作者的 FB https://www.facebook.com/ExpandeScientia/ 寫點科普，請給指教。

一小时 Python 入门-part 1

Python 是一个简洁易读的语言，学习者几乎可以立刻上手，也适用于大量的商业应用上。目前已超越 C/C++、Java，成为各大学课程中的主流入门程序语言，美国 Top 10 Computer Science (计算机科学) 系所中便有 8 所采用 Python 作为入门语言。

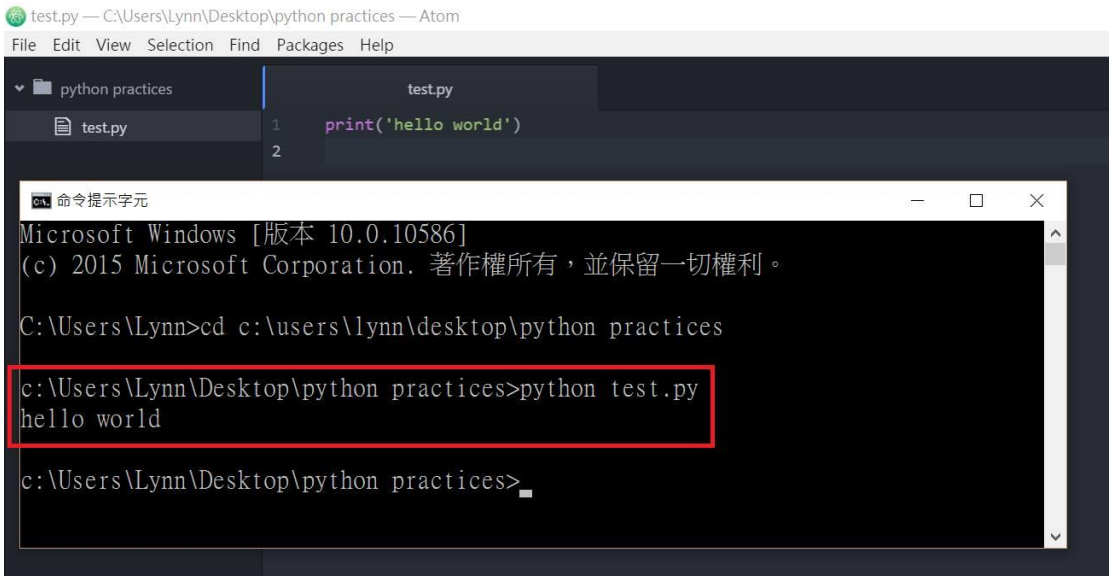
今天就让我们来试着写写看 Python 这个好玩的语言，并利用学习结果写个简单的爬虫程序，能够在 FB 自动发文、自动洗赞，统计出朋友的留言按赞数!

环境设置

首先可以至官网下载下载 Python: https://www.python.org/ 。下载完成后，你将获得: Python 直译器与 IDLE (Python 的 IDE)。身为一个 IDE、该有的都有了，只是由于有点丑，所以大家都不爱用。接下来如何执行程序有两个选项:

1. 使用主机 Terminal

让我们找一个写程序用的文本编辑器: Notepad++、Sublime Text、Vim、Atom…记事本也行，写完程序代码后可以打开 Terminal 、打上 python 檔名.py 执行看看，不过此举较不推荐新手用。



可以看到 Terminal 印出 hello world 字样了

2. 下载使用 Jupyter Notebook

如果要更方便一点的话可以下载 Jupyter，也是 Python 的 IDE，包含了编辑器和编译功能，非常适合 Python 的学习噢！安装方式请参考：
http://www.largitdata.com/course/29……

Python 基本语法与练习

好了，来实作看看吧！打开 Jupyter Notebook 应用程序、跳出 Jupyter Notebook 的编辑页面就可以来撰写我们的程序代码啦。

1. Print：永远是跟世界打声招呼的第一个程序

print 语法是在 print 函式中放进你要印的东西，比如 print(我要印这个)。说声 Hello World，跟程序的世界打声招呼吧！

```
In [5]: print('hello world')

hello world
```

第一个跟世界打招呼的程序!

print 的内容可以是变量(variable)：print(a)，数字(init)：print(1234)，或是字符串(string)：print(‘abc123’)。

2. Variable (变数)：一切 TypeError 的泉源

上面提到了变量，但这是什么意思呢？其实就是帮数据取个名字、把数据储存起来。比如：变量 = 数据，a = 123。当我们输入 a、就会找到 123。

变量虽然能储存数据，但数据有很多种类型、所以就会有数据型态(Type)。举例来说：

- a = 123 , type(a)会告诉我们 init (整数)。
- b = '456' , type(b)会告诉我们 str (字符串)。
- c = 8.70 , type(c)会告诉我们这是一个 float (浮点数)。

```
In [6]: a = 123
        type(a)

Out[6]: int

In [7]: b = '456'
        type(b)

Out[7]: str

In [8]: c = 8.70
        type(c)

Out[8]: float
```

数值可以做运算：

```
In [10]: a = 5
         b = 8
         print (a + b)
         print (a - b)
         print (a * b)
         print (a / b) #數學除法
         print (a // b) #無條件捨去除法
         print (a ** b) # a 的 b 次方
```

```
13
-3
40
0.625
0
390625
```

也可以使用各种你知道的函数：

```
In [11]: a = -3
         b = 4
         print(abs(a)) # a的絕對值
         print(max(a, b)) # a, b的最大值
         print(min(a,b)) # a, b的最小值
```

```
3
4
-3
```

但不同的形态运算要注意，不要把字符串和数值加在一起、否则会产生TypeError。

```
In [12]: a = 123
         b = '456'
         print(a+b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-2946636f9fe8> in <module>()
      1 a = 123
      2 b = '456'
----> 3 print(a+b)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

记得学会看错误讯息再回去看自己写错了什么。

那该怎么办呢?

别怕别怕，若要运算则需要做型态转换，可以用 int(), str(), float() 来转换变量型态。可以看到下图，int() 将字符串’456’转成整数、str() 将整数 123 转成字符串。

