأخطاء
BaseException: هو الأب
Exception: هو الأب المشتق منه الإستثناءات المعتادة
ImportError: محاولة استدعاء موديل
KeyboardInterrupt: عندما يقاطع المستخدم التنفيذ (غالبا ب ^C)
NameError: محاولة استدعاء identifier غير موجود
SyntaxError: كود بايثون خاطئ
IndexError: الوصول لترتيب غير موجود فى sequence معينة
KeyError: مفتاح غير موجود فى قاموس معين
IOError: مشاكل فى الدخل او الخرج IO ملف غير موجود مثلا
OverflowError: تعدى الحجم المسموح بيه لنوع معين
OSError: نظام التشغيل
AssertionError: خطأ نتج بسبب فشل فى assert expression

```
>>> 1==0
False
>>> assert 1==0
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    assert 1==0
AssertionError
```

أكتب اكواد افضل وعالجها بصورة افضل!

# *Chapter 8 (IO)*

هنتكلم فى الفصل دا عن التعامل مع ال files وتحديدا ال IO اختصارا ل Input/Output
ال file type بيعبر عن صف مسئول عن التعامل مع الملفات (ممكن يكون سوكيت او غيره بما ان كل شئ عبارة عن file)

فى اكثر من طريقة لإنشاء file object ودا عن طريق open او file class
open هتعمل ريترن ب file object وهى معرفة كالتالى

open(name, mode, buffering)

حيث name هو مسار الملف
mode هو بيعبر عن حالة الوصول (الملف مفتوح للقراءة ، للكتابة ، للإضافة ؟) الإفتراضى هو r للقراءة

r → قراءة فقط
w → كتابة فقط (بيتم محو كل البيانات الموجودة)
a → للإضافة من عند النهاية ،مع عدم محو البيانات
r+ → قراءة وكتابة
w+ → كتابة وقراءة
اى من ال modes السابقة اذا لحقته ب b اختصارا ل binary سيتم التعامل مع binary read, binary write, .. etc

ال buffering لتحديد هل سيتم عمل اى buffering فى حال التعامل مع الملف، الإفتراضى هو -1 (بيتم النقل لنظام التشغيل)

.close()

لغلق ال file object

.read(num=None)

بتقوم بقراءة عدد معين من البايتات وفى حال عدم تحديده بيتم قراءة الملف بكامله

.readline()

قراءة سطر واحد

.readlines()

قراءة كل السطور (على صورة list)

.tell()

بتخبرنا بالمكان الحالى

.fseek(offset, whence=0)

بتنقل المكان الحالى الى offset معين بعد ال whence
whence ربما تكون 0 (اى بداية الملف ) او 1 (المكان الحالى) او 2 (نهاية الملف)
عندنا ملف سميناه iotest.xcd فيه التالى 0123456789

.write(s)

يقوم بكتابة s فى الملف

.writelines(seq_of_strings)

يقوم بكتابه كل عناصر seq_of_strings فى الملف
--كقيامك بإستدعاء write على كل عنصر فى هذه ال sequence

.fileno()

الحصول على File Descriptor خاص بال file

.flush()

لتأكيد نقل ال buffer الداخلى كله على الملف

.name

للحصول على مسار الملف

.mode

للحصول على ال access mode

.encoding

للحصول على الإنكودينج

.closed

للتحقق من تحقق اغلاق المسار

```
FNAME="iotest.xcd"
txt="""
line 1
line 2
line 3
line 4
line 5
line 6
line 7
some text
yada yada yada!
"""
f=open(FNAME, "w")
print f.fileno()
print f.name
print f.mode
print f.closed
f.write(txt)
f.close()
print f.closed
```

فى المثال السابق قمنا بفتح ملف iotext.xcd للكتابة، وكتبنا فى داخله محتويات المتغير txt
الناتج

```
3
iotest.xcd
w
False
True
```

القراءة

```
f=open(FNAME, "r")
lines=f.readlines()
for line in lines:
    print "LINE = > ", line, #avoid printing a new line.
```

الناتج

```
LINE = >
LINE = >  line 1
```

```
LINE = >  line 2
LINE = >  line 3
LINE = >  line 4
LINE = >  line 5
LINE = >  line 6
LINE = >  line 7
LINE = >  some text
LINE = >  yada yada yada!
```

معاملات سطر الأوامر لبرنامجك

بكل بساطة مش هتحتاج غير ال argument vector ودى موجودة فى ال sys.argv
لو فاكر من السى

```
int main(int argc, char** argv){

}
```

فللوصول لل argv استخدم argv الموجودة بال sys module وللحصول على عددهم استخدم len(argv)

```
striky@striky-desktop:~$ python myecho.py hello world 123 "78 yay"
Number of arguments:  5
myecho.py
hello
world
123
78 yay
```

myecho.py

```
#!bin/python

from sys import argv # arguments vector.

print "Number of arguments: ", len(argv)

for arg in argv:
	print arg
```

## Python IO stuff

فى عندنا 2 modules مهمين هما os, os.path
ضيفهم كالتالى

```
import os
import os.path as op
```

`os.uname(...)`
`(sysname, nodename, release, version, machine)` بتعيد tuple مكونة من

```
>>> print os.uname()

('Linux', 'striky-desktop', '2.6.24-21-generic', '#1 SMP Tue Oct 21
23:43:45 UTC 2008', 'i686')
```

*os.getcwd()*

بتعيد المسار الحالى

```
>>> print os.getcwd()

/home/striky/workspace/pytut/src
```

os.getcwdu()

مثل سابقتها ولكن بتعمل ريترن ب unicode

os.environ

هى dictionary مخزن فيه متغيرات البيئة environment مثل ال HOME, LOGNAME, PATH,.. etc
للإطلاع عليهم

```
for key, val in os.environ.items():
   print key, " => ", val
```

`os.getenv(key, default)`

للحصول على مفتاح بعينه

للخصول على قيمة مفتاح ما فى الenviron

```
print os.getenv("HOME")
```

ال default سيتم اعادتها فى حالة عدم وجود المفتاح

انا root ؟

```
def isroot():
     return os.getenv("USER")=="root"
```

طبعا تقدر تستخدم ال keys, values methods الخاصة بال environ dictionary.

os.putenv(key, value)

إضافة key لل environment بقيمة value

os.unsetenv(key)

حذف key ما

os.chdir(to)

بتقوم بنقل المسار الحالى cwd الى to

```
print os.getcwd()
os.chdir("/home/striky")
print os.getcwd()
os.chdir("Music")
print os.getcwd()

#output
/home/striky/workspace/pytut/src
/home/striky
/media/s3/Music
```

لاحظ ان Music هنا symbolic link تحت home/striky/ وبيشير ل media/s3/Music/

os.listdir(path)

بتقوم بإعادة list من مكونات ال path

```
>>> p=os.getcwd() #/home/striky/workspace/pytut/src
>>> print os.listdir(p)
['userstringtest.py', 'iosess.py', 'gcombo.py', 'iohelpers.py',
'oopsample.py', 'iotest.xcd', 'gtk1.py', 'complpath.py']
```

os.link(src,dest)

بتنشئ hard link من src الى dest

os.symlink(src, dest)

بتنشئ symbolic link من src إلى dest

os.unlink(path)

حذف path

os.remove(path)

مثل os.unlink

```
os.rmdir(path)
```

لحذف مجلد معين

```
os.rename(src, dest)
```

اعادة التسمية

```
os.removedirs(path)
```

بتقوم بالحذف من اسفل لأعلى مثلا

```
os.removedirs('foo/bar/baz')
```

ستقوم بحذف مجلد baz اولا ثم bar ثم foo

```
chmod(path, mode)
```

بتقوم بتعديل ال mode على path

```
chown(path, uid, gid)
```

تحديد ال uid, gid على path معين

```
print os.sep # /
```

الفاصل العناصر المسار

/home/striky

```
print os.curdir #.
```

المجلد الحالى وهى ال "."

```
print os.altsep#None
```

حرف فاصل بديل

```
print os.pardir#..
```

المجلد الأب وهى ال ".."

```
print os.extsep#.
```

الفاصل للإمتدادات وهو ال "."

```
print os.pathsep#:
```

الفاصل فى متغير ال PATH وهنا:

```
print repr(os.linesep)#\n
```

الفاصل بين السطور وهنا هو ال "\n"

```
p=os.getcwd() #/home/striky/workspace/pytut/src
F=p+r'/'+"iohelpers.py"
print op.basename(F)

print op.isfile(F)
print op.islink(F)
print op.isabs(F)
print op.isdir(F)
print op.isdir(p)
print op.ismount("/media/s3")
print op.abspath(F)
print op.dirname(F)
print op.split(F)
```

```
print op.splitdrive(F)
print op.splitext(F)
print op.exists(F+"xx")#Nope!
print op.getatime(F) #last access time
print op.getmtime(F) #last modification time
print op.getsize(F)  #file size.

print op.join("/home", "striky", "Music")
```

الناتج

```
iohelpers.py
True
False
True
False
True
True
/home/striky/workspace/pytut/src/iohelpers.py
/home/striky/workspace/pytut/src
('/home/striky/workspace/pytut/src', 'iohelpers.py')
('', '/home/striky/workspace/pytut/src/iohelpers.py')
False
1226557143.0
1226557142.0
521
/home/striky/Music
('/home/striky/workspace/pytut/src/iohelpers', '.py')
```

exists(path)

هل المسار موجود ؟

isfile(path)

هل ال path ملف؟

isdir(path)

هل ال path مجلد؟

islink(path)

هل ال path عبارة عن link ؟

ismount(path)

هل هو عبارة عن نقطة ضم ؟

isabs(path)

هل هو المسار بالكامل ؟

basename(path)

القاعدة فى المسار

abspath(path)

المسار المطلق

dirname(path)

اسم المجلد

getatime(path)

الحصول على توقيت ال last access

getmtime(path)

الحصول على توقيت ال last modification

getctime(path)

الحصول على توقيت ال last change او ال last creation اذا كان على windows

getsize(path)

الحصول على مساحة path

join(a)

لدمج مكونات ال path بإستخدام الفاصل المناسب

split(path)

تقوم بإعادة tuple مكونة من ال dirname وال basename

splitdrive(path)

تقوم بإعادة tuple مكونة من ال drive, وباقى المسار

splitext(path)

تقوم بإعادة tuple مكونة من المسار كامل بدون الإمتداد و الإمتداد

## File Pointer

يمكن تكون مليت من إستخدام FileHandler.write

```
>>>F = open(fileName, 'w')
>>>print >> F, 'Hola'  #It will write the Hola word to the file that we opened
>>>F.close()
>>>F=open(fileName, 'r')
>>>for line in F.readlines(): print line
Hola
```

على فرض إنك هتعمل File ودا معناه إنك هتستخدم ال permission 'w'

```
>>> f = open('C:\\2.txt', 'w') #Open 2.txt for writing mode.
>>> print >> f, 'Hola!'  #Add 'Hola' to it
>>> print >> f, 'Hello!' # same
>>> print >> f, 'Using File Pointer !' #the same
>>> f.close() #closing the file handler.
>>> f = open('C:\\2.txt', 'r')  # Open in reading mode.
```

```
>>> for line in f.readlines(): #iterates through the file lines
    print line #printing each line.
Hola!
Hello!
Using File Pointer !
```

```
>>>f.close() #closing the handler.
```

طب تمام .. هنفتح ال File مرة تانية ولكن فى ال Append mode -وضع الإضافة

```
>>> f=open('C:\\2.txt', 'a')
>>> print >> f, 'Programming Fr34ks r0x!'
>>> print >> f, 'File pointers are',  #Note : this comma is used to avoid printing a new line.
>>> print >> f, ' so great' # added to the line 'File pointers are'
>>> f.close()
```

نقرا اللى مكتوب فى ال File بإننا نعمل Iteration بسيطة على ال filehandler.readlines method

```
>>> f=open('C:\\2.txt', 'r')
>>> for line in f.readlines():
        print line

Hola!
Hello!
Using File Pointer !
Programming Fr34ks r0x!
File pointer are  so great
```

# Chapter 9 (Modules/Packages:Charging a Battery)

بايثون مشهورة بعبارة batteries included فاللغة نفسها لاتقدم سوى ال syntax ولكن لإستخدامها تم تقسيم الخدمات الى ملفات خارجية بإسم modules واذا كانت modules مترابطة تم تحزيمها فى package على سبيل المثال بايثون لاتقوم بتوفير الدوال الخاصة بالرياضيات مباشرة

```
>>> cos(30)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
```

معنى السابق وجود خطأ فى السطر الأول من ال module "ابسط وحدة لتطبيق بايثون" ونوع الخطأ هو عدم وجود دالة ال cos فى مساحة البرنامج
اذا ؟ كيف اتعامل مع الدوال الرياضية ؟
يوجد فى مكتبات بايثون الأساسية modules لعمل معظم انواع التطبيقات قواعد بيانات شبكات حسابات معالجة بيانات .. الخ الخ

**Importing a Module:**
قم بإستدعاء ال module المسماة math وهى وحدة تشمل العديد والعديد للقيام بالعمليات الحسابية بإستخدام import

```
import math
```

هكذا قمنا بإستدعاء الوحدة لساحة البرنامج
لاحظ قد لاتكون ال module موجود فينتج ImportError

```
>>> try:
...     import mymath
... except ImportError, e:
...     print e
...
No module named mymath
```

تقدر تستخدم ال magical import لإستدعاء الموديلز ايضا

```
>>> msys=__import__('sys')
```

تقدر تستخدم as لعمل تسميه مختلفة( لموديل او احد عناصر موديل الخ) هنا للموديل مثل المثال السابق msys كبديل ل sys

```
>>> import sys as mysys
```

وهكذا

## Finding Nemo

اين توجد هذه الوحدات؟ هل يوجد شروط لإستدعاءها ام ماذا ؟
المفسر لايعلم عن كل ملف بايثون على جهازك ولكن هناك بعض الأماكن اللتى يبحث فيها قبل ان يرسل لك ال ImportError مثل المجلد الحالى او مجلد بايثون الإفتراضى او مجلد site-packages (يفضل استخدامه عند اضافة اى وحدات خارجية لبايثون)
للحصول على القائمة كاملة التى يبحث فيها المفسر استدعى sys.path

```
>>> import sys
>>> sys.path
['', '/usr/lib/python2.5/site-packages/Tempita-0.2-py2.5.egg', '/usr/lib/python2.5/site-packages/Mako-0.2.2-py2.5.egg',
....
….]
```

## First Module

يفضل دائما عند انشاء موديل او اى سكربت انك تنشئ هيدر مشابه للتالى

```
#####################################################
# Author: Ahmed Youssef
# License: GPL V3
# Module: firstmodule
# Purpose: Learning modules
# Date: 12-20-2008
#####################################################
```

انشئ ملف firstmodule.py وعرف فيها مجموعة من الدوال كالتالى

```
def aloha():
        print "Aloha!"

def adios():
        print "Adios!"
```

ايه الهدف من ال modules قلنا ؟ فصل التطبيق لأكثر من جزء لإمكانية استخدام خدماته اكثر من مرة وايضا

تسهيل تقسيم العمل مثال شخص يكون مسئول عن جزئية التعامل مع قاعدة البيانات وشخص اخر مسئول عن الواجهة وهكذا

اوكى كتبنا الموديل ماذا الآن ؟
انشئ ملف جديد وليكن greeter.py هيتم فيه استخدام الدوال aloha, adios الموجودة فى firstmodule.py

```
import firstmodule

firstmodule.aloha()
firstmodule.adios()
```

عند التنفيذ

```
striky@striky-desktop:~/workspace/pytut/src$ python greeter.py
Aloha!
Adios!
```

وهكذا

***using from***

ربما لاتريد ان تكتب اسم الموديل فى كل مرة او ربما ماتريده او دالة معينة فقط الخ

```
from firstmodule import aloha, adios

aloha()
adios()
```

او ربما تريد استدعاء جميع محتويات module ما قم فقط بتنفيذ جملة الإستدعاء
*from somemodule import **

حيث * تعنى جميع المحتويات

Reloading

ربما تستخدم بيئة تفاعلية او ربما نظام معين يكون عملية اعادة التشغيل فيه مكلفة او غيرهاا وقمت بتعديل module معينة قم بتنفيذ الدالة reload لإعادة تحميل ال module بالتعديلات الجديدة

### ***Walk on the main***

احترس من اختبار الكود فى ال module وإلا سيتم تنفيذه بمجرد استدعائها
اذا عدلت كود ال firstmodule.py الى

```
def aloha():
	print "Aloha!"

def adios():
	print "Adios!"
```

```
aloha()
```

وقمت بتنفيذ ال greeter.py ستجد الناتج كالتالى

```
striky@striky-desktop:~/workspace/pytut/src$ python greeter.py
Aloha!
Aloha!
Adios!
```

مع اننا استدعينا aloha مرة واحدة ولكن تم تنفيذها مرتان وذلك بسبب استدعائها فى ال firstmodule
والحل ؟ الااستطيع اختبار الكود ؟

اليس الأفضل ان تختبر اولا اذا كانت ال module هى اللتى يتم تنفيذها كملف رئيسى او مجرد مستدعاه فى ملف اخر ؟ كيف اعلم هذا ؟
بكل بساطة توفر لك ال modules متغير خاص بإسم __name__ يحوى اسم ال module الحالية وهو "__main__"

قم بإضافة شرط فى نهاية ال firstmodule كالتالى

```
if __name__=="__main__":
        #Testing...
        aloha()
```

وتم حل المشكلة عند تنفيذلك لل greeter.py سيتم طباعة Aloha! و Adios! وعند تنفيذك لل firstmodule فتكون هى السكربت الرئيسى اللذى سيتم تنفيذه او ال __main__ فيتم تنفيذ كود الإختبار او اناكان :)

*__It's all about __all__*
للتحكم فيما يمكن استدعاءه من وحدة ما تستطيع استخدام متغير خاص بإسم __all__
__all__=["aloha"]

## Packages

للآن جيد ماذا لو زاد عدد ال modules تستطيع بالتأكيد انشاء المئات من تلك الملفات ولكن يجب عليك تحزيم ال modules المترابطة مثلا اذا كنا ننشئ مشروع عن التعامل مع قواعد بيانات مختلفة oracle, sqlite, mysql وغيرهم
اليس الأفضل تجميعهم فى حزمة ما بإسم databases مثلا ؟ ويتم استدعاءها

```
import databases.mysql
```

او مثلا

```
from databases import mysql
```

```

```

بكل تأكيد هذا اكثر تنظيما

```
myfirstpackage/
        |-- __init__.py
        |-- mysql.py
        |-- oracle.py
        `-- sqlite.py
```

لدينا حزمة بإسم myfirstpackage وتشمل 4 ملفات
1- ال __init__ وفيه بيتم تحديد الوحدات اللتى نريد تحميلها مباشرة وربما بعض المتغيرات الأساسية ؟
2- 3 وحدات بإسم mysql.py, oracle.py, sqlite.py معرفين كالتالين

الملف mysql.py

```
def about():
        print "mysql module."
```

الملف oracle.py

```
def about():
        print "oracle module."
```

الملف sqlite.py

```
def about():
        print "sqlite module."
```

الملف __init__

```
print "__init__ myfirstpackage"
import mysql
```

```
VERSION="1.42.0"
```

لاحظ اننا قمنا بعمل import ل mysql مباشرة وحددنا متغير بإسم VERSION

انشئ ملف dbtester.py

```
import myfirstpackage

print dir(myfirstpackage)
print myfirstpackage.VERSION
myfirstpackage.mysql.about()
myfirstpackage.sqlite.about()
```

ستجد الناتج مشابه للتالى

```
striky@striky-desktop:~/workspace/pytut/src$ python dbtester.py
__init__ myfirstpackage
['VERSION', '__builtins__', '__doc__', '__file__', '__name__', '__path__', 'mysql']
1.42.0
mysql module.
Traceback (most recent call last):
  File "dbtester.py", line 6, in <module>
    myfirstpackage.sqlite.about()
AttributeError: 'module' object has no attribute 'sqlite'
```

وبكل تأكيد لن سيتم رفع استثناء AttributeError بسبب عدم استدعاء sqlite للساحة إلا اذا قمت بإضافتها يدويا

```
import myfirstpackage.sqlite
```

## Platform

كثيرا مانحتاج للحصول على معلومات عن النظام الذي يعمل عليه البرنامج (لإختبار التوافقية ، الإعتماديات او ربما العلم بالشئ)
تقدم لنا بايثون وحدة بإسم platform

مثال

```
striky@striky-desktop:~/workspace/pytut/src$ python platformreport.py
[architecture => ('32bit', 'ELF')]
[dist => ('debian', 'lenny/sid', '')]
[java_ver => ('', '', ('', '', ''), ('', '', ''))]
[libc_ver => ('glibc', '2.4')]
[mac_ver => ('', ('', '', ''), '')]
[machine => i686]
[node => striky-desktop]
[platform => Linux-2.6.27-9-generic-i686-with-debian-lenny-sid]
[processor => ]
[python_build => ('r252:60911', 'Oct  5 2008 19:24:49')]
[python_compiler => GCC 4.3.2]
[python_version => 2.5.2]
```

```
[python_version_tuple => ['2', '5', '2']]
[release => 2.6.27-9-generic]
[system => Linux]
[uname => ('Linux', 'striky-desktop', '2.6.27-9-generic', '#1 SMP Thu Nov 20 21:57:00 UTC 2008', 'i686', '')]
[version => #1 SMP Thu Nov 20 21:57:00 UTC 2008]
[win32_ver => ('', '', '', '')]
```

تستطيع بكل تأكيد كتابة كل function مثلا

```
platform.dist()
platform.machine()
platform.uname()
```

الخ الخ
ولكن ربما نستخدم حيلة صغيرة لإستدعاءهم جميعا ؟

```
import platform
for s in dir(platform):
        if not s.startswith("_"): #If it does not start with an underscore.
                f=getattr(platform, s)   #Fetch the attribute (should be a function..)
                try:
                        print "[%s => %s]"%(s, f()) #Prints attr, returned value
                except:
                        pass #Global catch for functions requires params(e.g popen).
```

هنا نقوم بعرض محتويات platform بإستخدام dir ونحصل على ال function object بإستخدام getattr من الوحدة وبإسم الدالة ونقوم بتنفيذها (مجرد استدعائها بعد الحصول عليها) وبس كدا

## *Chapter 10 (Databases)*

**Python/MySQL**

MySQLdb هى Interface بتسمحلك بالتعامل مع MySQL من خلال بايثون
اوكى القصة بدأت ان اتعمل wrap لل MySQL C APIs بصورة OO
فى امثلة لشكل ال APIs

http://mysql-python.sourceforge.net/MySQLdb.html#id5

جميل احنا تعاملنا كله من خلال ال MySQLdb وهى عملت wrap ل _mysql انترفيس لضمان التكافئ مع ال DB API specifications هتكون PEP 249

اولا ال connect

connect(...)
هى المسئولة عن انشاء الإتصال بقاعدة البيانات وبتعمل ريترن ب Connection Object لازم عشان ننشئ اتصال نحتاج شوية معلومات زى ال

- host

ودا بيعبر عن الهوست اللى هيتم الإتصال عليه(الإفتراضى localhost )

- user

اسم المستخدم (الإفتراضى المستخدم الحالى )

- passwd

الباسورد الخاص بإسم المستخدم (افتراضى لايوجد)

- db

قاعدة البيانات (الإفتراضى لأ)

- port

زى مانت عارف ال MySQL ليها server ودا البورت بيعبر عن ال TCP Port اللى بيستخدمه السرفر وافتراضيا 3306 (عدله لو قمت بتغييره!)

- ssl

–لإنشاء SSL Connection (ملحوظة :throws exception لو غير مدعم!)

- compress

–لتفعيل ال compression (الإفتراضى لأ)

- connect_timeout

–تحدد زمن ال timeout

- charset

–اذا تم اضافتها هيتم تضمين use_unicode=True

- sqlmode

–لتحديد ال sqlmode (يفضل تراجع MySQL documentation)

-

تقدر تحدد الكثير من الإعدادات كل اللى عليك تراجع ال MySQL Documentation

apilevel

بتحدد اى DB API مدعمة ؟ الحالى 2.0

1- اعمل import ل MySQLdb كالتالى

```
>>> import MySQLdb as ms
```

--انا خليت ms ك alias طبعا انت حر فى كيفية الإستدعاء

```
>>> ms.apilevel
'2.0'
```

```
>>> ms.threadsafety
```

```
1
```

حسنا ايه معنى ال threadsafety اصلا ؟
هى عبارة عن رقم بين [0, 3]
:0
بيعنى ان ال threads مش تقدر تشارك فى ال module
:1
ان ال threads تقدر تشارك فى ال module ولكن مش ال connections
-2
ان ال threads تقدر تشارك فى ال module وال connections ولكن مش ال cursors (هنتكلم عنها)
-3
اعلى شئ وهى امكانية المشاركة الكاملة فى ال module, connections و ال cursors

paramstyle
سترينج بيعبر عن طريقة التعامل مع ال queries من خلال المدخلات يعنى مثلا احيانا فى ناس بتستخدم ؟ (علامة استفهام) او طريقة %s او حتى استخدام الأرقام :1 و :2 وهكذا (حسب الترتيب) فالإفتراضى هو format

```
>>> ms.paramstyle
'format'
```

ال Exceptions/Errors المرتبطة هنا هما Errors مشتقين من Error
ينقسمو الى
1- IntefaceError ودا بيعبر عن ايرور(خطأ) فى الإنترفيس المستخدمة مش ال داتابيز
2- DatabaseError بيعبر عن ايرور(خطأ) فى قاعدة البيانات وتنقسم لكذا شئ اهمهم
DataError مشاكل مع الداتا
OperationalError ايرور(خطأ) اثناء تنفيذ عملية معينة
ProgrammingError فشل فى تنفيذ sql command معين
NotSupportedError عملية غير مدعمة!

1- ال Connection Objects
هى كائن بيتم اعادتها عند الإتصال بقاعدة بيانات وليها عدة ميثودز
1- close()
لغلق الإتصال
2-commit()
لتنفيذ ال transaction الحالى (مش هتفرق فى حال مش فى تدعيم لل transactions اصلا او ال auto commit مفعلة)
3-rollback()
الغاء ال transaction الحالى (نفس الملحوظة السابقة ولكن لاحظ فى حال انهاء الإتصال وعدم ال commit هيتم عمل rollback اتوماتيك!)

4- cursor()
5- set_sql_mode(sqlmode)
بتحدد ال sqlmode (يفضل تراجع ال MySQL documentation )
6- set_character_set(charset)
تحديد ال characterset

للحصول على cursor (هنتعرض ليه)

2- ال Cursor Objects

بكل بساطة طالما عندك اتصال بقاعدة بيانات يبقة انت محتاج * تتفاعل * معاها عن طريق تنفيذ SQL statements معينة فال cursor بيقوم بدور الوسيط بينكم بيسمحلك تنفذ SQL statements وبيسمحلك تتعامل مع ال rows الناتجة

ال Cursor ليه شوية fields و methods اهمهم

execute(sqlQuery, args)
بتقوم بتنفيذ Sql Statement على قاعدة البيانات وبيتم تجهيزها قبل التنفيذ ب args فى حال لو انت قررت تعمل late binding

where name=?

او

where name=:name

وهكذا

rowcount

عدد الصفوف الناتجة من تنفيذ اخر امر

callproc(proc, args)

لإستدعاء stored procedure!

fetchone()

الحصول على صف واحد من الناتج

fetchmany()
للحصول على عدد معين من الصفوف تم تحديده من خلال arraysize (تبع ال cursor وبتعبر عن عدد الصفوف)

arraysize

بتعبر عن رقم الصفوف اللى هيتم اعادته من خلال ال fetchmany method

fetchall()

للحصول على كل ال الصفوف الناتجة

rownumber

ال index الخاص ب ال cursor!

نختم بمثال (جزء من برنامج حالى استخدمت فيه MySQL ك backend)

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL auto_increment,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `state` tinyint(2) NOT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `username` (`username`)
) ;
```

محتاجين نطبق crud على الجدول دا من خلال Python/MySQL

Create/Read/Updata/Delete

1- استدعى MySQLdb

```
import MySQLdb as ms
```

2- انشئ صف DBMan

```
class DBMan(object):

   def __init__(self, dbname="pyim"):


      self._dbname=dbname
      self._sqlconnection=ms.connect(host="localhost",
                              user="root",
                              passwd="",
                              db="pyim")

       self._sqlcursor=self._sqlconnection.cursor()
```

لاحظ فى ال constructor محدين اسم ال database ك pyim
وحددنا البيانات وانشئنا Connection object بإسم self._sqlconnection
وحصلنا على cursor منه بإسم self._sqlcursor

إضافة مستخدم جديد

```
   def addUser(self, username, pwd, state=State.Offline):
         #State.Offline=1

      md5edpass=self._md5(pwd)
      sqlstmt="INSERT INTO users VALUES(NULL, '%s', '%s', %d)"%(username, md5edpass, state)
      try:
         self._sqlcursor.execute(sqlstmt)
         self._sqlconnection.commit()
      except Exception, e:
            print e.message
```

حذف مستخدم

```
   def deleteUser(self, username):

      sqlstmt="DELETE FROM users WHERE username='%s'"%username
      try:
         self._sqlcursor.execute(sqlstmt)
         self._sqlconnection.commit() #commit
      except Exception, e:
            print e.message
```

تسجيل دخول

```
    def login(self, username, pwd):

        md5edpass=self._md5(pwd)
        sqlstmt="SELECT username, password FROM users WHERE username='%s' AND password='%s'"%
(username, md5edpass)
        self._sqlcursor.execute(sqlstmt)
        if self._sqlcursor.fetchone():
            self.setState(State.Online, username)
```

تغيير الحالة

```
    def setState(self, state, username):

        sqlstmt="UPDATE users SET state=%d WHERE username='%s'"%(state, username)
        try:
            self._sqlcursor.execute(sqlstmt)
        except Exception, e:
            print e.message
```

عرض الكل

```
    def getAllUsers(self):

        sqlstmt="SELECT username, state FROM users"
        self._sqlcursor.execute(sqlstmt)
        for row in self._sqlcursor.fetchall():
            yield row[0], row[1]
```

ملحوظة: انا هنا ناقشت MySQLdb من خلال مفهوم ال DB API بمعنى ان نفس المبادئ هتلقيها ثابتة فى اى انترفيس هتستخدمها
ومازلنا منتظرين DB API 3

Refs:
MySQLdb 1.2.2 docs
Python DB API Specifications v2

## PySQLite

ماهى PySQLite؟
هى interface ل SQLite من خلال ال Python

للتحميل اضغط هنا

للتستيب مثل اى Lib

```
python setup.py install
```

لنبدأ
1- هنحتاج نعمل import لل module
PySQLite هتعملها import ك pysqlite2 ولكن ال lib دى بردو اللى يهمنا فيها هو sub-module بإسم dbapi2

الكتابة

2- هنحتاج نعمل Connection مع DB تمام ؟ ال db نفسها عبارة عن file عادى جدا -فى حال عدم وجوده هيتم إنشاء file جديد- فلعمل ال Connection هنحتاج نستخدم ال connect method الموجودة بال dbapi2

```
#!bin/python

from pysqlite2 import dbapi2 as SQLite
```

نعمل connect ال connect ميثود بتنشئ file فى حال عدم وجوده وإذا موجود هيتعمل return بيه

```
dbConnection=SQLite.connect("mydb.sqlite")
```

كدا انشأنا ال connection بنجاح

ملحوظة: تقدر تعمل Quick Access DB على ال Memory

```
memConnection=SQLite.connect(“:memory:”)
```

بعد ماأنشأنا ال Connection محتاجين نعمل Cursor عشان نستخدمه فى التعامل مع ال DB

```
cursor=dbConnection.cursor() #gets a cursor object..
```

نريد ان ننشئ Table وليكن بإسم Info ويشمل 3 Fields مثلا

id: integer, primary Key
name: varchar(50)
phone: varchar(10)

جميل يبقة هنحتاج SQL Statement

```
sqlStmt='CREATE TABLE info (id INTEGER PRIMARY KEY, name VARCHAR(50), phone VARCHAR(10))'
```

ولتنفيذ ال SQL Statement نستخدم ال execute method الخاصة بال cursor object

```
>>> cursor.execute(sqlStmt)
<pysqlite2.dbapi2.Cursor object at 0x0128B230
>
```

ندخل بعض ال داتا

```
>>> cursor.execute('INSERT INTO info VALUES(null, "ahmed youssef", "12345678")')
<pysqlite2.dbapi2.Cursor object at 0x0128B230>

>>> cursor.execute('INSERT INTO info VALUES(null, "3amer mohamed", "41234114")')
<pysqlite2.dbapi2.Cursor object at 0x0128B230>
```

نقدر ندخل ال fields كالتالى ..

```
>>> username="guru"
>>> phone   ="36987452"
```

كل اللى عليك تمرر علامة إستفهام وفى ال 2nd argument تخليها tuple مكونة من ال vars اللى عايز تدخلها ..

```
>>> cursor.execute('INSERT INTO info VALUES(null, ?, ?)', (username, phone)) #replaced...
<pysqlite2.dbapi2.Cursor object at 0x0128B230>
```

بعد ماعدلنا او اضفنا لازم نستدعى ال Commit method لحفظ التعديلات دى ..

```
>>> dbConnection.commit()
```

ملحوظة: إذا حبيت تخلى التعديلات يتم تنفيذها اوتوماتيك
ضيف فى ال connect ميثود الخاصة بإنشاء ال connection التالى

autocommit=1

فى حالة قيامك بتعديل ما وحبيت ترجع فيه بنستخدم ال rollback method

بعد إنتهائك اقفل ال cursor, connection

```
cursor.close()
dbConnection.close();
```

القراءة

كالعادة لازم نعمل connect على db وننشئ ال connection
ونعمل cursor object بإستخدم cursor ميثود الموجودة بال connection object
ننفذ بعض ال sql statements ولكن هنا هنخليها عبارة عن إستعلامات بسيطة

ننشئ ال connection

```
>>> dbConnection=SQLite.connect("mydb.sqlite") #reopen the db..
```

ننشئ cursor

```
>>> cursor=dbConnection.cursor()
>>> #let's query the db..
```

sql statement ليتم تنفيذها

```
>>> sqlStmt='SELECT * from info'
```

تنفيذ ال sqlStmt

```
>>> cursor.execute(sqlStmt)
```

fetchall هى ميثود بتعيد كل ال rows على صورة tuples ف list

```
>>> cursor.fetchall()
[(1, u'ahmed youssef', u'12345678'), (2, u'3amer mohamed', u'41234114'), (3, u'guru',
u'36987452')]
```

او تقدر تعمل شئ مشابه لكدا بإنك ت iterate على كل الصفوف اللى موجودة بال result

```
>>> for row in cursor:
        #id, name, phone
        print "----------"
        print "ID: ", row[0]
        print "Name: ", row[1]
        print "Phone: ", row[2]


----------
ID:  1
Name:  ahmed youssef
Phone:  12345678
----------
```

```
ID:  2
Name:  3amer mohamed
Phone:  41234114
----------
ID:  3
Name:  guru
Phone:  36987452
```

لاحظ إنك تقدر تتعامل معاها ب .next لأنها iterator

```
>>> cursor.next()
(1, u'ahmed youssef', u'12345678')
>>> cursor.next()
(2, u'3amer mohamed', u'41234114')
```

fetchmany(num)

بتعيد عدد معين من الصفوف

```
>>> ret=cursor.fetchmany(2)
>>> ret
[(1, u'ahmed youssef', u'12345678'), (2, u'3amer mohamed', u'41234114')]
```

fetchone()

بتعيد صف واحد

```
>>> one=cursor.fetchone()
>>> one
(3, u'guru', u'36987452')
```

جميل جدا .. طب وإذا حبيت اخزن user defined type ؟
بكل بساطة اعمل ال class بتاعك الأول

```
class Person(object):

    def __init__(self, name, phone):
        self.name=name
        self.phone=phone
```

ننشئ connection و cursor ولكن ننبه ال database انها تعمل parse لل declared types زى ال Person مثلا .. هنغير شوية ونتعامل مع ال memory

```
#create a connection.
```

```
memConnection=SQLite.connect(':memory:', detect_types=SQLite.PARSE_DECLTYPES)

#cursor
cursor=memConnection.cursor()
```

الآن ننشئ table بحيث إنه ياخد 2 fields وهم ال ID, information

```
cursor.execute("CREATE TABLE test (id INTEGER PRIMARY KEY, p person)")
```

جميل جدا .. ناقص إننا نحدد إزاى ال object الخاص بنا يتحول ل string وازاى نجمع ال data بتاعته تانى من ال string دا
ملحوظة: إحنا بنتكلم على مجرد text بإستخدم toString method مثلا ..مش serializing objects او Pickling

```
def adaptPerson(person):
    return "%s;%s" %(person.name, person.phone)
```

وكيفية التجميع .. بكل بساطة إحنا حولنا ال fields بتاعت ال Person object ل string ودمجناهم ب ;.. يبقة نقدر نجمعهم بإننا نفصل ال ; ونمرر ال قيم الخاصة بال fields دى لل Constructor وننشئ object منها

```
def convToPerson(text):
    name, phone=map(str, text.split(";"))
    return Person(name, phone)
```

بعد ماعملنا الميثودز الخاصة بالتحويل والتجميع .. كل اللى ناقص اننا نبلغ SQLite بكدا

```
SQLite.register_adapter(Person, adaptPerson)
SQLite.register_converter("person", convToPerson)
```

ننشئ بعض الكائنات

```
p1=Person("ahmed", "12345678")
p2=Person("rul3z", "89745632")
```

ونضيفهم للجدول

```
cursor.execute('INSERT INTO test VALUES(null, ?)', (p1, ))
cursor.execute('INSERT INTO test VALUES(null, ?)', (p2, ))
```

نجرب نستعلم عن الموجودين

```
#select..
cursor.execute('SELECT * from test')
for row in cursor:
   print row

#output:
(1, (ahmed;12345678))
(2, (rul3z;89745632))
```

نقفل ال cursor, connection

```
#clean-up
cursor.close()
memConnection.close()
```

الكود النهائى

```
#!bin/python

from pysqlite2 import dbapi2 as SQLite


#dbName='myobjDBTest.sqlite'
#create a connection.
#dbConnection=SQLite.connect(dbName, detect_types=SQLite.PARSE_DECLTYPES)
memConnection=SQLite.connect(':memory:', detect_types=SQLite.PARSE_DECLTYPES)
#cursor
cursor=memConnection.cursor()


class Person(object):

   def __init__(self, name, phone):
      self.name=name
      self.phone=phone

   def __repr__(self):
      return "(%s;%s)" %(self.name, self.phone)

#define a method to register it..

def adaptPerson(person):
   return "%s;%s" %(person.name, person.phone)

def convToPerson(text):
   name, phone=map(str, text.split(";"))
   return Person(name, phone)
```

```
SQLite.register_adapter(Person, adaptPerson)
SQLite.register_converter("person", convToPerson)

p1=Person("ahmed", "12345678")
p2=Person("rul3z", "89745632")


#create a test table..
cursor.execute("CREATE TABLE test (id INTEGER PRIMARY KEY, p person)")

#add
cursor.execute('INSERT INTO test VALUES(null, ?)', (p1, ))
cursor.execute('INSERT INTO test VALUES(null, ?)', (p2, ))

#select..
cursor.execute('SELECT * from test')
for row in cursor:
    print row

#clean-up
cursor.close()
memConnection.close()
```

وللمزيد راجع التالى :

http://www.devshed.com/c/a/Python/Us...ite-in-Python/
http://www.initd.org/tracker/pysqlite/wiki/basicintro
http://www.initd.org/pub/software/py...ined-functions

ملحوظة: sqlite3 اصبحت موديل اساسية فى بايثون تقدر تطبق نفس الفصل عليها

***ORMs***

مامعنى ORM ؟ هى اختصار ل Object Relational Mapping حيث تقوم بتمثيل بياناتك على هيئة objects من classes بدلا من صفوف من جداول
فبإختصار ال class بيعبر عن بنية الجدول وال object بيعبر عن كل صف فى الجدول
تستطيع ايضا من خلال ال ORMs ادارة العلاقات بين الجداول وبعضها يتيح الوراثة!!

***Storm***

هنبدأ ب storm وهى ORM مقدم من canonical

اولا بعض الأساسيات

```
>>> from storm.locals import *
```

ننشئ صف ليمثل لنا جدول للكتب

```
>>> class Book(object):
...     __storm_table__="book"
...     id=Int(primary=True)
...     name=Unicode()
...      npages=Int()
```

ننشئ قاعدة بيانات

```
>>> db=create_database("sqlite:")
```

ننشئ كائن مخزن (ليتعامل مع عناصر قاعدة البيانات)

```
>>> store=Store(db)
```

ننشئ الجدول المعبر عن الكتب

```
>>> store.execute("CREATE TABLE book (id INTEGER PRIMARY KEY, name VARCHAR, npages INTEGER)")
```

ننشئ كائن

```
>>> rbook.name=u"Introduction to Ruby"
>>> rbook.npages=210
>>> print rbook.id, rbook.name, rbook.npages
None Introduction to Ruby 210
```

فلنضيفه الآن

```
>>> store.add(rbook)
<__main__.Book object at 0xb78aecac>
>>> print rbook.id, rbook.name, rbook.npages
None Introduction to Ruby 210
>>>
>>> pybook.name=u"PyGuide"
>>> pybook.npages=230
>>> print pybook
<__main__.Book object at 0xb78ae80c>
>>> store.add(pybook)
<__main__.Book object at 0xb78ae80c>
```

الحصول على سجل ما

```
>>> pythonbook=store.find(Book, Book.name==u"PyGuide").one()
>>> pythonbook.name
u'PyGuide'
```

الطريقة one تحصل على صف واحد

او ربما البحث بال primary key الخاص به

```
>>> store.get(Book, 1).name
u'Introduction to Ruby'
```

الطريقة flush كالعادة لعمل flush

```
>>> store.flush()
>>> store.get(Book, 1).id
1
>>> pythonbook=store.find(Book, Book.name==u"PyGuide").one()
>>> pythonbook.id
2
```

الطريقة commit تستخدم لتحقيق اى تعديل على اى كائن
التعامل مع الكائنات افضل كثيرا من جمل SQL المملة وتساعد على تجنب الكوارث وتسهل امكانية النقل من قاعدة بيانات لأخرى بكل سهولة

للمزيد تابع https://storm.canonical.com/

### *SQLObject*

ايضا مثال رائع للل ORMs هو SQLObject مشابه ل storm

1- استدعاء المكونات ل sqlobject

```
>>> from sqlobject import *
>>>
```

2- انشاء Hub ليعالج الإتصال القادم من العنوان sqlite:/:memory: وهى مسار قاعدة بيانات sqlite مخزنة فى الذاكرة

```
>>> sqlhub.processConnection=connectionForURI('sqlite:/:memory:')
>>>
```

3- نكتب صف يعبر عن الجدول

```
>>> class Book(SQLObject):
...     title=StringCol()
...     npages=IntCol()
```

4- ننشئ الجدول

```
>>> Book.createTable()
[]
```

5- ننشئ كائنات من الصف Book

```
>>> rbbook=Book(title="Introduction to Ruby", npages=230)
>>> rbbook.title
'Introduction to Ruby'
>>> rbbook.npages
230
>>> rbbook
<Book 1 title="'Introduction to ..." npages=230>

>>> pybook="PyGuide"
>>> pybook=Book(title="PyGuide, npages=330")
>>> pybook=Book(title="PyGuide", npages=330)
>>> pybook
<Book 2 title='PyGuide' npages=330>
```

للحصول على كتاب ما بإستخدام ال id استخدم الطريقة get

```
>>> Book.get(1)
<Book 1 title="'Introduction to ..." npages=230>
```

```
>>> book=Book.get(2)
>>> book
<Book 2 title='PyGuide' npages=330>
```

للإستعلام استخدم الطريقة select

```
>>> books=Book.select()
>>> list(books)
[<Book 1 title="'Introduction to ...'" npages=230>, <Book 2 title='PyGuide'
npages=330>]
```

هنا حصلنا على جميع الكائنات من الصف Book

```
>>> rbbooks=Book.select(Book.q.title=="Introduction to Ruby")
>>> list(rbbooks)
[<Book 1 title="'Introduction to ...'" npages=230>]
```

هناك selectBy ايضا بديلة ل select تابع المثال التالى للحصول على كائنات الكتب اللتى عدد صفحاتها 230