```
In [13]: a = 123
         b = '456'
         print(a+int(b))
         print(str(a)+b)

         579
         123456
```

谨记：「Error 都是暂时的，没有解不掉的 Bug，莫忘心灵祥和。」

就算身边没有程序激励员也要保持好心情喔(啾咪^_<

3. Input：让用户输入数据呗

这边我们使用的是 Python 3 的环境，使用的是 input；如果环境是 Python 2 的朋友注意是用 raw_input 噢，Python 2 的 input 有别的意义。

input 的功能是让用户所有输入的数据会被当成字符串储存起来，需要时再作转型。举例而言：

- a = input()：它会等你输入，并以字符串(string)的形式存入 a 这个变数中。
- b = input('Your Name: ')：在屏幕上会出现 'Your Name: '的提示，并把数据以字符串(string)形式存入 b 这个变量中。

```
In [15]: a = input()
         print('a = ', a)

         b = input('Your Name: ')
         print('Your Name: ', b)
```

```
289
a =  289
Your Name: Lynn Chen
Your Name:  Lynn Chen
```

记得这个数字 289 是字符串的形式喔~

● 小练习

请写一支 Python 程序，能读取两个整数，并把他们的四则运算印出来。(提示：用 input()印出来的东西是字符串，记得转型噢!)

```
In [14]: a = int(input('整數1 = '))
         b = int(input('整數2 = '))
         print(a+b)
         print(a-b)
         print(a*b)
         print(a/b)
```

```
整數1 = 10
整數2 = 101
111
-91
1010
0.09900990099009901
```

4. If-Else 日常的逻辑判断

别说你没用过 if-else 函数式，我们每天都在面临 if else 的抉择——如果(if) 百货公司周年庆到了：那就是大采购的时间啦！否则(else)我们就晚点再买。

照样造句一下还有——如果 (if)手机快没电：我就乖乖回家充电！否则 (else)我就继续抓宝可梦。

来看一下 if-else 语法解说。if 写完之后接冒号、下一行缩排后开始写要做什么事情。注意 else 是否则的意思，就不用再写一次条件了。

```
In [2]: #我是PYTHON寫註解的方式~ 電腦不會讀到註解掉的地方

if 條件: #有些語言會用括號, Python使用冒號
    做什麼  #記得縮排, 可以按 tab, 或直接空四行
    繼續做什麼
else: #else後面不能再寫條件
    做什麼
```

但如果我们不只有两层（周年庆就去买东西、否则就等下次），可能有好几层条件呢?

比如：如果我们平板电池的电量还很高、就去抓宝可梦；电池电量快没了、就乖乖回家；如果电池电量普普通通、看看小说就好。

把这件事情用 Python 表达出来吧！

```
In [4]: battery = 50
if battery > 80:
    print('來去抓寶可夢囉')
elif battery < 30:
    print('乖乖回家吧')
else: #30~80之間 #記得else後面還是不能放條件噢!
    print('看看小說就好')
```

```
看看小說就好
```

电池只有 50，在 30~80 之间，因此看看小说就好！

if-else 主要用在逻辑运算的判断上：

- > 大于
- < 小于
- >= 大于等于
- <= 小于等于
- == 等于
- !== 不等于

或是布尔值(Boolean)：

- true 真的
- false 假的
- and 且
- or 或
- not 非

不小心买到 Samsung Note 7、听说电池快没电时容易有爆炸的风险，怎么办?! 利用逻辑运算来提醒我们这个危机吧！

In [5]:
```
phone = 'iPhone'
battery = 5
if phone == 'Samsung Note 7' and battery < 10: #邏輯判斷的等於要用 == 兩個等於
    print('要爆炸了，塊陶啊~~~')
elif phone != 'Samsung Note 7' and battery < 10:
    print('不是Note7就好，記得充電喲寶貝')
elif not phone == 'Samsung Note 7' or battery > 90:
    print('不是Note7或電池飽飽的，沒有爆炸的危機!')
else:
    print('我也不知道你發生什麼事了')
```

```
不是Note7就好，記得充電喲寶貝
```

● 小练习

写一支 Python 整数机，第一步让用户输入想要做的符号运算，比如「+，-，*，/」，第二步让使用者输入’整数 1’和 ‘整数 2’，最后让这两个整数进行运算。如果输入的运算符号不是「+，-，*，/」，便输出「错误」。

In [1]:
```
x = input('運算符號: ')
a = int(input('整數1: '))
b = int(input('整數2: '))

if x == ('+'):
    print(a+b)
elif x == ('-'):
    print(a-b)
elif x == ('*'):
    print(a*b)
elif x == ('/'):
    print(a/b)
else:
    print('錯誤')
```

```
運算符號: *
整數1: 11
整數2: 23
253
```

5. For Loop 转圈圈吧哈姆太郎

如果要把 0 到 9 的数字印出来，应该怎么做呢?

嗯当然你可以这样做…

```
In [2]: print(0)
        print(1)
        print(2)
        print(3)
        print(4)
        print(5)
        print(6)
        print(7)
        print(8)
        print(9)
```

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

但如果今天是要从 0 印到 100 呢？（´・ω・`）…

我们之所以写程序，就是为了要让程序能自动化地帮忙做一些事情，这样显得太没有效率了。因此在这边我们需要写一个「循环」(Loop)。首先来介绍一个最基础的 Python 循环「For Loop」。

比起痛苦地写十行 print，引进 For 循环后、区区两行便能解决了：

```
In [4]: for i in range(0,10):
            print(i)

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

不是很看得懂程序代码的意思吗？别紧张，这边来解释一下这个式子的涵义。若要枚举出从 0 到 n-1 的数字，写法是：

for [变量名称] in range(n): (缩排) print([变量名称])

来试试看：

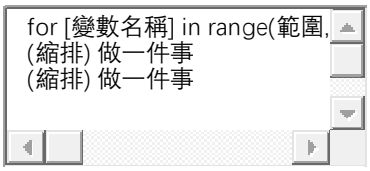
```
In [5]: for i in range(10):
            print(i)

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

在这里的 n=10，是不是把 0~9 (n-1)的数字都印出来了呢？

到底 range 又是什么意思呢？

事实上 range(n) = range(0, n)，简单来说就是个产生器，你可以在这个 range 范围中拿数据、拿完到 n-1 为止。以 range(3)为例，第一次拿到资料 0、第二次拿到资料 1、第三次拿到数据 2，然后就结束了。

让我们重新看一下 for 函数结构：



1 for [变量名称] in range(范围, 比如从数字 X 到 Y):
2 (缩排) 做一件事
3 (缩排) 做一件事

了解的话，就让我们来做个小练习吧！

● 小练习

请输出一个九九表。(提示 1: for loop 里面也可以有第二个 for loop 噢!)

In [31]:
```
for i in range(1, 10):
    for j in range(1, 10):
        print(i*j)
```

```
1
2
3
4
5
6
7
8
9
2
4
6
8
10
12
14
16
18
```

进阶版：如果不希望 print 出来的结果换行且要有空格，可以使用 end=’ ’；若要再换行可使用 end=’\n’。

In [3]:
```
for i in range(1, 10):
    for j in range(1, 10):
        print(i*j, end=' ')
    print(end='\n')
```

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

事实上 range 的数列是有间距的喔！

range 的结构是：range(起点, 终点, 间距)，其中的间距预设为 1 、比如 range(0, 6) = range(0, 6, 1) 。代表从 0 到 5，每次加 1 的意思。

分别印出：

(1) 从 1 到 10、每次加 2 的数列；

(2) 从 10 到 2、每次-3 的数列 吧！

```
In [34]: for i in range(1, 10, 2):
             print(i)

1
3
5
7
9
```

```
In [36]: for j in range(10, 1, -3):
             print(j)

10
7
4
```

● 小练习

请做出一支能猜数字的程序：每次让使用者猜一个整数，若猜对就输出 Bingo；使用者最多可以猜 3 次。(提示：Bingo 后可以使用 break 来离开循环)

In [53]:
```
ans = 29  #猜數字的解答

for guessChance in range(0, 3):
    guess = int(input())
    if ans == guess:
        print('Bingo!')
        break
    else:
        print('猜錯了')
print('遊戲GG囉')
```

```
23
猜錯了
4
猜錯了
17
猜錯了
遊戲GG囉
```

6. List：能无限储存数据的格子

想象一下，若今天有三笔数据要计算时，我们会这样写：

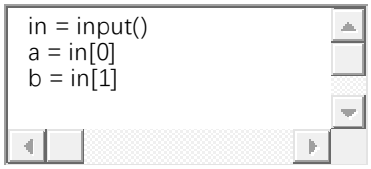
1 a = 1

2 b = 2

3 c = 3

… 那如果你有成千上万笔数据呢?

再想象一下，如果不知道用户要输入多少数据又要怎么办呢?




```
1 in = input()
2 a = in[0]
3 b = in[1]
```

这样真的行吗…

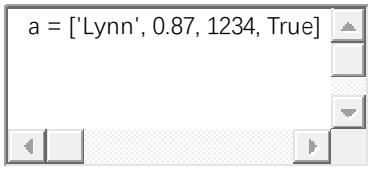
「神说要有了 List，所以有了 List。」- 来自深海大菠萝里的神秘讯息。

继续运用你的想象力！我们现在有个异次元里的柜子，它有无限个格子，第一格可以放东西、第二格也可以放东西…以此类推。

柜子里可以放任何东西，我们会用一个数字代表东西放在柜子中的第几层格子，这个数字就叫做 Index（索引）。

把柜子的概念换成 Python 的 List 语法，就会变成：放在第 0 层的变数、放在第 1 层的变量… 这个数字就是 Index(索引)，而变量就是我们塞进去的数据。

一个 Python 的 List 长相如下，以中括号[] 包起来、逗号，分隔：

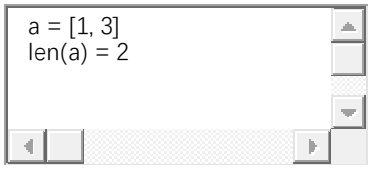



```
1 a = ['Lynn', 0.87, 1234, True]
```

这里的 a 就是一个 List，第 0 格放的是 Lynn 这个字符串、第 1 格放的是 0.87 这个浮点数、第 2 格放的是 1234 这个整数、第 3 格放的是 True 这个布尔值。

有了格子的概念后，我们可以使用 **a[第几格]** 来取出变量，比如 a[0] 代表’Lynn’、a[3] 代表 True。可以看到由于 a[4] 并不存在，因此会出现 IndexError 的警示。

要如何知道一个 List 有多长呢？我们能使用 **len([list 变量名称])**。



1 a = [1, 3]
2 len(a) = 2

在拿到 List 的长度后，让我们能来玩一点变化吧！比如 List 加上上面教学的 For Loop 看看会发生什么事：

```
In [6]: a = ['Lynn', 0.87, 1234, True]
        for i in range(0, len(a)):
            print(a[i])

Lynn
0.87
1234
True
```

若要更简洁一点的话，由于 in 这个关键词有「在… 之中」的意思，in range(0, 3)代表在 0~2 之中。因此能改成：

```
In [8]: a = ['Lynn', 0.87, 1234, True]
        for i in a:
            print(i)