```
>>> pybook
<Book 2 title='PyGuide' npages=330>
>>> pybook.npages=230
>>> pages230=Book.selectBy(npages=230)
>>> list(pages230)
[<Book 1 title="'Introduction to ...'" npages=230>, <Book 2 title='PyGuide'
npages=230>]
```

للمزيد تابع http://www.sqlobject.org/

# *Chapter 11 (Parsing Markups)*

## XMLing with Python

ملفات ال xml من اهم الملفات اللى بنتعامل معاها بصورة شبه يومية وبايثون من انسب الحلول للتعامل معاها..
فى اكتر من باكيج للتعامل مع ال Markups
http://docs.python.org/lib/markup.html

على فرض عندنا ملف كالتالى

```
<?xml version="1.0"?>

<!DOCTYPE books SYSTEM "books.dtd">
<?xml-stylesheet type="text/xsl" href="books.xsl"?>


<books>
   <book id="1">
      <name>Introduction to Python</name>
      <author>Ahmed Youssef</author>
      <price>80</price>
   </book>
   <book id="2">
      <name>Introduction to Java</name>
      <author>Wael Muhammed</author>
      <price>130</price>
   </book>
   <book id="3">
      <name>Introduction to Ruby</name>
      <author>Ahmed Youssef</author>
      <price>70</price>
   </book>
   <book id="4">
         <name>Introduction to Linux Programming</name>
         <author>Ahmed Mostafa</author>
         <price>90</price>
   </book>
</books>
```

فى root وهى ال books tag
وليها ابناء كل واحد بإسم book
كل book tag ليه attributes ؟ ايوة كل book لية id معين
داخل كل book بيشمل name, author, price tags لإسم الكتاب والكاتب والسعر

من الملف دا نريد ان نحصل على اسم كل كتاب ومجموع السعر بتاعهم

## 1- minidom

فى implementation خفيفة ل DOM بإسم minidom هنستدعيها كالتالى

```
import xml.dom.minidom as md #(parse, parseString..)
```

فى عندنا دالتين مهمين وهم parse, parseString
parse للتعامل مع file
parseString للتعامل مع string
والإتنين هيدولك ريترن ب document object

ال node object
هو يعتبر الأب لكل العناصر ملف ال xml وليه ميثودز/صفات مهمة

nodeType

بتعبر عن النوع هل هى text node, element, comment,document,.. etc

parentNode

رفرنس للأب (ماعدا ال document root) وللل attrs هتكون ديما None

previousSibling

ال node السابقة لل node الحالية إلا اذا كانت هى الأولى

nextSibling

ال node التالية إلا اذا كانت ال node الحالية هى الأخيرة

childNodes

جميع ال nodes اللى داخل ال node الحالية

hasChildNodes()

هل فى nodes داخلها ؟

firstChild

اول ابن

lastChild

اخر ابن

hasAttributes()

هل فيها attributes ؟

appendChild(child)

إضافة ابن جديد

insertBefore(child, befored)

بتضيف child قبل ال befored وفى حال عدم وجوده بيتم إضافته فى النهاية

removeChild(child)

حذف ابن child

normalize()

ربط ال text nodes المتقاربة

ال document object بيعبر عن الملف وليه ميثودز/صفات مهمة زى

documentElement # used as a property

ودى بتعبر عن ال root element وفى مثالنا هنا هى books

getElementsByTagName(tagName) #tagName

بتدور على tagName معين فى كل الأبناء وابنائهم وهكذا وتديلك ريترن ب element object

createElement(tagName)

لإنشاء tag جديد

createComment(comment)

لإنشاء تعليق داخلى

createAttribute(attr)

لإنشاء صفة attribute

ملحوظة فى بعض الميثودز بنفس الإسم ولكن اخرها NS ودى لربطها مع namespace ما وبتاخد nsURI كمعامل ليها.

ال Element Object بيعبر عن عنصر معين فى الملف وليه ميثودز مهمة زى

tagName  #used as a property

بتعيد الإسم المجرد لل element

getElementsByTagName*

مشابهه للموجودة بال document object

hasAttribute(attrName)

هل بيحتوى على attribute ؟

getAttribute(attrName)

بيعيدلك قيمة attribute معينة بإسم attrName

setAttribute(attrName, val)

بيربط attribute معينة attrName ليها قيمة val بالعنصر

removeAttribute(attrName)

لحذف attribute معينة attrName (مش بيرفع اى exception !)

ملحوظة فى بعض الميثودز بنفس الإسم ولكن اخرها NS ودى لربطها مع namespace ما وبتاخد nsURI كمعامل ليها.

مجموعة ال exceptions
http://docs.python.org/lib/dom-exceptions.html

طيب تمام
1- استدعى ال minidom

```
import xml.dom.minidom as md #(parse, parseString..)
```

2- انشئ ال document object سواء بإستخدام parse او parseString حسب تخزينك لملف ال xml

```
doc=md.parse("books.xml")
```

3- احصل على ال document root و اعرضه واحصل على كل tag قيمته book واطبعه

```
def inspectBooks():
   global doc
   print "Root Element: ", doc.documentElement.tagName
   books=doc.getElementsByTagName("book")
```

```
for book in books:
        if book.hasAttribute("id"): #id and it should have one!
                print "ID: ",book.getAttribute("id")
        for child in book.childNodes:
                if child.nodeType==child.ELEMENT_NODE:
                        if child.tagName=="name":
                        child.normalize()
                        print "Book: ",child.firstChild.data
```

تمام ال doc هنا -متغير عام- global variable

```
global doc
```

الحصول على ال document root هنا جالنا ريترن ب Element object واحنا نريد ال tagName
doc.documentElement.tagName

نحصل على كل العناصر اللى tagName بتاعها book

```
books=doc.getElementsByTagName("book")
```

نعمل loop على كل عنصر فيها

```
for book in books:
```

اذا كان فيه id attribute (لمجرد عرض المثال)

```
        if book.hasAttribute("id"): #id and it should have one!
                print "ID: ",book.getAttribute("id")
```

طيب ولطباعة اسم الكتاب؟ لاحظ انه متخزن فى ال name tag
بسيطة جدا نعمل loop على كل الأبناء فى لا book element ونشوف النوع اذا كان ELEMENT NODE و ال tagName بتاعه هو name نطبعه

```
                        if child.tagName=="name":
                        child.normalize()
                        print "Book: ",child.firstChild.data
```

ملحوظة لل nodes انواع كتير element, comment, text, .. etc

ال firstChild دا بيعبر عن ال text node اللى فى ال name tag وال data بتدى ريترن بال string اللى جواها

<name> text node ... </name

الحصول على الثمن الكلى

```
def getTotalSum():
   global doc
   thesum=0
   prices=doc.getElementsByTagName("price")
   for price in prices:
      price.normalize()
      thesum += int(price.firstChild.data) #TO int.
   return thesum
```

نحصل على كل ال price elements

<price>numeric_value</price>

ونحول القيمة ل int وبس ونضيفها على ال thesum وبعد مانخلص نعمل الريترن بيها

ناتج التنفيذ ل

```
   inspectBooks()
   print "Total Sum: ", getTotalSum()
#output

Root Element:  books
ID:  1
Book:  Introduction to Python
ID:  2
Book:  Introduction to Java
ID:  3
Book:  Introduction to Ruby
ID:  4
Book:  Introduction to Linux Programming
Total Sum:  370
```

## 2- SAX

بيعتمد على ال events بمعنى انه بيديلك خبر كل مايبدأ عنصر او يبدأ ال content اللى داخله وهكذا يمكن تشوفه اعقد شوية لكن انا عن نفسى من محبى استخدامه

1- استدعى اللى هنستخدمه

```
from xml.sax import make_parser, parseString
from xml.sax.handler import ContentHandler
```

ال ContentHandler هو مفتاحنا السحرى فيه ميثودز event handlers بيتعمل ليها استدعاء عند حدوث حدث معين

startDocument()

بيتم استدعائها مرة واحدة عند بداية الملف

endDocument()

بيتم استدعائها مرة واحدة عند نهاية الملف

startElement(name, attrs)

بيتم استدعائها عند بداية قراءة كل عنصر el

**<el [attr1=val1, attr2=val2, ... attrN=valN]>**CONTENT</el>

characters(content)

بيتم استدعائها عن بداية قراءة محتوى العنصر

<el [attr1=val1, attr2=val2, ... attrN=valN]>**CONTENT**</el>

endElement(el)

بيتم استدعائها عند نهاية قراءة عنصر el

<el [attr1=val1, attr2=val2, ... attrN=valN]>CONTENT</**el**>

فى بعض الميثودز بتنتهى ب NS ودى فى حالة التعامل مع namespace

اللى Attributes ماهى الا mapping او dictionary مضاف ليها بعض الميثودز مثل

getLength()

للحصول على عددهم

getNames()

الحصول على اسم كل attribute

getType()

للحصول على النوع وهى عادة CDATA

getValue(attrName)

الحصول على القيمة المرافقة لل attribute المسماة attrName

نرجع للمثال
1- هنستدعى الميثودز/الصفوف المستخدمة

```
from xml.sax import make_parser, parseString
from xml.sax.handler import ContentHandler
```

ال make_parser هتعيد لينا XML reader
parseString لقراءة ال xml من string
parse هى ميثود تبع ال XML reader object بتاخد مسار ملف (نفس parse,parseString اللى تحدثنا عنهم فى minidom)
ContentHandler صف مختص بمعالجة المحتوى (لاحقا)
2- ملف ال xml ك string مخزن

```
xmldoc="""<?xml version="1.0"?>


<books>
   <book id="1">
      <name>Introduction to Python</name>
      <author>Ahmed Youssef</author>
      <price>80</price>
   </book>
   <book id="2">
      <name>Introduction to Java</name>
      <author>Wael Muhammed</author>
      <price>130</price>
   </book>
   <book id="3">
      <name>Introduction to Ruby</name>
      <author>Ahmed Youssef</author>
      <price>70</price>
   </book>
   <book id="4">
         <name>Introduction to Linux Programming</name>
         <author>Ahmed Mostafa</author>
         <price>90</price>
   </book>
```

```
</books>
"""
```

3- انشئ صف جديد مشتق من ال ContentHandler

```
class BooksHandler(ContentHandler):
```

ملحوظة اى ميثود مش هتعملها override مش تكتبها فى معالج المحتوى - ContentHandler- ..

```
def __init__(self):

    self._total=0 #Sum of prices.
    self._curel=None
    self._curid=None
    self._booksInfo=[]
     self._authors=[]
```

ايه المتغيرات دى كلها ؟
self._total للتخزين المجموع الكلى للأسعار
self._curel لتخزين اسم العنصر اللى بيتم معالجته
self._curid لتخزين اخر id تم قرائته
self._booksInfo تخزين معلومات عن الكتاب مكونة من ال name, id
self._authors تخزين اسماء الكتاب

```
def getTotal(self):
    return self._total

def getBooksInfo(self):
    return self._booksInfo

def getAuthors(self):
     return self._authors
```

عرفنا getters للوصول للمتغيرات الداخلية
ملحوظة يفضل تستخدم properties مع lambda

```
booksinfo=property(fget=lambda self: self._booksInfo)
authors=property(fget=lambda self: self._authors)
total=property(fget=lambda self: self._total)
```

```
def startDocument(self):
    #print "Starting Document."
        pass
```

لو حبيت تضيف اى رسالة او اى حاجة على هواك يتم تنفيذها عند بداية قراءة الملف

```
 def endDocument(self):
    #print "Ending Document."
        pass
```

نفس السابقة ولكن عند انتهاء القراءة

```
 def startElement(self, el, attrs):

    #print "Starting ", el
    self._curel=el
        if el=="book":
                #get the id..
                self._curid=attrs.getValue("id") #attrs["id"]
```

هنا هيتم الإستدعاء عند بداية قراءة كل عنصر el والصفات الخاصة بيه attrs
1- نخزن العنصر الحالى فى ال self._curel

```
      self._curel=el
```

2- نختبر اذا كان العنصر الحالى هو book فليه attribute بإسم id فنحصل عليها ونخزنها كآخر id لآخر كتاب تم قرائته فى ال reader

```
        if el=="book":
                #get the id..
                self._curid=attrs.getValue("id") #attrs["id"]
```

طبعا تقدر تحصل عليها كأنك بتتعامل مع dict مش بإستخدام .getValue ميثود

```
def characters(self, content):
    if content.strip():

                if self._curel=="price":
                   #print "In Price.."
                   try:
                      self._total += int(content)
                   except:
                      pass
                elif self._curel=="name":
```

```
            self._booksInfo +=[(content, self._curid)]
        elif self._curel=="author":
            self._authors +=[content]
        else:
            pass
```

هنا هيتم استدعائها عن قراءة المحتوى للعنصر الحالى
وبناءا على العنصر الحالى هنتعامل سواء اذا كان ثمن او اسم الكتاب او الكاتب

4- ننشئ كائن من معالج المحتوى الجديد BooksHandler

```
bh = BooksHandler()
```

ننشئ XML reader ونمرر ليه ال handler الجديد والفايل اللى هيتعالج او نستخدم parseString ونمرر ليها كائن معالج المحتوى(bh)

```
p = make_parser( )
p.setContentHandler(bh)
p.parse(open("books2.xml"))
```

او نستخدم parseString نحدد ال string اللى هيتعالج وال هاندلر (bh)

```
parseString(xmldoc, bh)
```

للمزيد عن
http://docs.python.org/lib/module-xml.sax.html
ومش تنسى http://www.saxproject.org/

**ايهما استخدم ؟**

```
همم DOM بيتعمد على انشاء tree للملف ودا شاق جدا للملفات اللى حجمها كبير!
من ناحية اخرى SAX بيعتمد على ال events ودا اسلوب فعال جدا
```

## 3- Expat undercover

Expat مكتبة سى سريعة لمعالجة ملفات ال XML وتم عمل wrapper ليها فى بايثون

1- استدعى الموديلز اللازمة

```
import xml.parsers.expat as exp
```

2- انشئ صف جديد بنفس فكرة ال ContentHandler

```
class ParsingHandler(object):

   def __init__(self, xml):
      self._curel=None
      self._curattrs=None
      self._inbook=False
      self._books=[]
      self._thesum=0

      self._p=exp.ParserCreate()
      self._p.StartElementHandler=self.__startElement
      self._p.EndElementHandler=self.__endElement
      self._p.CharacterDataHandler=self.__charsDataHandler
       self._p.Parse(xml)
```

لاحظ عندنا متغيرات لمتابعة العنصر الحالى والصفات الحالية ليه وهل احنا داخل ال book tag او لأ واسماء الكتاب والمجموع الكلى

القسم التانى متعلق بال parser
1- انشئ XMLParserType object بإستخدام ParserCreate
2- اربط ال handlers المختصين ببداية كل عنصر ونهايته والمحتوى ب handlers انت هتجهزهم لاحقا
3- عالج ال xml بإستخدام ال Parse method

انشئ getters

```
   def getTotalSum(self):
      return self._thesum

   def getBooksInfo(self):
      return self._books

   def printBooksInfo(self):
      for book in self._books:
         print book
```

عرف ال handlers الخاصين بنا اللى اسندناهم للمعالجينالأساسين لل self._p parser

```
   def __startElement(self, el, attrs):
      print "Starting: ",el, attrs
      if el=="book":
```

```
            self._inbook=True
        self._curel=el
        self._curattrs=attrs

    def __charsDataHandler(self, data):
        if data.strip():
            if self._inbook and self._curel=="name" :
                self._books += [data]
            elif self._curel=="price" :
                self._thesum += int(data)
            else:
                pass

    def __endElement(self, el):
        if el=="book":
            self._inbook=False
          self._curel, self._curattrs=None,None
```

الإستخدام

```
if __name__=="__main__":
    p=ParsingHandler(xmldoc)
    print "Total sum: ", p.getTotalSum()
    p.printBooksInfo()
```

لاحظ ان ال xmldoc هو string بيعبر عن ملف ال xml اللى هيتم معالجته

ناتج التنفيذ

```
Total sum:  370
Introduction to Python
Introduction to Java
Introduction to Ruby
Introduction to Linux Programming
```

## HappyMapper

ضع فى حسبانك ملف كالتالى

```
<?xml version="1.0"?>
```

```
<computer>
     <library>
          <books>
              <book id="1">
                  <name>Introduction to Python</name>
                  <author>Ahmed Youssef</author>
                  <price>80</price>
              </book>
              <book id="2">
                  <name>Introduction to Java</name>
                  <author>Wael Muhammed</author>
                  <price>130</price>
              </book>
              <book id="3">
                  <name>Introduction to Ruby</name>
                  <author>Ahmed Youssef</author>
                  <price>70</price>
              </book>
              <book id="4">
              <name>Introduction to Linux Programming</name>
              <author>Ahmed Mostafa</author>
              <price>90</price>
              </book>
          </books>
     </library>
</computer>
```

فى الجزئيات السابقة تحدثنا عن DOM, SAX وكلام كتير اخر .. مارأيك فى هذا الكود ؟

```
for book in computer.library.books:
   print book.id,  book.name, '', by '', book.author

totalsum=sum([int(str(book.price)) for book in computer.library.books])
print "SUM: ", totalsum
```

اليس اسهل كثيرا ؟
الناتج

```
1 Introduction to Python , by  Ahmed Youssef
2 Introduction to Java , by  Wael Muhammed
3 Introduction to Ruby , by  Ahmed Youssef
4 Introduction to Linux Programming , by  Ahmed Mostafa
SUM:  370
```

ماذا عن ال attributes -السمات او الصفات-؟
تستطيع الوصول اليها ايضا من خلال اسمها مباشرة

او ملف من amazon مشابه للتالى

```
<?xml version="1.0" encoding="UTF-8"?>
<ItemSearchResponse xmlns="http://webservices.amazon.com/AWSECommerceService/2005-10-
05">
  <OperationRequest>
    <HTTPHeaders>
      <Header Name="UserAgent">
      </Header>
    </HTTPHeaders>
    <RequestId>16WRJBVEM155Q026KCV1</RequestId>
    <Arguments>
      <Argument Name="SearchIndex" Value="Books"></Argument>
      <Argument Name="Service" Value="AWSECommerceService"></Argument>
      <Argument Name="Title" Value="Ruby on Rails"></Argument>
      <Argument Name="Operation" Value="ItemSearch"></Argument>
      <Argument Name="AWSAccessKeyId" Value="dontbeaswoosh"></Argument>
    </Arguments>
    <RequestProcessingTime>0.064924955368042</RequestProcessingTime>
  </OperationRequest>
  <Items>
    <Request>
      <IsValid>True</IsValid>
      <ItemSearchRequest>
        <SearchIndex>Books</SearchIndex>
        <Title>Ruby on Rails</Title>
      </ItemSearchRequest>
    </Request>
    <TotalResults>22</TotalResults>
    <TotalPages>3</TotalPages>
    <Item>
      <ASIN>0321480791</ASIN>
    <DetailPageURL>http://www.amazon.com/gp/redirect.html%3FASIN=0321480791%26tag=ws
%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/ASIN/0321480791%253FSubscrip
tionId=dontbeaswoosh</DetailPageURL>
      <ItemAttributes>
        <Author>Michael Hartl</Author>
        <Author>Aurelius Prochazka</Author>
        <Manufacturer>Addison-Wesley Professional</Manufacturer>
        <ProductGroup>Book</ProductGroup>
        <Title>RailsSpace: Building a Social Networking Website with Ruby on Rails (Addison-
Wesley Professional Ruby Series)</Title>
      </ItemAttributes>
    </Item>
  </Items>
</ItemSearchResponse>
```

وتريد الحصول على بعض العناصر اليس كذلك ؟

```
print rt.OperationRequest.HTTPHeaders.Header.Name
print rt.OperationRequest.Arguments[0].Name
print rt.OperationRequest.RequestProcessingTime
print rt.Items.TotalPages
print rt.Items.TotalResults
print rt.Items.Item.ASIN
 print rt.Items.Item.DetailPageURL
```

اليس اسهل كثيرا ؟

```
UserAgent
SearchIndex
0.064924955368042
3
22
0321480791
http://www.amazon.com/gp/redirect.html%3FASIN=0321480791%26tag=ws
%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/ASIN/0321480791%253FS
ubscriptionId=dontbeaswoosh
```

او ربما مهتم ب Twitter ؟
على فرض لدينا هذا الملف

```
<?xml version="1.0" encoding="UTF-8"?>
<statuses type="array">
  <status>
    <created_at>Sat Aug 09 05:38:12 +0000 2008</created_at>
    <id>882281424</id>
    <text>I so just thought the guy lighting the Olympic torch was falling
when he began to run on the wall. Wow that would have been catastrophic.</
text>
    <source>web</source>
    <truncated>false</truncated>
    <in_reply_to_status_id>1234</in_reply_to_status_id>
    <in_reply_to_user_id>12345</in_reply_to_user_id>
    <favorited></favorited>
          <user>
            <id>4243</id>
            <name>John Nunemaker</name>
            <screen_name>jnunemaker</screen_name>
            <location>Mishawaka, IN, US</location>
```

```
            <description>Loves his wife, ruby, notre dame football and iu
basketball</description>

<profile_image_url>http://s3.amazonaws.com/twitter_production/profile_imag
es/53781608/Photo_75_normal.jpg</profile_image_url>
            <url>http://addictedtonew.com</url>
            <protected>false</protected>
            <followers_count>486</followers_count>
          </user>
  </status>
</statuses>
```

وتريد الحصول على كل ما تحت ال user

```
    statuses.status.user.inspect_me()
```

ستجد الناتج

```
Attrs:
Tags:
     id  =>  4243
     name  =>  John Nunemaker
     screen_name  =>  jnunemaker
     location  =>  Mishawaka, IN, US
     description  =>  Loves his wife, ruby, notre dame football and iu
basketball
     profile_image_url  =>
http://s3.amazonaws.com/twitter_production/profile_images/53781608/Photo_7
5_normal.jpg
     url  =>  http://addictedtonew.com
     protected  =>  false
     followers_count  =>  486
```

اكيد مازالت تحتاج للعديد من الإختبارات وبإستخدامك هيتم تحسينها

كيفية الإستخدام
1- استدعاء happymapper

```
import happymapper
```

2- استخدام الدالة get_root اللتى تأخذ معامل xmlfile وهو مسار ملف ال xml وتعيد لك ال root tag
طيب دا بالنسبة للملفات الخارجية ماذا عن النصوص الداخلية ؟ قرأت الصفحة فى متغير داخلى او ماشابه ماذا تفعل ؟
تستطيع استخدام الدالة get_root_document التى تأخذ معامل doc وهو محتوى ملف XML وتعيد لك ال root tag
ملحوظة الملفات المستخدمة فى الشرح amazon.xml و twitter.xml مأخوذه من هنا
http://railstips.org/2008/11/17/happymapper-making-xml-fun-again

للحصول على HappyMapper
http://programming-fr34ks.net/pfsoft/happymapperstable.tar.gz

## HTMLing with Python

على فرض انك قرأت الفصل السابق قم بتنفيذ البرنامج التالى

--سكربت يقوم بقراءة صفحة من الإنترنت ويقوم بالحصول على جميع اللينكات فيها اعتمد على النموذج التالى

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

from HTMLParser import HTMLParser as HP
import urllib2 as ulib
import sys

def fetchdatafrom(url):
   return ulib.urlopen(url).read()

def as_unicode(data):
   return data.decode("cp1256").encode("utf-8")


class PageParser(HP):

        def __init__(self):
                self._ina=False
                self._links=[]

        links=lambda self: self._links

        def handle_start_tag(self, tag, attrs):
                pass

        def handle_data(self, data):
                pass

        def handle_endtag(self, tag):
                pass

def getlinks(url):

        htmlsrc=fetchdatafrom(url)
        p=PageParser()
        p.feed(htmlsrc)
        return p.links()
```

طريقة الإستخدام مشابهه لتلك مع SAX
حيث تعيد تعريف الطرق handle_starttag و handle_data و handle_endtag للتعامل مع وسوم ال HTML
الدالة fetchdatafrom تقوم بإعادة كود الصفحة اليك على صورة string
الدالة as_unciode تقوم بتحويل ال cp1256 الى unicode (ربما اذا اردت ان تعالج ال data j تستطيع الإستفادة منها )
ال PageParser هو صف يشتق ال HTMLParser ويتم التعامل داخله مثلما تعاملنا مع الصفوف المشتقة ContentHandler
ولإطعامه السورس نستخدم الطريقة feed
الدالة getlinks تقوم بالحصول على الروابط من الطريقة links اللتى تعيد لنا الروابط اللتى تم قرائتها

## Beautiful Soup

هى HTML/XML parser بايثونية وتعالج ايضا الملفات المكتوبة بطريقة سيئة ولاتجعلك تقلق من الإنكودينج لمعالجة ملفات ال HTML استخدم الصف BeautifulSoup واذا اردت معالجة ملفات XML استخدم BeautifulStoneSoup

حل المطلوب السابق بإستخدام BeautifulSoup

```
#!bin/python

import BeautifulSoup as bs
import urllib2 as ulib

def fetchdatafrom(url):
   return ulib.urlopen(url).read() or "\n"

def getzetcodemain():
   return fetchdatafrom('http://zetcode.com')

soup=bs.BeautifulSoup(getzetcodemain())
for el in soup.findAll('a'):
   # [0][0] is href.
   print "[url=%s]%s[/url]"%(el.attrs[0][1], el.contents)
```

تنتظرك رحلة رائعة مع الوثائق الخاصة بيها
http://www.crummy.com/software/BeautifulSoup/documentation.html

تدريب

استفيد من السكربتات السابق فى تنفيذ التالى
انشاء مفهرس للمنتديات يأخذ قسم معين كبداية وتقوم بتحديد عدد الصفحات المطلوبة ويقوم بفتحها واستخلاص اللينكات والعناوين لها

# Chapter 12 التفاعل مع برامج اخرى

للآن تستطيع كتابة سكربتات جميلة بالبايثون ولكن ايضا قد نحتاج لإدخال بعض البيانات لبرنامج معين من خلال سطر الأوامر

sys.argv هى list تشمل كل المعاملات اللتى تم ارسالها لبرنامجك

```
#echo .py
from sys import argv

print "ARGV: ", argv

for i, arg in enumerate(argv):
        print "Argv[%d]: %s"%(i, arg)
```

لاحظ ان اول معامل فى ال argv سيكون دائما هو اسم البرنامج
يتم الفصل بين كل معامل بإستخدام مسافة
لدمج اكثر من معامل ضعهم بين علامتى تنصيص

```
striky@striky-desktop:~/workspace/pytut/src$ python echo.py Hello
ARGV:  ['echo.py', 'Hello']
Argv[0]: echo.py
Argv[1]: Hello
striky@striky-desktop:~/workspace/pytut/src$ python echo.py "Hello World"
ARGV:  ['echo.py', 'Hello World']
Argv[0]: echo.py
Argv[1]: Hello World
```

ماذا عن enumerate ؟

```
هى دالة تقوم بإعادة index (لعدد الدورات) وقيمة من container (مثلا list ك argv)
```

***Gimme usage!***

تطبيق جيد ايضا ان تضع دالة بإسم usage توضح كيفية استخدام البرنامج
قم دائما بإختبار عدد المعاملات اللتى تم ارسالها للسكربت فأى عدد غير مقبول قم بعرض ال usage

```
def usage():
        """My fancy usage helper"""
        ........


def consoleMain():
        if len(argv) != DEFINED_LENGTH:
```

```
                GIMME_USAGE

if __name__=="__main__":
        consoleMain()
```

***Forget about usage GIMME optparser!***

بايثون كالعادة توفر لك العديد والعديد لمساعدتك فتوفر لك اكثر من وحدة لمعالجة معاملات سطر الأوامر لاحظ الإستخدام التالى

```
striky@striky-desktop:~$ python mufhrs.py -f http://linuxac.org/forum/forumdisplay.php?f=23 -l 1 -u 3 -s 1 -t vb > pgfihrsx2.txt
```

ياإلهى كيف تدير كل هذه المعاملات الغير معقولة؟
انتبه جيدا لأن هذا هو جزء من حل السكربت المطلوب منك سابقا

هنا مثلا لفهرسة قسم فى منتدى نريد اقل ترتيب للصفحة واكثر ترتيب ومقدار الزيادة (ستفيدك كثيرا اذا قررت محاولة فهرسة منتدى SMF) ونوع المنتدى نخبر السكربت بهذا عن طريق تحديد اسم للمعامل وقيمة له مثلا
-s لمقدر الزيادة

وتكون قيمتها هى المعامل التالى لها 1

-u عدد الصفحة المطلوب الإنتهاء عندها

وقيمتها 3

-t لنوع المنتدى

وقيمته vb

وهكذا ، او ربما استخدام الصيغة المطولة

--step=1
--upper=3
--ftype=vb

لاحظ ان الترتيب ليس هاما!!
اكيد اخذ العديد من الشروط والإختبارات (شكرا لبايثون والوحدة optparse لقد اخذت الكثير من الجهد عن عاتقنا)
هذا الجزء من حل السكربت المطلوب

```
def consoleMain():

    optsparser=OptionParser()
    optsparser.add_option("-f", "--forum", dest="forumlink", help="Forum Section")
    optsparser.add_option("-l", "--lower", dest="lower", help="Lowest page")
    optsparser.add_option("-u", "--upper", dest="upper", help="Upper page")
    optsparser.add_option("-s", "--step", dest="step", help="Step")
    optsparser.add_option("-t", "--type", dest="ftype", help="Forum type (e.g) vb")
    options, args=optsparser.parse_args() #defaulted to sys.argv[1:]
    #print options, "====",args
    forumlink=optsparser.values.forumlink
    lower=int(optsparser.values.lower)
    upper=int(optsparser.values.upper)
    forumtype=optsparser.values.ftype.lower()
```

```
step=int(optsparser.values.step)
```

1- يجب استدعاء ال الوحدة optparse او الصف OptionParser مباشرة للإختصار

```
from optparse import OptionParser
```

2- انشاء كائن من OptionParser

```
optsparser=OptionParser()
```

3- اضافة اسماء المعاملات بإستخدام الطريقة add_option واللتى تأخذ معاملات عديده اهمها
1- الصورة المختصرة لإسم الإختيار -f
2- الصورة الطويلة (الكاملة) لإسم الإختيار -forum
3- اسم من اختيارك للحصول على قيمته وليكن forumlink مثلا
4- قسم المساعدة الخاص بال option

```
optsparser.add_option("-f", "--forum", dest="forumlink", help="Forum Section")
optsparser.add_option("-l", "--lower", dest="lower", help="Lowest page")
optsparser.add_option("-u", "--upper", dest="upper", help="Upper page")
optsparser.add_option("-s", "--step", dest="step", help="Step")
optsparser.add_option("-t", "--type", dest="ftype", help="Forum type (e.g) vb")
```

عند استدعاءك للبرنامج فى وضع المساعدة help يتم تنفيذ شئ مشابه للتالى

```
Options:
  -h, --help            show this help message and exit
  -f FORUMLINK, --forum=FORUMLINK
                        Forum Section
  -l LOWER, --lower=LOWER
                        Lowest page
  -u UPPER, --upper=UPPER
                        Upper page
  -s STEP, --step=STEP  Step
  -t FTYPE, --type=FTYPE
                        Forum type (e.g) vb
striky@striky-desktop:~/workspace/pytut/src$ python mufhrs.py --help
Usage: mufhrs.py [options]

Options:
  -h, --help            show this help message and exit
  -f FORUMLINK, --forum=FORUMLINK
                        Forum Section
  -l LOWER, --lower=LOWER
```

```
                Lowest page
 -u UPPER, --upper=UPPER
                Upper page
 -s STEP, --step=STEP  Step
 -t FTYPE, --type=FTYPE
                    Forum type (e.g) vb
```

رائعة اليس كذلك ؟
للحصول على قيم الإختيارات

```
 forumlink=optsparser.values.forumlink
 lower=int(optsparser.values.lower)
 upper=int(optsparser.values.upper)
 forumtype=optsparser.values.ftype.lower()
  step=int(optsparser.values.step)
```

ماهذا ؟ كيف  جعلت بايثون اسماء الإختيارات كمتغيرات خاصة بالكائن ؟
ج: بعض سحر setattr :)

تدريب:

*قم بكتابة اداة مشابهة ل cat بإستخدام بايثون علما بأن ال stdin, stdout, stderr ستجدهم فى الوحدة sys

os.system
لتنفيذ اوامر خاصة بالنظام توجد الدالة system فى الوحدة os  والتى تعيد ايضا ال exit status (اللتى يعيدها البرنامج عند انتهاءه لتشير لنجاح او حدوث خطأ اثناء التنفيذ)
على فرض لدينا هذا السكربت exitstatus.py

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-


def who():
        name=raw_input("Name: ")
        if name != "Ahmed":
                print "Not Ahmed"
                exit(1)
        else:
                print "Welcome"


who()
```

وقمنا بتنفيذ

```
striky@striky-desktop:~/workspace/pytut/src$ python exitstatus.py
Name: Ahmed
Welcome
striky@striky-desktop:~/workspace/pytut/src$ echo $?
0
```

المتغير $? يشمل ال exit code الخاص بالبرنامج

**تنبيه:**

```
-- استخدم دائما ال Exceptions
```

كثير من الأحيان نحتاج لتنفيذ اوامر والحصول على الخرج الخاص بها
فى هذا المثال سنقوم بتنفيذ الأمر cat على الملف السابق ونقوم بقراءته داخل السكربت
قم بإستدعاء الوحدة subprocess الخاصة بتنفيذ برامج فرعية داخل البرنامج (كبديل ل system و مشابهها)

```
>>> import subprocess as sb
>>> ret=sb.call(['cat', 'exitstatus.py'])
#!/usr/bin/env python
#-*- coding:utf-8 -*-


def who():
	name=raw_input("Name: ")
	if name != "Ahmed":
		print "Not Ahmed"
		exit(1)
	else:
		print "Welcome"

who()
>>> ret
0
```

الطريقة call تقوم بتنفيذ امر ما فى list حيث اول عنصر هو الأمر والباقى هو المعاملات اللتى يأخذها البرنامج

```
>>> catoutput=sb.Popen(["cat", "exitstatus.py"], stdout=sb.PIPE).communicate()[0]
>>> print catoutput
#!/usr/bin/env python
#-*- coding:utf-8 -*-


def who():
	name=raw_input("Name: ")
	if name != "Ahmed":
		print "Not Ahmed"
		exit(1)
```

```
        else:
                print "Welcome"


who()
```

نبدأ من اليمين لليسار الطريقة communicate تعيد لنا tuple تشمل الخرج والخطأ فنأخذ العنصر الأول وهو الخرج (ناتج تنفيذ العملية)
Popen هو صف يأخذ اول عنصر args الأمر والمعاملات
stdout, stdin, stderr هى الخرج والإدخال والخطأ الخاصين بالعملية ويأخذو قيم None او PIPE (قيمة خاصة لتعلم بوجوب فتح انبوبة pipe) او file object او رقم ليعبر عن file descriptor موجود
shell قيمة منطقية ( True لتعبر عن تنفيذ البرنامج من خلال الشيل او False ليتم التنفيذ من خلال excevp (لاتهتم الآن) )

## ConfigParser

موديل ConfigParser لمعالجة ملفات ال .ini "لاعلاقة لها بالرجيسترى!"
ملف ال .ini نوع قديم من وصف البيانات ومستخدم بكثرة فى التطبيقات القديمة نسبيا
مثال

```
[program]
name = SVM
version = 0.2.4
license = GPLv3

[author]
name = Ahmed Youssef
email = guru.python@gmail.com
```

هنا فى الملف يوجد 2 sections او قسمين الأول program والثانى author كل منهم بيحوى keys/values "فى بيسموها اوبشنز" براحتك
مثلا ال name هو key تحت ال program section وقيمته SVM
احفظ الملف السابق وليكن tst.cfg

1- استدعى ال ConfigParser موديل

```
from ConfigParser import *
```

2- انشئ كائن

```
cp=SafeConfigParser() #create an object of SafeConfigParser
```

لاحظ ان فى كذا صف RawConfigParser, ConfigParser, SafeConfigParser فال RawConfigParser هو الأب واشتقه ال ConfigParser وهو الأب ل SafeConfigParser لذا قم بإستخدامه دائما

3- قم بقراءة الملف

```
cp.read("tst.cfg") #read by filename.
```

ال configparser object اللى انشئناه cp بيقوم بقراءة الملف بإستخدام ال read method

.add_section(section)

لإضافة سكشن جديد

.set(section, key, value)

لإضافة key جديد تحت القسم section وله قيمة value

.sections()

للحصول على جميع الأقسام

.has_section(section)

هل يوجد قسم بإسم section ؟

.get(section, key)

للحصول على قيمة ل key تحت section

.options(section)

للحصول على كل ال options تحت section معين

.has_option(section, option)

هل section يحوى option بإسم option ؟

.items(section)

الحصول على list مكونة من tuples بتشمل key, value مثلا

```
[('name', 'Ahmed Youssef'), ('email', 'guru.python@gmail.com')]
```

.write(fp)

كتابة الملف سواء على ال stdout او فى ملف ما.. الخ الخ
فى مجموعة من ال Errors مثلا NoSectionError, ParsingError, DuplicateSectionError, NoOptionError ودى فى حال محاول الوصول لقسم او اختيار غير موجود او محاولة التكرار او خطأ فى معالجة الملف "كتابة بصورة غير سليمة" وغيرهم..

من الحاجات اللى تهمك.. السكاشن -الأقسام- وال Options

```
SECTCRE = re.compile(
    r'\['                       # [
    r'(?P<header>[^]]+)'              # very permissive!
    r'\]'                       # ]
    )
OPTCRE = re.compile(
    r'(?P<option>[^:=\s][^:=]*)'         # very permissive!
    r'\s*(?P<vi>[:=])\s*'               # any number of space/tab,
                                # followed by separator
                                # (either : or =), followed
                                # by any # space/tab
    r'(?P<value>.*)$'                   # everything up to eol
    )
```

راجع ConfigParser.py للإطلاع على المزيد

## Replacer

ربما لديك مجموعة كبيرة من الملفات تحتاج القيام بتعديل سريع عليها جميعا ؟
فى هذه الجزئية سننشئ سكربت للتعديل على تلك الملفات بإستخدام
1- القواميس dict
2- ال recursion ( وهى استدعاء الدالة لنفسها)

حسنا هاهو السكربت

```
#!bin/python


#Imports#
import os
import os.path as op
import time
from sys import argv



class Replacer(object):

    def __init__(self, parentDir, dic={}, exts=[]):
        self.__parentDir=parentDir
        self.__dic=dic
        self.__exts=exts
        self.__replacingTimes=0


    def __getExt(self, s):
        #idx=s.rfind(".")
        #ext=s[idx:]
        #return ext
            return op.splitext(s)[1]

    def replace(self):
        start=self.__parentDir
        for e in os.listdir(start): #foreach entry in os.listdir..
            path=start+op.sep+e
            if os.path.isfile(path):
                ext=self.__getExt(path)
                if not ext in self.__exts:
                    continue #Re-Loop..
                #replace..
                f=file(path, "r")
                src=f.read()
                f.close()
                for key in self.__dic:
                    self.__replacingTimes += src.count(key)
                    src=src.replace(key, self.__dic[key])
```

```
            f=file(path, "w")
            f.write(src)
            f.close()
        elif os.path.isdir(path):
            #RECURSE..
            rep=Replacer(path, self.__dic, self.__exts)
            rep.replace()
        else:
                continue
```

الفكرة هى اعطاء مجلد لتعديل الملفات اللتى بداخله
وإذا كان داخله مجلد يتم فتح ذلك المجلد للتعديل على مافى داخله وهكذا

نبدأ ب

```
def __init__(self, parentDir, dic={}, exts=[]):
    self.__parentDir=parentDir
    self.__dic=dic
    self.__exts=exts
     self.__replacingTimes=0
```

تحديد مجلد الأب parentDir
تحديد الكلمات القديمة والجديدة فى قاموس ليتم استبدالهم
تحديد الإمتدادات القابل العمل عليها

الحصول على امتداد ملف

كما ذكرنا تستطيع استخدام os.path.splitext اللتى تعيد لنا قائمة مكونة من اسم الملف والإمتداد

```
def __getExt(self, s):
    #idx=s.rfind(".")
    #ext=s[idx:]
    #return ext
    return op.splitext(s)[1]
```

تستطيع ايضا كتابة ذلك يدويا بإستخدام الطريقة rfind للحصول على ترتيب ال نقطة "." من اليمين -حتى لاتقع فى مشكلة مع ملفات مثل h1.ext1.ext2 – وتحسب الحروف من ذلك الترتيب الى النهاية
نأتى الى الطريقة replace

```
def replace(self):
    start=self.__parentDir
    for e in os.listdir(start): #foreach entry in os.listdir..
        path=start+op.sep+e
        if os.path.isfile(path):
```

```
        ext=self.__getExt(path)
        if not ext in self.__exts:
            continue #Re-Loop..
        #replace..
        f=file(path, "r")
        src=f.read()
        f.close()
        for key in self.__dic:
            self.__replacingTimes += src.count(key)
            src=src.replace(key, self.__dic[key])

        f=file(path, "w")
        f.write(src)
        f.close()
    elif os.path.isdir(path):
        #RECURSE..
        rep=Replacer(path, self.__dic, self.__exts)
        rep.replace()
    else:
            continue
```

فى هذه الطريقة نعمل كالتالى
1- الحصول على قائمة بالمدخلات فى المجلد الرئيسى

```
for e in os.listdir(start): #foreach entry in os.listdir..
    path=start+op.sep+e
```

2- اختبار ماإذا كان المسار ملفا

```
if os.path.isfile(path):
        ext=self.__getExt(path)
```

3- اختبار اذا كان ذلك الملف يحوى امتداد مقبولا (محدد للإستبدال) وإلا نعود الى بداية الدوارة بإستخدام continue

```
        if not ext in self.__exts:
            continue #Re-Loop..
```

5- اذا كان ملفا يحوى امتداد مقبول للإستبدال يتم فتحه لقراءة محتواه وغلقه واستبدال القيم القديمة بالجديدة

```
        #replace..
        f=file(path, "r")
        #Note: Not reading line by line as I've never met a more than 1MB text file!
        src=f.read()
        f.close()
        for key in self.__dic:
```

```
            self.__replacingTimes += src.count(key)
                src=src.replace(key, self.__dic[key])
```

6- فتح الملف للكتابة وكتابة ذلك المحتوى مرة اخرى وغلقه

```
            f=file(path, "w")
            f.write(src)
                f.close()
```

7- اذا كان المسار مجلدا فيتم فتحه (على اساس انه المجلد الرئيسى واعادة تنفيذ ماسبق بإستدعاء الطريقة replace وهذا مايسمى بال recursion

```
        elif os.path.isdir(path):
            #RECURSE..
            rep=Replacer(path, self.__dic, self.__exts)
            rep.replace()
        else:
                continue
```

التعامل مع المستخدم

```
def consoleMain():
        ##python replacer.py root [old] [new] [exts]

        root, oldones, newones, exts=argv[1:5]
        oldones=oldones.split(",") #use comma in between.
        newones=newones.split(",") #use comma in between.
        exts=exts.split(",")        #use comma in between.
        if len(oldones)==len(newones):
                dic=dict(zip(oldones, newones))
                #print "DIC:", dic
                #print "EXTS:", exts
                start=time.time()
                rep=Replacer(root, dic, exts)
                rep.replace()
                end=time.time()
                print "Time: ", (end-start)
        else:
                print "len(keys)!=len(values)"
                exit(1)

        start=time.time()
        rep=Replacer(root, {old:new}, [exts])
        rep.replace()
        end=time.time()
```

```
    print "Time: ", (end-start)
```

تابع لمزيد من التوضيح جزئية استخدام ال optparse

## Parsing CSV Files

CSV هى اختصار ل comma separated values من الأسم واضح انها تستخدم فى تمثيل قيم مع فصلها بإستخدام الفاصلة (ال comma) فى صفوف (تستخدم عادة فى استيراد او تصدير بيانات ما ربما قاعدة بيانات مثلا؟)
لاحظ ممكن يكون الفاصل مجرد مسافة او سلاش / او او او ولكن الأشهر هو ال فاصلة
مثال

ahmed, 19, m
ayman, 20, m

وقد تحتوى على صف اول يمثل الهيدر (يشمل عناوين الأعمدة)

name, age, sex
ahmed, 19, m
ayman, 20, m

على فرض لدينا ملف بإسم somefile.csv وفيه البيانات التالية

```
ahmed, m, 19
wael, m, 20
radwa, f, 19
gina, f, 21
ayman, m, 20
```

استدعى ال csv module كالتالى

```
import csv
```

قم بإنشاء reader object "مسئول عن القراءة للملف ومعالجته" بإستخدام cvs.reader()

```
reader=csv.reader(open("somefile.csv", "rb")) #default dialect. #b as a catch for win32.
```

تقدر تستخدم ال for loop مع ال reader فهى تقوم بعمل yield لكل صف يتم قراءته وللحصول على رقم الصف استخدم ال line_num

```
for row in reader:
    print row, " at: ", reader.line_num
```

فلنقم بتحسين المثال بعض الشئ

```
import csv

f=open("somefile.csv", "rb")
try:
    reader=csv.reader(f) #default dialect. #b as a catch for win32.
```

```
    for row in reader:
        print row, " at: ", reader.line_num


except Exception, ex:
    print ex.message
finally:
     f.close()
```

اذا اردت ربط الصف بقاموس وذلك بتحديد اسماء الأعمدة كالتالى

```
    reader=csv.DictReader(f, fieldnames=("name", "age", "sex")) #default dialect. #b as a catch for
win32.

    for row in reader:
        print row, " at: ", reader.line_num
         print row["name"] #the name column
```

للكتابة الموضوع سهل ايضا
بتنشئ writer (كاتب) من csv.writer
تكتب ال header (اسماء الاعمدة) بإستخدام الطريقة writer.writerow
وتعمل دوارة على المدخلات لكتابة كل صف
على سبيل المثال

```
import csv

f=open("somefile1.csv", "w")
try:

    writer=csv.writer(f)
    inputrows=(
          (1, "ahmed", "ahmedf1@gmail.com"),
          (2, "ayman", "aymanf2@gmail.com"),
          (3, "smsm", "smsm@yahoo.com")
          )

    headers=("id", "user", "email")
    writer.writerow(headers)

    for row in inputrows:
        writer.writerow(row)

except Exception, ex:
    print ex.message

finally:
```

```
f.close()
```

للمزيد حول ال CSV راجع وثائق بايثون

# Chapter 12 (Networking)

فصل اكبر من ان يغطيه كتاب مثل هذا و لمقدمة عن ال sockets راجع مقالة ويكيبديا
http://en.wikipedia.org/wiki/Internet_socket

## Simple Server

```
#simpleserver.py
import socket

class EchoServer(object):

	def __init__(self, host='', port=51002):
		self._host, self._port=host, port
		self._endpoint=(host, port) #host, addr
		self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
		self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

	def start(self):
		self.sock.bind(self._endpoint)
		self.sock.listen(1)
		print "Server running on: ", self._port
		self.handle_request()

	def handle_request(self):

		while True: #Waits for a client.
			clientsock, addr=self.sock.accept()
			#clientfile=clientsock.makefile('rw', 0) #Create a file-like object.
			print "Connection from: ", addr

			clientsock.sendall(str(addr)+" you are connected to server...")

			while True: #communication loop



				#clientfile.write(str(addr)+" you are connected to server.\n")
				msg=clientsock.recv(8092)
				if msg:
					print ">> ", msg
				clientsock.sendall(msg)
				#msg=clientfile.readline().strip() #clean it up.
				#print "Recieved: ", msg
				#clientfile.write("Got: "+msg+"\n")

			#Cleaning UP
```

```
                        #clientfile.close()
                        clientsock.close()




if __name__=="__main__":
        try:
                es=EchoServer()
                es.start()
        except KeyboardInterrupt:
                exit()
```

فى هذا الكود انشأنا صف جديد بإسم EchoServer

```
class EchoServer(object):

        def __init__(self, host='', port=51002):
                self._host, self._port=host, port
                self._endpoint=(host, port) #host, addr
                self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
```

1- قمنا بتحديد ال endpoint (الهوست والبورت)
2- انشأنا TCP Socket
3- قمنا بتفعيل استخدام SO_REUSEADDR لإلغاء شغل البورت عنده ايقاف السرفر

قمنا بتعريف طريقة start لبدأ السرفر

```
        def start(self):
                self.sock.bind(self._endpoint)
                self.sock.listen(1)
                print "Server running on: ", self._port
                self.handle_request()
```

1- عمل bind (ربط بال endpoint اللتى تم تحديدها)
2- تجهيز وتشغيل ال tcp listener بإستخدام الطريقة listen
3- نستدعى الطريقة handle_request اللتى سيتم فيها التعامل مع العميل
مكونة من حلقتين الأولى للتعامل مع العملاء المنتظرين والثانية لمعالجة عميل ما

```
        def handle_request(self):

                while True: #Waits for a client.
                        clientsock, addr=self.sock.accept()
```

```
                #clientfile=clientsock.makefile('rw', 0) #Create a file-like object.
                print "Connection from: ", addr

                clientsock.sendall(str(addr)+" you are connected to server...")

                while True: #communication loop


                        msg=clientsock.recv(8092)
                        if msg:
                                print ">> ", msg
                        clientsock.sendall(msg)

                #Cleaning up
                clientsock.close()
```

الطريقة accept تقبل اتصالا وتعيد لنا كائن socket وعنوان
تستطيع استخدام الطريقة makefile لإنشاء file-like object للتعامل مع ال socket
الطريقة sendall لإرسال رسالة
الطريقة recv للحصول على الرسالة القادمة (ويتم تحديد حجمها عن طريق معامل ال bufsize)

## Simple Client

```
#simpleclient.py
import socket

class SimpleClient(object):

        def __init__(self, endpoint=('127.0.0.1', 51002)):
                self._endpoint=endpoint
                self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                self.sock.connect(self._endpoint)

        def start(self):

                while True:

                        data=self.sock.recv(8096)
                        if not data:
                                break
                        print data

                        msg=raw_input("> ")
                        if not msg:
                                break
                        self.sock.send(msg)
                self.sock.close()
```

```
if __name__=="__main__":
	try:
		sc=SimpleClient()
		sc.start()
	except KeyboardInterrupt:
		exit()
```

هنا انشأنا صف جديد SimpleClient

```
	class SimpleClient(object):

	def __init__(self, endpoint=('127.0.0.1', 51002)):
		self._endpoint=endpoint
		self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
		self.sock.connect(self._endpoint)
```

نقوم بعمل اتصال مع السرفر (تحديد ال endpoint) بإستخدام الطريقة connect

الطريقة start تقوم بعمل حلقة الإتصال مع السرفر

```
	def start(self):

		while True:

			data=self.sock.recv(8096)
			if not data:
				break
			print data

			msg=raw_input("> ")
			if not msg:
				break
			self.sock.send(msg)
		self.sock.close()
```

الخ الخ

## SocketServer

هى موديل فيها تجميع للصفوف الشائعة فمثلا لاحاجة لكتابة الأكواد السابقة لمجرد انشاء TCPServer او UDPServerوهكذا ولكن الأساس ثابت وهناك بعض المتغيرات اللتى يمكن اعادة تعريفها (التعامل مع العميل على سبيل المثال)
سرفر

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

from SocketServer import TCPServer, StreamRequestHandler

class MyStreamRequestHandler(StreamRequestHandler):

	def handle(self):
		print "Got connection from: ", self.client_address
		self.wfile.write(str(self.client_address)+" you are connected to server.")
		#Communication loop...


		while True:

			msg=self.request.recv(1024)
			if not msg:
				break
			print ">> ", msg
			#Send it back...
			self.request.send(msg)
		print "Done handling..."


def go(endpoint=('', 52002)):
	addr=endpoint
	tcpServer=TCPServer(addr, MyStreamRequestHandler)
	tcpServer.allow_reuse_address=1
	print "Server started..."
	tcpServer.serve_forever() #inf. loop

if __name__=="__main__":
	try:
		go()
	except KeyboardInterrupt:
		exit()
```

عميل

```
import socket

class SimpleClient(object):

	def __init__(self, endpoint=('127.0.0.1', 52002)):
		self._endpoint=endpoint
		self.sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
		self.sock.connect(self._endpoint)

	def start(self):
		#self.sock.sendall("Hey you!!")
		while True:

			data=self.sock.recv(1024)
			if not data:
				break
			print data

			msg=raw_input("> ")
			if not msg:
				break

			self.sock.sendall(msg)
		#self.sock.close()



if __name__=="__main__":
	try:
		sc=SimpleClient()
		sc.start()
	except KeyboardInterrupt:
		exit()
```

**MixIns**

تستطيع بكل سهولة ان تجعل سرفرك يعالج اكثر من عميل سواء بإستخدام ال Threading او ال Forking
وذلك بإشتقاقك للـ ThreadingMixIn او ال ForkingMixIn الموجودة فى SocketServer module

```
class MyServer(ThreadingMixIn,TCPServer):
	pass #Done!
```

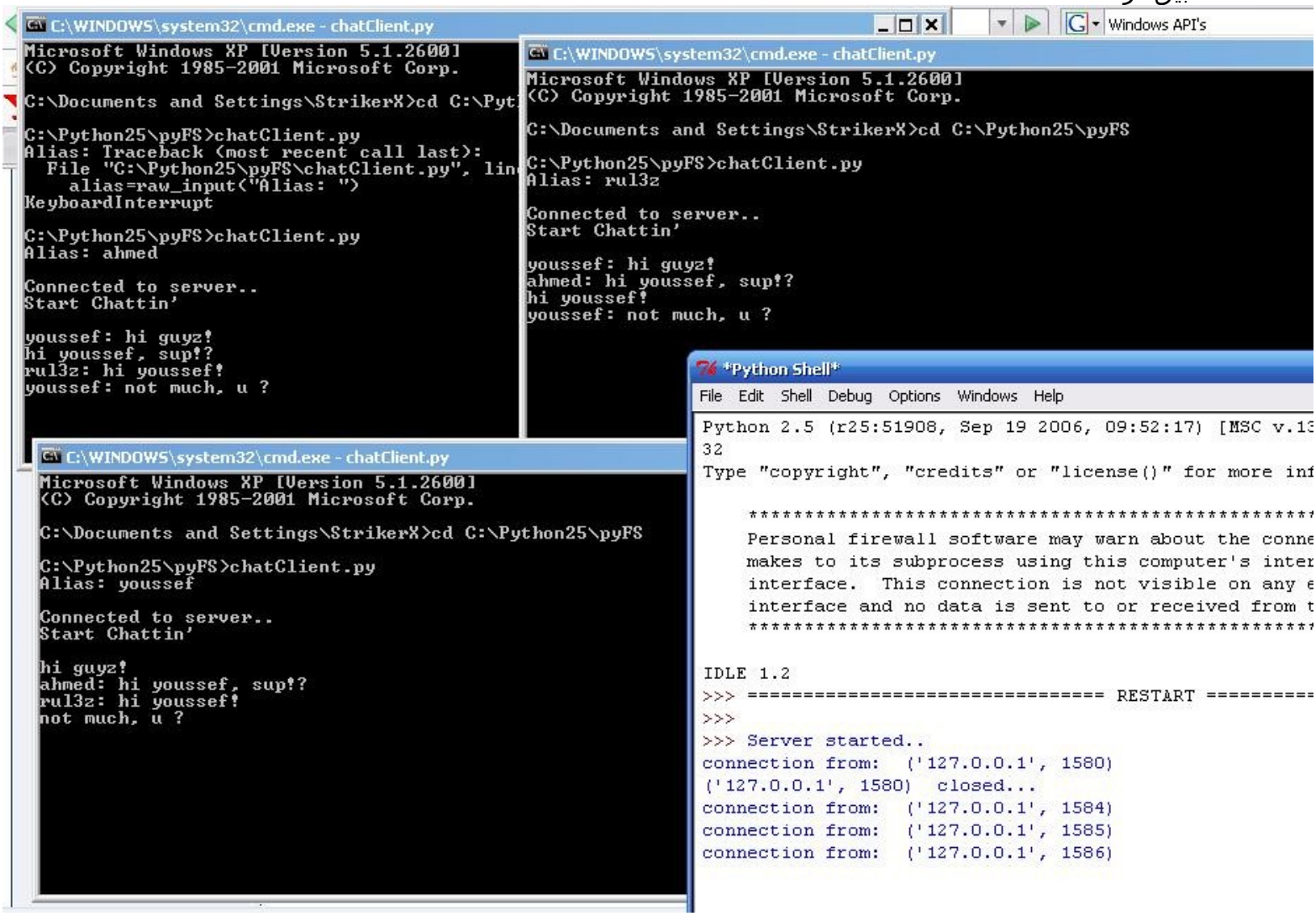
او هكذا

```
class MyServer(ForkingMixIn, TCPServer):
```

```
        pass #Done!
```

الفرق ان ال Forking يتم معالجة كل عميل فى بروسيس -عملية- جديدة بينما ال Threading يتم معالجتها داخل نفس العملية ولكن بخيط جديد (تعدد مهام مثل محرر النصوص اللذى تكتب فيه ويقوم بالترقيم وتصحيح الأخطاء الإملائية والعديد من هذه العمليات فى ان واحد)

تطبيق دردشة




السرفر

```
#!bin/python

import socket
import threading


class ChatServer(object):
    '''Indexer...'''
```

```
def __init__(self, port):

    self.port=port
    addr=('', self.port)
    self._bufsize=2048

    self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) #Quick restarts.
    self.listener.bind(addr)
    self.alSocks=[]
    #self.tListening=threading.Thread(target=self.listeningHandler, args=[])
    #self.tListening.start()

        self.listeningHandler() #Start listening...



def listeningHandler(self):
    self.listener.listen(5)
    print "Server started.."
    while True:
        clientSocket, clientAddr=self.listener.accept()
        #Handle the client in a new thread...
        self.tHandleClient=threading.Thread(target=self.clientHandler, args=[clientSocket])
        self.tHandleClient.start()


def clientHandler(self, clientSocket):

    self.alSocks += [clientSocket]
    print "connection from: ", clientSocket.getpeername()
    self._bufsize=2048
    try:
        while True:

            data=clientSocket.recv(self._bufsize)
            if not data:
                break
            #handle sending all recieved in another thread..
            #serverToAll=threading.Thread(target=self.serverToAll, args=[clientSocket, data])
            #serverToAll.start()
            self.serverToAll(clientSocket, data)

    except Exception:
        #don't act
        print clientSocket.getpeername(), " closed..."
    finally:
        self.alSocks.remove(clientSocket)
        clientSocket.close()

def serverToAll(self, currentClient, data):
```

```
        try:
            for sock in self.alSocks:
                if not sock == currentClient:
                    sock.send(data)
                else:
                    pass
        except Exception, e:
            print e


if __name__=="__main__":

    chatServer=ChatServer(8030)
```

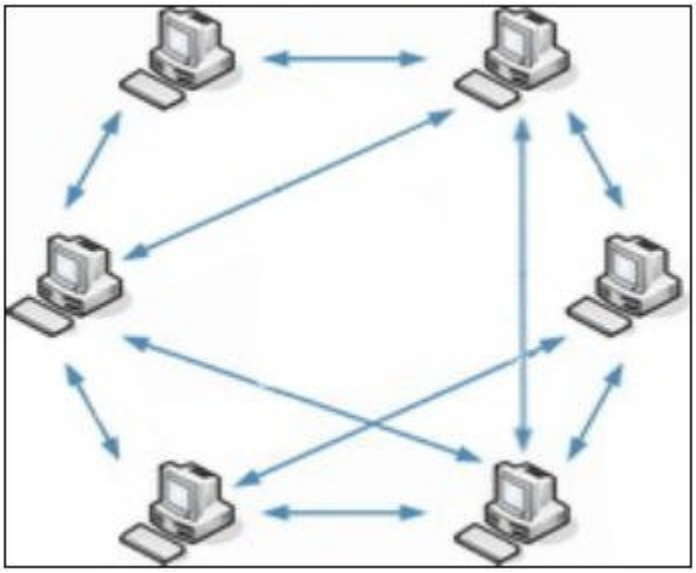
ملحوظة لأنشاء خيط جديد فى برنامجك قم بإستخدام الصف threading.Thread لأنشاء كائن وقم بتحديد ال target وهى الميثود اللتى سيتم تنفيذها بصورة خارجية فى ذلك الثريد و args هى عبارة عن list تحوى المعاملات (args) الخاصة بتلك الميثود

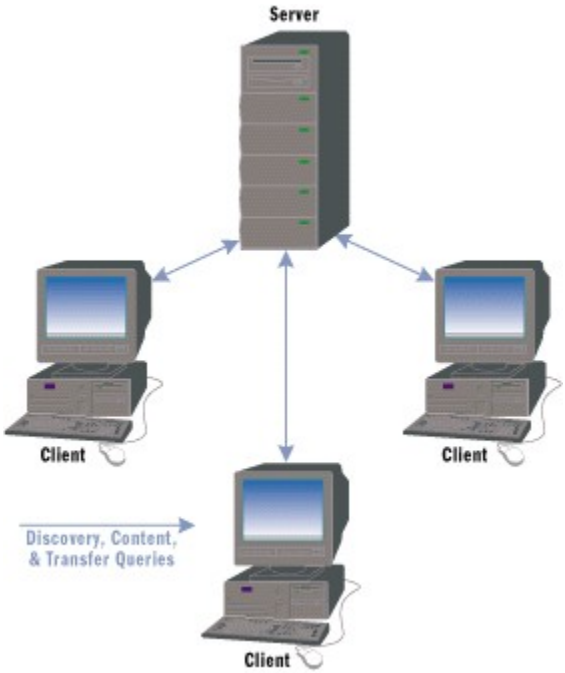
العميل

```
#!bin/python

import socket
import threading


class Peer(object):

    def __init__(self, serverAddr=('localhost', 8030), alias="anonymouse"):

                self.serverAddr=serverAddr
                self.tcpClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                self.alias=alias
                self._bufsize=2048
                self.tcpClient.connect(self.serverAddr)

                print "\nConnected to server.."
                #self.tClientToServer=threading.Thread(target=self.clientToServerHandler, args=[])
                #self.tClientToServer.start()
                        self.clientToServerHandler()

    def clientToServerHandler(self):
                print "Start Chattin' \n"
                while True:

                    data=raw_input()
                    msg=alias+": "+data
                    if not data:
                        break
                    serverToClient=threading.Thread(target=self.serverToClientHandler, args=[])
```

```
                serverToClient.start()
                #self.serverToClientHandler()

                self.tcpClient.send(msg)



    def serverToClientHandler(self):

            while True:
                data=self.tcpClient.recv(self._bufsize)
                if not data:
                    break
                print data


if __name__=="__main__":

            alias=raw_input("Alias: ")
            peer=Peer(alias=alias)
```

***App with a BUG***





Server-Based Network وهو بيعتمد إن يكون فى Server يعمل بدور وسيط بين الأجهزة وبنطلق عليه Discovery Server او Indexer لأنه بيكون على Database بتتخزن فيها معلومات كل Peer من حيث ال Shared Files و ال Alias و .. إلخ

ملحوظة: ال Model الثانى مش يعتبر P2P بصورة كاملة

اولا هنعمل Module بسيطة نعمل بيها Encapsulation لما سبق

```
class PeerInfo(object):
    def __init__(self, alias, sharedFiles, listeningPort):
        self.alias=alias
        self.sharedFiles=sharedFiles
        self.listeningPort=listeningPort
    def __str__(self):
        sb="Alias: " + self.alias
        sb += "\nFiles: " + str(self.sharedFiles)
        sb +="\nListens At:" + str(self.listeningPort)
        return sb

def serialize(obj):
    '''Serialize an object to a string...'''
    return marshal.dumps(obj)
def deserialize(objString):
    '''Deserialize an object string...'''
    return marshal.loads(objString)
if __name__=="__main__":
    p=PeerInfo("ahmed", [1, 2, 3 ,4], 80)
    print p
    print "alias: ",p.alias
    print "files: ", p.sharedFiles
    print "port : ", p.listeningPort
```

ال peerInfo class هو class هنخزن فيه بيانات المستخدم عشان تكون اسهل فى التعامل والإستعلام وهى ال alias, sharedFiles, ListeningPort
هنعمل 2 methods بسيطين جدا لمعالجة ال Objects اللى هتتبعت على السوكيت
وهما serialize و deserialize

بصورة مبدأية لما نفكر فى ال Discovery Server لازم يكون فيه شوية مميزات
1- ان يتم تسجيل بيانات اى حد يعمل Connect والبيانات دى بتشمل Alias, SharedFiles, ListeningPort
2- نقدر نستعلم عن الملفات الموجودة على ال Clients الآخرين

3- كل Client يقدر يعدل على بياناته
4- يقدر يعرض كل ال Clients الآخرين
5- يقدر يعدل على ال SharedFiles

6- اقصى عدد يقدر يتصل بالسرفر
7- وطبعا لازم يقدر يتعامل مع اكتر من Client فى نفس الوقت
8- .....
وهكذا
ملحوظة: ممكن تعمل Clients ليهم صلاحيات اعلى "مثلا اللى مشترين pro للمنتج بتاعك"

اولا إحنا مش هنعمل DB لأننا مش منتج كبير .. احنا يدوب بنعرض الفكرة .. فال DB بتاعتنا هتكون عبارة عن Dictionary او HashTable او غيرهم حسب اللغة اللى إنت جاى منها

```
{'clientIP:port' : peerInfo Object}
```

اولا هنعمل socket object ونعمل set لل options بتاعته "اهم شئ REUSEADDR وهى بتسمحلك بإستخدام ال port مرة اخرى مباشرة فى حال إنك قفلت السرفر لسبب ما "وغالبا هنا الإختبار للبرنامج"
ملحوظة : مع إننا هندعم ال Chat بين المستخدمين ولكننا هنستخدم TCP مش UDP -لأننا هنتعامل مع نقل لملفات-

```
(self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM
(self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1
```

ال Class هيبدأ بالصورة دى .. وهنحدد فيه ال port اللى هيشتغل عليه السرفر .. واقصى عدد للمستخدمين والأوامر المدعمة
/register : لتسجيل ال client
/setSharedFiles : لتعديل ال ملفات اللى معمولها share
/setNick : لتغيير ال Nick او ال alias
/showall : لعرض جميع المستخدمين
/query : للإستعلام عن ملف ما

```
class DiscoveryServer(object):
    '''Indexer...'''
    def __init__(self, port, maxPeers=5):
        self.port=port
        addr=('', self.port)
        self.maxPeers=maxPeers
       self.supportedCommands=["/register", "/setNick", "/setSharedFiles",
"/showall", "/query"]
        self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)
```

```
        self.listener.bind(addr)
        self.tListening=threading.Thread(target=self.listeningHandler,
args=[])
        self.tListening.start()
        self.alSocks=[]
       # {clientAddr:peerInfo Object}
        self.db={}
        self.log=[]
                                                        self.BUF=2048
```

لاحظ مش هسجل log فى البرنامج .. هسيبهالك واجب او فرصة إنك تعدل شئ لشئ احسن وهكذا .. بحيث إنك تبقة مشارك
self.alSocks هى List هنضيف فيها كل ال Sockets المفتوحة "هتعرف بعد شوية"

لاحظ فى السطر دا

```
   self.tListening=threading.Thread(target=self.listeningHandler, args=[])
```

إننا انشأنا thread للميثود listeningHandler وال args اللتى سنمررها [] List فاضية
لأنها مش بتاخد args
ولبدأ ال thread بنستخدم start method
ال implementation الخاص بال method دى

```
  def listeningHandler(self):
     self.listener.listen(self.maxPeers)
     print "Server Started..."
     while True:
        clientSocket, clientAddr=self.listener.accept()
        print "Gotta a connection from", clientAddr
        tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
        tClientHandling.start()
     clientSocket.close()
```

بنخلى السرفر يبدأ فى عملية ال listening
وبعد كدا بنعمل forever loop بنتعامل فيها مع اى client بيعمل connect وبمجرد مايعمل Connect نهندله فى thread جديد بإستخدام method بإسم clientHandler

```
           tClientHandling=threading.Thread(target=self.clientHandler,
args=[clientSocket])
```

ملحوظة accept method بتدى return ب tuple مكونة من ال clientSocket, clientAddress

```
def clientHandler(self, clientSocket):
    self.alSocks += [clientSocket]
    formatedAddress=clientSocket.getpeername()[0]+":"+str(clientSocket.getpeername()[1])
    objString=""
    try:
      while True:
          objString=clientSocket.recv(self.BUF)
          if not objString:
              break
          data=deserialize(objString)
          #print data
          tAnalyzeData=threading.Thread(target=self.analyzeData, args=[data, clientSocket])
          tAnalyzeData.start()

          objString=""

    except Exception, e:
      print "E: ", e
      print clientSocket.getpeername(), " closed.."
      self.alSocks.remove(clientSocket)
      del self.db[formatedAddress]
```

ال method دى خاصة بالتعامل مع ال Client وكل اللى بتعمله فى الحقيقة هى إنها بتضيف ال ClientSocket إلى ال alSocks list اللى بال server class وتبدأ thread جديد تحلل فيه الداتا الجاية من ال Client
وطالما هتحلل data وفى thread جديد يبقة لازم نعمل method تبقة مسؤلة عن عملية ال تحليل او ال parsing لل داتا واكيد ال method دى هتاخد argument وهى ال data واللى ارسل الداتا وهو ال clientSocket

```
tAnalyzeData=threading.Thread(target=self.analyzeData, args=[data, clientSocket])
          tAnalyzeData.start()
```

جميل جدا .. تحليل الداتا فى ال analyzeData method

```
def analyzeData(self, data, clientSocket):
      formatedAddress=clientSocket.getpeername()
[0]+":"+str(clientSocket.getpeername()[1])
      try:
          if isinstance(data, tuple): #registering...
              pInfo=PeerInfo(data[1], data[2], data[3]) #(register,
alias, files, port)
```

```
                print "Registering: ", pInfo.alias
                print pInfo
                self.db[formatedAddress]=pInfo #peerInfo object..
                print self.db
            if isinstance(data, list):
                try:
                     #split the sender's alias..
                     #recvd=['tina: /showall']
                     recvd=data[0].split(": ")[1]
                     cmd=recvd.split(" ")[0]
                     # test cmd...
                     if not cmd in self.supportedCommands:
                         self.sendToAll(data, clientSocket)
                     else:
                         if cmd=="/showall":
                             self.showAllTo(clientSocket)
                         if cmd=="/query":
                             fileName=recvd.split(" ")[1]
                             self.queryFile(fileName, clientSocket)
                         if cmd=="/setNick":
                             self.setNick(formatedAddress, recvd.split(" ")
[1])

                except Exception,e :
                     print "Error: ", e



        except Exception, e:
                  print "Data: ", data
                  print "Error: ", e
                  self.alSocks.remove(clientSocket)
```

انا اتبعت طريقة بسيطة شوية هنا وهو إنى بعت تسجيل البيانات على صورة tuple مكونة من
(register”, files=[], listeningPort/“)
وحولتها ل peerInfo object
وبعد كدا بعض الإختبارات لل command نفسه لو كان query نعمل كذا لو كان كذا نعمل كذا وإذا مش كان موجود فى الأوامر المدعمة بال Server تبقة مجرد رسالة تتبعت لكل الأهل والأحباب

فى حال إن حصل اى ايرور(خطأ) يتم حذف ال Client من ال alSocks لنعرف إنه غير active او متصل بالسرفر حاليا
طب فى حال لو إن الداتا اللى إتبعتت مجرد مسج عادية ؟ يعنى هنحتاج نبعتها لكل المستخدمين ماعدا بالطبع اللى ارسلها مش كدا ؟
نحصل على كل المستخدمين منين ؟
اها تمام من ال alSocks اللى بتعبر عن كل ال Clients المتفاعلين مع السرفر حاليا
**فشطة**

```
def sendToAll(self, msg, clientEx):
        print "Message Recieved: ", msg
        try:
            for sock in self.alSocks:
                if not sock==clientEx:
                    sock.send(serialize(msg))
                else:
                    pass
        except Exception, e:
                                                print "Error: ", e
```

جميل جدا.. فى حال لو المستخدم عايز يستعرض المتسخدمين الموجودين "اكيد رد السرفر هيكون ليه لوحده مش كدا ؟"super
فهتكون ال showallTo method كالتالى

```
def showAllTo(self, clientSocket):

        data="\n---------------------------------------\nOnline Users:\n"
        for addr, pInfo in self.db.items():
            data += pInfo.alias + " -> " +addr +"\n"
        data +="\n---------------------------------------\n"
        print data
                                        clientSocket.send(serialize(data))
```

ملحوظة انا فضلت اعمل serialize لكل الداتا اللى هتتبعت حتى لو strings عشان مش اقعد اتعب نفسى فى ال de-bugging واعملهم de-serialize فى الطرف التانى

لمعالجة امر تغيير ال Nick وهو /setNick

```
    def setNick(self, to, newNick):
        self.db[to].alias=newNick
        print "Nick Changed..."
                                                    print self.db[to]
```

جميل جدا ناقص ال Querying files
اللى هيطلب الإستعلام هو واحد مش كل الناس فلازم نمرر ال socket بتاعه فى ال method + نبعتله ال داتا

```
def queryFile(self, fileName, clientSocket):
        print "Querying: ", fileName
        data=""
        for addr, pInfo in self.db.items():
            if fileName in pInfo.sharedFiles:
                data += "\n"+addr + " | " + pInfo.alias + " => " +
fileName
                data += "\n\t" + pInfo.alias + " Listens at: "+
str(pInfo.listeningPort)
        print data
                                        clientSocket.send(serialize(data))
```

كدا انهينا ال Discovery Server او ال Indexer

ننقل على ال Client او ال Peer
ال Peer لازم يتفاعل مع ال Server و يقدر يحمل ملفات من ال Peers التانين و ان يكون فى peers يقدرو يحملو منهم بردو فكدا هنعمل client ليتعامل مع السرفر ونعمل internal server يكون مسئول عن عملية ال دونلود او ال fetching

ال Constructor بتاعه هياخد
1- alias
2- list of sharedFiles
3- ال endPoint الخاصة بالسرفر
ال endPoint هى ال Server IP + port

بمجرد مايتم ال connection مع ال server نهندله فى thread جديد

```
def __init__(self, alias, serverAddr=(), sharedFiles=[]):
```

```
  self.alias=alias
  self.serverAddr=serverAddr
  self.sharedFiles=sharedFiles
  self.tcpClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  self.tcpClient.connect(self.serverAddr)
  self.BUF=1024
  self.listeningPort=rnd.randint(8081, 10000)
  self.pInfo=PeerInfo(self.alias, self.sharedFiles, self.listeningPort)

  print "\nConnected to server..."
  self.tClientToServer=threading.Thread(target=self.clientToServerHandler, args=[])
  self.tClientToServer.start()
```

رائع جدا
بس بردو لازم نجهز لل Peers اللى هيحاولو يعملو fetch ل files من عندنا فهنعمل server object وبردو نشغله فى thread جديد صح كدا ؟

```
#listen for connections in background..
  self.addr=('', self.listeningPort)
  self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  self.listener.bind(self.addr)
 self.tListening=threading.Thread(target=self.listeningHandler, args=[])
  self.tListening.start()
```

جميل بما إنى على جهاز واحد فهتحصل مشكلة انى مش هقدر اعمل set ل port معين اربطه بالسرفر لأن بكل بساطة كل ال Peers هيحاولو يربطو على السرفر دا .. فبكل بساطة هنستخدم randint method الموجودة ب random module ونحصل على اى random int بس لاحظ إنه يكون فوق ال 1024 وبردو نبلغ بيه ال Discovery Server

```
self.listeningPort=rnd.randint(8081, 10000)
```

اولا ال registerAtServer Method

```
 def registerAtServer(self):
  msg=("/register", self.alias, self.sharedFiles, self.listeningPort)
  self.tcpClient.send(serialize(msg))
```

وهى اول ميثود هيتم إستدعائها بمجرد إنى اعمل connect على السرفر

```
def clientToServerHandler(self):
   print "Start Chatting.."
   #first register the files...
   self.registerAtServer()
   while True:
     tServerToClient=threading.Thread(target=self.serverToClientHandler, args=[])
     tServerToClient.start()
     data=raw_input()
     if not data: continue
     if data.lower=="exit": exit()

     if data.split(" ")[0]=="/fetch":
       fileneeded=data.split(" ")[1]
       addr=data.split(" ")[2]
       tFetchFile=threading.Thread(target=self.fetchFile, args=[addr, fileneeded])
       tFetchFile.start()
     else:
       msg=self.alias+": "+data
       self.tcpClient.send(serialize([msg]))
```

بما إن ال client مش ليه غير وظيفة واحدة بس اللى مش هتتعامل مع ال server وهى ال fetching فهنختبرها الأول إذا هى المطلوبة او لأ.. فى حال آه نبدأها فى thread جديد خاص بالتعامل معاها لو لأ تبقة مجرد رسالة او امر زى إستعلام او تغيير بيانات فهنبعته للسرفر والسرفر يحلله

نهدل رسايل ال server فى thread هيستخدم ال serverToClientHandler method

```
 def serverToClientHandler(self):
   while True:
     data=deserialize(self.tcpClient.recv(self.BUF))
     if not data: break
     if isinstance(data, list): #data ['tina: hi']
       print data[0]
     else:
       print data
```

عملية ال fetching بكل بساطة هتتعامل معاها كالتالى

```
def fetchFile(self, addr, fileneeded):
   #addr is formated => addr:listeningPort
   endPoint=addr.split(":")[0], int(addr.split(":")[1])
   fetchTCPClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   fetchTCPClient.connect(endPoint)
   fetchTCPClient.sendall(serialize(("/fetch", fileneeded)))
   tDownloadFile=threading.Thread(target=self.downloadFile, args=[fetchTCPClient, fileneeded])
   tDownloadFile.start()
```

ونبدأ فيها thread جديد للل download بيستخدم ال downloadFile method و هيمرر ال tcpClient, الفايل المطلوب
ك args ليها

```
def downloadFile(self, fetchTCPClient, fileneeded):
   f=file(fileneeded, "wb")
   while True:
     try:
       buf=fetchTCPClient.recv(self.BUF)
       if not buf:
         break
       f.write(buf)
     except EOFError, eofErr:
       print "EOFError: ", eofErr
     except Exception, e:
       print "Error: ", e
       break
   print "File Downloaded!"
   f.close()
   fetchTCPClient.close()
```

جميل جدا احنا كدا شبه خلصنا .. ناقص شئ واحد بس وهو التعامل مع ال clients اللى نريد ان يحملو فايلات مننا
-خليك فاكر إننا already مشغلين thread خاص بال listening للطلبات

```
self.tListening=threading.Thread(target=self.listeningHandler, args=[])
   self.tListening.start()
```

جميل ال thread دا بيستخدم listeningHandler method وال method دى مش هتاخد args

```
 def listeningHandler(self):
   self.listener.listen(5)
   while True:
     clientSocket, clientAddr=self.listener.accept()
     tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
     tClientHandling.start()
```

محتاجين إنها تكون Multi-threaded عشان مش نربط نفسنا مع مستخدم واحد بس
فهنعمل thread جديد يعالج كل client يعمل connect على ال internal server

```
def clientHandler(self, clientSocket):
   rcvd=clientSocket.recv(self.BUF)
   data=deserialize(rcvd)
   if isinstance(data, tuple):
```

```
    if data[0]=="/fetch": #go on..
      fileneeded=data[1] #(/fetch, fileneeded, from)
      print "File Request: ", fileneeded
      f=file(fileneeded, "rb")
      while True:
        try:
          buf=f.read(self.BUF)
          if not buf:
            break
          clientSocket.send(buf)
        except Exception, e:
          print "Error: ", e
          break

      f.close()
      clientSocket.close()
      print "Copied!"
```

وبس كدا

```
if __name__=="__main__":
  alias=raw_input("Alias: ")
  sharedFiles=os.listdir(os.getcwd())
  peer=Peer(alias, ('localhost', 8080), sharedFiles)
```

طبعا تقدر تظبطها بحيث إنك تمرر ال addr الخاص بال server من ال command line او حتى من prompt

الأكواد

```
Discovery Server
Code:
#!bin/python
import socket
import sys
import os
import threading
from utils import serialize, deserialize, PeerInfo
class NotSupportedCommand(Exception):
  pass
class DiscoveryServer(object):
  '''Indexer...'''
```

```
def __init__(self, port, maxPeers=5):
  self.port=port
  addr=('', self.port)
  self.maxPeers=maxPeers
  self.supportedCommands=["/register", "/setNick", "/setSharedFiles", "/showall", "/query"]
  self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  self.listener.bind(addr)
  self.tListening=threading.Thread(target=self.listeningHandler, args=[])
  self.tListening.start()
  self.alSocks=[]
 # {clientAddr:peerInfo Object}
  self.db={}
  self.log=[]
  self.BUF=2048

def listeningHandler(self):
  self.listener.listen(self.maxPeers)
  print "Server Started..."
  while True:
    clientSocket, clientAddr=self.listener.accept()
    print "Gotta a connection from", clientAddr
    tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
    tClientHandling.start()
  clientSocket.close()
def clientHandler(self, clientSocket):
 self.alSocks += [clientSocket]
 formatedAddress=clientSocket.getpeername()[0]+":"+str(clientSocket.getpeername()[1])
 objString=""
 try:
   while True:
     objString=clientSocket.recv(self.BUF)
     if not objString:
        break
     data=deserialize(objString)
     #print data
     tAnalyzeData=threading.Thread(target=self.analyzeData, args=[data, clientSocket])
     tAnalyzeData.start()

     objString=""

 except Exception, e:
   print "E: ", e
   print clientSocket.getpeername(), " closed.."
   self.alSocks.remove(clientSocket)
   del self.db[formatedAddress]
```

```
def analyzeData(self, data, clientSocket):
 formatedAddress=clientSocket.getpeername()[0]+":"+str(clientSocket.getpeername()[1])
 try:
   if isinstance(data, tuple): #registering...
     pInfo=PeerInfo(data[1], data[2], data[3]) #(register, alias, files, port)
     print "Registering: ", pInfo.alias
     print pInfo
     self.db[formatedAddress]=pInfo #peerInfo object..
     print self.db
   if isinstance(data, list):
     try:
       #split the sender's alias..
       #recvd=['tina: /showall']
       recvd=data[0].split(": ")[1]
       cmd=recvd.split(" ")[0]
       # test cmd...
       if not cmd in self.supportedCommands:
         self.sendToAll(data, clientSocket)
       else:
         if cmd=="/showall":
           self.showAllTo(clientSocket)
         if cmd=="/query":
           fileName=recvd.split(" ")[1]
           self.queryFile(fileName, clientSocket)
         if cmd=="/setNick":
           self.setNick(formatedAddress, recvd.split(" ")[1])

     except Exception,e :
       print "Error: ", e


 except Exception, e:
      print "Data: ", data
      print "Error: ", e
      self.alSocks.remove(clientSocket)

def queryFile(self, fileName, clientSocket):
  print "Querying: ", fileName
  data=""
  for addr, pInfo in self.db.items():
    if fileName in pInfo.sharedFiles:
      data += "\n"+addr + " | " + pInfo.alias + " => " + fileName
      data += "\n\t" + pInfo.alias + " Listens at: "+ str(pInfo.listeningPort)
  print data
  clientSocket.send(serialize(data))

def showAllTo(self, clientSocket):
```

```
    data="\n---------------------------------------\nOnline Users:\n"
    for addr, pInfo in self.db.items():
      data += pInfo.alias + " -> " +addr +"\n"
    data +="\n---------------------------------------\n"
    print data
    clientSocket.send(serialize(data))

  def sendToAll(self, msg, clientEx):
    print "Message Recieved: ", msg
    try:
      for sock in self.alSocks:
        if not sock==clientEx:
          sock.send(serialize(msg))
        else:
          pass
    except Exception, e:
      print "Error: ", e
  def setNick(self, to, newNick):
    self.db[to].alias=newNick
    print "Nick Changed..."
    print self.db[to]

if __name__=="__main__":
  discoveryServer=DiscoveryServer(8080)
```

Utils:

```
import marshal

class PeerInfo(object):
  def __init__(self, alias, sharedFiles, listeningPort):
    self.alias=alias
    self.sharedFiles=sharedFiles
    self.listeningPort=listeningPort
  def __str__(self):
    sb="Alias: " + self.alias
    sb += "\nFiles: " + str(self.sharedFiles)
    sb +="\nListens At:" + str(self.listeningPort)
    return sb

def serialize(obj):
  '''Serialize an object to a string...'''
  return marshal.dumps(obj)
def deserialize(objString):
  '''Deserialize an object string...'''
```

```
  return marshal.loads(objString)
if __name__=="__main__":
  p=PeerInfo("ahmed", [1, 2, 3 ,4], 80)
  print p
  print "alias: ",p.alias
  print "files: ", p.sharedFiles
  print "port : ", p.listeningPort
```

Peer

```
#!bin/python
import socket
import sys
import os
import threading
from utils import serialize, deserialize, PeerInfo
import random as rnd

class Peer(object):
  def __init__(self, alias, serverAddr=(), sharedFiles=[]):

    self.alias=alias
    self.serverAddr=serverAddr
    self.sharedFiles=sharedFiles
    self.tcpClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.tcpClient.connect(self.serverAddr)
    self.BUF=1024
    self.listeningPort=rnd.randint(8081, 10000)
    self.pInfo=PeerInfo(self.alias, self.sharedFiles, self.listeningPort)

    print "\nConnected to server..."
    self.tClientToServer=threading.Thread(target=self.clientToServerHandler, args=[])
    self.tClientToServer.start()
    #listen for connections in background..
    self.addr=('', self.listeningPort)
    self.listener=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    self.listener.bind(self.addr)
    self.tListening=threading.Thread(target=self.listeningHandler, args=[])
    self.tListening.start()

  def registerAtServer(self):
    msg=("/register", self.alias, self.sharedFiles, self.listeningPort)
    self.tcpClient.send(serialize(msg))
```

```
 def clientToServerHandler(self):
  print "Start Chatting.."
  #first register the files...
  self.registerAtServer()
  while True:
   tServerToClient=threading.Thread(target=self.serverToClientHandler, args=[])
   tServerToClient.start()
   data=raw_input()
   if not data: continue
   if data.lower=="exit": exit()

   if data.split(" ")[0]=="/fetch":
    fileneeded=data.split(" ")[1]
    addr=data.split(" ")[2]
    tFetchFile=threading.Thread(target=self.fetchFile, args=[addr, fileneeded])
    tFetchFile.start()
   else:
    msg=self.alias+": "+data
    self.tcpClient.send(serialize([msg]))
 def serverToClientHandler(self):
  while True:
   data=deserialize(self.tcpClient.recv(self.BUF))
   if not data: break
   if isinstance(data, list): #data ['tina: hi']
    print data[0]
   else:
    print data
 def fetchFile(self, addr, fileneeded):
  #addr is formated => addr:listeningPort
  endPoint=addr.split(":")[0], int(addr.split(":")[1])
  fetchTCPClient=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  fetchTCPClient.connect(endPoint)
  fetchTCPClient.sendall(serialize(("/fetch", fileneeded)))
  tDownloadFile=threading.Thread(target=self.downloadFile, args=[fetchTCPClient, fileneeded])
  tDownloadFile.start()
 def downloadFile(self, fetchTCPClient, fileneeded):
##  try:
##   f=file(fileneeded, "wb")
##   while True:
##       dataRcvd=fetchTCPClient.recv(self.BUF)
##       if not dataRcvd: break
##       f.write(dataRcvd)
##
##   print "File Downloaded!"
##
##  except Exception, e:
##   print "Error: ", e
```

```
##
##   finally:
##     print "Closing the file.."
##     fetchTCPClient.close()
##     f.close()
    f=file(fileneeded, "wb")
    while True:
      try:
        buf=fetchTCPClient.recv(self.BUF)
        if not buf:
          break
        f.write(buf)
      except EOFError, eofErr:
        print "EOFError: ", eofErr
      except Exception, e:
        print "Error: ", e
        break
    print "File Downloaded!"
    f.close()
    fetchTCPClient.close()


  def listeningHandler(self):
    self.listener.listen(5)
    while True:
      clientSocket, clientAddr=self.listener.accept()
      tClientHandling=threading.Thread(target=self.clientHandler, args=[clientSocket])
      tClientHandling.start()

  def clientHandler(self, clientSocket):
    rcvd=clientSocket.recv(self.BUF)
    data=deserialize(rcvd)
    if isinstance(data, tuple):
      if data[0]=="/fetch": #go on..
        fileneeded=data[1] #(/fetch, fileneeded, from)
        print "File Request: ", fileneeded
        f=file(fileneeded, "rb")
        while True:
          try:
            buf=f.read(self.BUF)
            if not buf:
              break
            clientSocket.send(buf)
          except Exception, e:
            print "Error: ", e
            break

        f.close()
```

```
        clientSocket.close()
        print "Copied!"



if __name__=="__main__":
  alias=raw_input("Alias: ")
  sharedFiles=os.listdir(os.getcwd())
  peer=Peer(alias, ('localhost', 8080), sharedFiles)
```