Lynn
0.87
1234
True
```

● 小练习

现在有一个 list a = [1, 3, 5, 7, 9]，请对每一个元素都平方后印出来，且须将 a 也变成 [1, 9, 25, 49, 81]。

In [22]:
```
a = [1, 3, 5, 7, 9]
for i in range(0, len(a)):
    print(a[i]*a[i])
    a[i] = a[i]*a[i]
print(a)
```

```
1
9
25
49
81
[1, 9, 25, 49, 81]
```

来玩玩更多 list 的操作吧！

- 建立一个空的 list: a = list() 或是 a = []都可以
- 动态增加元素：
  - list.append(x): 把变数 x 塞到 list 的最后面
  - list.insert(i, x): 把变量 x 塞到 i 这个位置上
  - list.pop(): 把 list 的最后一格丢掉
  - list.pop(i): 把 list 的第 i 格丢掉
  - list.remove(x): 会把第一个出现的变量 x 拿掉
  - list.clear(): 把 list 内的资料全部清光光
- 与常见函数的结合：
  - max(list): 找出 list 中最大值
  - min(list): 找出 list 中最小值
  - sum(list): 找出 list 数字总和

In [27]:
```
#來舉辦一場眾星雲集的演唱會吧
singers = ['Justin Bieber', 'Taylor Swift', 'Lady Gaga', 'Katy Perry']
singers.append('Sia') #Sia來唱水晶燈壓軸
print(singers)

singers.insert(2, 'Britney Spears') #增加小甜甜布蘭妮在第三位跳舞
print(singers)
```

```
['Justin Bieber', 'Taylor Swift', 'Lady Gaga', 'Katy Perry', 'Sia']
['Justin Bieber', 'Taylor Swift', 'Britney Spears', 'Lady Gaga', 'Katy Perry', 'Sia']
```

In [29]:
```
#新電影選卡司囉
cast = ['Colin Firth', 'Christian Bale', 'Angelina Jolie', 'Anne Hathaway']
cast.pop() #小安演太爛只好踢掉
print(cast)

cast.pop(2) #裘莉離婚去了只好踢掉
print(cast) #女演員都沒了QQ
```

```
['Colin Firth', 'Christian Bale', 'Angelina Jolie']
['Colin Firth', 'Christian Bale']
```

In [32]:
```
sweet_dinner = ['小提琴', '妹子', '牛排', '妹子', '蠟燭']
sweet_dinner.remove('妹子') #第一個妹子走了
print(sweet_dinner)
sweet_dinner.remove('妹子') #第二個妹子也走了
print(sweet_dinner)

#乾~都沒妹子了還要吃嗎QQ 收東西洗洗睡啦
sweet_dinner.clear()
print(sweet_dinner)
```

```
['小提琴', '牛排', '妹子', '蠟燭']
['小提琴', '牛排', '蠟燭']
[]
```

- 小练习

五次数学段考的成绩分别为 10、30、50、70、90 分，算出平均后，老师发现大家考太烂、只好将成绩开根号再乘上 10（提示：成绩0.5*10），再算出一个新平均。**

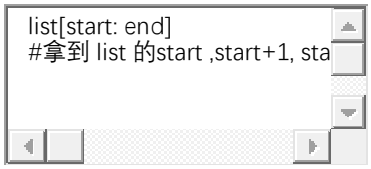
请印出：1. 五次成绩；2. 平均成绩；3. 五次新成绩；4. 新分数的平均。

In [67]:
```
math_grades = [10, 30, 50, 70, 90]
print('五次成績: ', end='')
for grade in math_grades:
    print(grade, end=' ')
print(end='\n')
average = sum(math_grades)/len(math_grades)
print('平均成績:', average)

print('新成績:', end='')
for grade in range(0, len(math_grades)):
    math_grades[grade] = math_grades[grade]**0.5*10
    print(math_grades[grade], end=' ')
print(end='\n')
new_average = sum(math_grades)/len(math_grades)
print('新平均成績:', new_average)
```

```
五次成績: 10 30 50 70 90
平均成績: 50.0
新成績:31.622776601683796 54.772255750516614 70.71067811865476 83.66600265340756 94.86832980505139
新平均成績: 67.12800858586283
```

以为就这样结束了吗？List 还有切片（Slice） 的功能哦大大！

Python 的 slice 功能让我们能拿出 List 柜子的一部份。



1 list[start: end]
2 #拿到 list 的 start ,start+1, start+2, ..., end-2, end-1

```
In [71]: greeting = ['Hi', 'Hello', 'Hey', 'Yo', 'Howdy', 'Morning']
         print(greeting[0:5])
         print(greeting[2:4])

         ['Hi', 'Hello', 'Hey', 'Yo', 'Howdy']
         ['Hey', 'Yo']
```

针对 slice 语法，让我们深入介绍一些细节部分：

- list[start: end]， start 和 end 都可以省略不写
- start 的预设为 0
- end 的预设为 len(list)
- liest[:end]代表 0~end-1
- list[start:]代表 start~len(list)-1
- list[:]代表 0~len(list)-1

```
In [72]: greeting = ['Hi', 'Hello', 'Hey', 'Yo', 'Howdy', 'Morning']
         print(greeting[ :4])
         print(greeting[2: ])
         print(greeting[ : ])

         ['Hi', 'Hello', 'Hey', 'Yo']
         ['Hey', 'Yo', 'Howdy', 'Morning']
         ['Hi', 'Hello', 'Hey', 'Yo', 'Howdy', 'Morning']
```

● 小练习

来办场 Party 吧！输入十个整数、存入一个名为 people 清单中（表示我们的宴客人数)；之后会有五次询问，每次会输入列表开始和结束的位置，再输出从开始到结束位置的总和。

In [1]:

```
people=[]
for i in range(0, 10):
    i = int(input('輸入數字: '))
    people.append(i)
print(people)
print(end='\n')

for questions in range(0, 5):
    a = int(input('頭在哪裡: '))
    b = int(input('尾在哪裡: '))
    print('總和: ', sum(people[a:b]))
    print(end='\n')
```

```
輸入數字: 2
輸入數字: 3
輸入數字: 6
輸入數字: 8
輸入數字: 6
輸入數字: 3
輸入數字: 5
輸入數字: 8
輸入數字: 1
輸入數字: 2
[2, 3, 6, 8, 6, 3, 5, 8, 1, 2]

頭在哪裡: 0
尾在哪裡: 6
總和:  28

頭在哪裡: 1
尾在哪裡: 5
總和:  23

頭在哪裡: 8
尾在哪裡: 11
總和:  3

頭在哪裡: 3
尾在哪裡: 4
總和:  8

頭在哪裡: 2
```

还没结束喔！

一小时 Python 入门-part 2

延续上一篇【一小时 Python 入门-part 1】，让我们接下去介绍 Python 的 dict 功能吧！

7. Dict 一个靠卷标能查到东西的字典

在上一部分中我们提到，List 能动态增加长度、塞任何型态的数据（字符串、整数、浮点数），并能透过 Index 来取得一层层柜子中的数据。

但如果卷标不是按照柜子顺序行吗？比如给各个数据一个专属的卷标名称、再根据卷标名称取出数据？

再一次运用想象力，我们有另外一种柜子，上面会贴有标签名称，比如放在「冰箱」标签中的东西，或是放在「抽屉」标签中。

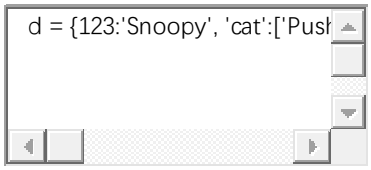
这个标签代表着东西放在哪边，称为 key（键值）。标签可以是任何东西、里面什么都能放。

把这个概念移到 Python 中的 Dict 语法吧！

Dict 其实就是 Dictionary（字典）的意思，我们有可以放在 key 是 'box' 里的变量、也可以有放在 key 是 123 里的变量，key 只能是不能修改的型态，包括字符串(str)、整数(int)与浮点数(float)。

变量则和 List 一样什么都能放，比如变量可以是一个 List。

来看看一个 dict 语法吧，dict 由大括号{}包住、元素以 key:value 的 key-value pair 组成（key 后面接冒号和变数 value），并以由逗点，隔开。




```
1 d = {123:'Snoopy', 'cat':['Pusheen', 'Kitty']}
```

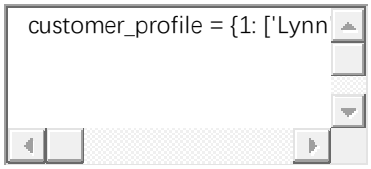
注意：dict 在输出时并不保证 key 的顺序。

```
In [2]: d = {'one':1, 'two':2, 'three':3}
        print(d)

        {'one': 1, 'three': 3, 'two': 2}
```

我们一样能用 len(dict) 来找出 dict 的大小：

```
In [4]: profile = {'name': 'Lynn', 'birth': 1205, 'id': 235}
        print(len(profile))

        3
```

但如果 dict 中有 list 呢？长度会变成多少？



1 customer_profile = {1: ['Lynn', 1205, 'college'], 'id': 235}

印出来就知道啦。事实上是一样的，我们只看 key 的数量。

```
In [5]: customer_profile = {1: ['Lynn', 1205, 'college'], 'id': 235}
        print(len(customer_profile))

        2
```

另外，类似 list 可以用 index 来取值、比如 list[index]，dict 同样能用 key 来取值。当取到不存在的 key 时，相对于 list 中的 IndexError，dict 会显示 KeyError：

```
In [7]: customer_profile = {1: ['Lynn', 1205, 'college'], 'id': 235}
        print(customer_profile[1])
        print(customer_profile['id'])
        print(customer_profile['hey'])

        ['Lynn', 1205, 'college']
        235

        ---------------------------------------------------------------------------
        KeyError                                  Traceback (most recent call last)
        <ipython-input-7-d966e968c6ba> in <module>()
              2 print(customer_profile[1])
              3 print(customer_profile['id'])
        ----> 4 print(customer_profile['hey'])

        KeyError: 'hey'
```

这边来介绍一下 dict 的语法：

- 建立与删除
  - dict={}: 建立空的 dict

    - del dict[key]: 删除特定的 key-value pari
- 增加与更新
    - dict[key]=value: 如果 key 不存在，会增加这组 K-V；如果 key 已存在，会更新这组 K-V。
- for … in
    - for [变量名称] in dict:每一回合会拿到一把在 dict 里的 key, 就可以用 dict[key] 拿到对应的变量。

● 小练习

让使用者在名为 table 的 dict 中，输入五组的 key-value pair，key 是字符串、value 是数字，最后把这个 dict 印出来。(提示：dict 没有 append，直接用增加或更新(InsertOrUpdate)的方式)

In [2]:
```
table = {}

for inputTimes in range(0, 5):
    k = input('輸入字串: ')
    v = int(input('輸入整數: '))
    table[k] = v
    print(end='\n')
print(table)
```

```
輸入字串: eat
輸入整數: 24

輸入字串: 3
輸入整數: 17

輸入字串: Lynn
輸入整數: 19

輸入字串: hello_world
輸入整數: 265

輸入字串: 5578
輸入整數: 201

{'3': 17, 'Lynn': 19, '5578': 201, 'hello_world': 265, 'eat': 24}
```

除了用 key 找 value，我们也可以反查，用 value 找出相对应的 key：

```
In [8]: d = {'one': 1, 'two': 2, 'three':3, 'four':4, 'five':5}
target = 2
for key in d:
    if d[key] == target:
        print('我找到了! key = ', key)
        break
    else:
        print('繼續找吧孩子...')
```