جميل كدا انهيت البرنامج ولكن فى **trick** اكيد لأننا كاتبين عنوان الفصل App with a Bug
فى عدة مشاكل فى التطبيق
1- استخدام marshal (يفضل استخدم pickle لعمل serialize للكائنات) ولكن بما ان التطبيق فورى فمش ليها تأثير
2- الإستخدام المكثف للثريدينج (يفضل انشاء الثريدات كالتالى
1- ثريد معالجة كل عميل
وتحليل الداتا فى نفس الثريد وليس ثريد آخر (ربما لن تشعر بالفرق الآن ولكن عند محاولة نقل التطبيق الى قاعدة بيانات)
وبالنسبة للعميل كذلك لاتقم بتحليل الداتا فى ثريد منفصل الا اذا كنت تعلم ماذا تفعل

يفضل دائما استخدام ال stdlib لضمان عدم الوقوع فى مشكلات فمثلا للثريدينج تستطيع استخدام ForkingMixIn او ThreadingMixIn

إضافات للتطبيق
- استخدام قاعدة بيانات (ربما sqlite3 مثلا او MySQL)
2- وفر خاصية ال resume للملفات اللى بتتحمل
3- فعل ال LOG "استفيد من الرسايل اللى بيتعملها print" على السرفر
4- استخدم ال Discovery Server ك Web Service
5- واجهة رسومية بسيطة
6- اوامر اكتر لهندلة ال PeerInfo وامتيازات خاصة

## Implementing Enums

لاحظ استخدامنا فى الفصل السابق لأوامر فى التعامل بين السرفر والعملاء مثل /setSharedFiles او /query الخ الخ
ولكن لفصل افضل للتطبيق تستطيع انشاء Enums

احيانا بنحتاج اننا نجمع مجموعة من البيانات تحت اسم معين مثلا ايام الأسبوع (احد اتنين ثلاثاء .. الخ ) والألوان (ابيض ازرق اخضر .. الخ)

مثلا فى باسكال

```
type
  TDay = (Saturday=1, Sunday=2, Monday=3, Tuesday=4, Wednesday=5, Thursday=6, Friday=7);
```

الكود المشابه ليه فى بايثون ممكن يكون

```
(Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday=range(1, 7
```

ففى كذا طريقة فى بايثون
وممكن تعرفهم كالتالى

```
class MyDays(object): #as rawenum
    sunday, monday, tuesday=range(3)
class MyKVDays(object):
    sunday, monday, tuesday=0, 9, 2
```

ممكن نعمل class كالتالى

```
class RawEnum(object):
   def __init__(self, start, *enum): #do we need to set a start, end, step? very fancy...
      self._kw=dict(zip(enum, range(start, start+len(enum))))
      counter=start
      for arg in enum: #assuming it doesn't exist.
```

```
        self.__setattr__(arg, start)
        counter += 1
    __str__=lambda self: str(self._kw)    #Can be solved with an ordered dict.
```

وهنا بيتحدد فيه قيمة البداية لل enum فقط

```
    Colors=RawEnum(5, 'white', 'black', 'blue')
    print Colors.white
    print Colors
```

او اننا نستخدم **kwargs

```
class KVEnum(object):
    def __init__(self, **kwargs):
        self.__kw=kwargs
        for k, v in kwargs.items():
            self.__setattr__(k, v)
    __str__=lambda self: str(self.__kw)
```

بحيث نقدر نستخدمها كالتالى

```
    Days=KVEnum(sunday=0, monday=9, tuesday=2)
    print Days
    print Days.monday
```

## FTPing

بايثون بتقدملك موديلز عديدة لمهام كثيرة مثل ftplib (للقيام بعمليات العميل الخاصة ببروتوكول FTP)
استخدامها مباشر
1- استدعاء الموديل

```
import ftplib
```

2- بيانات الدخول

```
HOST="YOUR_HOST"
USER="YOUR_USERNAME"
PASSWD="YOUR_PASSWORD"
```

3- انشاء كائن من الصف FTP

```
ftp=ftplib.FTP()
```

4- انشاء الإتصال بإستخدام الطريقة connect اللتى تأخذ معاملات ال host و رقم البورت (افتراضيا 21)

```
(ftp.connect(HOST, 21
```

5- الدخول بإستخدام الطريقة login وتأخذ معاملات user, password

```
ftp.login(USER, PASSWD)
```

6- التعامل الخاص بيك

بعض الطرق

getwelcome()

لعرض رسالة الترحيب

rename(old, new)

لإعادة تسمية old ب new

cwd(path)

تغيير مجلد العمل الحالى current working directory

pwd()

مسار مجلد العمل الحالى

mkd(path)

انشاء مجلد path

delete(f)

حذف الملف f

rmd(d)

حذف المجلد d

size(f)

الحصول على مساحة ملف f

quit()

ارسال رسالة QUIT

close()

لإنهاء الإتصال

set_pasv(boolean)

هل نوع الإتصال سلبى ام لأ ؟ passive mode

retrbinary( command, callback[, maxblocksize[, rest]])

للحصول على ملف RETR f واستدعاء callback على كل block عند اكتمال تحميلها

storbinary(cmd, file[, block])

تخزين ملف STOR file ما مع تحديد مساحة قطع النقل لكل مرة

abort()

الغاء عملية نقل ملف

dir(p)

عرض ال listing الخاصة ب p

للمزيد راجع الوثائق الرسمية او راجع سورس الموديل ftplib.py

## XMLRPC what?

اذا لم تكن مهتما ب XML-RPC فكن حرا للإنتقال للفصل القادم
فى ابسط الصور XML-RPC هو بروتوكول Remote Procedure Call عبر بروتوكول HTTP
للمزيد http://en.wikipedia.org/wiki/XML-RPC

نوضح الموضوع بمثال بسيط

```
#!bin/python


from SimpleXMLRPCServer import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler
from SocketServer import ThreadingMixIn

class Greeter(object):

    def hi(self):
        """Returns hi message"""
        return "Hi"

    def bye(self):
        """Returns bye message"""
        return "Bye"

    def say(self, what):
        """Returns Simone says message"""
        return "Simone says: "+ what

class MyServer(ThreadingMixIn, SimpleXMLRPCServer):

    pass

def test():
    addr=('', 40002)
    srvr=MyServer(addr, SimpleXMLRPCRequestHandler)
    srvr.register_instance(Greeter())
    srvr.register_introspection_functions()
    srvr.serve_forever()
    #print "Started..."

if __name__=="__main__":
    test()
```

للوهلة الأولى سيتبادر لذهنك اننا ننشئ سرفر متعدد الخيوط (ThreadingMixIn واستخدمنا بدل ال SocketServer  صف مشابة وهو

؟ SimpleXMLRPCServer

نعم بالفعل

ولدينا صف عادى جدا

```
class Greeter(object):

    def hi(self):
        """Returns hi message"""
        return "Hi"

    def bye(self):
        """Returns bye message"""
        return "Bye"

    def say(self, what):
        """Returns Simone says message"""
        return "Simone says: "+ what
```

كل ماهنالك اننا نريد ان نتيح امكانية استخدام طرق ذلك الصف عبر ال HTTP
لكائن SimpleXMLRPCServer بعض الطرق مثل register_instance اللتى تقوم بتسجيل ذلك الكائن على السرفر
و register_introspection_functions تقوم بتسجيل بعض الدوال لمعرفة مايتعلق بذلك الكائن مثل
system.listMethods (للحصول على الطرق الخاصة به ) و system.methodSignature للحصول على توقيع الطريقة
(اسمها والمعاملات وال retrun ) و methodHelp للحصول على نص المساعدة الخاص بالطريقة وذلك بوضع
اسمها كمعامل لهذه الطريقة

```
    srvr.register_instance(Greeter())
    srvr.register_introspection_functions()
```

قم بتشغيل ال سرفر ونأتى للعميل

```
import xmlrpclib

s = xmlrpclib.ServerProxy('http://localhost:40002')
print s.system.listMethods()
print s.hi()
print s.bye()
print s.say("Something")


print s.system.methodHelp('say')
```

1- ننشئ كائن ServerProxy نحدد فيه عنوان السرفر ورقم البورت اللذى ينصت اليه

```
s = xmlrpclib.ServerProxy('http://localhost:40002')
```

2- للحصول على قائمة الطرق المتاحة على السرفر نستخدم system.listMethods
3- استغلال الطرق المتاحة مثل استدعاء الطرق hi او bye

```
print s.hi()
print s.bye()
```

4- ايضا استغلال الطرق اللتى قد تأخذ معاملات ما مثل say

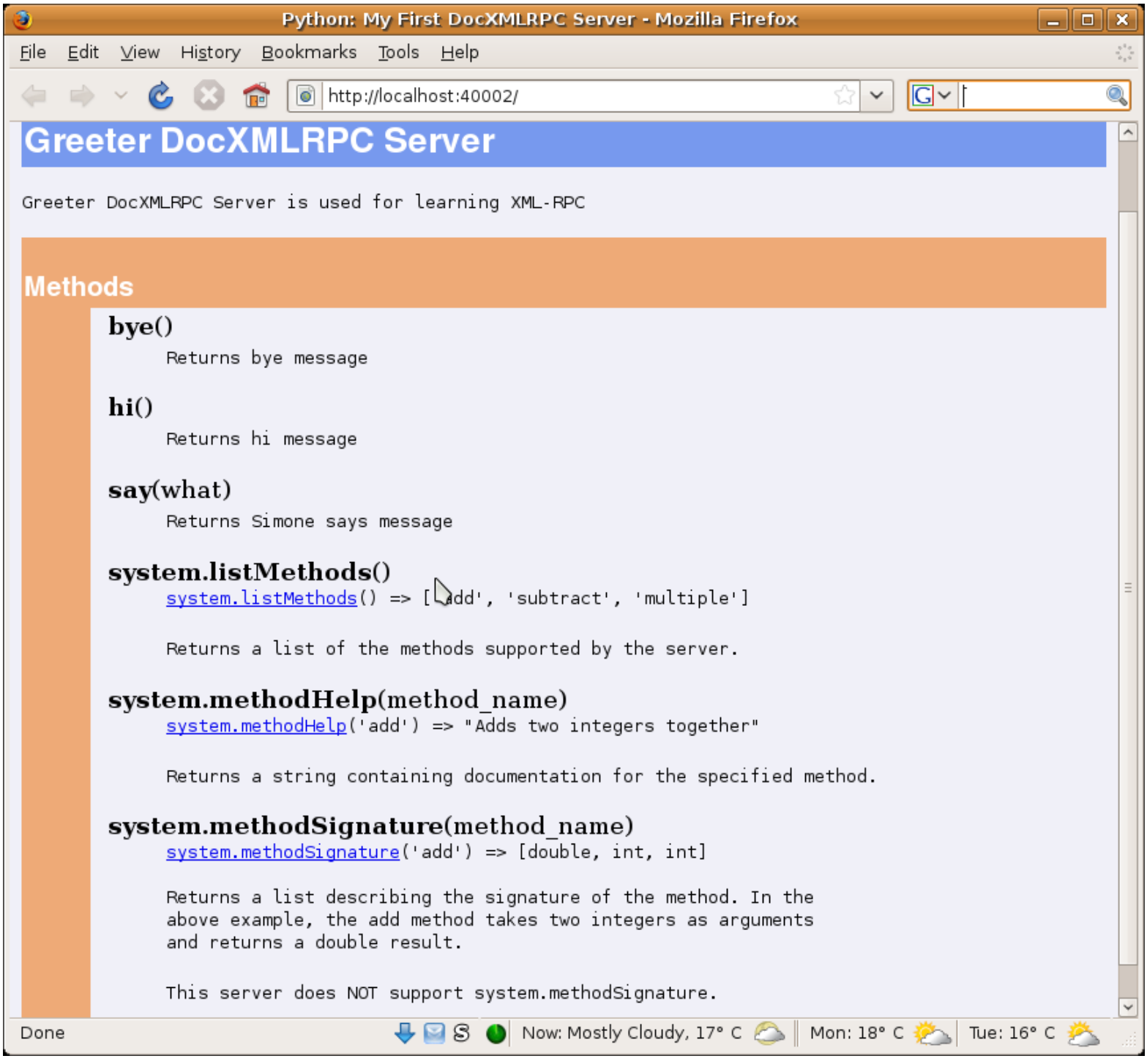
```
print s.say("Something")
```

سيكون الناتج

```
striky@striky-desktop:~/workspace/pytut/src$ python xmlrpcclient1.py
['bye', 'hi', 'say', 'system.listMethods', 'system.methodHelp', 'system.methodSignature']
Hi
Bye
Simone says: Something
Returns Simone says message
```

يوجد صف اخر وهو المفضل استخدامه DocXMLRPCServer وذلك للمساعدة فى عرض ال documentation ايضا

## DocXMLRPCServer



مثل سابقه ولكن له بعض الطرق الإضافية مثل set_server_title لوضع العنوان و set_server_name لوضع اسمه فى اعلى الصفحة و set_server_documenation لوضع وثيقة خاصة به

```
from DocXMLRPCServer import DocXMLRPCServer, DocXMLRPCRequestHandler
from SocketServer import ThreadingMixIn

class Greeter(object):

    def hi(self):
        """Returns hi message"""
        return "Hi"

    def bye(self):
        """Returns bye message"""
        return "Bye"

    def say(self, what):
        """Returns Simone says message"""
        return "Simone says: "+ what

class MyServer(ThreadingMixIn, DocXMLRPCServer):

    pass

def test():
    addr=('', 40002)
    srvr=MyServer(addr, DocXMLRPCRequestHandler)
    ##server methods...
    srvr.set_server_title("My First DocXMLRPC Server")
    srvr.set_server_name("Greeter DocXMLRPC Server")
    srvr.set_server_documentation("Greeter DocXMLRPC Server is used for learning XML-RPC")
    srvr.register_instance(Greeter())
    srvr.register_introspection_functions()
    srvr.serve_forever()
    #print "Started..."

if __name__=="__main__":
    test()
```

تستطيع الإستفادة من هذا السرفر فى لغات اخرى مثل روبى مثلا

```
##Ruby
striky@striky-desktop:~$ irb
irb(main):001:0> require "xmlrpc/client"
=> true
irb(main):002:0> require "pp"
=> true
irb(main):004:0> s=XMLRPC::Client.new2("http://localhost:40002")
=> #<XMLRPC::Client:0xb7aa2a50 @user=nil, @proxy_port=nil, @auth=nil, @cookie=nil, @create=nil,
@port=40002, @http=#<Net::HTTP localhost:40002 open=false>, @proxy_host=nil,
@http_last_response=nil, @parser=nil, @timeout=30, @path="/RPC2", @password=nil,
@http_header_extra=nil, @use_ssl=false, @host="localhost">
irb(main):005:0> s
=> #<XMLRPC::Client:0xb7aa2a50 @user=nil, @proxy_port=nil, @auth=nil, @cookie=nil, @create=nil,
@port=40002, @http=#<Net::HTTP localhost:40002 open=false>, @proxy_host=nil,
@http_last_response=nil, @parser=nil, @timeout=30, @path="/RPC2", @password=nil,
@http_header_extra=nil, @use_ssl=false, @host="localhost">
irb(main):006:0> s.call('hi')
=> "Hi"
irb(main):007:0> s.call('bye')
=> "Bye"
irb(main):008:0> s.call('say', "Hello, World!")
=> "Simone says: Hello, World!"
```

## Quote of the Day

هننشئ فى المثال دا سرفس مشابهه ل "اقتباس اليوم"
وهو سرفر يعمل فى الخلفية ويرسل اقتباس عشوائى لكل عميل ثم يغلق الإتصال معه

1- ملف ال quotes.txt

```
Never pretend to a love which you do not actually feel, for love is not
ours to command.  --Alan Watts
To love deeply in one direction makes us more loving in all others.
--Anne-Sophie Swetchine
There is always some madness in love. But there is also always some reason
in madness. --Friedrich Nietzsche
Life is wasted on the living. --Douglas Adams
Life is just a chance to grow a soul. - A. Powell
```

2- ال quoter وهو موديل للحصول على الإقتباسات من ملف ما

```
#!bin/python

from __future__ import with_statement
import random as rnd


def get_quotes(f="quotes.txt"):
   ###quotes are separated by \n
   with open(f) as fh:
      lines=fh.read()
      quotes=lines.split("\n")
      return quotes

def get_random_quote(quotes=get_quotes()):
   return rnd.choice(quotes)

if __name__=="__main__":
   print get_random_quote()
```

كما نرى السطر الأول بيستدعى ال with_statement من المستقبل :d الإمكانات اللتى تم تقرير اضافتها
للإصدارات الأحدث من بايثون والموديل random للحصول على اختيار عشوائى
*الدالة get_quotes متغيرة فى f وهو مسار الملف اللذى يحوى الإقتباسات وجعلنها quotes.txt افتراضيا
لاحظ فى إستخدام with تم استبدال try/except لأنها تتم داخليا

نحصل على الإقتباسات "كل سطر يحوى اقتباس" (ممكن تعمل strip للتأكيد على عدم وجود اسطر زائدة فى الملف

*الدالة get_random_quote متغيرة فى quotes وهو قائمة list بالإقتباسات وجعلناها افتراضيا قيمة get_quotes() ، تستخدم للحصول على اختيار عشوائى من quotes ويتم ذلك بإستدعاء الدالة choice اللتى تعيد لنا اقتباس عشوائى متتابعة (قائمة) وهنا فى المثال هى quotes

3- السرفر quotd

```
from SocketServer import TCPServer, StreamRequestHandler, ThreadingMixIn
import threading
import quoter
from django.utils import daemonize

class MyServer(ThreadingMixIn,TCPServer):
    pass

class MyStreamRequestHandler(StreamRequestHandler):

    def handle(self):

        self.request.send(self.get_quote()+"\r\n")

    def get_quote(self):
        return quoter.get_random_quote()

def go(endpoint=('', 58000)):
    addr=endpoint
    tcpServer=MyServer(addr, MyStreamRequestHandler)
    tcpServer.allow_reuse_address=1
    print "Server started..."
    tcpServer.serve_forever() #inf. loop




if __name__=="__main__":
    try:
        daemonize.become_daemon()
        go()
    except KeyboardInterrupt:
        exit()
```

الكود بسيط جدا عند اتصال اى عميل يتم الحصول على اقتباس عشوائى ويتم ارساله
بس دا كود عادى جدا زى اللى سبق؟

بالفعل ولكن يختلف فى وجود الوحدة daemonize من django.utils
وتنفيذ الدالة become_daemon منها
بكل بساطة هذا هو الكود المطلوب منك للتحويل الى daemon!
تعالى نلقى نظرة على daemonize.become_daemon

```
import os
import sys

if os.name == 'posix':
    def become_daemon(our_home_dir='.', out_log='/dev/null',
                      err_log='/dev/null', umask=022):
        "Robustly turn into a UNIX daemon, running in our_home_dir."
        # First fork
        try:
            if os.fork() > 0:
                sys.exit(0)     # kill off parent
        except OSError, e:
            sys.stderr.write("fork #1 failed: (%d) %s\n" % (e.errno, e.strerror))
            sys.exit(1)
        os.setsid()
        os.chdir(our_home_dir)
        os.umask(umask)

        # Second fork
        try:
            if os.fork() > 0:
                os._exit(0)
        except OSError, e:
            sys.stderr.write("fork #2 failed: (%d) %s\n" % (e.errno, e.strerror))
            os._exit(1)

        si = open('/dev/null', 'r')
        so = open(out_log, 'a+', 0)
        se = open(err_log, 'a+', 0)
        os.dup2(si.fileno(), sys.stdin.fileno())
        os.dup2(so.fileno(), sys.stdout.fileno())
        os.dup2(se.fileno(), sys.stderr.fileno())
        # Set custom file descriptors so that they get proper buffering.
        sys.stdout, sys.stderr = so, se
else:
    def become_daemon(our_home_dir='.', out_log=None, err_log=None, umask=022):
        """
        If we're not running under a POSIX system, just simulate the daemon
        mode by doing redirections and directory changing.
        """
        os.chdir(our_home_dir)
        os.umask(umask)
        sys.stdin.close()
```

```
    sys.stdout.close()
    sys.stderr.close()
    if err_log:
        sys.stderr = open(err_log, 'a', 0)
    else:
        sys.stderr = NullDevice()
    if out_log:
        sys.stdout = open(out_log, 'a', 0)
    else:
        sys.stdout = NullDevice()

class NullDevice:
    "A writeable object that writes to nowhere -- like /dev/null."
    def write(self, s):
        pass
```

اذا لم تفهم الكود السابق من خبرة سابقة فى برمجة لينكس لاتقلق يكفيك استخدام الدالة مباشرة

استخدم telnet localhost 58000

```
striky@striky-desktop:~/workspace/pytut/src/nettut/quoter$ telnet localhost 58000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Life is just a chance to grow a soul. - A. Powell
Connection closed by foreign host.
```

# *Chapter 13 (Python on the WEB)*

## Grok

هو اطار عمل مبنى على مكتبات Zope

التستيب

بكل سهولة

```
easy_install grokproject
```

وبعد كدا

```
striky@striky-desktop:~/workspace/pytut/src/nettut$ mkdir groktut
striky@striky-desktop:~/workspace/pytut/src/nettut$ cd groktut/
```

وقم بتنفيذ grokproject والحقه بإسم المشروع

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut$ grokproject FirstProject
```

اسم المدير وليكن grok

```
Enter user (Name of an initial administrator user): grok
```

كلمة السر (لن تظهر تم استخدام getpass لإلغاء ظهورها) اكتبها grok

```
Enter passwd (Password for the initial administrator user):
Downloading info about versions...
Creating directory ./FirstProject
Invoking zc.buildout...
Develop: '/home/striky/workspace/pytut/src/nettut/groktut/FirstProject/.'
Installing eggbasket.
Getting distribution for 'grok==0.14.1'.
eggbasket: Distributions are not installed. A tarball will be downloaded.
eggbasket: Distributions are not installed. A tarball will be downloaded.
eggbasket: Downloading http://grok.zope.org/releaseinfo/grok-eggs-0.14.1.tgz ...
eggbasket: Downloading http://grok.zope.org/releaseinfo/grok-eggs-0.14.1.tgz ...
eggbasket: Finished downloading.
eggbasket: Finished downloading.
eggbasket: Extracting tarball contents...
eggbasket: Extracting tarball contents...
eggbasket: Installing eggs to /home/striky/.buildout/eggs which will take a while...
eggbasket: Installing eggs to /home/striky/.buildout/eggs which will take a while...
```

```
Getting distribution for 'grok==0.14.1'.
Got grok 0.14.1.
```

الآن انت جاهز

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut$ cd FirstProject/
```

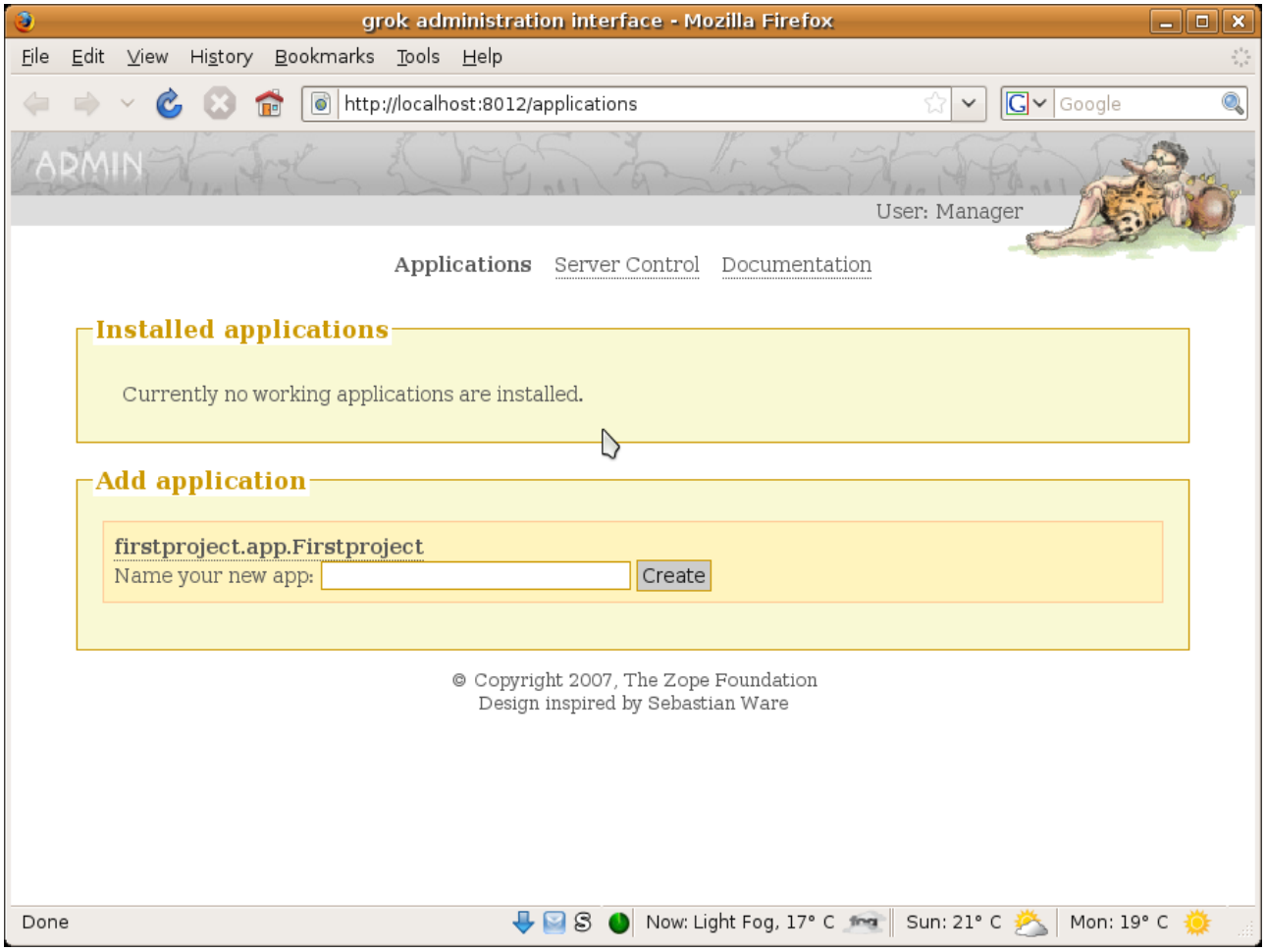
اعرض ال tree

```

striky@striky-desktop:~/workspace/pytut/src/nettut/groktut/FirstProject$ tree
.
|-- bin
|   |-- buildout
|   |-- i18nextract
|   |-- i18nmergeall
|   |-- i18nstats
|   |-- test
|   `-- zopectl
|-- bootstrap.py
|-- buildout.cfg
|-- develop-eggs
|   `-- FirstProject.egg-link
|-- parts
|   |-- app
|   |   |-- debugzope
|   |   |-- runzope
|   |   `-- site.zcml
|   |-- data
|   |-- i18n
|   |   `-- configure.zcml
|   |-- test
|   `-- zopectl
|       |-- zdaemon.conf
|       `-- zope.conf
|-- setup.py
|-- src
|   |-- FirstProject.egg-info
|   |   |-- PKG-INFO
|   |   |-- SOURCES.txt
|   |   |-- dependency_links.txt
|   |   |-- entry_points.txt
|   |   |-- not-zip-safe
|   |   |-- requires.txt
|   |   `-- top_level.txt
|   `-- firstproject
|       |-- __init__.py
|       |-- app.py
|       |-- app.txt
|       |-- app_templates
```

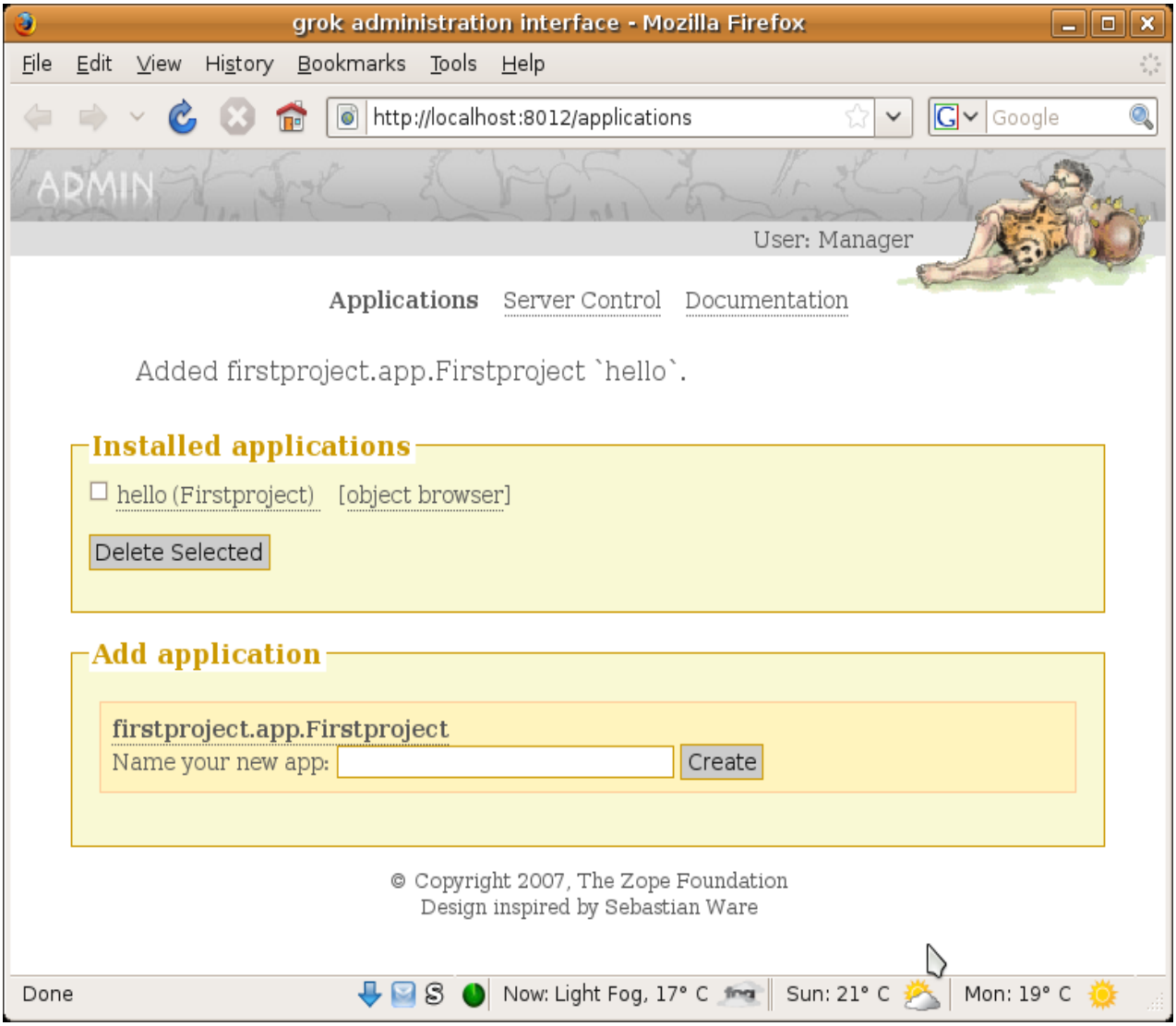
```
|       |   `-- index.pt
|       |-- configure.zcml
|       |-- ftesting.zcml
|       |-- static
|       |   `-- README.txt
|       `-- tests.py
`-- versions.cfg

13 directories, 32 files
```

الهيكلية واضحة ملفات تنفيذيه و templates وملفات static واعدادات الخ الخ

قم بتشغيل zope



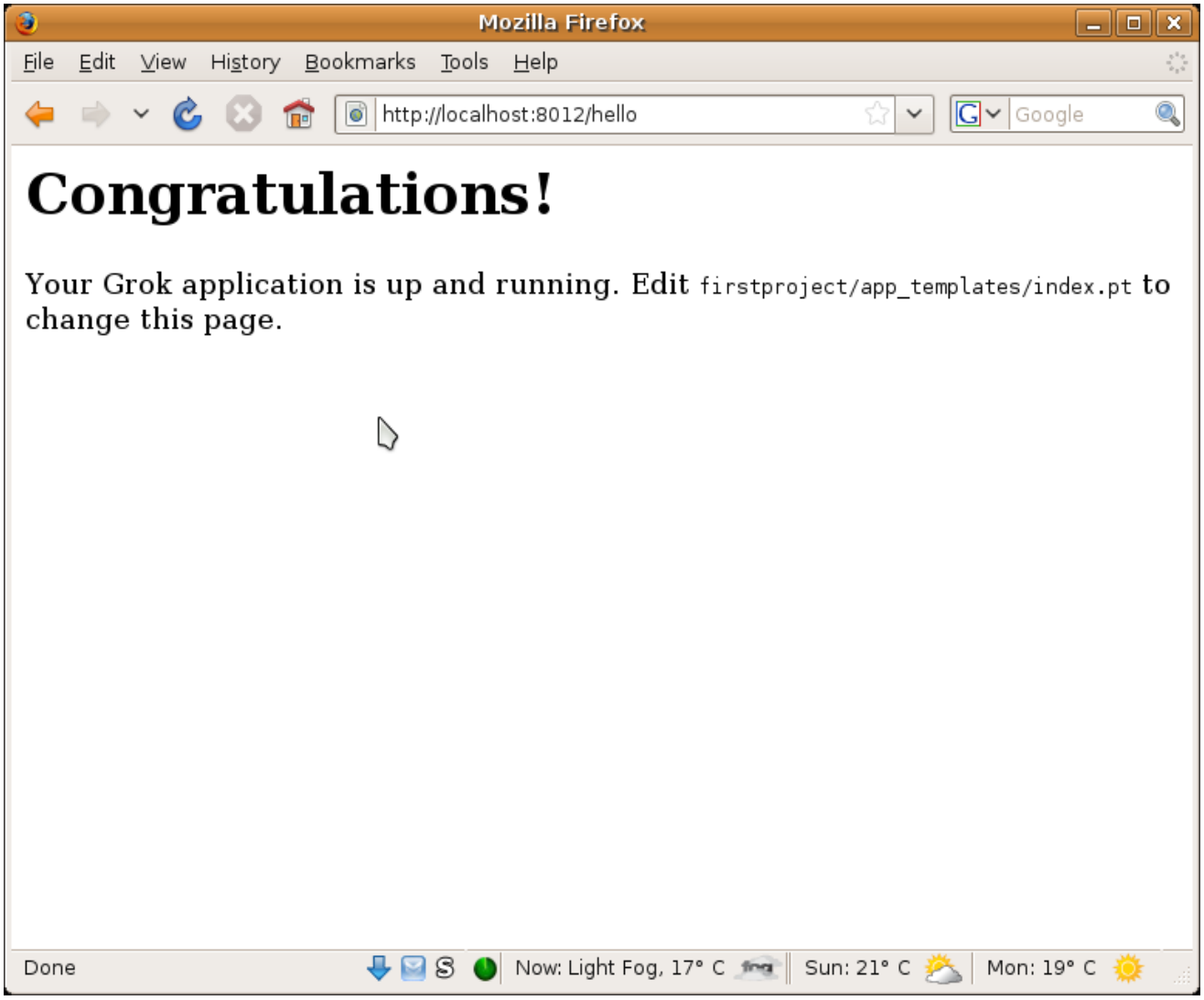

قم بتسمية التطبيق مثلا hello




اكتب فى المتصفح ذلك العنوان

localhost:8012/hello

ستجد امامك نافذة مشابهه لهذه تخبرك بمسار ال templates لتعديلها وهى فى المسار /firstprojct/app_templates/ index.pt




الهيكلية كالتالى

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut/FirstProject/src/firstproject$ tree
.
|-- __init__.py
|-- __init__.pyc
|-- app.py
|-- app.pyc
|-- app.txt
|-- app_templates
|   `-- index.pt
|-- configure.zcml
|-- ftesting.zcml
|-- static
|   `-- README.txt
`-- tests.py
```

2 directories, 10 files

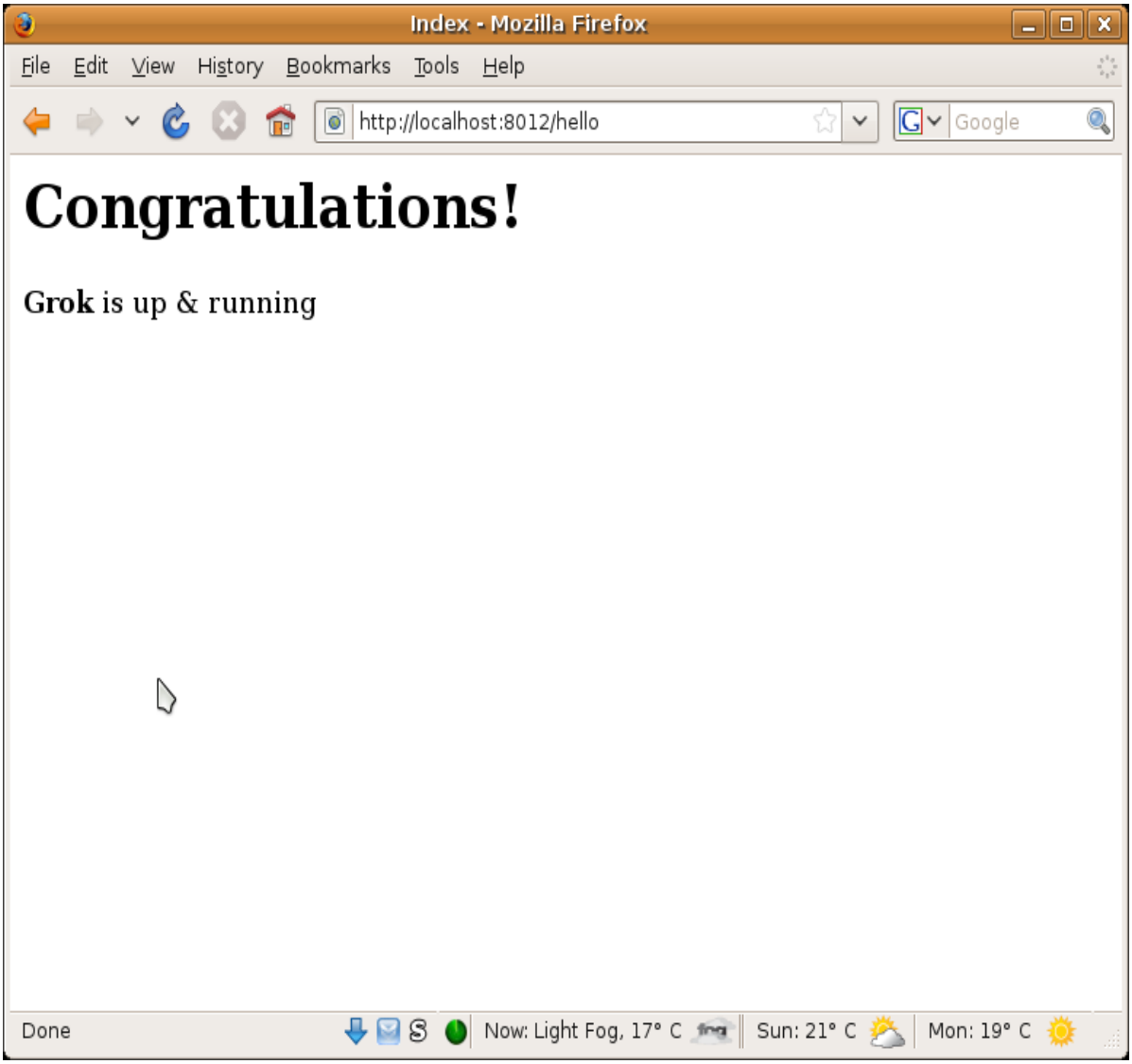
قم بفتح ذلك الملف index.pt فى ال app_templates ستجده مشابه للتالى

```
<html>
<head>
</head>
<body>
  <h1>Congratulations!</h1>

  <p>Your Grok application is up and running.
  Edit <code>firstproject/app_templates/index.pt</code> to change
  this page.</p>
</body>
</html>
```

قم بتعديله للتالى مثلا

```
<html>
        <head>
                <title>Index</title>
        </head>
        <body>
          <h1>Congratulations!</h1>

                <p>
                        <b>Grok</b> is up & running
                </p>
        </body>
</html>
```

وقم بعمل رفرش او اذهب لذلك العنوان localhost:8012/hello



الآن التطبيق لديه view واحدة وهى index التى يتم عند استدعاءها عرض ال template فى الملف index.pt
افتح الملف app.py ستجده مشابه للتالى

```
import grok

class Firstproject(grok.Application, grok.Container):
   pass

class Index(grok.View):
   pass # see app_templates/index.pt
```

قم بإضافة 2 views وهما Hi, Bye

```
class Hi(grok.View):
        pass #renders app_templates/hi.pt

class Bye(grok.View):
        pass #renders app_templates/bye.pt
```

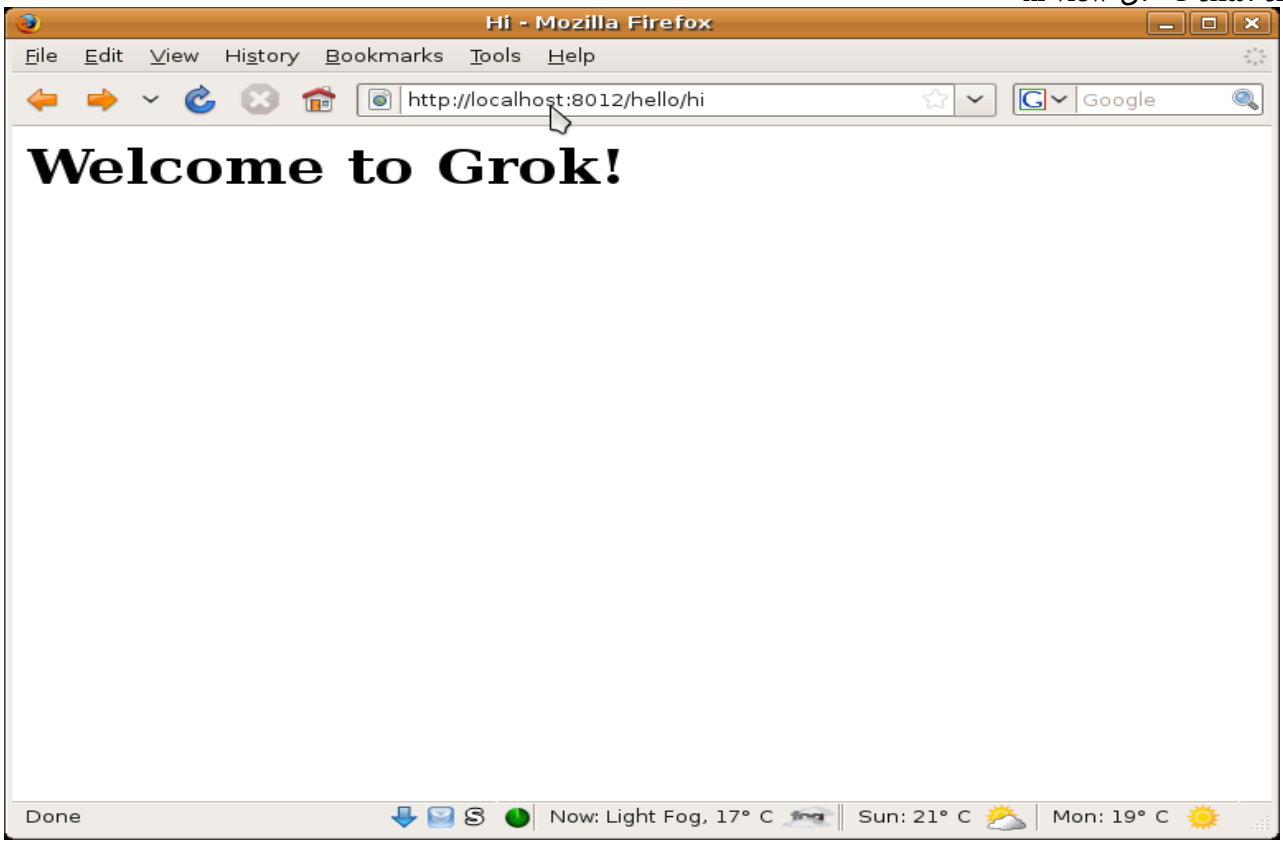
قم بإنشاء الملفات hi.pt و bye.pt ليتم عرضها عند استدعاء تلك ال views
الملف hi.pt

```
<html>
        <head>
                <title>Hi</title>
        </head>
        <body>
                <h1>Welcome to Grok!</h1>
        </body>
</html>
```

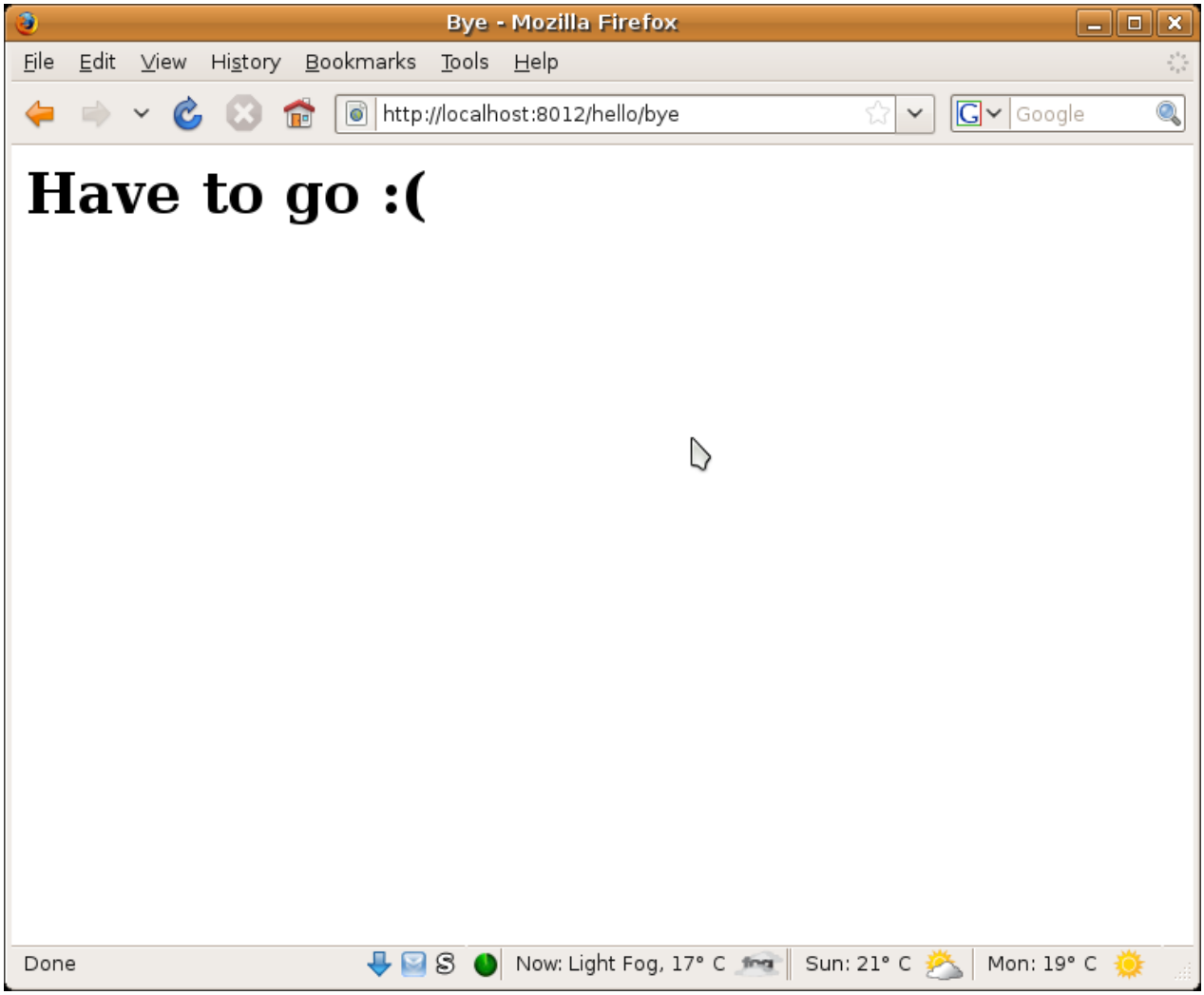
الملف bye.pt

```
<html>
        <head>
                <title>Bye</title>
        </head>
        <body>
                <h1>Have to go :(</h1>
        </body>
</html>
```

عند استدعاء ال hi view




عند استدعاء ال bye.view




ملحوظة: قد تريد احيانا تغيير رقم البورت الإفتراضى ل zope بدل من 8080 (اللذى قد تكون اسندته لخدمة اخرى او غيره)
كل ماعليك هو اعداد ملف buildout.cfg فى القسم [zopectl]
بإضافة ذلك السطر

```
address = localhost:8012
#controlling the listening port, re-run buildout script.
```

وقم بتشغيل ملف ال buildout

```
striky@striky-desktop:~/workspace/pytut/src/nettut/groktut/FirstProject$ bin/buildout
Develop: '/home/striky/workspace/pytut/src/nettut/groktut/FirstProject/.'
Uninstalling zopectl.
Updating eggbasket.
Updating app.
Updating data.
Installing zopectl.
Generated script '/home/striky/workspace/pytut/src/nettut/groktut/FirstProject/bin/zopectl'.
Updating i18n.
```

```
The recipe for i18n doesn't define an update method. Using its install method.
i18n: setting up i18n tools
Updating test.
```

الى اين الآن ؟
تستطيع الذهاب الى http://grok.zope.org وتقوم بقراءة الوثائق الخاصة وتنشئ بعض التطبيقات الحقيقية

## Webpy

فريمورك بسيطة وممتازة وغير معقدة لاتحتاج منك الكثير من المفاهيم ومبنية من اجل البساطة (ربما اذا اكملت الكتاب للآن تستطيع ان تفهم كودها المصدرى )

للتستيب

easyinstall web.py

او قم بتحميلها من الموقع الرسمى http://webpy.org
وقم بتشغيل سكربت ال setup

```
python setup.py install
```

ابسط تطبيق

```
import web

urls = (
    '/(.*)', 'index'
)


class index:
    def GET(self, name):
        return 'Hello, World!'

app = web.application(urls, globals())

if __name__ == "__main__":
    app.run()
```

1- استدعاء webpy

```
import web
```

2- انشاء ال urls على صورة tuple

```
'/(.*)', 'index'
```

على سبيل المثال هنا هيتم توجيه اى طلب الى ال view المسماة index اللتى قمنا بتعريفها كالتالى

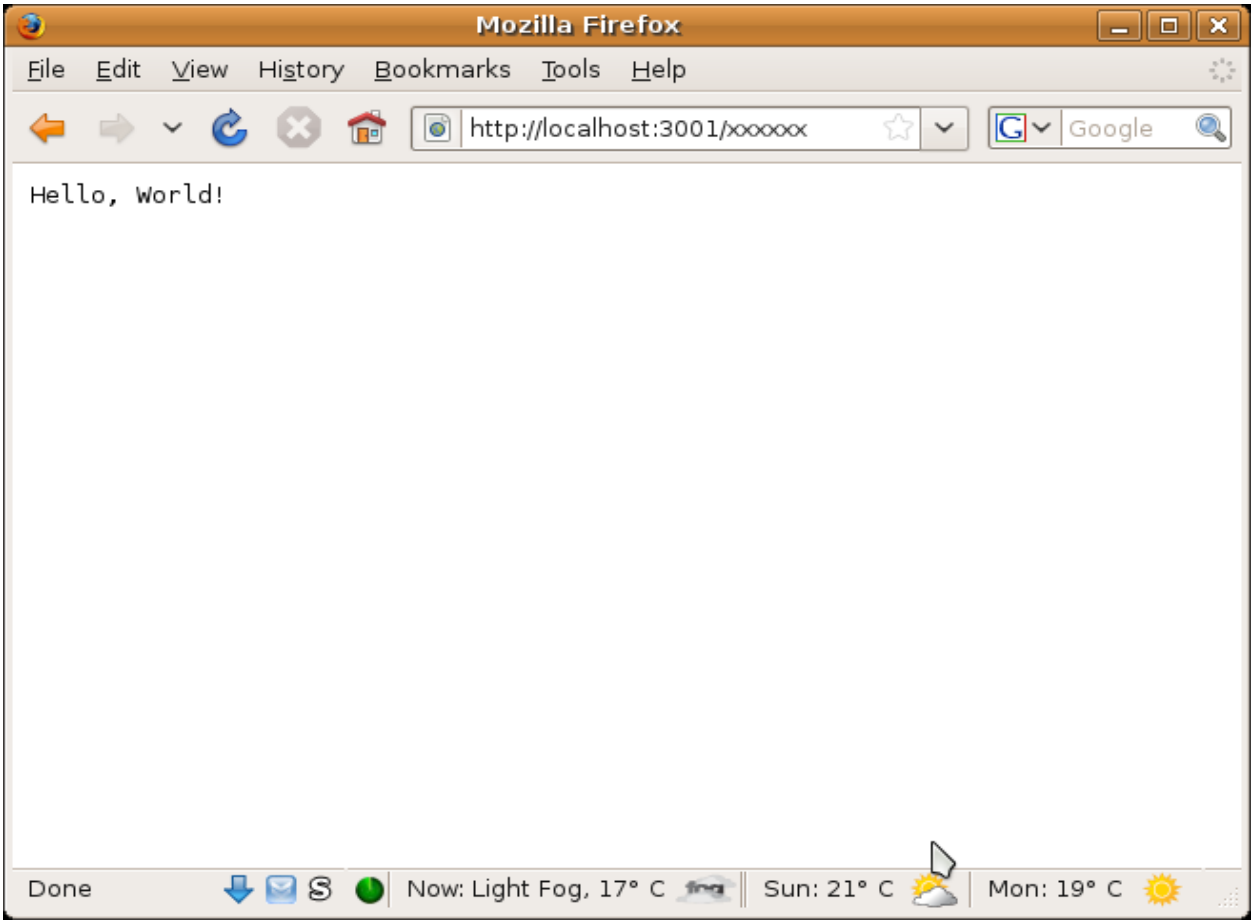
```
class index:
    def GET(self, name):
        return 'Hello, World!'
```

اخيرا انشاء عنصر التطبيق

```
app = web.application(urls, globals())
```

وتشغيله بإستخدام الطريقة run

```
    app.run()
```



لاحظ هنا تشغيلنا للسرفر على البورت 3001 وذلك بإضافة ذلك المعامل الى السكربت عند التشغيل والاسيكون البورت 8080

striky@striky-desktop:~/workspace/pytut/src/nettut/webpytut$ python hello.py 3001

تجد العديد من الأمثلة هنا
*http://webpy.org/src*

مثال على انشاء بلوج بسيط بإستخدام webpy
تجده هنا

http://k4ml.com/wiki/python/webpy/simpleblog

## The Big Three

هناك 3 اطارات تتصدر عالم بايثون فى الويب وهم Django و TurboGears و Pylons

Pylons هى اطار حديث نسبيا ويجمع افضل مافى العوالم python, ruby, perl واستفاد كثيرا من تجارب الإطارات السابقة

http://pylonshq.com/

**Hello World: Pylons**

```
striky@striky-desktop:~/workspace/pytut/src/nettut$ paster create -t pylons helloworld
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-i686.egg/dispatch/__init__.py:98:
Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/CherryPy-2.3.0-py2.5.egg/cherrypy/lib/profiler.py:54: UserWarning: Your
installation of Python doesn't have a profile module. If you're on Debian, you can apt-get python2.4-profiler
from non-free in a separate step. See http://www.cherrypy.org/wiki/ProfilingOnDebian for details.
  warnings.warn(msg)
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:239: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:263: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:281: Warning: 'as' will become a reserved keyword in Python 2.6
Selected and implied templates:
  Pylons#pylons  Pylons application template

Variables:
  egg:      helloworld
  package:  helloworld
  project:  helloworld
Enter template_engine (mako/genshi/jinja/etc: Template language) ['mako']:
Enter sqlalchemy (True/False: Include SQLAlchemy 0.4 configuration) [False]:
Enter google_app_engine (True/False: Setup default appropriate for Google App Engine) [False]:
Creating template pylons
Creating directory ./helloworld
  Recursing into +package+
    Creating ./helloworld/helloworld/
    Copying templates/default_project/+package+/__init__.py_tmpl to ./helloworld/helloworld/__init__.py
    Recursing into config
      Creating ./helloworld/helloworld/config/
      Copying templates/default_project/+package+/config/__init__.py_tmpl to
./helloworld/helloworld/config/__init__.py
      Copying templates/default_project/+package+/config/deployment.ini_tmpl_tmpl to
./helloworld/helloworld/config/deployment.ini_tmpl
      Copying templates/default_project/+package+/config/environment.py_tmpl to
./helloworld/helloworld/config/environment.py
      Copying templates/default_project/+package+/config/middleware.py_tmpl to
./helloworld/helloworld/config/middleware.py
      Copying templates/default_project/+package+/config/routing.py_tmpl to
./helloworld/helloworld/config/routing.py
    Recursing into controllers
```

```
    Creating ./helloworld/helloworld/controllers/
    Copying templates/default_project/+package+/controllers/__init__.py_tmpl to
./helloworld/helloworld/controllers/__init__.py
    Copying templates/default_project/+package+/controllers/error.py_tmpl to
./helloworld/helloworld/controllers/error.py
  Recursing into lib
    Creating ./helloworld/helloworld/lib/
    Copying templates/default_project/+package+/lib/__init__.py_tmpl to
./helloworld/helloworld/lib/__init__.py
    Copying templates/default_project/+package+/lib/app_globals.py_tmpl to
./helloworld/helloworld/lib/app_globals.py
    Copying templates/default_project/+package+/lib/base.py_tmpl to ./helloworld/helloworld/lib/base.py
    Copying templates/default_project/+package+/lib/helpers.py_tmpl to ./helloworld/helloworld/lib/helpers.py
  Recursing into model
    Creating ./helloworld/helloworld/model/
    Copying templates/default_project/+package+/model/__init__.py_tmpl to
./helloworld/helloworld/model/__init__.py
  Recursing into public
    Creating ./helloworld/helloworld/public/
    Copying templates/default_project/+package+/public/bg.png to ./helloworld/helloworld/public/bg.png
    Copying templates/default_project/+package+/public/index.html_tmpl to
./helloworld/helloworld/public/index.html
    Copying templates/default_project/+package+/public/pylons-logo.gif to
./helloworld/helloworld/public/pylons-logo.gif
  Recursing into templates
    Creating ./helloworld/helloworld/templates/
  Recursing into tests
    Creating ./helloworld/helloworld/tests/
    Copying templates/default_project/+package+/tests/__init__.py_tmpl to
./helloworld/helloworld/tests/__init__.py
    Recursing into functional
      Creating ./helloworld/helloworld/tests/functional/
      Copying templates/default_project/+package+/tests/functional/__init__.py_tmpl to
./helloworld/helloworld/tests/functional/__init__.py
    Copying templates/default_project/+package+/tests/test_models.py_tmpl to
./helloworld/helloworld/tests/test_models.py
  Copying templates/default_project/+package+/websetup.py_tmpl to ./helloworld/helloworld/websetup.py
 Copying templates/default_project/MANIFEST.in_tmpl to ./helloworld/MANIFEST.in
 Copying templates/default_project/README.txt_tmpl to ./helloworld/README.txt
 Copying templates/default_project/development.ini_tmpl to ./helloworld/development.ini
 Recursing into docs
  Creating ./helloworld/docs/
  Copying templates/default_project/docs/index.txt_tmpl to ./helloworld/docs/index.txt
 Copying templates/default_project/ez_setup.py to ./helloworld/ez_setup.py
 Copying templates/default_project/setup.cfg_tmpl to ./helloworld/setup.cfg
 Copying templates/default_project/setup.py_tmpl to ./helloworld/setup.py
 Copying templates/default_project/test.ini_tmpl to ./helloworld/test.ini
Running /usr/bin/python setup.py egg_info
```

سيطلب منك 3 اجابات
1- ال template engine والإفتراضى هو mako

2- دعم sqlalchemy فى التطبيق والإفتراضى لا
3- تجهيز خيارات ل Google App Engine

الهيكلية

```
|-- MANIFEST.in
|-- README.txt
|-- development.ini
|-- docs
|   `-- index.txt
|-- ez_setup.py
|-- helloworld
|   |-- __init__.py
|   |-- config
|   |   |-- __init__.py
|   |   |-- deployment.ini_tmpl
|   |   |-- environment.py
|   |   |-- middleware.py
|   |   `-- routing.py
|   |-- controllers
|   |   |-- __init__.py
|   |   `-- error.py
|   |-- lib
|   |   |-- __init__.py
|   |   |-- app_globals.py
|   |   |-- base.py
|   |   `-- helpers.py
|   |-- model
|   |   `-- __init__.py
|   |-- public
|   |   |-- bg.png
|   |   |-- index.html
|   |   `-- pylons-logo.gif
|   |-- templates
|   |-- tests
|   |   |-- __init__.py
|   |   |-- functional
|   |   |   `-- __init__.py
|   |   `-- test_models.py
|   `-- websetup.py
|-- helloworld.egg-info
|   |-- PKG-INFO
|   |-- SOURCES.txt
|   |-- dependency_links.txt
|   |-- entry_points.txt
|   |-- not-zip-safe
|   |-- paster_plugins.txt
|   |-- requires.txt
|   `-- top_level.txt
|-- setup.cfg
|-- setup.py
`-- test.ini
```

ننشئ controller (وهو مايحوى ال actions مثل ال views اللتى فى grok او webpy)

```
striky@striky-desktop:~/workspace/pytut/src/nettut/helloworld$ paster controller hello
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-i686.egg/dispatch/__init__.py:98:
Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/CherryPy-2.3.0-py2.5.egg/cherrypy/lib/profiler.py:54: UserWarning: Your
installation of Python doesn't have a profile module. If you're on Debian, you can apt-get python2.4-profiler
from non-free in a separate step. See http://www.cherrypy.org/wiki/ProfilingOnDebian for details.
  warnings.warn(msg)
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:239: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:263: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:281: Warning: 'as' will become a reserved keyword in Python 2.6
Creating /home/striky/workspace/pytut/src/nettut/helloworld/helloworld/controllers/hello.py
Creating /home/striky/workspace/pytut/src/nettut/helloworld/helloworld/tests/functional/test_hello.py
striky@striky-desktop:~/workspace/pytut/src/nettut/helloworld$
```

تمام افتح الملف فى المسار helloworld/controllers/hello ستجده مشابه للتالى

```
import logging

from pylons import request, response, session, tmpl_context as c
from pylons.controllers.util import abort, redirect_to

from helloworld.lib.base import BaseController, render
#from helloworld import model

log = logging.getLogger(__name__)

class HelloController(BaseController):

    def index(self):
        # Return a rendered template
        #   return render('/template.mako')
        # or, Return a response
        return 'Hello World'
```

يتم التعامل مع العناوين كالتالى

mysite.com/controller/view

مثلا عن ارسال

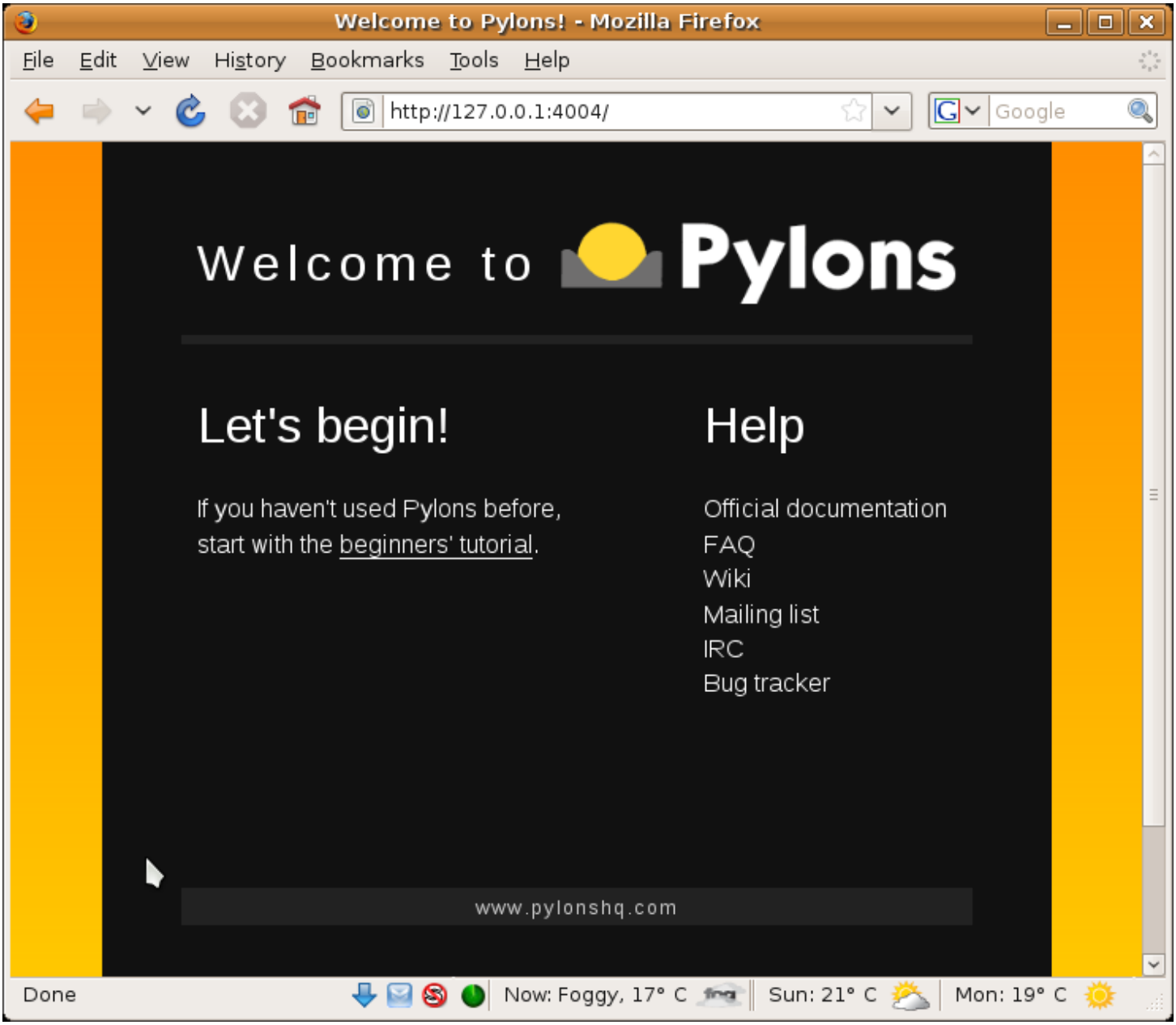
mysite.com/hello/index

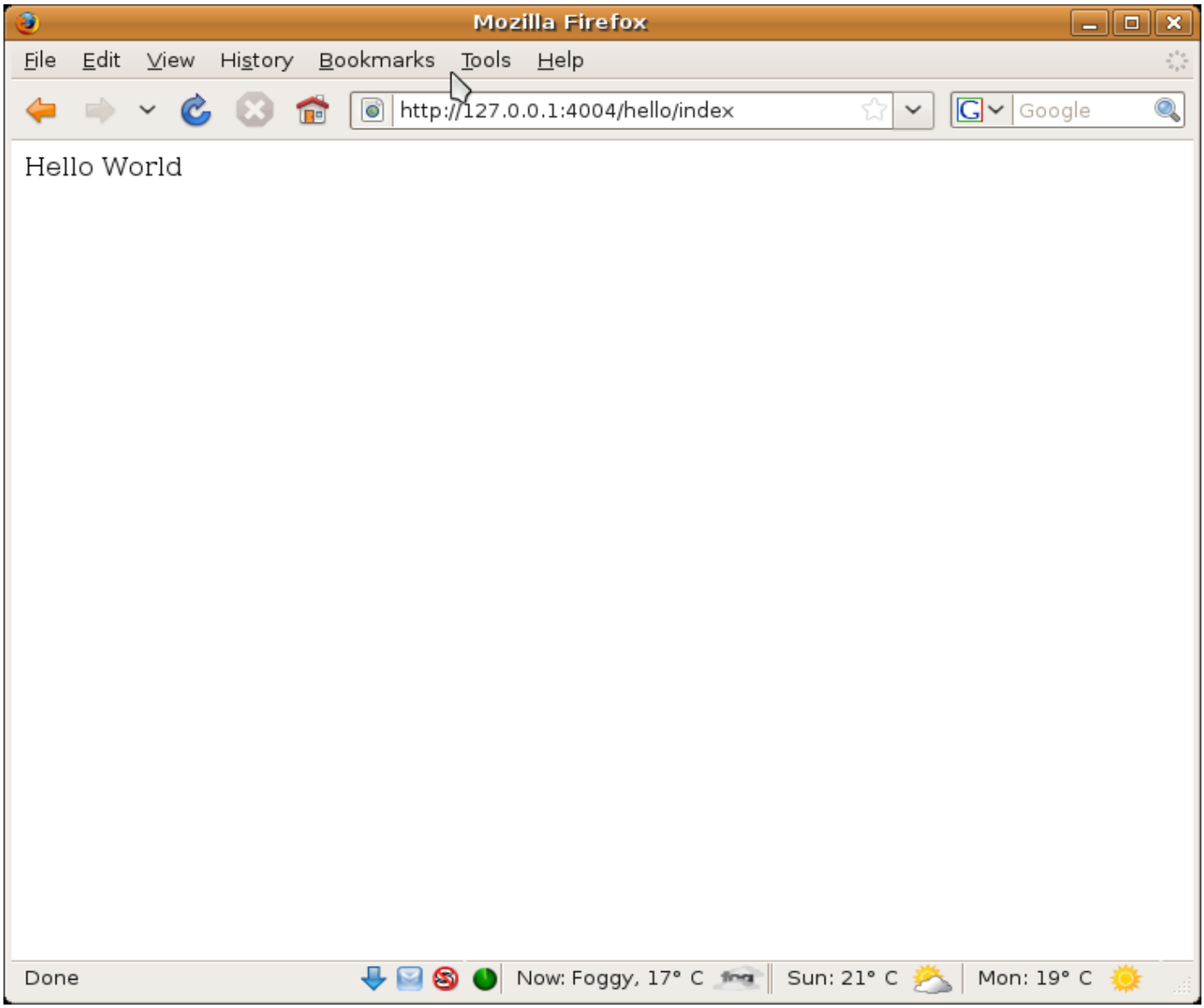
فيتم استدعاء الكنترولر (المقسم) hello ومنه يتم اختيار ال action المناسبة
قم بتشغيل السرفر

```
striky@striky-desktop:~/workspace/pytut/src/nettut/helloworld$ paster serve --reload development.ini
```

ملف ال development.ini يحوى معلومات عن البيئة كالهوست والبورت  ومتغيرات التطبيق الخ الخ

الآن اکتب hello/index فی العنوان




Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1:4004/hello/index
Google
Hello World
Done
Now: Foggy, 17° C
Sun: 21° C
Mon: 19° C

## TurboGears

تربوجيرز هى اطار عمل رائع يقوم على ربط التقنيات الحالية فى عالم بايثون للخروج بأفضل نتيجة فللتعامل مع قواعد البيانات يتم استخدام SQLAlchemy او SQLObject وللتعامل مع ال templates يتم استخدام kid وهكذا

لتستيب تربوجيرز قم اولا بتحميل الحزمة من الموقع http://turbogears.org/
راجع صفحة التستيب http://docs.turbogears.org/1.0/Install

قم بتحميل سكربت tgsetup.py وتشغيله
http://www.turbogears.org/download/tgsetup.py

نفذ سكربت التستيب

```
striky@striky-desktop:~/Desktop$ sudo python tgsetup.py

[sudo] password for striky:
Sorry, try again.
[sudo] password for striky:
TurboGears Installer
Beginning setuptools/EasyInstall installation and TurboGears download

Searching for TurboGears==1.0.8
Reading http://www.turbogears.org/download/
Reading http://pypi.python.org/simple/TurboGears/
Reading http://www.turbogears.org
Reading http://www.turbogears.org/
Reading http://www.turbogears.org/download/filelist.html
Best match: TurboGears 1.0.8
Downloading http://files.turbogears.org/eggs/TurboGears-1.0.8-py2.5.egg
Processing TurboGears-1.0.8-py2.5.egg
removing '/usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg' (and everything under it)
creating /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg
Extracting TurboGears-1.0.8-py2.5.egg to /usr/lib/python2.5/site-packages
Removing TurboGears 1.0.7 from easy-install.pth file
Adding TurboGears 1.0.8 to easy-install.pth file
Installing tg-admin script to /usr/bin

Installed /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg
Reading http://files.turbogears.org/eggs/
Processing dependencies for TurboGears==1.0.8
Searching for Extremes>=1.1
Reading http://pypi.python.org/simple/Extremes/
Best match: Extremes 1.1
Downloading http://pypi.python.org/packages/2.5/E/Extremes/Extremes-1.1-
py2.5.egg#md5=4015e2546295858558cca16faca5f34f
Processing Extremes-1.1-py2.5.egg
Moving Extremes-1.1-py2.5.egg to /usr/lib/python2.5/site-packages
Adding Extremes 1.1 to easy-install.pth file
```

```
Installed /usr/lib/python2.5/site-packages/Extremes-1.1-py2.5.egg
Searching for PyProtocols>=1.0a0dev-r2302
Reading http://pypi.python.org/simple/PyProtocols/
Reading http://peak.telecommunity.com/PyProtocols.html
Reading http://peak.telecommunity.com/dist/
Best match: PyProtocols 1.0a0dev-r2302
Downloading http://files.turbogears.org/eggs/PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg
Processing PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg
Moving PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg to /usr/lib/python2.5/site-packages
Adding PyProtocols 1.0a0dev-r2302 to easy-install.pth file

Installed /usr/lib/python2.5/site-packages/PyProtocols-1.0a0dev_r2302-py2.5-linux-i686.egg
Finished processing dependencies for TurboGears==1.0.8
```

***Hello World: TG***

لإنشاء تطبيق سريع كل ماعليك هو تنفيذ tg-admin quickstart

```
striky@striky-desktop:~/workspace/pytut/src/nettut$ tg-admin  quickstart
/usr/lib/python2.5/site-packages/CherryPy-2.3.0-py2.5.egg/cherrypy/lib/profiler.py:54: UserWarning: Your
installation of Python doesn't have a profile module. If you're on Debian, you can apt-get python2.4-profiler
from non-free in a separate step. See http://www.cherrypy.org/wiki/ProfilingOnDebian for details.
  warnings.warn(msg)
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-i686.egg/dispatch/__init__.py:98:
Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:239: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:263: Warning: 'as' will become a reserved keyword in Python 2.6
/usr/lib/python2.5/site-packages/RuleDispatch-0.5a1.dev_r2506-py2.5-linux-
i686.egg/dispatch/predicates.py:281: Warning: 'as' will become a reserved keyword in Python 2.6
Enter project name: hello
Enter package name [hello]: hello
Do you need Identity (usernames/passwords) in this project? [no]
Selected and implied templates:
  TurboGears#tgbase      tg base template
  TurboGears#turbogears  web framework

Variables:
  egg:             hello
  elixir:          False
  identity:        none
  package:         hello
  project:         hello
  sqlalchemy:      False
  sqlobject:       True
  sqlobjectversion:  SQLObject>=0.10.1
Creating template tgbase
```

```
Creating directory ./hello
  Recursing into +einame+.egg-info
    Creating ./hello/hello.egg-info/
    Copying PKG-INFO to ./hello/hello.egg-info/PKG-INFO
    Copying paster_plugins.txt to ./hello/hello.egg-info/paster_plugins.txt
    Copying sqlobject.txt_tmpl to ./hello/hello.egg-info/sqlobject.txt
  Recursing into +package+
    Creating ./hello/hello/
    Copying __init__.py to ./hello/hello/__init__.py
    Copying release.py_tmpl to ./hello/hello/release.py
    Recursing into static
      Creating ./hello/hello/static/
      Recursing into css
        Creating ./hello/hello/static/css/
Skipping file /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg/turbogears/qstemplates/qsbase/
+package+/static/css/empty_tmpl
      Recursing into images
        Creating ./hello/hello/static/images/
        Copying favicon.ico to ./hello/hello/static/images/favicon.ico
        Copying tg_under_the_hood.png to ./hello/hello/static/images/tg_under_the_hood.png
        Copying under_the_hood_blue.png to ./hello/hello/static/images/under_the_hood_blue.png
      Recursing into javascript
        Creating ./hello/hello/static/javascript/
Skipping file /usr/lib/python2.5/site-packages/TurboGears-1.0.8-py2.5.egg/turbogears/qstemplates/qsbase/
+package+/static/javascript/empty_tmpl
    Recursing into templates
      Creating ./hello/hello/templates/
      Copying __init__.py to ./hello/hello/templates/__init__.py
Creating template turbogears
  Recursing into +package+
    Copying commands.py_tmpl to ./hello/hello/commands.py
    Recursing into config
      Creating ./hello/hello/config/
      Copying __init__.py to ./hello/hello/config/__init__.py
      Copying app.cfg_tmpl to ./hello/hello/config/app.cfg
      Copying log.cfg_tmpl to ./hello/hello/config/log.cfg
    Copying controllers.py_tmpl to ./hello/hello/controllers.py
    Copying json.py_tmpl to ./hello/hello/json.py
    Copying model.py_tmpl to ./hello/hello/model.py
    Recursing into static
      Recursing into css
        Copying style.css to ./hello/hello/static/css/style.css
      Recursing into images
        Copying header_inner.png to ./hello/hello/static/images/header_inner.png
        Copying info.png to ./hello/hello/static/images/info.png
        Copying ok.png to ./hello/hello/static/images/ok.png
    Recursing into templates
      Copying login.kid to ./hello/hello/templates/login.kid
      Copying master.kid to ./hello/hello/templates/master.kid
      Copying welcome.kid to ./hello/hello/templates/welcome.kid
    Recursing into tests
      Creating ./hello/hello/tests/
      Copying __init__.py to ./hello/hello/tests/__init__.py
```

```
    Copying test_controllers.py_tmpl to ./hello/hello/tests/test_controllers.py
    Copying test_model.py_tmpl to ./hello/hello/tests/test_model.py
  Copying README.txt_tmpl to ./hello/README.txt
  Copying dev.cfg_tmpl to ./hello/dev.cfg
  Copying sample-prod.cfg_tmpl to ./hello/sample-prod.cfg
  Copying setup.py_tmpl to ./hello/setup.py
  Copying start-+package+.py_tmpl to ./hello/start-hello.py
  Copying test.cfg_tmpl to ./hello/test.cfg
Running /usr/bin/python setup.py egg_info
Manually creating paster_plugins.txt (deprecated! pass a paster_plugins keyword to setup() instead)
Adding TurboGears to paster_plugins.txt
running egg_info
paster_plugins not set in setup(), but hello.egg-info/paster_plugins.txt exists
writing requirements to hello.egg-info/requires.txt
writing hello.egg-info/PKG-INFO
writing top-level names to hello.egg-info/top_level.txt
writing dependency_links to hello.egg-info/dependency_links.txt
writing entry points to hello.egg-info/entry_points.txt
reading manifest file 'hello.egg-info/SOURCES.txt'
writing manifest file 'hello.egg-info/SOURCES.txt'
striky@striky-desktop:~/workspace/pytut/src/nettut$
```

لإختيار البورت اللذى تريد الإنصات عليه قم بتحرير ملف dev.cfg حيث يشمل اعدادات التطبيق

```
server.socket_port=40003
```

مسار قاعدة بيانات sqlite

```
sqlobject.dburi="sqlite://%(current_dir_uri)s/devdata.sqlite"
```

اذا قمت بعمل اى جداول .. الخ
قم بتشغيل التطبيق بإستخدام ال start script وهنا ستجد اسمه start-hello.py
افتح متصفحك وحدد العنوان localhost:40003 او غيره اذا قمت بإعداد البورت  ستشاهد صفحة مثل هذه


بنفس فلسفة pylons ستجد المتحكمات (controllers) فى ملف controllers.py

```
#controllers.py
import turbogears as tg
from turbogears import controllers, expose, flash
# from hello import model
# import logging
# log = logging.getLogger("hello.controllers")

class Root(controllers.RootController):
   @expose(template="hello.templates.welcome")
   def index(self):
      import time
      # log.debug("Happy TurboGears Controller Responding For Duty")
      flash("Your application is now running")
      return dict(now=time.ctime())
```

وهذا هو ال controller الرئيسى وتم كشفه لل template فى المسار hello/templates/welcome.kid

ملحوظة لكشف اى action ل template ما استخدم @expose وضيف ليها معامل template وقيمته = مسار ال template المطلوب

ملف welcome.kid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:py="http://purl.org/kid/ns#"
   py:extends="'master.kid'">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" py:replace="''"/>
<title>Welcome to TurboGears</title>
</head>
<body>

  <div id="sidebar">
    <h2>Learn more</h2>
    Learn more about TurboGears and take part in its
    development
    <ul class="links">
      <li><a href="http://www.turbogears.org">Official website</a></li>
      <li><a href="http://docs.turbogears.org">Documentation</a></li>
      <li><a href="http://trac.turbogears.org/turbogears/">Trac
        (bugs/suggestions)</a></li>
      <li><a href="http://groups.google.com/group/turbogears"> Mailing list</a> </li>
    </ul>
    <span py:replace="now">now</span>
  </div>
  <div id="getting_started">
    <ol id="getting_started_steps">
      <li class="getting_started">
        <h3>Model</h3>
        <p> <a href="http://docs.turbogears.org/1.0/GettingStarted/DefineDatabase">Design models</a> in the
<span class="code">model.py</span>.<br/>
         Edit <span class="code">dev.cfg</span> to <a
href="http://docs.turbogears.org/1.0/GettingStarted/UseDatabase">use a different backend</a>, or start with a
pre-configured SQLite database. <br/>
         Use script <span class="code">tg-admin sql create</span> to create the database tables.</p>
      </li>
      <li class="getting_started">
        <h3>View</h3>
        <p> Edit <a href="http://docs.turbogears.org/1.0/GettingStarted/Kid">html-like templates</a> in the
<span class="code">/templates</span> folder;<br/>
       Put all <a href="http://docs.turbogears.org/1.0/StaticFiles">static contents</a> in the <span
class="code">/static</span> folder. </p>
      </li>
      <li class="getting_started">
        <h3>Controller</h3>
        <p> Edit <span class="code"> controllers.py</span> and <a
href="http://docs.turbogears.org/1.0/GettingStarted/CherryPy">build your
         website structure</a> with the simplicity of Python objects. <br/>
         TurboGears will automatically reload itself when you modify your project. </p>
      </li>
```

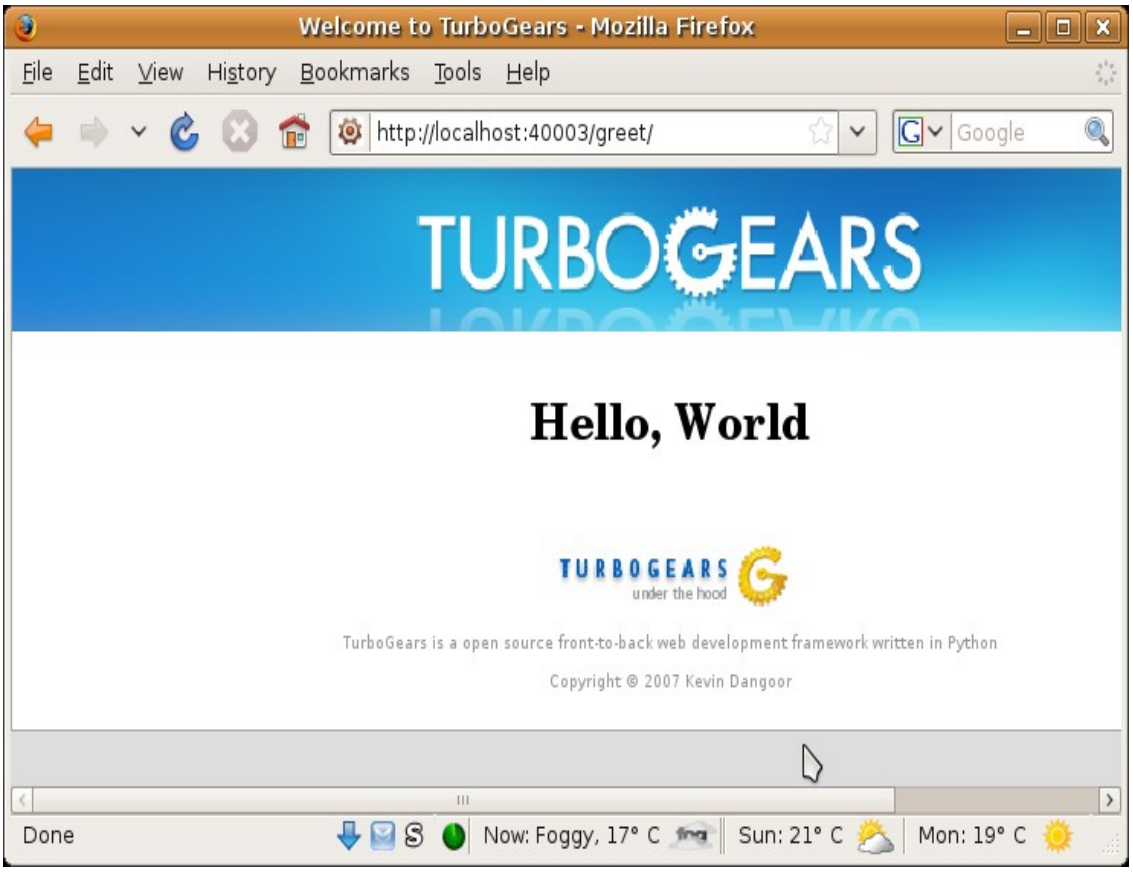
```
    </ol>
    <div class="notice"> If you create something cool, please <a
href="http://groups.google.com/group/turbogears">let people know</a>, and consider contributing something
back to the <a href="http://groups.google.com/group/turbogears">community</a>.</div>
  </div>
  <!-- End of getting_started -->
</body>
</html>
```

تعالى نجرب اضافة action جديد وليكن greet
بكل بساطة ضيفه فى ال controllers.py كطريقة لل RootController

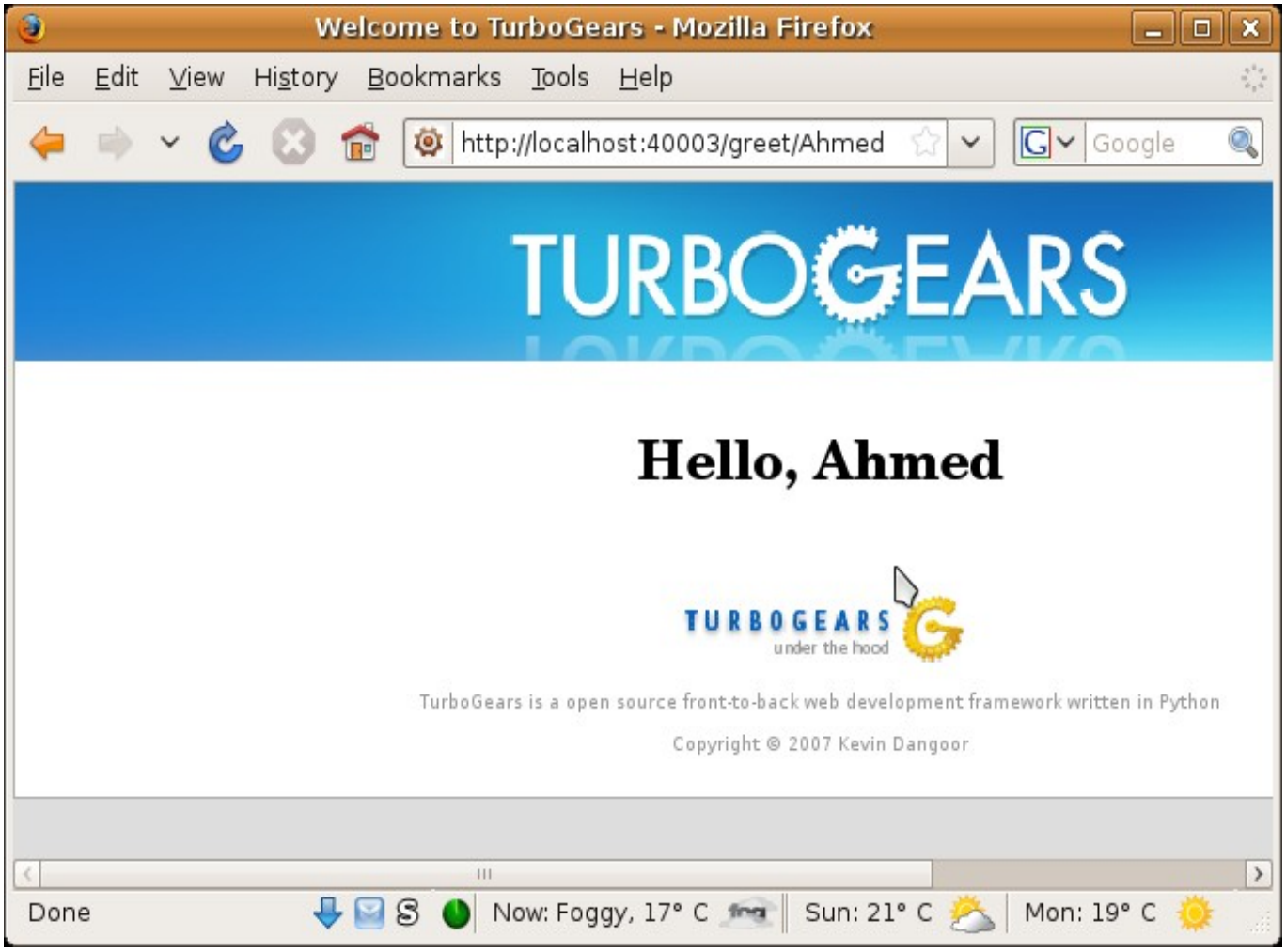
```
    @expose(template='hello.templates.greet')
    def greet(self, who='World'):
            return dict(g='Hello, '+who)
```