```
我找到了! key =  two
```

或是检查 dict 中有没有包含特定的 key：

```
In [9]: d = {'one': 1, 'two': 2, 'three':3, 'four':4, 'five':5}
print('one' in d)
print(1 in d)
```

```
True
False
```

若想拿到一个由所有 key 组成的 list、或一个由所有 value 组成的 list，可以使用 dict.keys()或是 dict.values()。结合反查的方法，可以检查一个特定的 key 或 value 在这个 dict 中存不存在。

```
In [10]: d = {'one': 1, 'two': 2, 'three':3, 'four':4, 'five':5}
print(0 in d.keys())
print(1 in d.values())
print('one' in d.keys())
print(9 in d.values())
```

```
False
True
True
False
```

那如果有了 key，想要把 dict 中相对应的数据取出来应该要怎么做呢？作法有两个：

- d[key]: 这个做法相对不安全，key 如果不存在的话就会出现 KeyError (如同上面我们所示范的)。
- d.get(key, default_value): 是比较安全的作法，如果 key 不存在的话就会回传 default_value (下面例子中的 deafult_value 就是'找不到耶')。

```
In [11]: d = {'one': 1, 'two': 2, 'three':3, 'four':4, 'five':5}
         print(d['one'])
         print(d.get('two', '找不到耶'))
         print(d.get(2, '找不到耶'))
```

```
1
2
找不到耶
```

- 小练习

一年一度的世界歌王大赛来啦！身为评审的你，请输入五个歌手的名字与成绩。接下来会有五次查询的机会，每次查询都可以让观众输入一个名字、来查看心爱的歌手。

如果歌手不在名单中，请输出’这个人没参赛哟’；如果歌手在名单中，请输出该歌手的名字与成绩。

```
In [1]: competition = {}
        for judge in range(0, 5):
            name = input('歌手名: ')
            grade = int(input('成績: '))
            competition[name] = grade
            print(end='\n')
        print(competition)
        print(end='\n')

        for audience in range(0, 5):
            query = input('你要查誰呢: ')
            print(competition.get(query, '這個人沒參賽喲'))
            print(end='\n')
```

```
歌手名: Rihanna
成績: 84

歌手名: Adele
成績: 92

歌手名: Taylor Swift
成績: 70

歌手名: Ed Sheeran
成績: 86

歌手名: Adam Lambert
成績: 82

{'Rihanna': 84, 'Taylor Swift': 70, 'Adele': 92, 'Ed Sheeran': 86, 'Adam Lambert': 82}

你要查誰呢: Adele
92

你要查誰呢: Katy Perry
這個人沒參賽喲

你要查誰呢: Adam Lambert
```

最终专案：FB 狂赞士

呼~好不容易走到这一步了，有没有很兴奋呢!!!!!! 接下来我们将开始撰写人生中第一个爬虫程序啦~

首先我们得到 Facebook Developer(开发者中心) 网站上点选工具及支持，进入 Graph API Explorer。

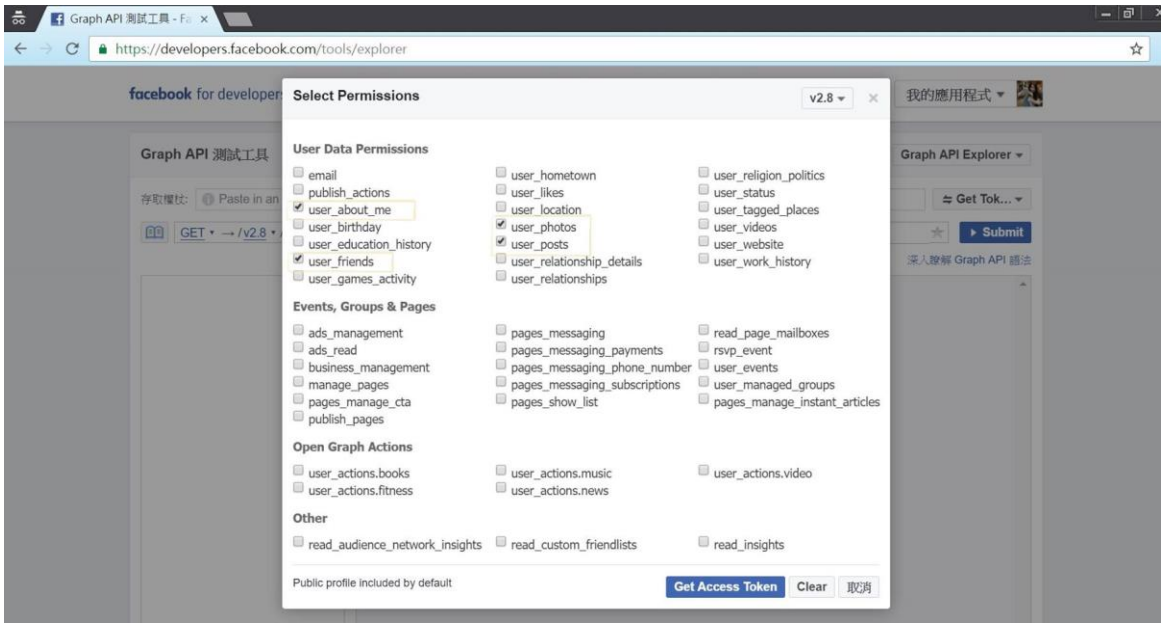
什么是 Graph API? 其实大家想要使用 Facebook API 的目的大多都是希望能够撷取与用户相关的一些对象 (人、网页、相片…)与 连结关系(按赞、留言、标签…)。

Facebook 将这个概念称之为 Social Graph，而存取这些关系的 API，就称之为 Graph API。透过 Graph API，任何你需要 Facebook 提供的信息都能从中拿到。

进入 Graph API Explorer 后拿取 FB Access Token。这个 Token 代表你的身分，因此请谨记：绝对不要外流 Token!!!

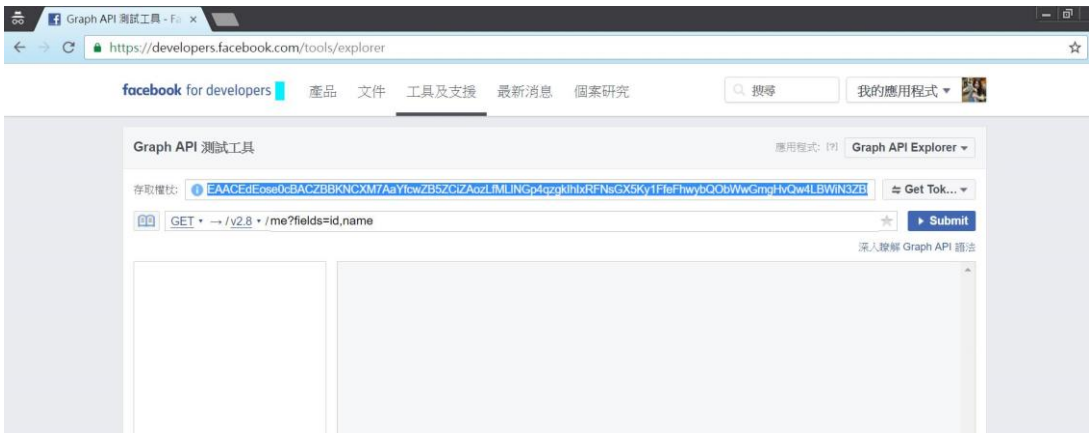
来设定一下权限吧，点击 Get User Access Token：

点选要开启的权限，这边我们设定 user_about_me、user_friends、user_photos 和 user-posts；也可以根据你想分析的内容勾选。最后按下 Get Access Token。



如果想要自动发文和按赞的话，就要点选 publish_actions 噢!

最后将那行超长的「存取令牌」字符串给复制下来：



由于拿到了权限，我们可以在 GET 这边输入一些 API 的字符串，按下 Submit 后就可以看到一些个人讯息，比如/me 查看个人档案、/me/friends 查看 FB 好友、/me/posts 查看贴文纪录。

如果没有出现相对应数据的话，表示 Get Access Token 刚刚勾选的时候没有打开权限，再回去确认一下即可。

接下来的 code 比较多，我们切换回来编辑器（Editor）这边、等程序代码写完后再使用终端机（Terminal）把档案启动~ 做法是 python 檔名.py。

在使用 Graph API 之前，先安装 Request Package，做法是打开 Terminal，输入**$pip install requests** 即可。

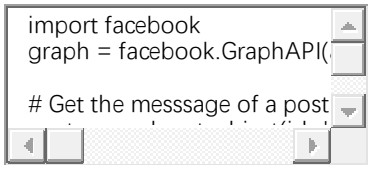
接下来就把 facebook 这个 module 载入进档案中，故 Editor 的最上方输入 import facebook，并放进你刚刚在 Graph API 网站上复制的 token：