ماهذا ؟ ايه معنى who ؟
بكل بساطة هى معامل يتم استدعائه بعد اسم ال action فى العنوان مثلا
localhost:40003/hello/Ahmed
فتصبح قيمة who هى Ahmed وفى حال عدم تحديدها تكون قيمتها World
نعيد dict من تلك ال action يشمل المتغيرات اللتى ستصبح مكشوفة فى ال template ليتم استخدامها فيه

عند الإستدعاء بدون معاملات



بإستخدام معاملات




يوجد اطارات عمل اكثر من رائعة مثل Django التى لم نتعرض لها فى الكتاب وايضا Web2Py وغيرها
تجد قائمة بأهم اطر العمل هنا
http://wiki.python.org/moin/WebFrameworks

# *Chapter 14 (Extending Python)*

ماهى ال Extensions ؟
هى امتدادات لل Python مكتوبة بال C او C-Like مثل ال C++ على سبيل المثال

سهل كتابة extensions لل Python ولكن مالهدف ؟
الهدف إنك تضيف built-in modules لل Python مكتوبة بال C بهدف السرعة مثلا او إضافة built-in types او عمل encapsulation لل C lib functions او System Calls او حتى إخفاء ال source code الخاص بيك :)

المتطلبات: خبرة جيدة بال C و Python
اول شئ بكل تأكيد هو إننا هنستخدم Python API/C وذلك بضم python.h لل project

ملحوظة: قم بضم python.h قبل اى header اخر. جميل نبدأ ب Hello, World :)
1- انشئ ملف helloMod.c
2- اكتب التالى

```
#include <Python.h>

static PyObject* hola(PyObject* self, PyObject* args)
{


   if (!PyArg_ParseTuple(args, "", NULL))
      return NULL;

   printf("Hola!");

   Py_RETURN_NONE;
}

static PyMethodDef HolaMethods[] =
{
    {"hola", hola, METH_VARARGS, "prints Hola\n"},
    {NULL, NULL, 0, NULL}
};

PyMODINIT_FUNC

inithola(void)
{
    (void) Py_InitModule("hola", HolaMethods);
}
```

نبدأ ب module بسيطة وهى hola هنشرح سطر سطر
اولا نعمل include ل python api header كالتالى

```
#include <Python.h>
```

اى Object فى Python تقدر تعرفه ك Pointer ل ب PyObject
احنا الأول نريد ان نعرف function بسيطة تطبع كلمة Hola!

```
static PyObject* hola(PyObject* self, PyObject* args)
```

لاحظ self هو Pointer فى حال لو ال PyObject Class ودى هتكون ال method تبعه. لكن لو Function يبقة self هيكون NULL . نريد ان يتم إستخدام ال Function كالتالى

```
>>> hola.hola()
Hola!
```

لاحظ شئ إن ال Function مش هتاخد اى argument و مش ليها return او ال Return ب NONE

فالأول نختبر هل فى arguments تم تمرريها لل Function او لأ

```
if (!PyArg_ParseTuple(args, "", NULL))
    return NULL;
```

PyArg_ParseTuple هى Function بتسخدم فى عمل Parse او تحليل لل Arguments وفيها بيتم تحويل ال Python Values ل C Values زى ماهنشوف فى مثال قادم.
args هى ال arguments اللى هتتمرر لل Function
هى بتعبر عن ال Data Type الخاص بال Argument مثلا s او i وهكذا

s: String
i: Integer
.. etc

NULL هنا بيعبر عن ال متغيرات اللتى هتاخد القيم اللتى تم تمرريها لل args -هنطلع عليها اكثر فى مثال قادم-

```
 return NULL;
```

للخروج مباشرة من تنفيذ ال Function
بعد كدا نيجى لل ال Function هتعمله وهو طباعة كلمة Hola بإستخدام printf

```
printf("Hola!");
```

واخيرا زى ماقلنا اننا مش نريد ان اى return من ال Function فهنعمل Py_RETURN_NONE

```
    Py_RETURN_NONE;
```

هنحتاج نعرف ال methodTable وهو عبارة عن array بتشمل معلومات عن ال Function زى ال name, address وال Documentation الخاصة بيها وهكذا

```
static PyMethodDef HolaMethods[] =
{
    {"hola", hola, METH_VARARGS, "prints Hola"},
    {NULL, NULL, 0, NULL}
};
```

اول field هو ال name الخاص بال function
التانى هو ال function نفسها
التالت بيعبر عن ان ال Arguments اللتى هتتمرر هى Python-Level Arguments وغالبا بنستخدم METH_VARAGS
الرابع هو الوصف الخاص بال function
ونحجز المكان التانى فى ال Array ب

```
{NULL, NULL, 0, NULL}
```

نعمل Initialize لل Module بتاعتنا كالتالى

```
PyMODINIT_FUNC
inithola(void)
{
    (void) Py_InitModule("hola", HolaMethods);
}
```

PyMODINIT_FUNC هى إختصار ل Python Module Initializer Function و فيها بيتم تجهيز ال Module
بإستدعاء Py_InitModule وهى Function وهى اللتى بتقوم بالتجهيز بالفعل وبتاخد 2Arguments
1- اسم ال Module
2- ال Methods Table

جميل جدا.. كدا كتبنا اول Module خاصة بينا!
هنحتاج نضم ال Extension لل Python ولكن إزاى ؟

بكل بساطة افتح ال Editor المفضل عندك وهنعمل setup script بال Python بإستخدام Distutils

```
from distutils.core import setup, Extension

modExt = Extension('hola', sources = ['hola.c'])
```

```
setup (name = 'HolaPackage',
      version = '1.0',
      description = 'Simple demo',
      ext_modules = [modExt])
```

الخطوات سهلة وسلسة كالتالى:
1- إستدعينا setup, Extension من Disutils.core
2- عملنا Extension Object كالتالى

```
modExt = Extension('hola', sources = ['hola.c'])
```

وفيه بنحدد ال source واسم ال extension
3- تجهيز ال setup script بإننا نسجل فيه إسم ال Package و الإصدار والوصف و ال extensions كالتالى

```
setup (name = 'HolaPackage',
      version = '1.0',
      description = 'Simple demo',
      ext_modules = [modExt])
```

كل ماعليك هو

Python setup.py build

وبعد كدا تعمل Install لل Package كالتالى

Python setup.py install

نجرب ال Hola Module كالتالى
1- اعمل اى Test Script وليكن HolaTest.py
2- اعمل import ل hola Module كالتالى

```
import hola
```

3- استدعى ال hola function كالتالى

```
hola.hola()
#output:
Hola!
```

4- لنوضيح ال return الخاص بال Function اكتب

```
print hola.hola()
#Output:
Hola!
None
```

بعد ماطلعنا على الأساسيات نجرب نكتب module فيها function بتقبل argument ك name و age وتطبعهم و Function اخرى لحساب القيمة المطلقة لرقم وواحدة تقسم عددين وواحدة تعمل return ب Tuple, Dictionary
الفكرة بإختصار:
1- نعرف ال Functions
2- نضمهم لل Methods Table
3- نعمل Initialize لل Module
4- نجهز ال setup script
5- نعمل Build و Install لل Module
6- نستخدم ال Module عن طريق TestScript مثلا كالتالى

1- تعريف ال Functions

ال Hola Function

```
static PyObject* hola(PyObject* self, PyObject* args)
{

   const char* name;
   int age;

   if (!PyArg_ParseTuple(args, "si", &name, &age))
      return NULL;

   printf("Name: %s", name);
   printf("Age: %i", age);


   Py_RETURN_NONE;
}

 if (!PyArg_ParseTuple(args, "si", &name, &age))
```

لاحظ إننا هنا توقعنا إن هيتمرر لل Function التالى s وهى string و i وهى integer
وقمنا بإسناد هذه القيم ل name و age

ملحوظة: انت لن تقوم بالتعديل على name فافضل تعريف إنه يكون const فيكون تعريفه كالتالى

```
const char* name;

MyABS Function

static PyObject* myabs(PyObject* self, PyObject* args)
{

   int number;

   if (!PyArg_ParseTuple(args, "i", &number))
```

```
    return NULL;

  if (number<0){
    number=-number;
  }
  return Py_BuildValue("i", number);

}
```

لاحظ اننا توقعنا إن هيتمرر للFunction التالى i وهو integer وبيعبر عن الرقم
اسندنا القيمة الى number (التحويل من Python Value إلى C Value)

ال Return لازم يكون عبارة عن Python Value (او PyObject) وهو ال Return Type الخاص بال myabs Function
كما لاحظت، فبالتالى هنحتاج نحول من ال C Value إلى Python Value ودا هيتم عن طريق إستخدام
Py_BuildValue
وهنا تم تحديد إن هيتم عمل return ل i وهو Integer وقيمته مساوية ل number

holaDict Function

```
static PyObject* holaDict(PyObject* self, PyObject* args)
{

  const char* key;
  int value;

  if (!PyArg_ParseTuple(args, "si", &key, &value))
    return NULL;

  return Py_BuildValue("{s:i}", key, value); //Returns a Dict.
}
```

لاحظ ان بيتم إعادة Dictionary Object واحنا حددنا كدا بالجزئية دى{s:i}
إذا حبيت تعمل return ب List فكل ماعليك هو إنك تعدل ال Format للتالى
return Py_BuildValue("[s,i]", key, value); //Returns a List object
وإذا حبيت تعمل return ب Tuple فكل ماعليك هو إنك تعد ال Format كالتالى (s,i)

hola Tuple Function

```
static PyObject* holaTuple(PyObject* self, PyObject* args)
{
  const char* name;
  int age;


  if (!PyArg_ParseTuple(args, "si", &name, &age))
    return NULL;
```

```
    return Py_BuildValue("(s,i)", name, age); //Returns a Tuple
}
```

divTwo Function

```
static PyObject* divTwo(PyObject* self, PyObject* args)
{

    int first;
    int second;
    int result;

    if (!PyArg_ParseTuple(args, "ii", &first, &second))
        return NULL;

    printf("First: %i\n", first);
    printf("Second: %i\n", second);

    if(second==0){  //DivByZeroError!
        PyErr_SetString(PyExc_ZeroDivisionError, "DivByZero");
        return NULL; //Get out!
    }
    result = first/second;

    return Py_BuildValue("i", result);
}
```

إذا كان المقسوم عليه يساوى 0 يبقة فى Error! ونقدر نبلغ ال (المفسر (Interpreter)) بيه بإستخدام PyErr_SetString
نوع ال Error هو PyExc_ZeroDivisionError وال message هتكون DivByZero

2- الضم لل Methods Table كالتالى

```
static PyMethodDef SimpleModuleMethods[]=
{
    {"hola", hola, METH_VARARGS, "prints name and age"},
    {"myabs", myabs, METH_VARARGS, "returns the abs of a number"},
    {"divTwo", divTwo, METH_VARARGS, "DIV 2 "},
    {"holaTuple", holaTuple, METH_VARARGS, "returns a tuple"},
    {"holaDict", holaDict, METH_VARARGS, "returns a dict"},

    {NULL, NULL, 0, NULL}
};
```

لاحظ إن آخر عنصر فى ال Array هو حاجز..
3- عمل Initialize لل Module كالتالى بنستدعى فيها ال Py_InitModule Function كالتالى

```
PyMODINIT_FUNC
```

```
initsimplemodule(void)
{
    (void) Py_InitModule("simplemodule", SimpleModuleMethods);
}
```

4- ال Setup Script كالتالى

```
from distutils.core import setup, Extension

modExt = Extension('simplemodule', sources = ['simplemodule.c'])

setup (name = 'SimpleModPackage',
       version = '1.0',
       description = 'hola, myabs',
       ext_modules = [modExt])
```

بنوضح فيه اسم ال Package والإصدار والوصف وال extension اللذى بيشمل اسم ال module وال source

5- عمل Build و Install كالتالى

```
python setup.py build
python setup.py install
```

6- عمل Test Script واختبار ال Module كالتالى

```
#!bin/python

import simplemodule as sm

sm.hola("Ahmed", 18)
print sm.myabs(-10) #10
print sm.myabs(7) #7
print sm.holaDict("python", 1)
print sm.holaTuple("ahmed", 999)
print sm.divTwo(2, 0)
#Output:
Name: Ahmed
Age: 18
10
7
{'python': 1}
('ahmed', 999)
First: 2
Second: 0
Traceback (most recent call last):
File "C:\Python25\Projects\exten\smTest.py", line 10, in <module>
print sm.divTwo(2, 0)
ZeroDivisionError: DivByZero
```

References:

1- Extending Python
2- Programming Python 3rd Edition

Related:

1- Style Guide for C Code
2- SWIG
3- CXX

## *Chapter 15 (GUI)*

## PyGTK

هنتناول فى الجزئية دى مقدمة فى GTK


نريد ان نعمل window -نافذة-مشابهه لدى ونخليها متسنتره (فى منتصف الشاشة ) اول ماتنشئ
1- استدعى ال gtk

```
import gtk
```

2-انشئ class يورث ال gtk

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()
```

3- هنا بنقول ان ال window دى TOPLEVEL مش POPUP
بنستدعى ال __init_comp -طريقة لإنشاء الواجهة-
.show_all
الطريقة show_all بتعرض كل ال components داخل ال Window وهنا مش فى غيرها بس اتعود تستخدمها لأنك هتبقة تحط ويدجتس كتير جواها
.init_comp

```
    def __init_comp(self):
```

```
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
```

```
.set_title(new_title)
```

بتستخدم فى تغيير ال title على النافذة

```
.set_position(pos)
```

بتستخدم هنا لتحديد ال مكان الخاص بالنافذة
وليها عدة قيم زى

```
gtk.WIN_POS_CENTER
```

بتسنتر النافذة عند انشاءها

```
gtk.WIN_POS_CENTER_ALWAYS
```

هيتم سنترتها عند اى تغيير فى الsize

```
gtk.WIN_POS_MOUSE
```

هيتم اظهار النافذة عند مكان الماوس الحالى

```
.set_size_request(h,w)
```

لتحديد ارتفاع وعرض النافذة

```
if __name__=="__main__":
    w=Window()
    gtk.main()
```


هنا عندنا نافذة وفيها button واحد مكتوب عليه click me
طيب جميل

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
```

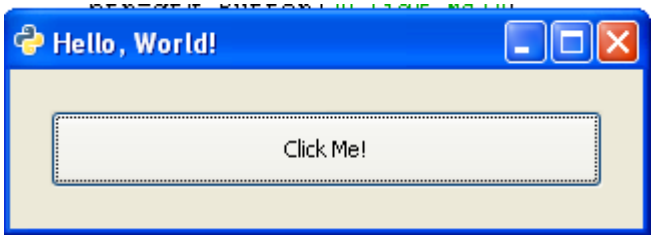
```
        self.set_border_width(20)
         self.set_position(gtk.WIN_POS_CENTER)
         btn=gtk.Button("Click Me!")
         self.add(btn)
```

التعريف العام لل button

```
  button = gtk.Button(label=None, stock=None)
```

تقدر تتحكم فى ال border_width بإستخدام

```
.set_border_width(width)
```




نريد ان يظهر مسج لطيفة كلمانضغط على ال button دا

```
    def __on_btn_clicked(self, widget, data):
        md=gtk.MessageDialog(self, gtk.DIALOG_DESTROY_WITH_PARENT,
gtk.MESSAGE_INFO, gtk.BUTTONS_OK, "Hi!")
        print widget
        print data
        md.run()
        md.destroy()
```

دى اسمها callback يعنى طريقة هيتم تنفيذها عند حدوث شئ معين زى الضغط على
نيجى لل MessageDialog
اول معامل هو ال parent
تانى معامل فيه خصائص الديالوج
gtk.DIALOG_MODAL

لو اه فهو اللذي هيصطاد اى ايفنت يحصل من الكيبورد (يمنع الوصول لل نافذة الأصلية الإ بعد انهائه)

gtk.DIALOG_DESTROY_WITH_PARENT

هيتقفل فى حال قفل ال parent

gtk.DIALOG_NO_SEPARATOR

مش هيظهر خط فاصل بين الرسالة وال buttons بتوع الرسالة

تالت معامل هو نوع المسج

هل معلومة او تحذير او سؤال او خطأ

`gtk.MESSAGE_INFO`

معلومة

`gtk.MESSAGE_WARNING`

تحذير

`gtk.MESSAGE_QUESTION`

سؤال

`gtk.MESSAGE_ERROR`

خطأ

تمام كدا بس انا شغلت الكود ومش فى حاجة حصلت :)
فعلا لأننا لسه مش ربطنا ال `callback` بال `signal`
بكل بساطة اكتب التالى

```
        btn.connect("clicked", self.__on_btn_clicked, None)
```

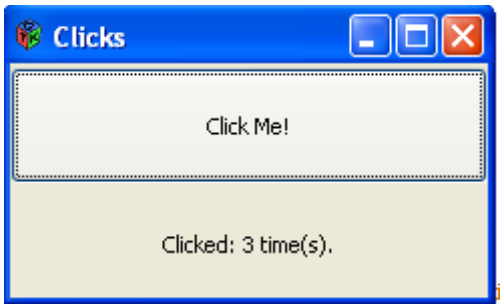

وبس كدا طبعا انت مدايق من None وايه المتغيرات اللتى تم تعريفها انا فى ال `callback` دى اصلا ؟ ايه ?? `widget, data`

طيب تمام جدا قبل ماتسألنى السؤال دا تقدر تعمل حاجة حلوة قوى بإنك تجرب `print` على `widget` وعلى `data` فى ال `callback`

```
<gtk.Button object (GtkButton) at 0xb803f0>
None
```

ال `widget` دا ال `button` فى مثالنا
ال `data` هى ال None تقدر طبعا تستبدلها بأى حاجة المهم انها تكون شئ هيفيدك..
هنتعرف عليها اكتر لاحقا...

Clicks




لاحظ ان النافذة متقسمة لجزئين راسي اول جزء فيه button والتانى فيه label بيتكتب عليه عدد مرات الضغط على ال button
احنا ممكن نكتبها بالطريقة المعتادة وممكن نكتبها بطريقة كتير افضل بإستخدام ال OOP
تمام الأول عشان نخليهم متقسمين فى شكل معين افقى او رأسى بنستخدم Hbox (اختصار ل Horizontal Box )
او Vbox (اختصار ل Vertical Box) تمام ؟ قشطة

ال vertical box بياخد ال widgets او بوكسز تانية فى صورة rows صفوف -صندوق رأسى-

ال horizontal box بياخد ال widgets او البوكسز التانى فى صورة Columns عواميد -صندوق افقى-

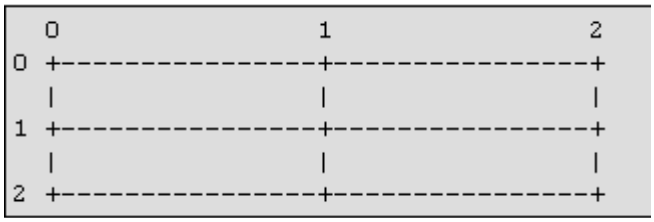
لإنشاء box بتبدأ اولا ب homogeneous ودى معناها هل كل الأجزء متساوية فى العرض او الطول ليها ترجمة بإسم متجانسة اعتقد مناسبة ؟ والمعامل التانى لتحديد عرض الفاصل

```
vbox=gtk.VBox(False, 2)
```

pack_start(child, expand, fill, padding)
لوضع ال widget من الشمال لليمين او من فوق لتحت "صورة فطرية!"
(pack_end(child, expand, fill, padding
لوضع ال widget من اليمين للشمال او من تحت لفوق وهى موجود لل Hbox وال Vbox

expand: هل عايزه يكبر مع اى زيادة فى حجم النافذة؟
fill: فى حال التصغير هل يتم اخفاء جزء منه؟وليس تصغيره
padding: الهامش حوله

-استخدام الجداول



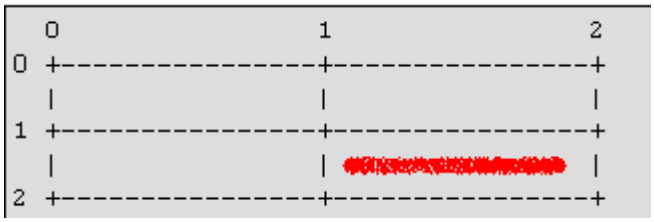

لإنشاء جدول بننشئه كالتالى

```
gtk.Table( rows, columns, homogeneous )
```

عدد الصفوف وعدد الأعمدة وهل متجانسين او لأ
لإضافة child بإستخدام ال attach method

```
.attach( child, left_attach, right_attach, top_attach, bottom_attach,
xoptions, yoptions, xpadding, ypadding)
```

child: هو الويدجت هيتم اضافته فى الجدول
left_attach : العمود على يسار المكان
right_attach: العمود على يمين المكان
top_attach: الصف فوق المكان
bottom_attach: الصف تحت المكان
مثال للتوضيح



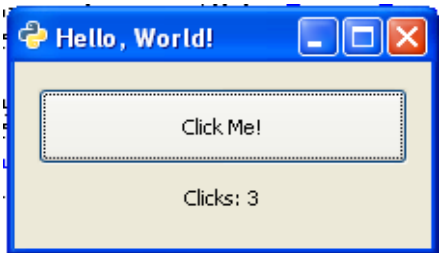

لو نريد ان نحط widget معين فى الكورنر اليمين من جدول زى مانت شايف 2X2
بيقع بين الخطين الرأسين 1و 2 وهما دول ال right_attach, left_attach
بيقع بين الخطين الأفقيين 1 و 2 وهما دول ال top_attach, bottom_attach
xoptions: الإختيارات ل x
gtk.FILL: لو الجدول اكبر من الويدجت فالويدجت هيتمدد ليشغل المساحة
gtk.EXPAND :هنا الجدول هيتمدد اذا كان فى مساحة فى ال window
gtk.SHRINK: اذا تم تصغير المساحة المتاحة للويدجت "مع تصغير الجدول مثلا" هيتم تصغيره

yoptions: الإختيارات ل y مشابهه ل x
xpadding: الهامش من ناحية ال x
ypadding: الهامش من ناحية ال y
لو انت مرضى مع الخيارات الأساسية تقدر تستخدم attach_defaults

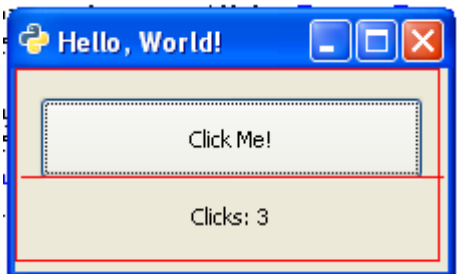
```
attach_defaults( child, left_attach, right_attach, top_attach,
bottom_attach )
```

ودى هتخليك تدخل ال left, right, top, bottom attach فقط والباقى هيكون بالإفتراضى لل x,y options
gtk.FILL | gtk.EXPAND
وال x,y padding هيكون 0
فين ال fixed ؟ موجود fixed لعرض ال ويدجتس بتحديد ال مكان على الفورم ولكن "استخدام السابق افضل من حيث حماية طريقة وضعك لل ويدجتس من حيث التمدد والإنكماش وكدا"

تعالى نعمل مثال clicks بصورة واضحة




التصميم




المستطيل الأحمر الكبير عبارة عن Vbox
وجواه صفين الصف الأول فيه button والتانى فيه label
الأول عندنا متغير __clicks بيعبر عن عدد الضغطات

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()
        self.__clicks=0
        self.show_all()

    def getClicks(self):
        return self.__clicks
```

```
    def setClicks(self, value):
        self.__clicks = value


    def delClicks(self):
        del self.__clicks

    clicks = property(getClicks, setClicks, delClicks, "Clicks's
Docstring")
```

ثانيا التصميم

```
def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)
```

هنا بنحدد خصائص النافذة title,position,border_width

```
        mvbox=gtk.VBox(False, 0)
```
بننشئ Vertical Box عشان نضم فيه button, label
```
        btnclicks=gtk.Button("Click Me!")
```
بننشئ ال Button مكتوب عليه Click Me!
```
        lbl=gtk.Label("Clicks: ")
```
بننشئ Label مكتوب عليه Clicks:

```
        mvbox.pack_start(btnclicks, True, True, 2)
        mvbox.pack_start(lbl, True, True, 0)
```
بنضيف ال btnclicks, lbl لل mvbox
```
        self.add(mvbox)
```
بنضيف ال mvbox لل window
```
        btnclicks.connect("clicked", self.__on_btnclicks_clicked, lbl,
None)
```
بنربط ال clicked signal الخاصة ب btnclicks بطريقة بإسم
__on_btnclicks_clicked

صممناها كالتالى

```
    def __on_btnclicks_clicked(self, widget, lblclicks, data):
        self.__clicks += 1
        print widget, lblclicks, data
        lblclicks.set_text("Clicks: "+str(self.__clicks))
```

المعامل الأول widget بيعبر عن ال receiver لل signal
التانى ال lblclicks بيعبر عن ال label اللذى نريد ان نغيره

التالت ال data بيعبر عن اى داتا اضافية
كل اللى هيحصل اننا هنزود عدد ال __clicks ونعدل ال تكست على الlblclicks
بإستخدام
```
.set_text(newtext)
```

الهيكلية

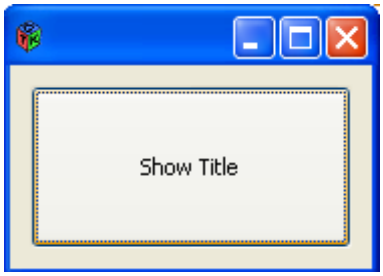
```
GObject
|
GtkObject
+GtkWidget
| +GtkMisc
| | +GtkLabel
| | | `GtkAccelLabel
| | +GtkArrow
| | `GtkImage
| +GtkContainer
| | +GtkBin
| | | +GtkAlignment
| | | +GtkFrame
| | | | `GtkAspectFrame
| | | +GtkButton
| | | | +GtkToggleButton
| | | | | `GtkCheckButton
| | | | | `GtkRadioButton
| | | | `GtkOptionMenu
| | | +GtkItem
| | | | +GtkMenuItem
| | | | +GtkCheckMenuItem
| | | | | `GtkRadioMenuItem
| | | | +GtkImageMenuItem
| | | | +GtkSeparatorMenuItem
| | | | `GtkTearoffMenuItem
| | | +GtkWindow
| | | | +GtkDialog
| | | | | +GtkColorSelectionDialog
| | | | | +GtkFileSelection
| | | | | +GtkFontSelectionDialog
| | | | | +GtkInputDialog
| | | | | `GtkMessageDialog
| | | | `GtkPlug
| | | +GtkEventBox
| | | +GtkHandleBox
| | | +GtkScrolledWindow
| | | `GtkViewport
| | +GtkBox
| | | +GtkButtonBox
| | | | +GtkHButtonBox
| | | | `GtkVButtonBox
| | | +GtkVBox
| | | | +GtkColorSelection
| | | | +GtkFontSelection
| | | | `GtkGammaCurve
```

```
| | | `GtkHBox
| | | +GtkCombo
| | | `GtkStatusbar
| | +GtkFixed
| | +GtkPaned
| | | +GtkHPaned
| | | `GtkVPaned
| | +GtkLayout
| | +GtkMenuShell
| | | +GtkMenuBar
| | | `GtkMenu
| | +GtkNotebook
| | +GtkSocket
| | +GtkTable
| | +GtkTextView
| | +GtkToolbar
| | `GtkTreeView
| +GtkCalendar
| +GtkDrawingArea
| | `GtkCurve
| +GtkEditable
| | +GtkEntry
| | `GtkSpinButton
| +GtkRuler
| | +GtkHRuler
| | `GtkVRuler
| +GtkRange
| | +GtkScale
| | | +GtkHScale
| | | `GtkVScale
| | `GtkScrollbar
| | +GtkHScrollbar
| | `GtkVScrollbar
| +GtkSeparator
| | +GtkHSeparator
| | `GtkVSeparator
| +GtkInvisible
| +GtkPreview
| `GtkProgressBar
+GtkAdjustment
+GtkCellRenderer
| +GtkCellRendererPixbuf
| +GtkCellRendererText
| +GtkCellRendererToggle
+GtkItemFactory
+GtkTooltips
`GtkTreeViewColumn
```

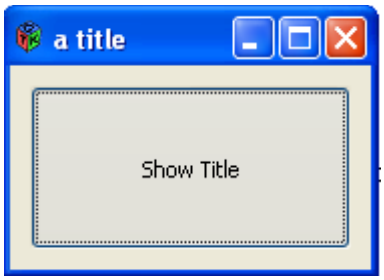
*Toggle Button*

```
toggle_button = gtk.ToggleButton(label=None)
```

مشابه لل check box ودا ليه حالتين True او False




هنا وهو مش متنشط يعنى False




هنا هو متنشط يعنى active
اول مايضغط عليه بيرسل toggled signal

```
import gtk


class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)

        mvbox=gtk.VBox(False, 0)
        togbtn=gtk.ToggleButton("Show Title")
       togbtn.set_active(True)
        togbtn.connect("toggled", self.__on_toggled)

        mvbox.pack_start(togbtn, True, True, 2)

        self.add(mvbox)
```

```
    def __on_toggled(self, widget):
        if self.title.strip():
            self.set_title(" ")
        else:
            self.set_title("Hello, World!")


if __name__=="__main__":
    w=Window()
    gtk.main()
```

*CheckButton*

الصورة العامة لإنشاءه

```
  check_button = gtk.CheckButton(label=None)
```



نفس نظام السابق ولكن بشكل مختلف ليس اكثر

```
import gtk


class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
```

```
        self.set_position(gtk.WIN_POS_CENTER)

        mvbox=gtk.VBox(False, 0)
        chkbtn=gtk.CheckButton("Show Title")
        chkbtn.set_active(True)
        chkbtn.connect("toggled", self.__on_toggled)

        mvbox.pack_start(chkbtn, True, True, 2)

        self.add(mvbox)

    def __on_toggled(self, widget):
        if self.title.strip():
            self.set_title(" ")
        else:
            self.set_title("Hello, World!")



if __name__=="__main__":
    w=Window()
    gtk.main()
```

*RadioButton*

الصورة العامة لإنشاءه

```
  radio_button = gtk.RadioButton(group=None, label=None)
```

لازم تظبط ال group دى عشان يشتغل بصورة سليمة فبكل بساطة خليها None لأول radio button وبعد كدا خليها ال radio button اللذي انشئ اول واحد!

هننشئ حاجة مشابهه لدى

التصميم


النافذة فيها 2 radio buttons وفيها Horizontal Separator و button

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()
        self.gender="Male"
        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        mvbox=gtk.VBox(False, 0)
        rd1=gtk.RadioButton(None, "Male")
        rd1.set_active(True)

        rd1.connect("toggled",self.__on_radio_toggled,  "Male")

        rd2=gtk.RadioButton(rd1, "Female")
        rd2.connect("toggled",self.__on_radio_toggled, "Female")

        mvbox.pack_start(rd1, False, False, 2)
        mvbox.pack_start(rd2, False, False, 2)

        mvbox.pack_start(gtk.HSeparator(), True, True, 0)
        btninfo=gtk.Button("OK!")
        btninfo.connect("clicked", self.__on_btninfo_clicked)

        mvbox.pack_start(btninfo, False, False, 3)
        self.add(mvbox)

    def __on_radio_toggled(self, w, data):
```

```
        self.gender=data

    def __on_btninfo_clicked(self, w):
        md=gtk.MessageDialog(self, gtk.DIALOG_DESTROY_WITH_PARENT,
gtk.MESSAGE_INFO, gtk.BUTTONS_OK, self.gender)
        md.run()
        md.destroy()
```

لاحظ هنا انشئنا Hseparator (فاصل افقى)وضفناه مباشرة

```
        mvbox.pack_start(gtk.HSeparator(), True, True, 0)
```

ال signal المهمة هنا هى toggled للـ radio buttons

*Adjustment*

هى مش ويدجت ولكن بيستخدم فى تخزين ونقل معلومات محددة لإعدادات ويدجتس معينة زى السكرولبارز والسبينرز والراينجز -الفترات- وغيرهم
لعمل adjustment object بننشئه كالتالى

```
Adjustment( value, lower, upper, step_increment, page_increment, page_size )
```

تقدر تعتبرها ك model لويدجت وهو بيعرضهالك فى ال view بتاعته على كل حال هنشوف
ال value القيمة الأساسية
ال lower اقل قيمة
ال upper اعلى قيمة
ال step_increment مقدار الزيادة

*Scale*

فى منها افقى وفي رأسى لإنشاء الأوبجكتس منها -تقدر تطلق عليها منزلق-

```
VScale( adjustment )
VScale( min, max, step )
HScale( adjustment );
HScale min, max, step );
```

يإما تمرر adjustment او تمرر اقل واكبر قيمة والزيادة
لإظهار القيمة مع ال scale او لأ تقدر تستخدم draw_value او set_draw_value وتديهم قيمة true او false فى حالة الإخفاء
-- هى true افتراضيا..

digits/set_digits

لتحديد عدد الأرقام بعد العلامة العشرية المرغوب فى ظهورها

set_value_pos(pos)

تقدر تستخدمهم لتحديد المكان اللذى سيظهر عليه قيمة ال value ودى بتاخد القيم

| |
|---|
| gtk.POS_LEFT<br>gtk.POS_RIGHT<br>gtk.POS_TOP<br>gtk.POS_BOTTOM |


مثلا لعمل المثال دا هننشئه كالتالى

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        adj=gtk.Adjustment(5,1, 101)
        hscale=gtk.HScale(adj)
        hscale.set_digits(0)


        mvbox=gtk.VBox(False, 0)
        mvbox.pack_start(hscale, True, True, 0)
        mvbox.pack_start(gtk.HSeparator(), True, True, 0)

        self.add(mvbox)
```

update_policy

بتحدد متى تعديل ال value بال adjustment الخاصة بال range widget وترسل ال value_changed signal بتاخد قيم مثل

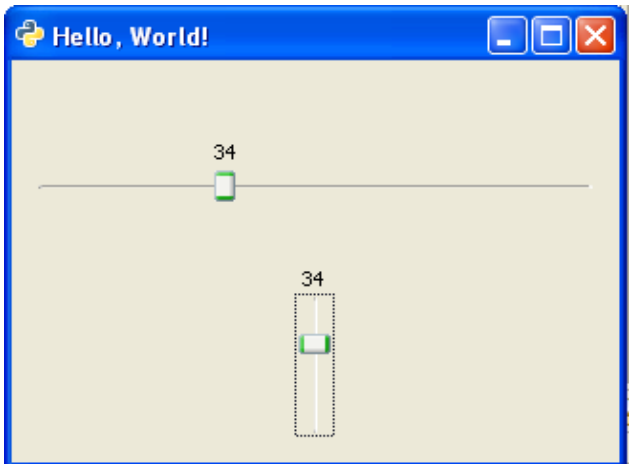
gtk.UPDATE_CONTINUOUS

بترسل عند حدوث اقل تغيير ممكن فى ال range

gtk.UPDATE_DISCONTINUOUS
بترسل لما المستخدم يسيب الماوس ويكون ال range ثابت
gtk.UPDATE_DELAYED

بترسل بمجرد ان المستخدم يترك الماوس او يتوقف عن الحركة لفترة صغيرة وبيتم التحكم فى ال policy من خلال
.set_update_policy(up_policy)

للحصول على ال adjustment الخاصة بال range استخدم
get_adjustment.
لتعديل ال adjustment استخدم
.set_adjustment(adj)




هنا عندنا 2 scales واحدة افقى والتانية رأسى والإتنين هنديلهم adjustment واحدة عشان التغيير فيهم يبقة على التوازى

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        adj=gtk.Adjustment(5,1, 101)
```

```
        hscale=gtk.HScale(adj)
        hscale.set_digits(0)
        vscale=gtk.VScale(adj)
        vscale.set_digits(0)

        mvbox=gtk.VBox(False, 0)
        mvbox.pack_start(hscale, True, True, 0)
        mvbox.pack_start(vscale, True, True, 2)

        self.add(mvbox)
```

*.set_digits(num)*

بتحدد عدد الأرقام المطلوبة بعد العلامة(خليناها 0)

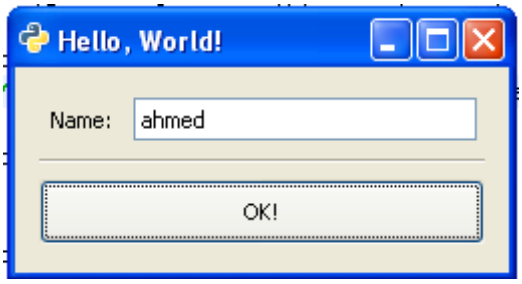

نيجى لمثال تانى هنا عندنا Scaleو SpinButton نريد ان نربطهم ان لما يتغير قيمة اى منهم يتعدل فى التانية هنا هنستخدم ال adjustment object للإتنين بحيث ان يتعدل قيمة ال value فيها للإتنين (بما انها ال model اللذى يعرضه كل من ال Scale, SpinButton)

كود المثال

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        adj=gtk.Adjustment(5,1, 101)
        hscale=gtk.HScale(adj)
        hscale.set_digits(0)
        spin=gtk.SpinButton(adj)
```

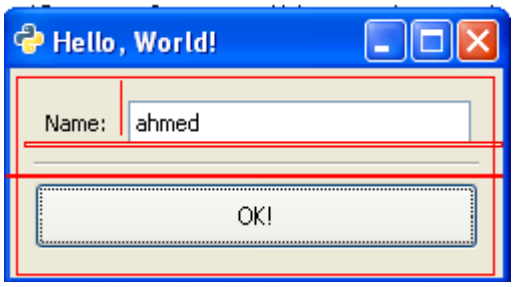
```
        spin.set_digits(0)

        mvbox=gtk.VBox(False, 0)
        mvbox.pack_start(hscale, True, True, 0)
        mvbox.pack_start(spin, True, True, 2)
        mvbox.pack_start(gtk.HSeparator(), True, True, 0)

        self.add(mvbox)
```

*Label/Entry*




ال Label بيستخدم لعرض تكست ما بدون الحاجة لتغييره من المستخدم
ال Text Entry مشابه لل تكست فيلد بيدخل فيه المستخدم قيمة مطلوبة مثلا...

التصميم




الكود

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()

        self.show_all()

    def __init_comp(self):
        self.set_title("Hello, World!")
```

```
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        mvbox=gtk.VBox(False, 0)

        lblname=gtk.Label("Name:")
        nameentry=gtk.Entry()

        hbox=gtk.HBox(False, 0)
        hbox.pack_start(lblname, True, True, 0)
        hbox.pack_start(nameentry, True, True, 4)

        mvbox.pack_start(hbox, True, True, 0)
        mvbox.pack_start(gtk.HSeparator(), True, True, 2)

        btnok=gtk.Button("OK!")
        btnok.connect("clicked", self.__on_btnok_clicked , nameentry)

        mvbox.pack_start(btnok, True, True, 2)
        self.add(mvbox)

    def __on_btnok_clicked(self, w,e, data=None):
        print e.get_text()
```

*Entry*

`.set_text(newtext)` لتغيير ال text

`.insert_text(text, _from)` بتضيف text من عند نقطة ال _from

`.select_region(_from,_to)` بتظلل منطقة معينة تبدأ من _from وتنتهى ب _to

`.set_max_length(maxlen)` بتحدد بيها اقصى عدد حروف لل entry

`.set_editable(bool)` هل يقدر المستخدم يعدل فيها؟

`Label:`

`.set_text(text)` تعديل ال text

`.get_text()` اعادة ال text

.set_justify(just) بتاخد قيم كالتالى

JUSTIFY_LEFT يسار

```
  JUSTIFY_RIGHT يمين
  JUSTIFY_CENTER المنتصف
```

```
.set_line_wrap(bool)
```

هل ي wrap السطور او لأ؟

```
.set_markup(markup)
```

تخزين markup

*gtk.Arrow*

بيستخدم ليشير الى اتجاه ما لبرنامج (بوضع رأس السهم)


```
arrow = gtk.Arrow(arrow_type, shadow_type)
arrow.set(arrow_type, shadow_type)
```

arrow_type بتعبر عن نوع السهم اعلى ، اسفل ، يمين ، يسار

```
ARROW_UP
ARROW_DOWN
ARROW_LEFT
ARROW_RIGHT
```

و shadow_type بتحدد نوع ال shadow

```
SHADOW_IN
SHADOW_OUT   # the default
SHADOW_ETCHED_IN
SHADOW_ETCHED_OUT
```

مثال

```
# Create an Arrow widget with the specified parameters
# and pack it into a button
def create_arrow_button(arrow_type, shadow_type):
    button = gtk.Button();
    arrow = gtk.Arrow(arrow_type, shadow_type);
    button.add(arrow)
    button.show()
    arrow.show()
    return button

class Arrows(object):
    def __init__(self):
        # Create a new window
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)

        window.set_title("Arrow Buttons")

        # It's a good idea to do this for all windows.
        window.connect("destroy", lambda x: gtk.main_quit())

        # Sets the border width of the window.
        window.set_border_width(10)

        # Create a box to hold the arrows/buttons
        box = gtk.HBox(False, 0)
        box.set_border_width(2)
        window.add(box)

        # Pack and show all our widgets
        box.show()

        button = create_arrow_button(gtk.ARROW_UP, gtk.SHADOW_IN)
        box.pack_start(button, False, False, 3)

        button = create_arrow_button(gtk.ARROW_DOWN, gtk.SHADOW_OUT)
        box.pack_start(button, False, False, 3)

        button = create_arrow_button(gtk.ARROW_LEFT, gtk.SHADOW_ETCHED_IN)
        box.pack_start(button, False, False, 3)

        button = create_arrow_button(gtk.ARROW_RIGHT,
gtk.SHADOW_ETCHED_OUT)
        box.pack_start(button, False, False, 3)

        window.show()
```

فى الدالة

```
# and pack it into a button
```

```
def create_arrow_button(arrow_type, shadow_type):
    button = gtk.Button();
    arrow = gtk.Arrow(arrow_type, shadow_type);
    button.add(arrow)
    button.show()
    arrow.show()
    return button
```

بنقوم بإنشاء button يحوى arrow قمنا بإنشاءه من خلال ال arrow_type, shadow_type واضفناه لل button بإستخدام ال add method

*gtk.Tooltips*

بتستخدم لتحديد ال tooltip (نص مساعد على الويدجت)
تنشئ كالتالى

```
tooltips = gtk.Tooltips()
```

تقوم بتحديد نص التلميح بإستخدام set_tip method

```
tooltips.set_tip(widget, tip_text)
```

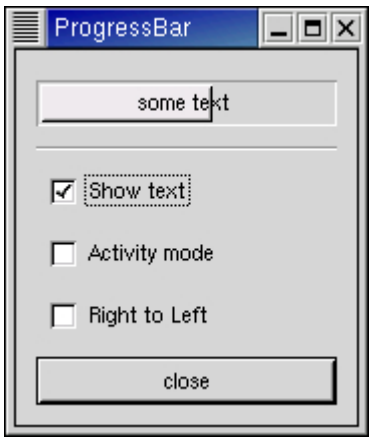
widget هو الويدجت المطلوب تحديد ال tip له
tip_text النص

فقط قم بإضافة ال create_arrow_button

```
class Tooltips:
    def __init__(self):
        # Create a new window
        window = gtk.Window(gtk.WINDOW_TOPLEVEL)

        window.set_title("Tooltips")

        # It's a good idea to do this for all windows.
        window.connect("destroy", lambda w: gtk.main_quit())

        # Sets the border width of the window.
        window.set_border_width(10)

        # Create a box to hold the arrows/buttons
        box = gtk.HBox(False, 0)
        box.set_border_width(2)
        window.add(box)

        # create a tooltips object
        self.tooltips = gtk.Tooltips()
```

```
        # Pack and show all our widgets
        box.show()

        button = create_arrow_button(gtk.ARROW_UP, gtk.SHADOW_IN)
        box.pack_start(button, False, False, 3)
        self.tooltips.set_tip(button, "SHADOW_IN")

        button = create_arrow_button(gtk.ARROW_DOWN, gtk.SHADOW_OUT)
        box.pack_start(button, False, False, 3)
        self.tooltips.set_tip(button, "SHADOW_OUT")

        button = create_arrow_button(gtk.ARROW_LEFT, gtk.SHADOW_ETCHED_IN)
        box.pack_start(button, False, False, 3)
        self.tooltips.set_tip(button, "SHADOW_ETCHED_IN")

        button = create_arrow_button(gtk.ARROW_RIGHT,
gtk.SHADOW_ETCHED_OUT)
        box.pack_start(button, False, False, 3)
        self.tooltips.set_tip(button, "SHADOW_ETCHED_OUT")

        window.show()
```

*gtk.ProgressBar*




هو ويدجت يستخدم لعرض تقرير عن الحالة

لإنشاءه

```
progressbar = gtk.ProgressBar(adjustment=None)
```

فى حال عدم تحديد ال adjustment هيتم انشاءها

.set_fraction(**fraction**)

لتحديد ال fraction وهى الكم المنتهى

.set_orientation(**orientation**)

لتحديد اتجاه ملء الزيادة

```
PROGRESS_LEFT_TO_RIGHT من اليسار لليمين
PROGRESS_RIGHT_TO_LEFT من اليمين لليسار
PROGRESS_BOTTOM_TO_TOP من اسفل لأعلى
PROGRESS_TOP_TO_BOTTOM من اعلى لأسفل
```

.get_text()

للحصول على النص الظاهر على ال progressbar

.set_text(to)

تحديد النص الظاهر على ال progressbar إلى to

.pulse()

لتشير حدوث تغيير فى ال progressbar

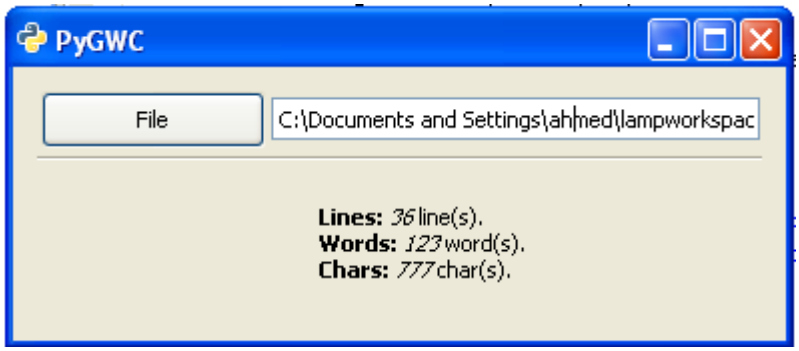
```
def progress_timeout(pbobj):

    if pbobj.activity_check.get_active():
        pbobj.pbar.pulse()
    else:
        # Calculate the value of the progress bar using the
        # value range set in the adjustment object
        new_val = pbobj.pbar.get_fraction() + 0.01
        if new_val > 1.0:
            new_val = 0.0
        # Set the new value
        pbobj.pbar.set_fraction(new_val)

    # As this is a timeout function, return TRUE so that it
    # continues to get called
    return True
```

.activity_check.get_active()

هل progressobject نشط اولأ

***PyGWC***

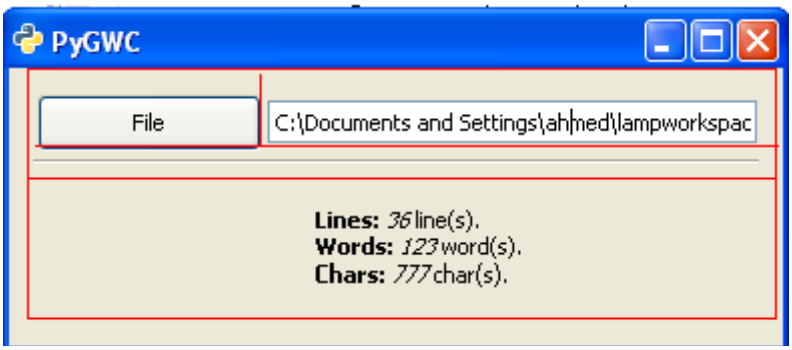
مثال لعد الأسطر والأحرف والكلمات

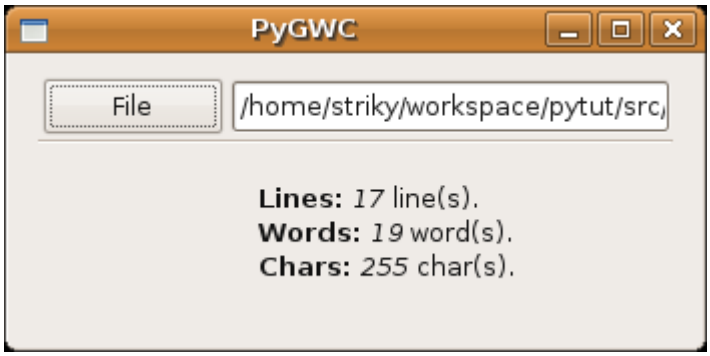



الأول نكتب الخدمة

```
class WordCounter(object):

    def __init__(self):
        pass

    def set_file(self, p):
        self.filepath=p
        fileobj=file(p, "r")
        self.txt=fileobj.read()

        fileobj.close()

    def get_nwords(self):
        return self.txt.count(" ")+1

    def get_nlines(self):
        return self.txt.count("\n")

    def get_nchars(self, count_spaces=True):
        if count_spaces:
            return len(self.txt)
        else:
            return len(self.txt)-self.get_nwords()

    def get_info_as_markup(self, spaces=True):
        s="""
        <b>Lines:</b> <i>%s</i> line(s).
        <b>Words:</b> <i>%s</i> word(s).
        <b>Chars:</b> <i>%s</i> char(s).
        """%(self.get_nlines(), self.get_nwords(),
self.get_nchars(spaces))
        return s
```

التـصميم




الكود

```
class Window(gtk.Window):

    def __init__(self):
        super(Window, self).__init__(gtk.WINDOW_TOPLEVEL)
        self.__init_comp()
        self.wc=WordCounter()

        self.show_all()

    def __init_comp(self):
        self.set_title("PyGWC")
        self.set_position(gtk.WIN_POS_CENTER)
        self.set_border_width(12)

        mvbox=gtk.VBox(False, 0)
        btnfile=gtk.Button("File")
        btnfile.connect("clicked", self.select_file)
```

```
        self.fileentry=gtk.Entry()
        self.fileentry.set_editable(False)
        hbox=gtk.HBox(False, 0)

        hbox.pack_start(btnfile, True, True, 2)
        hbox.pack_start(self.fileentry, True, True, 1)

        mvbox.pack_start(hbox, True, True, 0)
        mvbox.pack_start(gtk.HSeparator(), True, True, 2)

        self.lblinfo=gtk.Label()
        mvbox.pack_start(self.lblinfo, True, True, 0)

        self.add(mvbox)
```

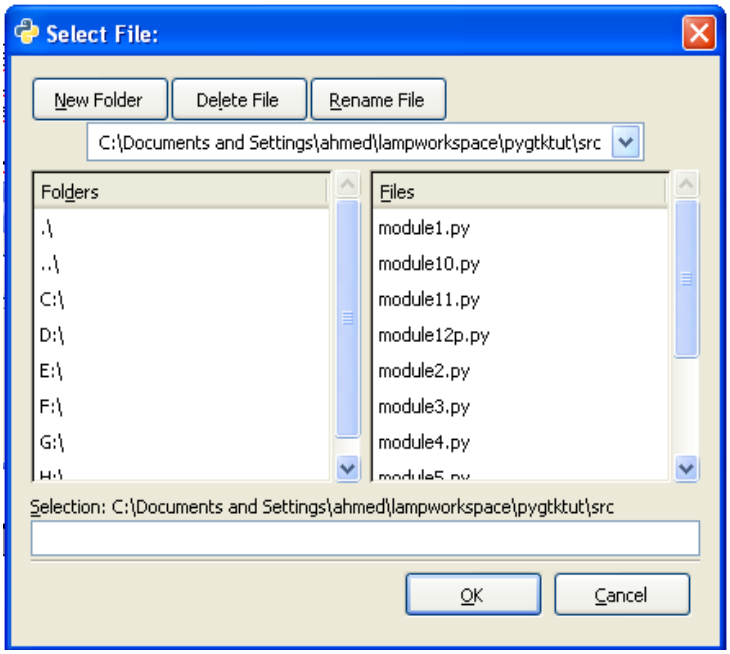
هنربط ال clicked signal الخاصة بال btnfile ب select_file callback المعرفة كالتالى

```
    def select_file(self,w):


        sel = gtk.FileChooserDialog("Open..",
                                    self,
                                     gtk.FILE_CHOOSER_ACTION_OPEN,
                                      (gtk.STOCK_CANCEL,
gtk.RESPONSE_CANCEL,
                                          gtk.STOCK_OPEN, gtk.RESPONSE_OK)
                                      )

        sel.set_default_response(gtk.RESPONSE_OK)

        res=sel.run()
        if res==gtk.RESPONSE_OK:
            self.wc.set_file(sel.get_filename())
            self.fileentry.set_text(sel.get_filename())
            self.lblinfo.set_markup(self.wc.get_info_as_markup(spaces=True
))

        else:
            print "Dialog with RESPONSE_CANCEL!"

        sel.destroy()
```

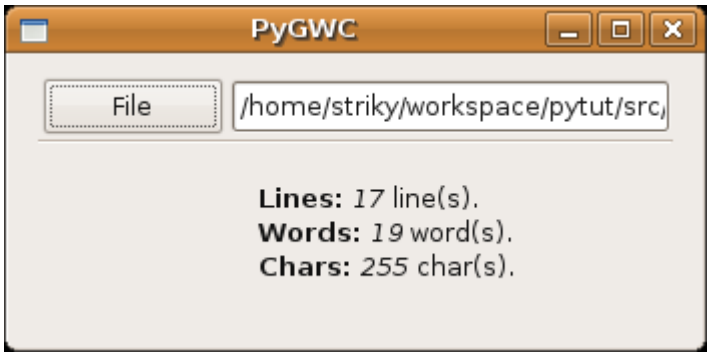
هنا بننشئ كائن من ال FilechooserDialog (لأختيار الملفات او الفولدرات) ونحدد العنوان بال

constructor
واخبرناه بعنوان النافذة ،ونوع فعلها (ACTION_OPEN لفتح الملفات وليست للحفظ) وانواع ردود الفعل العائدة منها وهى OK, CANCEL وايضا ال stocks

```
      sel = gtk.FileChooserDialog("Open",
                                  self,
                                   gtk.FILE_CHOOSER_ACTION_OPEN,
                                    (gtk.STOCK_CANCEL,
gtk.RESPONSE_CANCEL,
                                       gtk.STOCK_OPEN, gtk.RESPONSE_OK)
                                    )
```



يتم تدمير ال sel  ويتم تحدد اسم الفايل الخاص بال WordCounter object
نعدل التكست على ال fileentry لإسم الفايل
اخيرا نضع ال markup الناتج من ال wc.get_info_as_markup على ال lblinfo بإستخدام ال set_markup method




ملحوظة: لاتقم بإستخدام FileSelection (تم تعديل المثال ليستخدم FileChooserDialog)

*gtk.ComboBox*


قائمة منسدلة بتشمل مجموعة من الإختيارات ، فى مثالنا هنا مجموعة من اسماء التوزيعات

```
class Window(gtk.Window):



    def __init__(self):
        super(Window, self).__init__()
        self.__init_comp() #prepare components

        self.show_all()

    def __init_comp(self):

        self.vbox=gtk.VBox(False, 2)

        entries = ["Slackware", "Ubuntu", "Mandriva", "Debian"]
        cbo=gtk.combo_box_new_text()
        map(cbo.append_text, entries)

        cbo.connect("changed", self._on_cboselection_changed)
        self.vbox.pack_start(cbo, False, False)

        self.add(self.vbox)

    def _on_cboselection_changed(self, widget, *args):
        print widget.get_active_text()
```

الطريقة السريعة هى انشاءها بإستخدام combo_box_new_text (بتجهز الكومبو بتكست رندرر)
استدعى الطريقة append_text على كل المدخلات entries بإستخدام map

اربط ال changed signal ب on_cboselection_changed_
وللحصول على العنصر النشط فى القائمة استخدام get_active_text()

*Menus*

هننشئ فى المثال نافذة بشريط قوائم فيه قائمة واحدة File وبتشمل 3 عناصر open, save, quit

```
import gtk

class Window(gtk.Window):



    def __init__(self):
        super(Window, self).__init__()
        self.__init_comp() #prepare components

        self.show_all()

    def __init_comp(self):

        self.vbox=gtk.VBox(False, 2)

        mbar=gtk.MenuBar()
        self.vbox.pack_start(mbar, False, False, 2)

        self.connect("delete_event", lambda w,e: gtk.main_quit())

        file_item=gtk.MenuItem("File")
        fmenu=gtk.Menu()
        file_item.set_submenu(fmenu)

        open_item = gtk.MenuItem("Open")
        save_item = gtk.MenuItem("Save")
        quit_item = gtk.MenuItem("Quit")

        map(fmenu.append, [open_item,save_item, quit_item])

        open_item.connect("activate", self._on_item_activated, "Open")
        save_item.connect("activate", self._on_item_activated, "Save")
        quit_item.connect("activate", self._on_item_activated, "Quit")

        mbar.append(file_item)

        self.add(self.vbox)
```

خطوات متتالية لإنشاءا القوائم
1- شريط القوائم من الصف MenuBar وإضافته للصندوق الرأسى

```
        mbar=gtk.MenuBar()
        self.vbox.pack_start(mbar, False, False, 2)
```

2- العنصر الذى سيحوى القائمة وهنا هو File وينشئ من الصف MenuItem

```
        file_item=gtk.MenuItem("File")
```

3- القائمة من الصف Menu

```
        fmenu=gtk.Menu()
```

4- ضم القائمة

```
        file_item.set_submenu(fmenu)
```

5- إنشاء عناصر القائمة وإضافتهم بإستخدام الطريقة append

```
        open_item = gtk.MenuItem("Open")
        save_item = gtk.MenuItem("Save")
        quit_item = gtk.MenuItem("Quit")
```

```
 map(fmenu.append, [open_item,save_item, gtk.SeparatorMenuItem(),
quit_item])
```

لاحظ اننا انشأنا فاصل افقى SeparatorMenuItem ليفصل بين عنصرى فتح وحفظ وعنصر اغلاق

6- ربط ال activate signal بال callback المناسبة

```
        open_item.connect("activate", self._on_item_activated, "Open")
        save_item.connect("activate", self._on_item_activated, "Save")
        quit_item.connect("activate", self._on_item_activated, "Quit")
```

التى عرفناها كالتالى مثلا

```
    def _on_item_activated(self, w, action):
        print "Calling %s"%action
```

7- اخيرا ضم العنصر اللذى يحوى القائمة الى شريط القوائم

```
        mbar.append(file_item)
```

يوجد طريقة اسهل وهى بإستخدام ال UIManager توفر عليك الكثير من الكتابة

## Gladizer

تقدر تصمم واجهات ممتازة بإستخدام glade وبدون الحاجة لكتابة الكثير من الأكواد فيما يتعلق بشكل الواجهة بالإستعانة ب gladizer ، ملف ال glade. الواجهة الممثلة بصورة XML لاتعلم بايثون عنها شئ سوا انها ملف XML فيجب ان تخبرها بأنها وصف لواجهة برنامج ما وانك تريد ان تستخدمها فى تطبيقك. تستطيع القيام بذلك يدويا، او بإستخدام gladizer او اى اداة مشابهة مثل gladex, glade2py الخ

1- افتح glade وانشئ نافذة وقم بحفظ الملف ب hello.glade
هيكون الكود مشابه للتالى

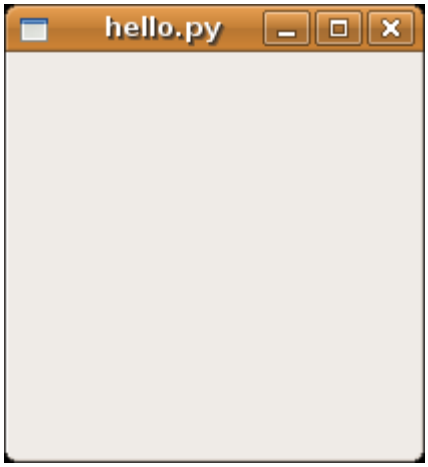
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE glade-interface SYSTEM "glade-2.0.dtd">
<!--Generated with glade3 3.4.5 on Tue Dec  2 06:07:51 2008 -->
<glade-interface>
  <widget class="GtkWindow" id="window1">
    <child>
      <placeholder/>
    </child>
  </widget>
</glade-interface>
```

استدعى gladizer ليقوم بالربط

```
striky@striky-desktop:~/Desktop$ gladizer.py -f hello.glade -p Python > hello.py
```

قم بالتنفيذ

```
striky@striky-desktop:~/Desktop$ python hello.py
```

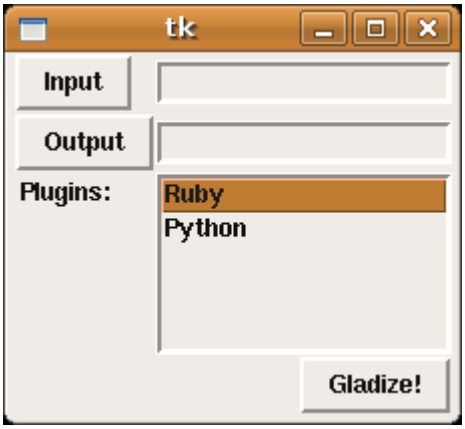


تصفح الكود الناتج من جلاديزر ستجده مشابه للتالى

```
#!bin/python

##CODE GENERATED by gladizer 1.2


import pygtk
pygtk.require('2.0')

import gtk, gtk.glade

class MainWindow(object):

        def __init__(self):


                #Widget tree..
                self.wTree=gtk.glade.XML('hello.glade')

                #connect signals and handlers.
                self.wTree.signal_autoconnect(self)

                self._window1=self.wTree.get_widget('window1')
                self._window1.show()
# run main loop

def main():
   mainwindow = MainWindow()
   #mainwindow.window1.show()
   gtk.main()

if __name__ == "__main__":
   main()
```

وبعد ذلك قم بإضافة ال callbacks كما تحب :)

تستطيع ايضا استخدام gladizer بواجهة رسومية من خلال gladizerguitk




صفحة gladizer
/http://sourceforge.net/projects/gladizer

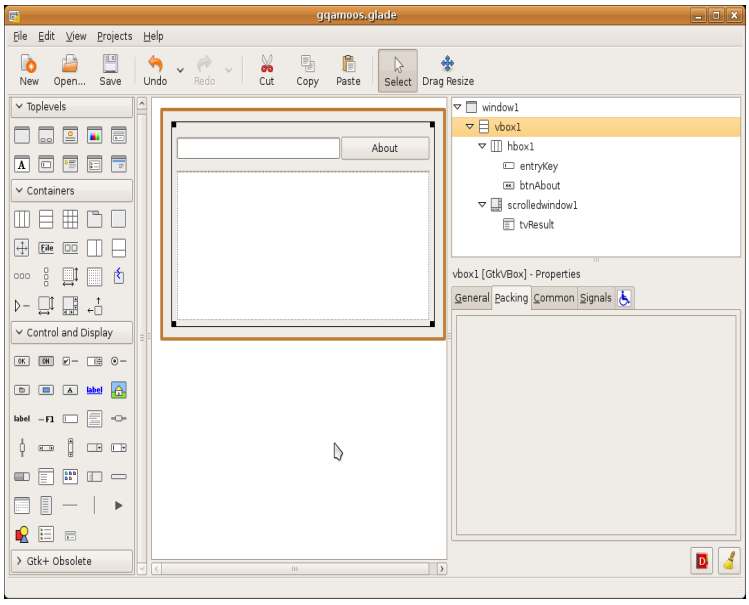
ملحوظة هناك بعض الأمثلة محسنة من http://pygtk.org/pygtk2tutorial/index.html
وهذا الفصل ليس إلا مقدمة لعالم GTK الرائع.

# Rad with Glade (Gqamoos)

هنشرح مثال على كيفية استخدام Glade مع PyGTK و Gladizer لإنشاء برنامج قاموس يعتمد على wordlist مع عرب ايز

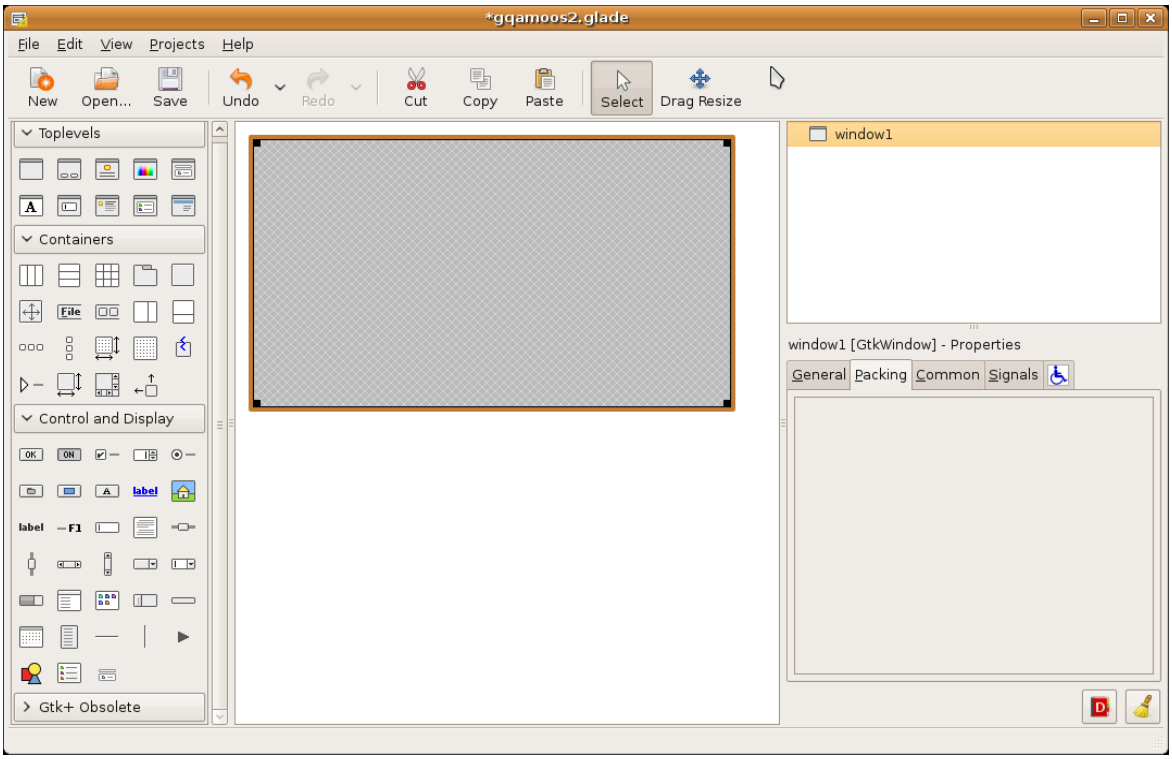
1- حمل ال wordlist
http://www.arabeyes.org/project.php?proj=Wordlist

2- افتح glade وانشئ ملف واحفظه ب gqamoos

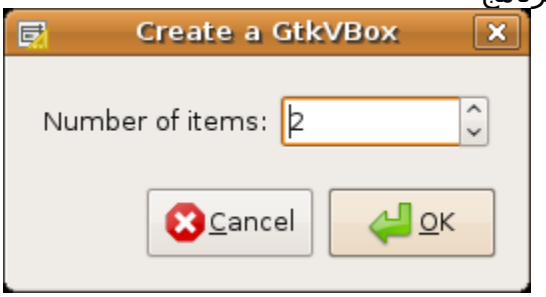



لاحظ الواجهة مقسمة للتالى

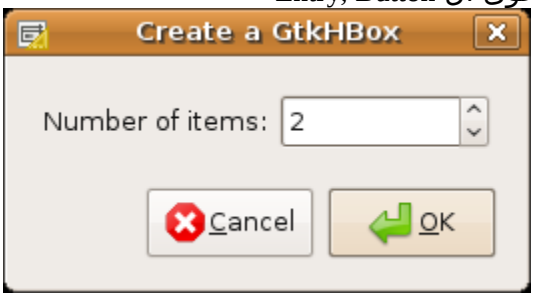
نافذة



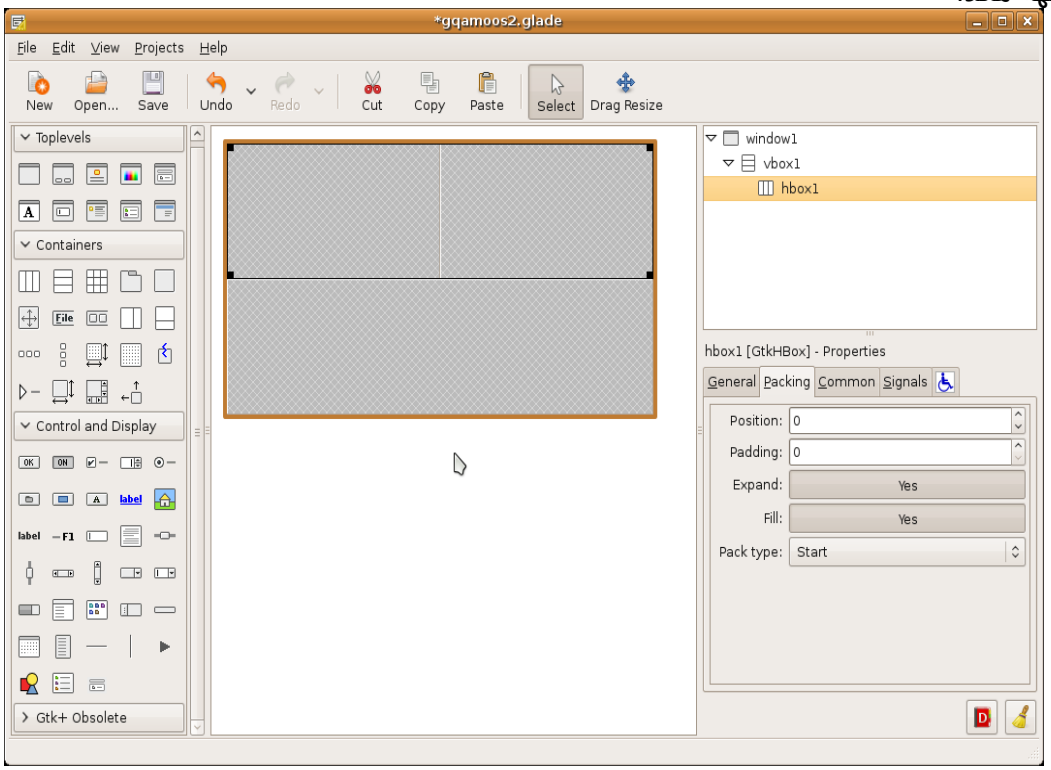

قم بإضافة Vbox من اللوحة (لصفين) على النافذة ليشمل القسم الأعلى TextEntry, Button
ال TextEntry للكلمة المراد البحث عنها
ال Button لعرض رسالة حول البرنامج



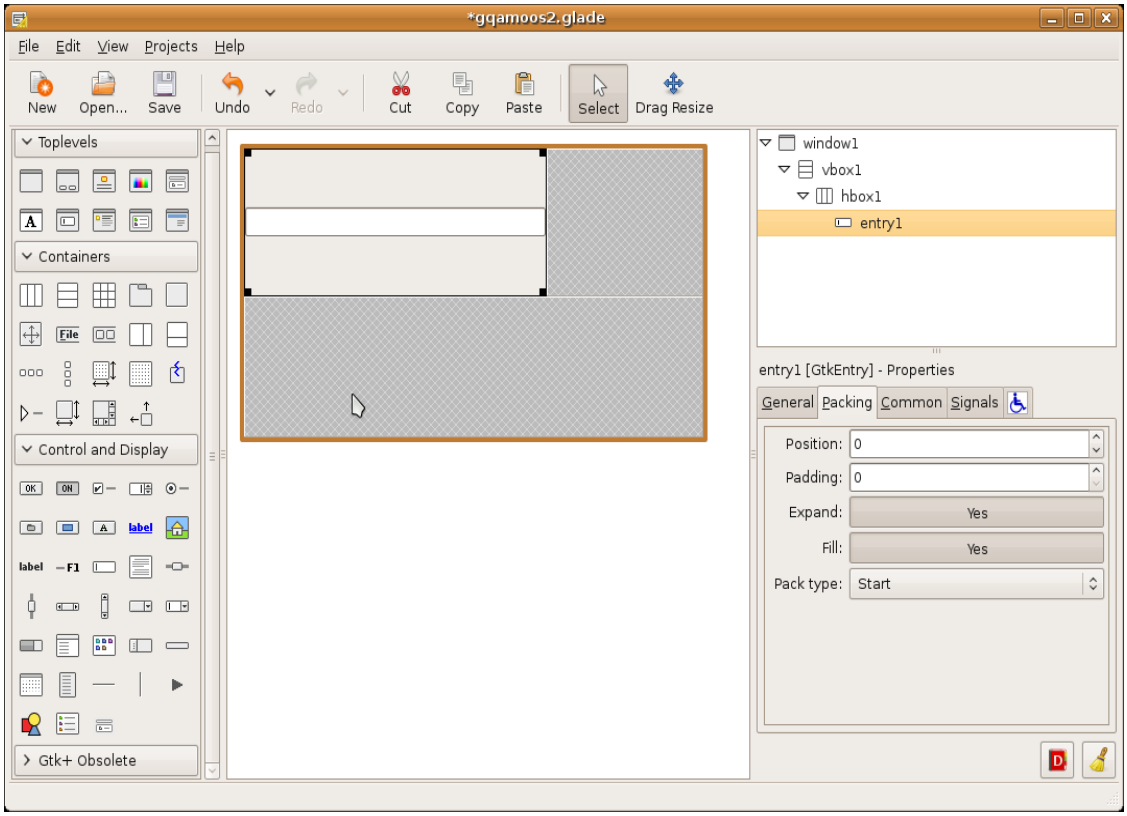

قم بإضافة صندوق افقى Hbox ليحوى ال Entry, Button



لتصبح الواجهة هكذا

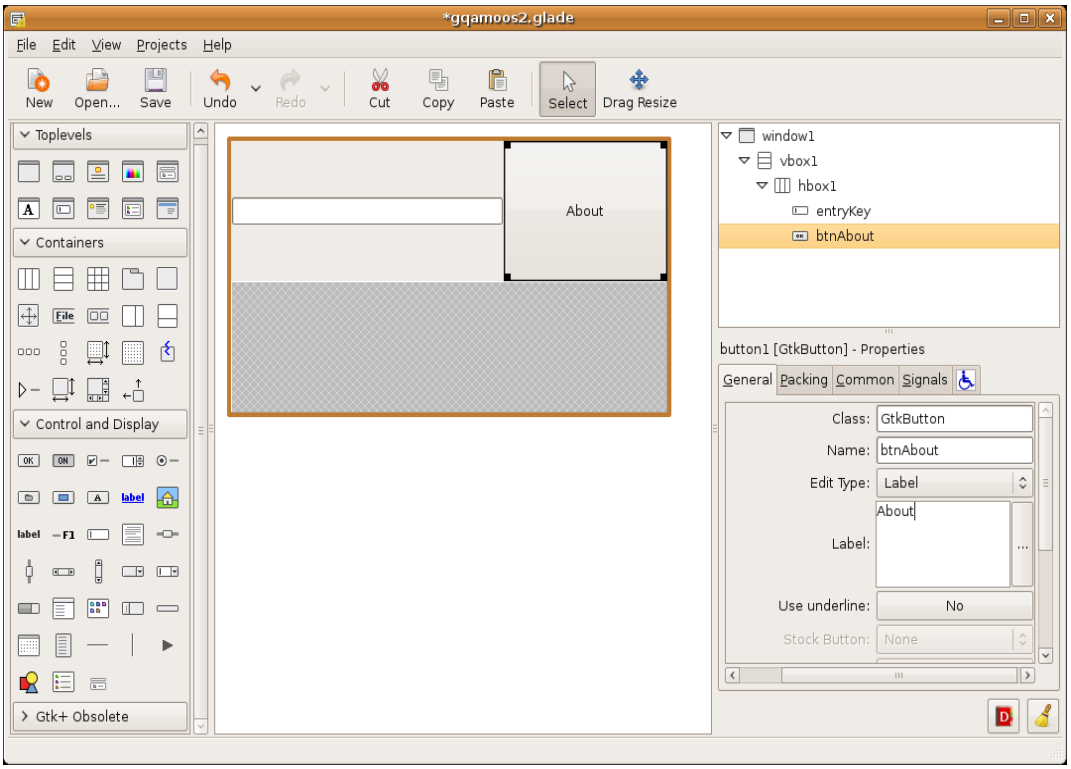



قم بإضافة ال Text Entry

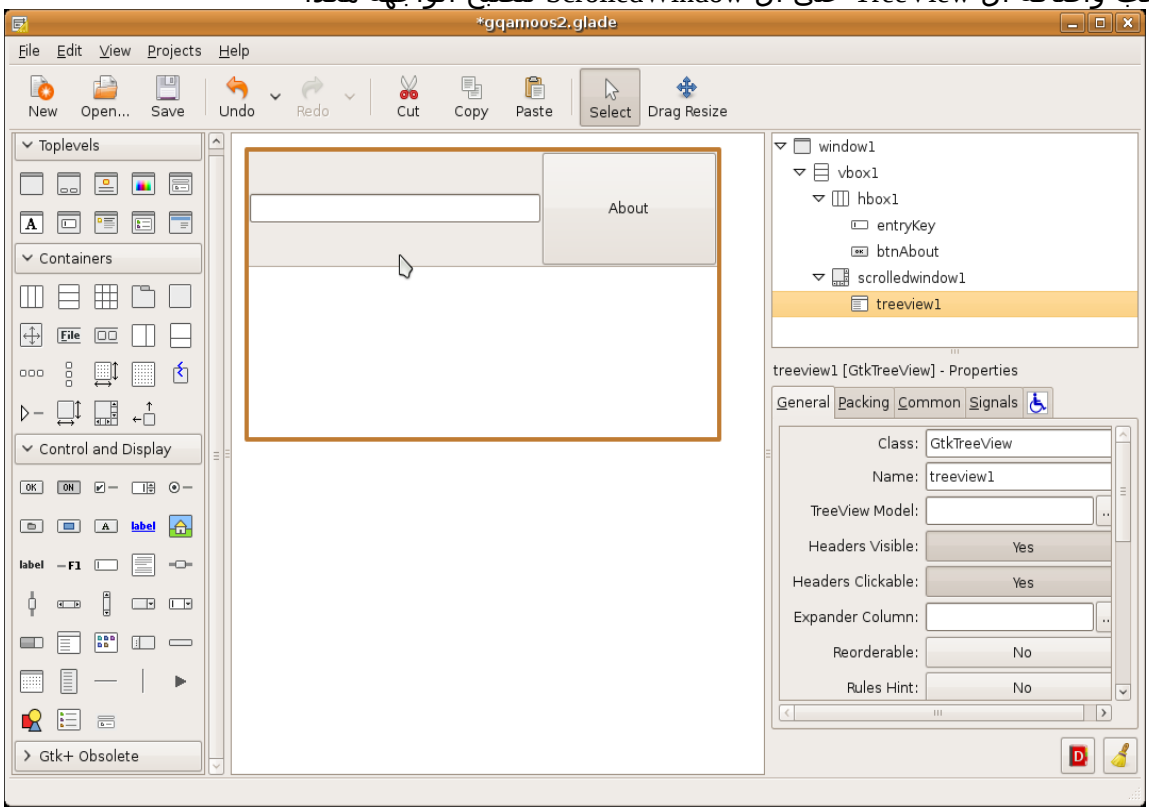


من قسم ال General بأسفل اليمين قم بتغيير اسم الويدجت ل entryKey

قم بإضافة زر Button من اللوحة على يمين ال Entry



فى القسم الأسفل اضف ScrolledWindow ليتيح لنا استخدام ال"Scrollbars" على ال TreeView
قم بسحب واضافة ال TreeView على ال ScrolledWindow لتصبح الواجهة هكذا

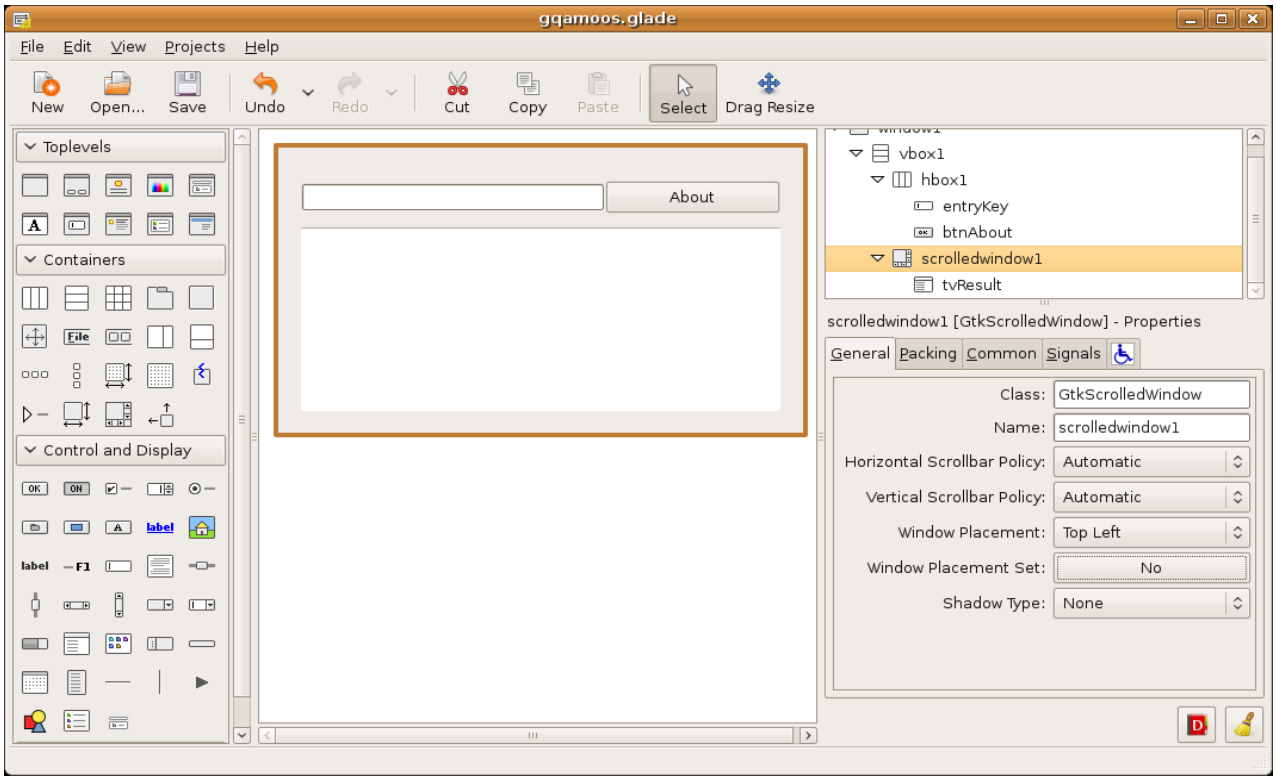


الآن سنقوم ببعض التعديلات على الخواص للويدجتس
قم بتعديل ال Label للـ btnAbout إلى About
ومن قسم ال signals قم بإضافة on_btnAbout_clicked للـ clicked signal
قم بتغيير ال Expand, Fill للـ Hbox الى القيم No, No

ولل entryKey قم بإضافة on_entryKey_changed للـ changed signal "ستجدها تحت فرع ال GtkEditable"

وايضا قم بتعديل اسم ال TreeView الى tvResult

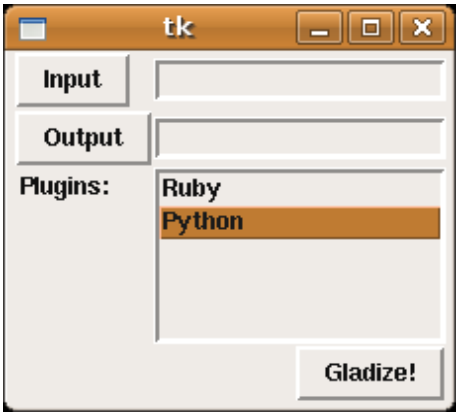
لتصبح الواجهة هكذا



تمام؟ كدا انهينا التصميم
قم بتشغيل gladizer على ملف gqamoos.glade ليكون الخرج هو gqamoos.py

```
striky@striky-desktop:~$ gladizer.py --file gqamoos.glade -p Python > gqamoos.py
```

او عن طريق gladizerguitk.py




اخيرا قم بتحرير الملف المشابه للتالى

```
#!bin/python

##CODE GENERATED by gladizer 1.2


import pygtk
pygtk.require('2.0')


import gtk, gtk.glade

class MainWindow(object):

     def __init__(self):


          #Widget tree..
          self.wTree=gtk.glade.XML('gqamoos.glade')

          #connect signals and handlers.
          self.wTree.signal_autoconnect(self)

          self._window1=self.wTree.get_widget('window1')
          self._vbox1=self.wTree.get_widget('vbox1')
          self._hbox1=self.wTree.get_widget('hbox1')
```

```
            self._entryKey=self.wTree.get_widget('entryKey')
            self._btnAbout=self.wTree.get_widget('btnAbout')
            self._tvResult=self.wTree.get_widget('tvResult')
            self._window1.show()


      def on_entryKey_changed(self, widget, *args):
            pass

      def on_btnAbout_clicked(self, widget, *args):
            pass


# run main loop

def main():
    mainwindow = MainWindow()
    #mainwindow.window1.show()
    gtk.main()

if __name__ == "__main__":
    main()
```

وذلك بإضافة ال looker service للبحث فى ال wordlist

```
import looker
```

وإضافة الوصول لأنواع البيانات الخاصة ب gobject مثل TYPE_STRING بإضافة ال gobject

```
import gobject
```

وإنشاء متغير داخلى من النوع looker.WordListLooker

```
self._wc=looker.WordsListLooker()
```

تجهييز ال TreeView بإستخدام الطريقة _init_tv

```
            self._init_tv()
```

الطريقة _init_tv معرفة كالتالى

```
      def _init_tv(self):
            encell=gtk.CellRendererText()
            encol=gtk.TreeViewColumn("English", encell, text=0)
            arcell=gtk.CellRendererText()
            arcol=gtk.TreeViewColumn("Arabic", arcell, text=1)
```

```
map(self._tvResult.append_column,[encol, arcol])
```

سيتم ال TreeView باسماء الأعمدة ونوعهم وإضافتهم وذلك من خلال اضافة الأعمدة encol, arcol وهم من النوع gtk.TreeViewColumn ويأخذون معاملات كالتالى
1- الهيدر "النص الظاهر"
2- ال CellRenderer وهو المسئول عن "رندرة" الخلية (مثل encell وهو "مرندر من النوع النصى" لعرض البيانات النصية)
3- الترتيب فى العرض من ال store تابع الطريقة _get_model()