```
1 import facebook
2 graph = facebook.GraphAPI(access_token='your_token', version='2.7')
```

尝试玩几个有趣的小专案！

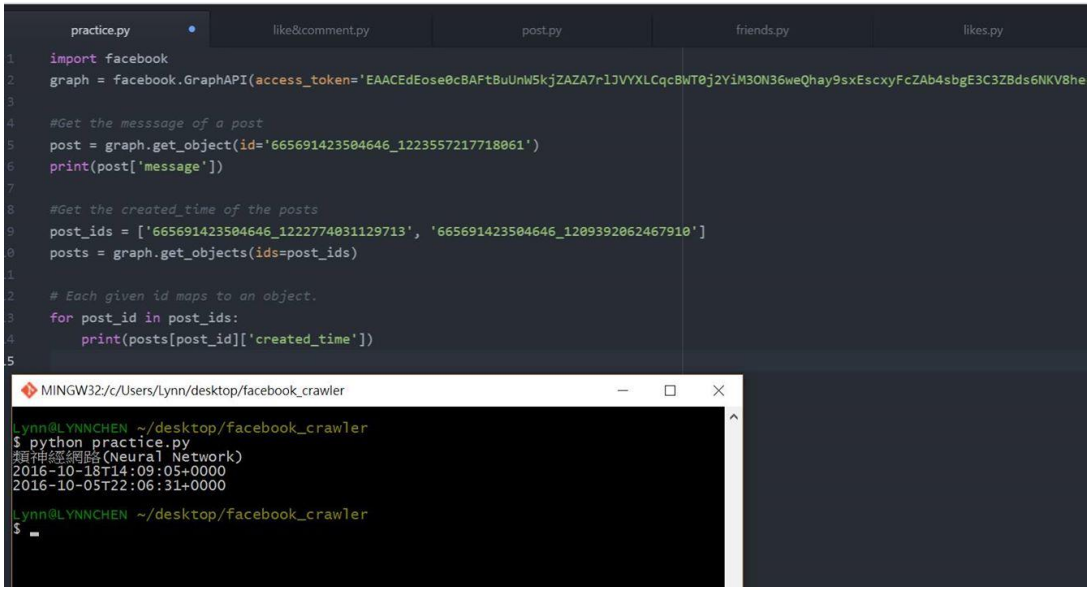
这边因为笔者的脑已经快爆炸了就纯粹附上程序代码和结果供大家参考，有问题再来询问呜呜呜呜 QQQQ（也可以直接去看 Graph API 的 Document，相当清楚！）

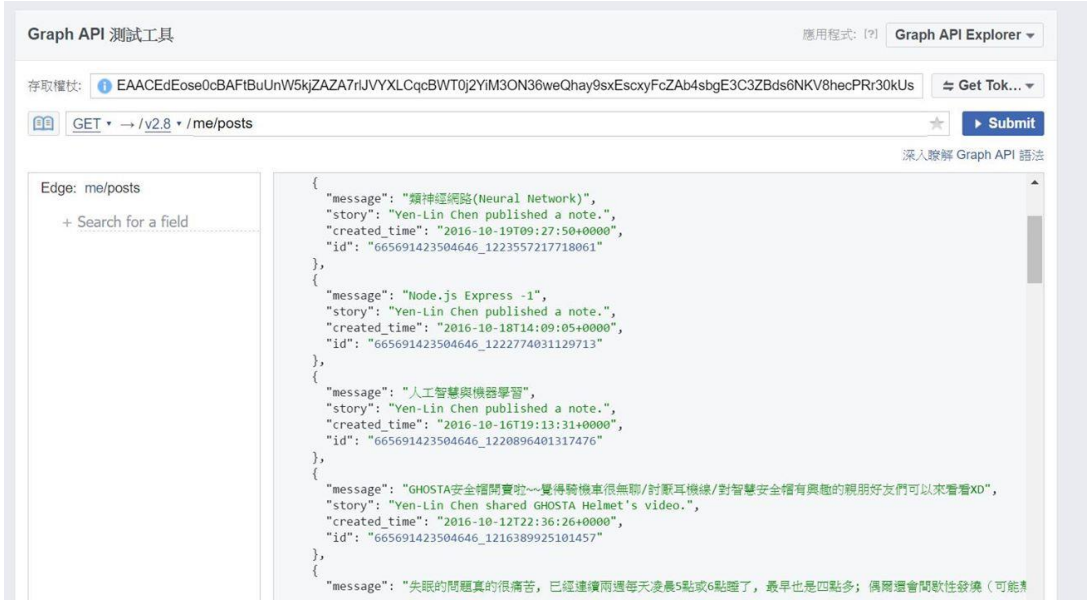
1. 查出 PO 文的 message，与创建 PO 文的时间