```
    def _get_model(self):
        lstore=gtk.ListStore(gobject.TYPE_STRING, gobject.TYPE_STRING)
        self._wc.searchKey=self._entryKey.get_text()
        for row in self._wc.lookup():
            iter=lstore.append([row[0], row[1]])

        return lstore
```

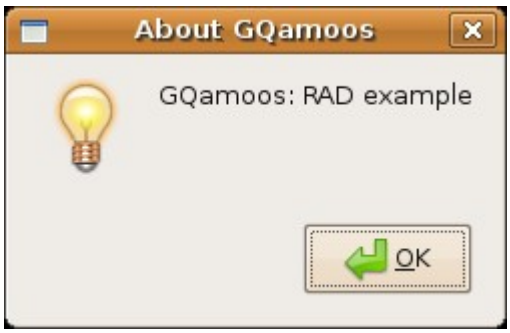
هنا نقوم بتحديد نوع البيانات المخزنة. تستطيع استخدام str بدلا من goject.TYPE_STRING
نحصل على النص الظاهر على ال entryKey من خلال الطريقة get_text ونقوم بإسناده لل searchKey property للـ self._wc "الأوبجكت المسئول عن استغلال ال looker service"
نقوم بإضافة نتائج البحث من خلال lstore.append فى داخل list ونقوم بإعادة ذلك ال store
نقوم بربط التغيير فى النص الظاهر على ال entryKey بتعديل ال model الخاص بال TreeView لتغيير نتائج البحث مع كل تغيير وذلك بتعديل on_entryKey_changed callback

```
    def on_entryKey_changed(self, widget, *args):
        self._tvResult.set_model(self._get_model())
```

نقوم بتحديد ال model بإستخدام الطريقة set_model الخاصة بال tvResult "ال TreeView"

تحديد رسالة حول البرنامج قم بتعديل ال on_btnAbout_clicked المسئولة عن معالجة ال clicked signal للزر btnAbout

```
    def on_btnAbout_clicked(self, widget, *args):
        md=gtk.MessageDialog(self._window1,
gtk.DIALOG_DESTROY_WITH_PARENT, gtk.MESSAGE_INFO, gtk.BUTTONS_OK,
"GQamoos: RAD example")
        md.set_title("About GQamoos")
        md.run()
        md.destroy()
```



قم بتشغيل البرنامج

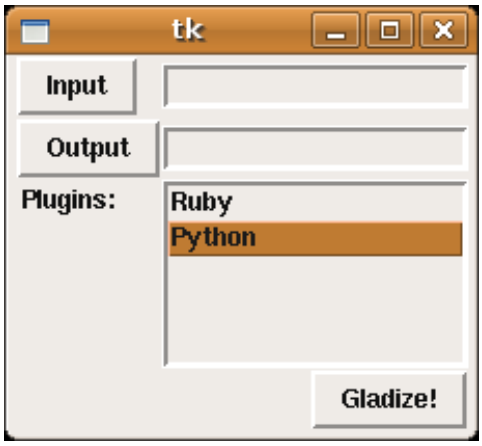

ستلاحظ اننا لم نحدد عنوان النافذة..تستطيع تعديل ذلك اما بإستخدام الطريقة set_title للل self._window1 او من خلال ملف glade
بتعديل Title للل window1 إلى Gqamoos


الكود النهائى

```
#!bin/python

##CODE GENERATED by gladizer 1.2


import pygtk
pygtk.require('2.0')

import looker

import gobject
import gtk, gtk.glade

class MainWindow(object):

     def __init__(self):


          #Widget tree..
          self.wTree=gtk.glade.XML('gqamoos.glade')

          #connect signals and handlers.
          self.wTree.signal_autoconnect(self)

          self._window1=self.wTree.get_widget('window1')
          self._vbox1=self.wTree.get_widget('vbox1')
          self._hbox1=self.wTree.get_widget('hbox1')
          self._entryKey=self.wTree.get_widget('entryKey')
          self._btnAbout=self.wTree.get_widget('btnAbout')
          self._tvResult=self.wTree.get_widget('tvResult')
          self._window1.show()
          self._wc=looker.WordsListLooker()

          self._init_tv()


     def _init_tv(self):
          encell=gtk.CellRendererText()
          encol=gtk.TreeViewColumn("English", encell, text=0)
          arcell=gtk.CellRendererText()
          arcol=gtk.TreeViewColumn("Arabic", arcell, text=1)
          map(self._tvResult.append_column,[encol, arcol])


     def _get_model(self):
          lstore=gtk.ListStore(str, str) #gobject.TYPE_STRING,
gobject.TYPE_STRING
          self._wc.searchKey=self._entryKey.get_text()
          for row in self._wc.lookup():
               iter=lstore.append([row[0], row[1]])

          return lstore
```

```
     def on_entryKey_changed(self, widget, *args):
          self._tvResult.set_model(self._get_model())

     def on_btnAbout_clicked(self, widget, *args):
          md=gtk.MessageDialog(self._window1,
gtk.DIALOG_DESTROY_WITH_PARENT, gtk.MESSAGE_INFO, gtk.BUTTONS_OK,
"GQamoos: RAD example")
          md.set_title("About GQamoos")
          md.run()
          md.destroy()
          #pass

# run main loop

def main():
    mainwindow = MainWindow()
    #mainwindow.window1.show()
    gtk.main()
if __name__ == "__main__":
    main()
```

## HowTo GladizerguiTK ?




سكربت جلاديزر تم التعرض ليه بالأعلى وتم فيه كتابة واجهة بإستخدام tKInter
لاحظ الواجهة مقسمة لشبكة (صفوف واعمدة)
الصف الأول يشمل عمودين زر input و entry
الصف الثانى يشمل عمودين زر output و entry
الصف الثالث يشمل عمودين label و listbox تحوى عناصر ال plugins المتاحة
الصف الرابع زر Gladize

1- قم بإستدعاء الوحدات اللازمة

```
from Tkinter import *
import tkFileDialog as tfd
import tkMessageBox as tmd
```

كل مكونات Tkinter و وحدة tkFileDialog و tkMessageBox
ننشئ صف جديد يشتق ال Frame class

```
class GladizerWindow(Frame):

    def __init__(self, master=None):

        Frame.__init__(self, master)

        self._inputfile=StringVar()
        self._outputfile=StringVar()
        self._usedplugin=None

        self._init_comps()
```

لاحظ ال self._inputfile, self._outputfile هم objects من الصف StringVar() اللذى سنقوم بربطه بال مسارات المطلوبة
استدعاءنا للطريقة _init_comps لإنشاء الواجهة اللتى ذكرناها بالأعلى
ويتم التعامل مع القيمة المخزنة بإستخدام الطرق get, set

```
def _init_comps(self):


    inp=Button(self, text="Input", command=self._getinputfile)
    inp.grid(row=0, sticky=W+N)

    self.inpentry=Entry(self, textvariable=self._inputfile)
    self.inpentry.grid(row=0, column=1, sticky=E)

    out=Button(self, text="Output", command=self._getoutputfile)
    out.grid(row=1, sticky=W+N)


    self.outentry=Entry(self, textvariable=self._outputfile)
    self.outentry.grid(row=1, column=1, sticky=E)

    Label(self, text="Plugins:").grid(row=2, column=0, sticky=N+W)
    #plugins list.
    self.lbplugins=Listbox(self, height=5, selectmode=BROWSE, relief=SUNKEN)
    self.lbplugins.grid(row=2, column=1, columnspan=3, sticky=E+S)
    if len(getplugins())>=1:
        for plugin in getplugins():
            self.lbplugins.insert(0, plugin)

    self.lbplugins.select_set(0)

    start=Button(self, text="Gladize!", command=self._gladize).grid(row=3, columnspan=2, sticky=E+S)
    self.grid()
```

لإنشاء زر من الصف Button

```
    inp=Button(self, text="Input", command=self._getinputfile)
```

يأخذ المشيد مجموعة من المعاملات
text هى النص الظاهر
ال command هى ال callback اللتى سيتم تنفيذها
الطريقة grid لتحديد مكانة وهى تأخذ عدة معاملات لتحديد الصف والعمود والإزاحة الخ  مثل row و column, columnspan
لإنشاء Entry

```
    self.inpentry=Entry(self, textvariable=self._inputfile)
```

لاحظ ربطنا لل textvariable بالمتغير self._inputfile بحيث ان يكون التغيير ديناميكى

لإنشاء ال Listbox

```
    self.lbplugins=Listbox(self, height=5, selectmode=BROWSE, relief=SUNKEN)
```

نحددها على الشبكة (الصف والعمود. الخ)

```
    self.lbplugins.grid(row=2, column=1, columnspan=3, sticky=E+S)
```

نضيف العناصر لها بإستخدام الطريقة insert وتأخذ اول معامل المركز الذى سيتم الإدخال عنده والثانى هو القيمة المدخلة

```
    if len(getplugins())>=1:
        for plugin in getplugins():
            self.lbplugins.insert(0, plugin)
```

نحدد الإختيار الإفتراضى بإستخدام الطريقة select_set اللتى تأخذ معامل قيمته المركز

```
    self.lbplugins.select_set(0)
```

تأخذ عدة معاملات مثل الإرتفاع ونوع الإختيار ولاحظ ايضا relief (اللتى جعلنا قيمتها SUNKEN)لجعلها غائصة

ال callbacks للزرين input, output

```
def _getinputfile(self, *args):

    self._inputfile.set(tfd.askopenfilename())
    #self.inpentry.delete(0, END)
    #self.inpentry.insert(0, self._inputfile)


def _getoutputfile(self, *args):

    self._outputfile.set(tfd.asksaveasfilename())
    #self.outentry.delete(0, END)
    #self.outentry.insert(0, self._outputfile)
```

تقدم لنا الوحدة tkFileDialog الدوال اللازمة للتعامل مع الملفات (اختيار ملف مثلا)
نقوم بإستخدام tfd.askopenfilename() للحصول على مسار الملف المطلوب فتحه ونخزن قيمته بإستخدام الطريقة set الخاصة بال StringVar object
وبنفس الكيفية نحصل على مسار ملف الحفظ بإستخدام tfd.asksaveasfilename()

تقدم لنا الوحدة tkMessageBox الدوال اللازمة للتعامل مع الرسائل مثل showerror, showinfo,.. etc

```
def showerror(self, msg):
    tmd.showerror("Error", msg)
```

لعرض رسالة خطأ showerror

```
tmd.showinfo("Gladized", "%s gladized successfully!"%self._outputfile.get())
```

لعرض رسالة معلومات showinfo
لاحظ الطريقة get الخاصة بال StringVar للحصول على القيمة المخزنة

```
def main():
```

```
    root=Tk()           #Master
    gw=GladizerWindow(root) #GladizerWindow
    root.mainloop()     #Entering the mainloop

if __name__=="__main__":
    main()
```

اخيرا نعرض النافذة وندخل التطبيق

```
    root=Tk()           #Master
    gw=GladizerWindow(root) #GladizerWindow
```

ونبدأ دائرة الأحداث بإستخدام الطريقة mainloop

```
    root.mainloop()     #Entering the mainloop
```

*تطبيق: قم بإنشاء واجهة رسومية لسكربت replacer

**Qt**

ماذا عن Qt ؟
تمت ترجمة الجزئية الخاصة بها من ZetCode وستجدها متوفرة للتصفح اونلين او تحميل ملف ال pdf سيتم توزيعها مع الكتاب
وللمزيد من التعمق تستطيع جزئية تصميم الواجهات الرسومية بإستخدام Qt/Ruby (لن تحتاج منك الكثير من الجهد) وايضا قراءة الأمثلة المرفقة مع PyQt4

**wxPython**

تم الغاء ضمها لهذا الإصدار (ربما الإصدار القادم)

**tKinter**

ليست هناك نية للتعرض لها عبر مثال Howto GladizerGUItk (يفضل عدم استخدامها لأى مشاريع مستقبلية GTK او wxPython او Qt) هم افضل الإختيارات المتاحة لك

**ماذا عن Py3K ؟**
**سيبقى الكتاب عن Python 2.5 وربما فى التحديث القادم يتم وضع اهم التغييرات**
**تستطيع الإطلاع عليها من هنا**
**http://docs.python.org/3.0/whatsnew/3.0.html**

# الخاتمة

## لقد فعلتها!

مبروك انهاءك الكتاب!

## الى اين ؟

تستطيع الإتجاة لمجالات عديدة فبايثون لغة عامة اي تستخدم فى كثير من المجالات وقد وفر لك الكتاب اساس جيد لذلك المشوار مثل التعامل مع بايثون و معالجة البيانات (XML, INI, Databases,.. etc ) وايضا كيفية انشاء واجهات ( استخدام GTK و Glade )وايضا تعلمت قدرا جيدا من اساسيات انشاء تطبيقات الشبكات
تم توفير الكثير من المصادر حول بايثون وانشاء التطبيقات الرسومية منها
Mono IronPython WinForms Tutorial
PyGTK Tutorial
--للتعامل مع PyQt4 قم بفتح الملف المرفق مع الكتاب اللذي يشمل ترجمة الدورة الشهيرة من ZetCode فى استخدام PyQt4 وايضا لاغنى عن الأمثلة المرفقة مع حزمة PyQt4 .. تستطيع الحصول عليها من هنا ايضا
http://ojuba.org/wiki/doku.php/docs/pyqt4

اتجه الى صفحة الوثائق http://python.org/doc/

## مصادر اخرى

ارشح لك الكتب التالية لإستمرار المشوار

Learning Python 3rd
كتاب اقل مايقال عنه انه ممتاز يحقق لك الرسوخ فى اللغة بطريقة لاتتخيلها

Programming Python, 3rd Edition
الجزء الثانى (المتقدم) ل Learning Python 3rd
يتناول برمجة النظام الملفات والخيوط والعمليات وايضا الواجهات الرسومية بإستخدام tKInter وعدة تطبيقات مكتملة كمحرر نصوص ومستعرض صور ولعبة وغيره وجزئية متعمقة فى الشبكات من ناحية ال client side وال server side وبعض التعرض لهياكل البيانات ومعالجة النصوص والكثير

Core Python Programming, 2nd
بإختصار هو الكتاب الأفضل فى رأيى كمرحلة تانية لكتاب Learning Python او تكملة ل Programming Python 3rd يناقش الكثير والكثير من الموضوعات البايثونية بدرجة متعمقة

Foundations of Agile Python Development
كتاب سيجعلك مبرمج بايثون افضل للأبد

Foundations of Python Network Programming
اذا حبيت ابدا برمجة الشبكات فذلك الكتاب هو اختيارك الأول ليس خاص بالشبكات فقط ولكن ايضا ببعض الأجزاء عن الويب وتعدد المهام والويب سرفسز وايضا جزئيات متعلقة ب POP, IMAP, SMTP

Wrox Beginning Python

كتاب رائع ياخذك للناحية العملية وانشاء تطبيقات مفيدة واهتمام مكثف بال XML وال Networking ولكن لاتأخذه للإستفادة من الأساسيات فكتاب Learning Python افضل كثيرا فى هذه الجزئية

Twisted Network Programming Essentials
الكتاب الوحيد اللذى يغطى twisted بتعمق (هو حلك الأفضل لكتابة تطبيقات شبكات امنة وعالية الجودة بإستخدام twisted )

Manning wxPython in Action
الكتاب الخاص ب wxPython ومقسم لمقدمة عن wxPython ومميزاتها والمخطط الأساسى واساسيات انشاء الوجهات  واستخدام متقدم كالطباعة وبعض الويدجات المتقدمة

الكتاب والأمثلة اللتى تأتى مع الوثائق هم خيارك الأول لعالم wxPython الرائع

Rapid GUI Programming with Python and Qt
الكتاب رقم 1 للتعامل مع Python, Qt مقسم ايضا لعدة اجزاء الأساسيات كمقدمة عامة وبعض التطبيقات الخفيفة وايضا استخدام Qt Designer واساسيات استخدام المخططات وقواعد البيانات والطباعة والأحداث الخ الخ  واستخدام متقدم كتعدد الخيوط

**لاتنس** قسم المقالات الخاص ب Programming-Fr34ks
**http://programming-fr34ks.net/smf/articles-12/**

## شكرا!

ايها القارئ لإختيارك الكتاب لمشوارك مع بايثون

## اهداء

اهدى هذا الكتاب اللذى اتمنى ان اكون قد وفقت فى عرضه الى والدى وإلى جميع اصدقائى لن اذكر اسماء حتى لاانسى احدا

ملحق 1: استخدام Py2EXE

Py2EXE لايفضل استخدامه سوى فى حالة التحزيم وعدم معرفة المستخدم بايثون وبكل المتطلبات للبرنامج الخ الخ ولكن ليس من اجل اخفاء الكود، على كل حال

1) قم بتحميل وتستيب
http://www.py2exe.org

2) فى محررك المفضل اكتب مايشابه التالى

```
#!bin/python
print "Programming Fr34ks r0x !"
```

3) احفظ الملف وليك pf.py وقم بتجربته حتى لايكون هناك اخطاء
4) اكتب سكربت التستيب

```
#!bin/python
from distutils.core import setup
import py2exe
setup(console=['pf.py'])  #pf.py is our script name
```

5) احفظ ملف التستيب وشغله

```
%>python setup.py py2exe
```

6) انقل لمجلد ال dist
7) شغله

```
%>pf.exe
Programming Fr34ks r0x !
```

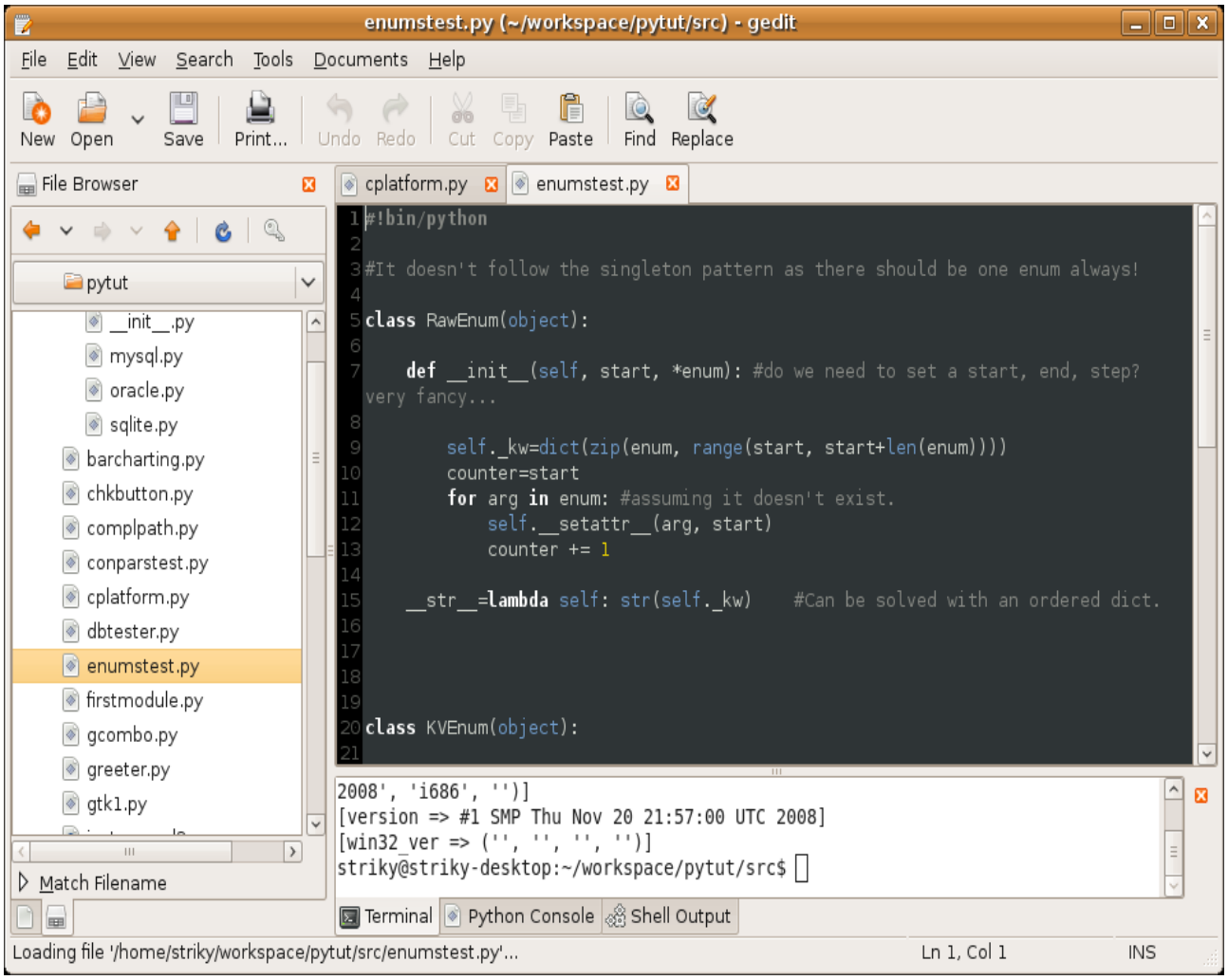
ملحق 2: محررات النصوص وبيئات التطوير

*VIM*

محرر قوى جدا ومشهور على نظم UNIX-LIKE اكتر (لايكفى كتاب للكلام عنه)
http://en.wikipedia.org/wiki/Vim_(text_editor)

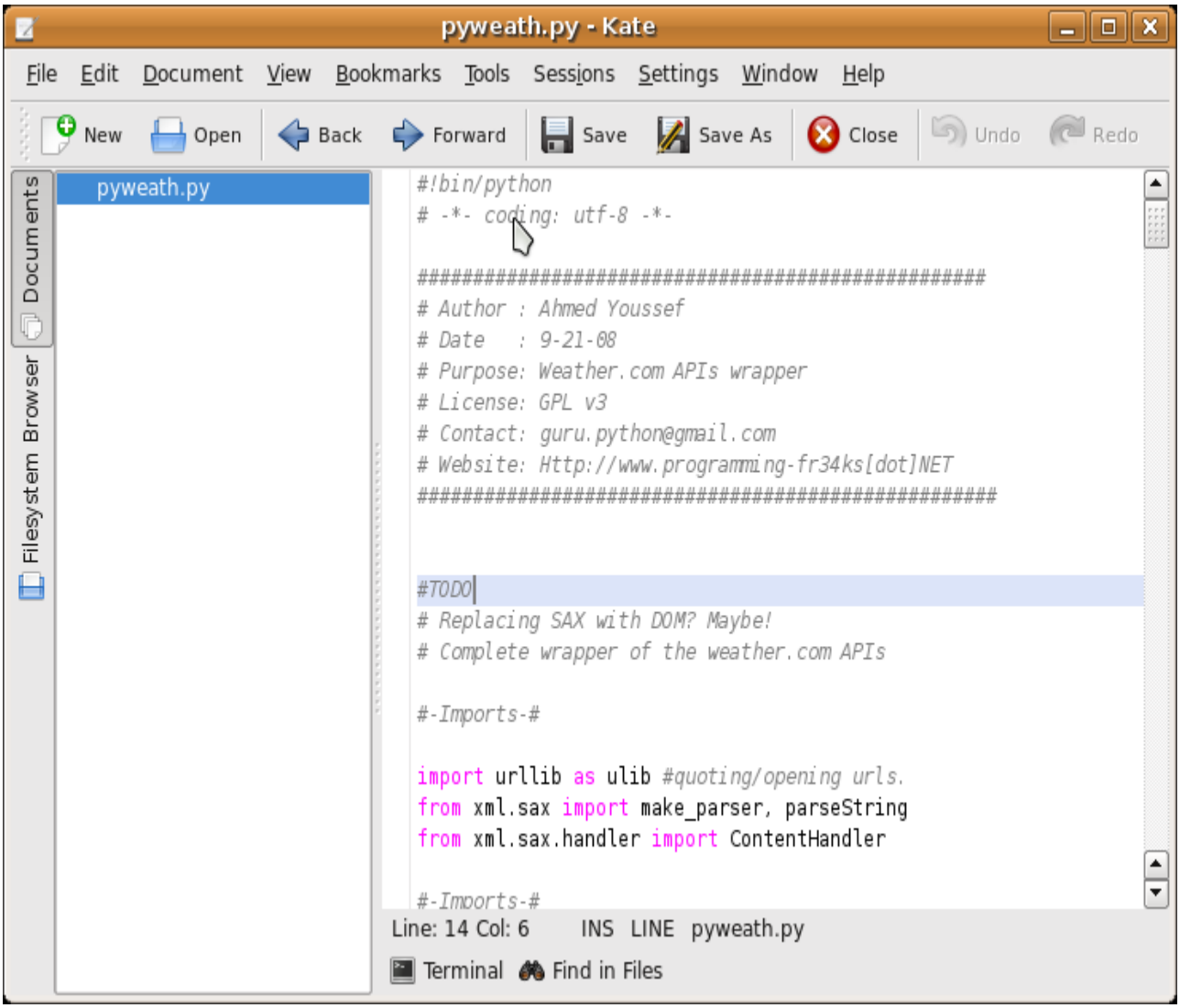
*Gedit(Gnome Editor)*

محرر افتراضى لواجهة Gnome بيدعم العديد من الإضافات كتلوين الكود المصدرى والإكمال التلقائى والترقيم وغير عدد كبير من الإضافات



*Kate(KDE Advanced Editor)*

محرر افتراضى لواجهة KDE مميزات مثل سابقه




*Komodo*

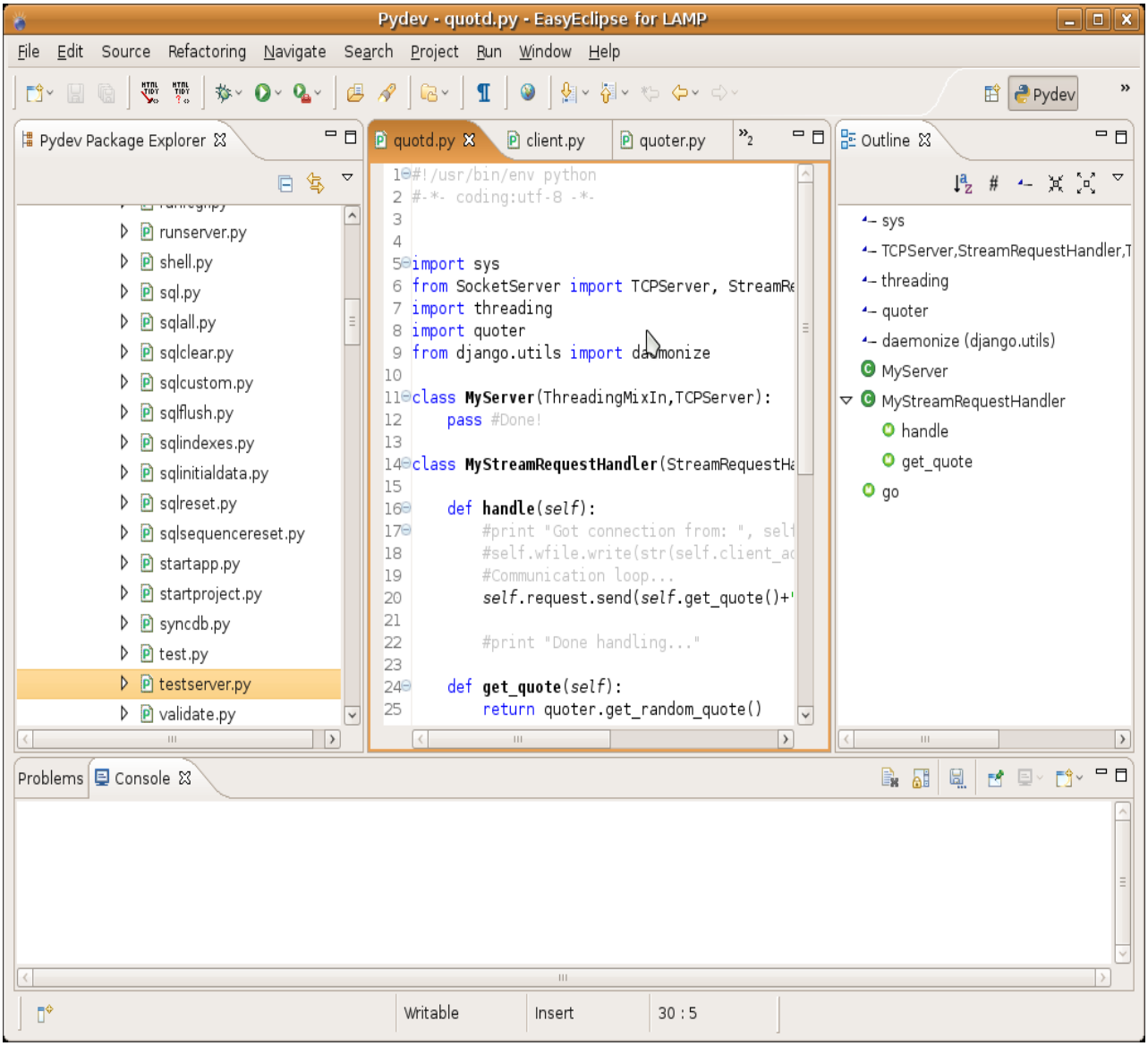
*IDE* غير مجانية

*OpenKomodo(KomodoEdit)*

مبنية على المكونات المجانية المتاحة من Komodo

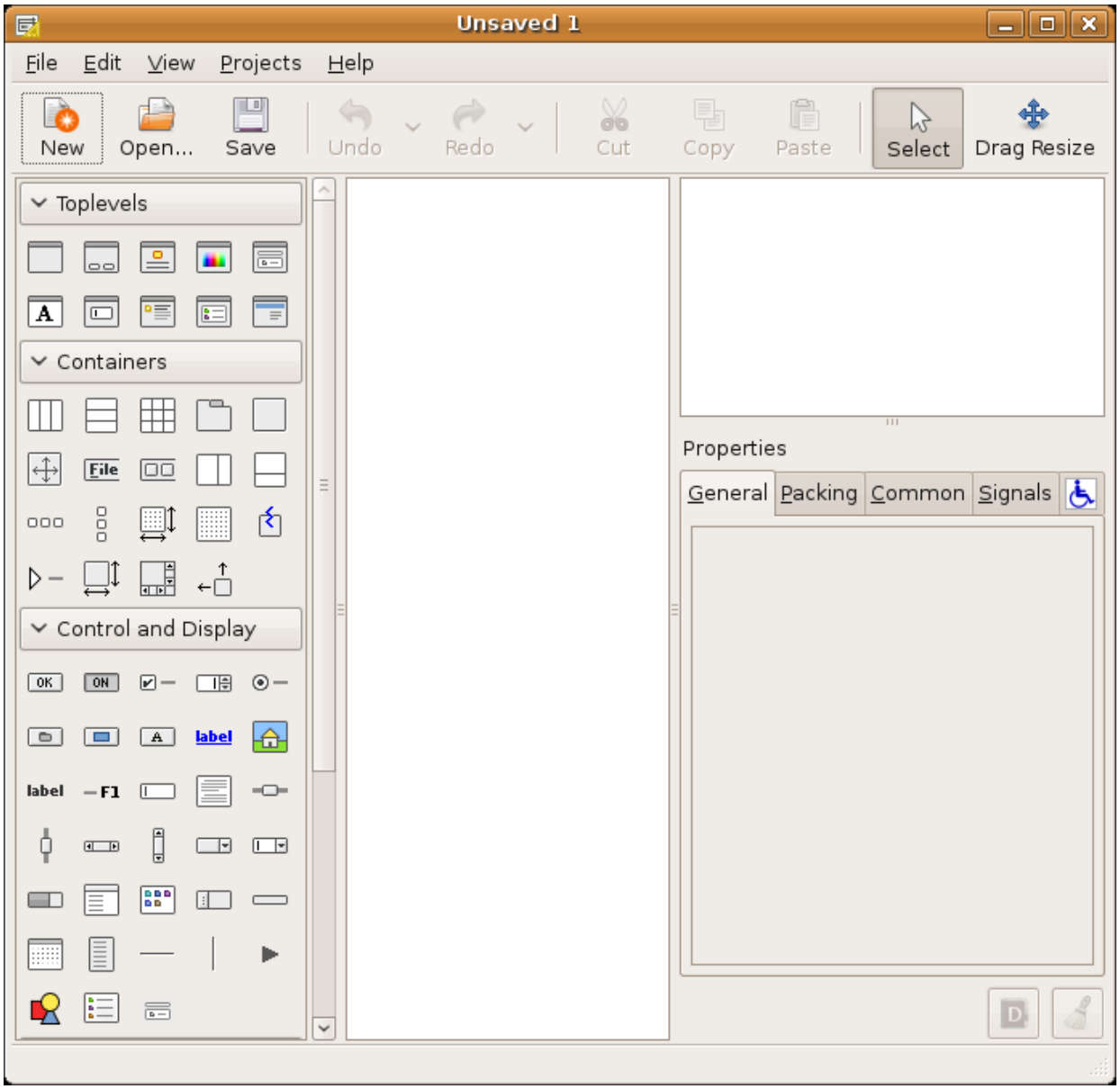
*PyDev*
*http://pydev.sourceforge.net/*

التعامل على حجم اكبر من احتياجات اكثر من ال refractoring و quick fixes و غيرها..



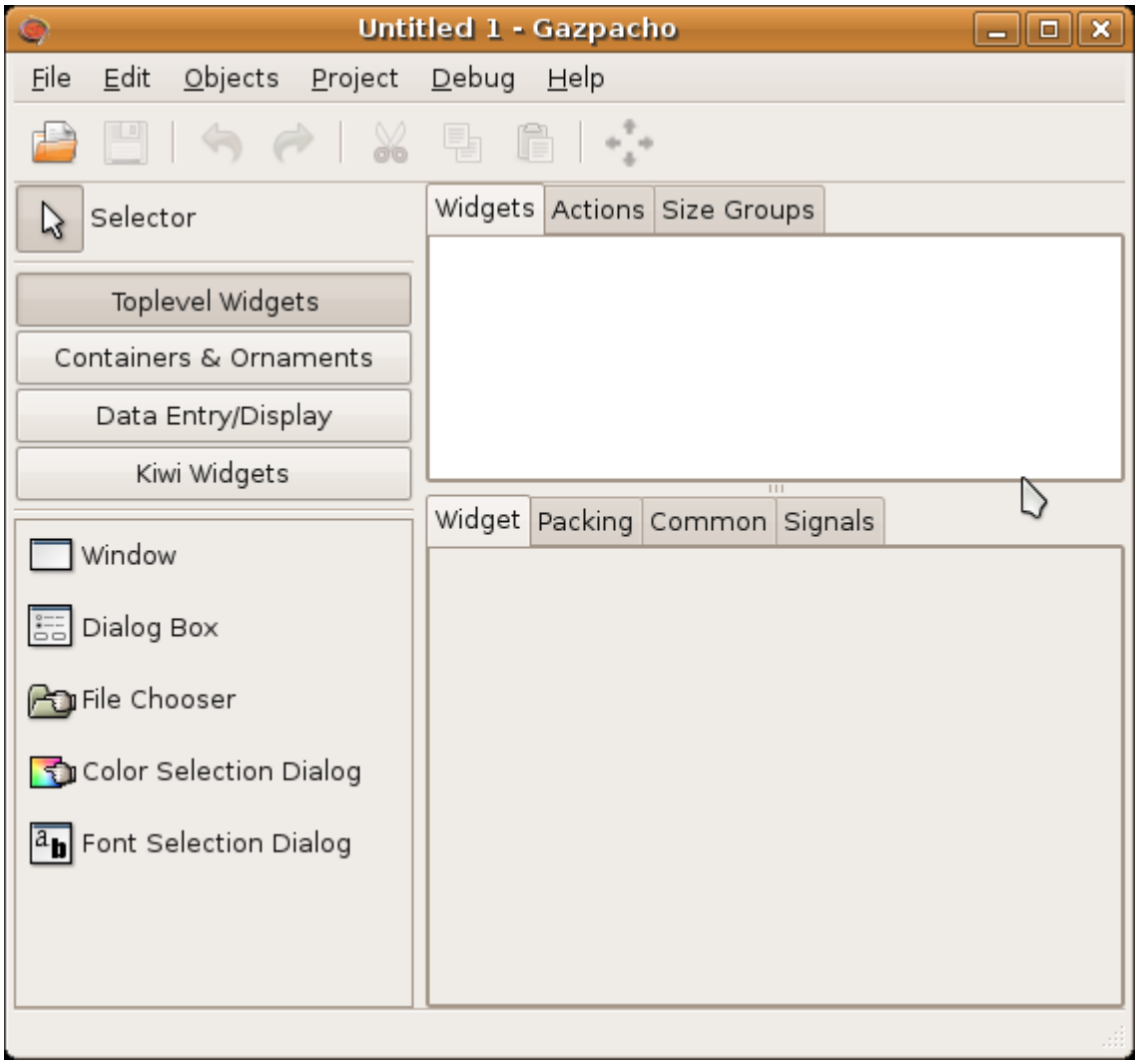
*Glade*

مصمم للواجهات مختص ب GTK ذكرنا كيفية استخدامه فى Gqamoos

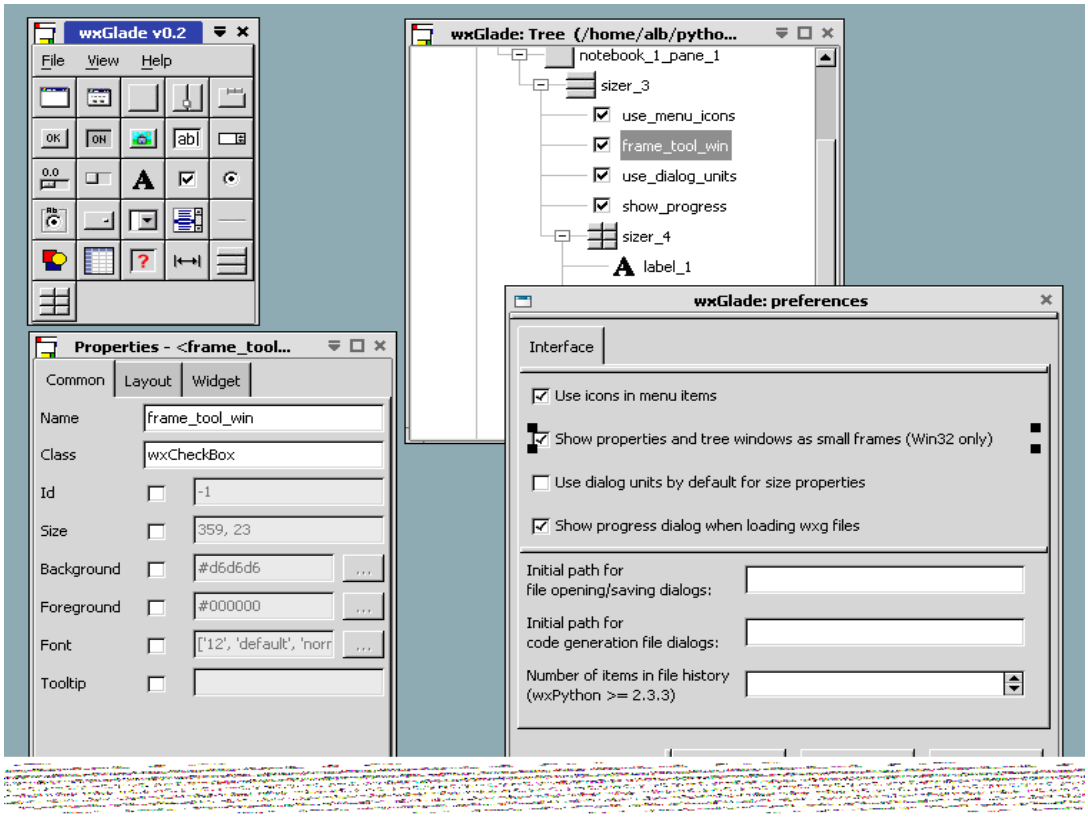


*Gazpacho*
*http://gazpacho.sicem.biz/*

الجيل الجديد من glade مكتوب بيثون و gtk



*wxGlade*
*http://wxglade.sourceforge.net/*

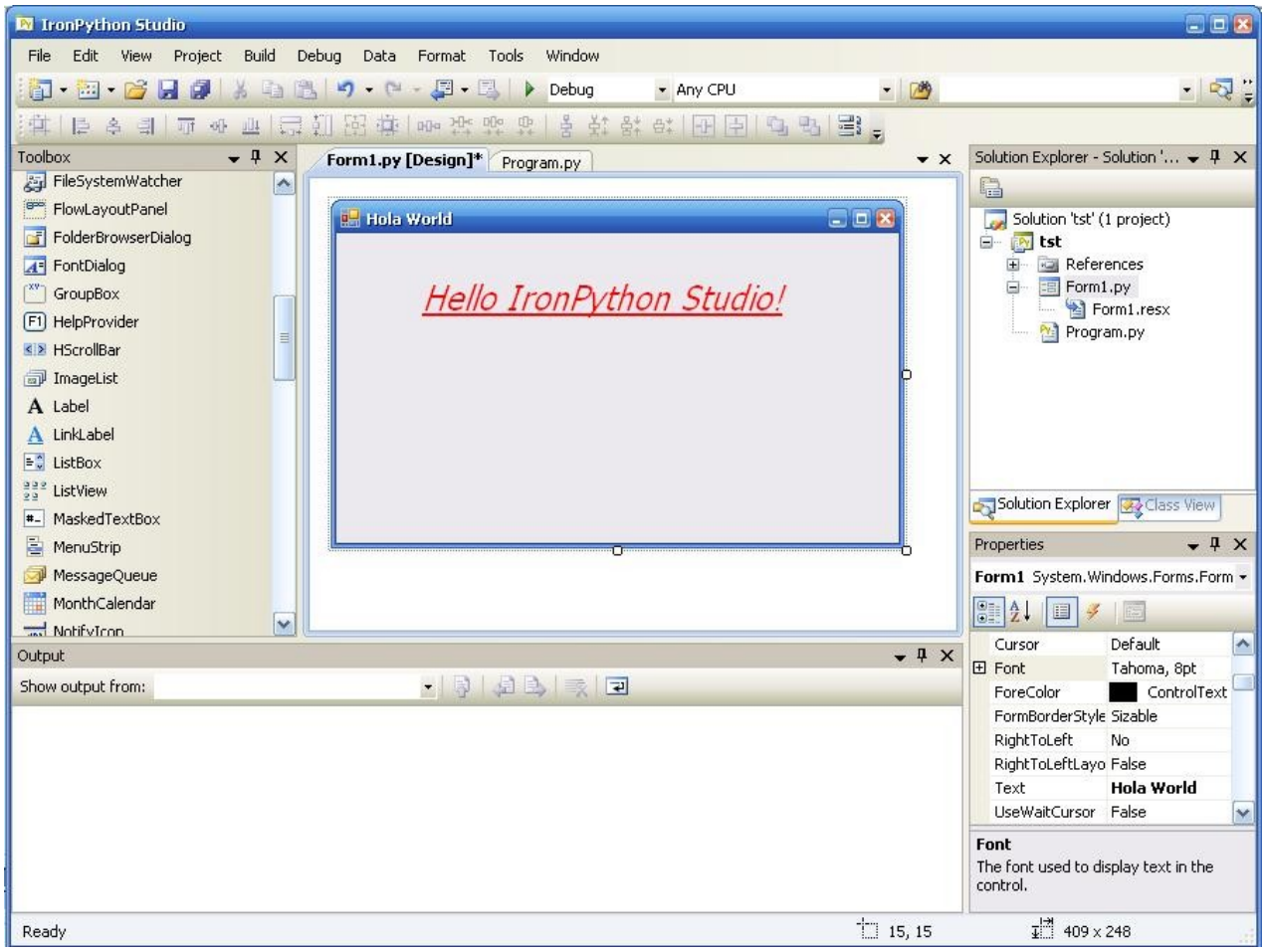



الصورة مأخوذة من الموقع الرسمى
مشابه ل Glade ولكن ل wx

*BoaConstructor*
*http://boa-constructor.sourceforge.net/*

*IDE* متكاملة محرر نصوص ومصمم واجهات (مشابهه لدلفى) .. الخ الخ
للأسف المشروع شبة متوقف

*IronPython Studio*




IronPython Studio هى IDE متكاملة ومجانية للغة بايثون مبنية على Visual Studio 2008 Shell runtime

http://www.codeplex.com/IronPythonStudio

Quick installation guide:

1- قم بتحميل وتستيب Visual Studio 2008 Shell Isolated Mode Redistributable package
2- اذهب الى المجلد "C:\VS 2008 Shell Redist\Isolated Mode"
3- شغل الملف "vs_shell_isolated.enu.exe"
4- حمل وستب IronPython Studio