```
1  import facebook
2  graph = facebook.GraphAPI(access_token='your_token', version='2.7')
3
4  # Get the messsage of a post
5  post = graph.get_object(id='665691423504646_1223557217718061')
6
7  print(post['message'])
8  # Get the created_time of the posts
9
10 post_ids = ['665691423504646_1222774031129713',
11 '665691423504646_1209392062467910']
12 posts = graph.get_objects(ids=post_ids)
13
14 # Each given id maps to an object.
15 for post_id in post_ids:
   print(posts[post_id]['created_time'])
```

对照一下… 文章”id”是”665691423504646_1222774031129713”的 message 是”类神经网络(Neural Network)”，而另外两篇文章 id 的 created_time 也成功印在 Terminal 上了！噢耶!





2. 自己帮自己按赞和 PO 文


```
1 import facebook
2 graph = facebook.GraphAPI(access_token='your_token', version='2.7')
```

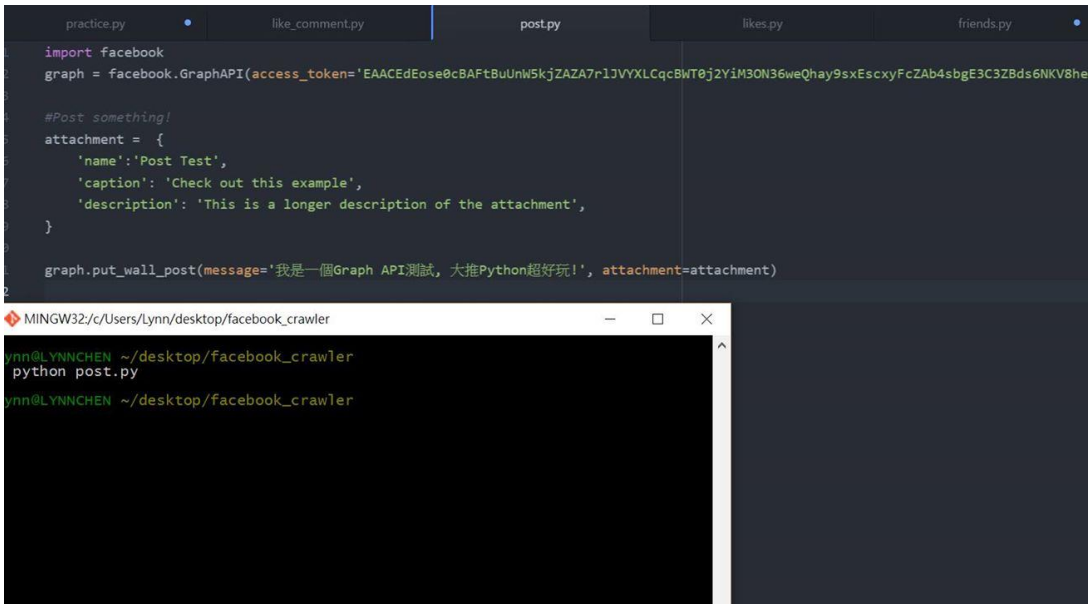
```
3
4 #Put a like!  我是 Node.js 那篇网志!
5 graph.put_like(object_id='665691423504646_1222774031129713')
6
7 # Put a comment!  我是 GHOSTA Helmet 影片分享
8 graph.put_comment(object_id='665691423504646_1216389925101457',  message=' 我 是
  一个 Graph API 测试~~~')
```

得澄清这真的不是我自己先按赞或自己去留言的 XDD” 一样成功了!

3. 自动 PO 文


```
1  import facebook
2  graph = facebook.GraphAPI(access_token='your_token', version='2.7')
3
4  # Post something!
5  attachment =  {
6      'name':'Post Test',
7      'caption': 'Check out this example',
8      'description': 'This is a longer description of the attachment',
9  }
10
```

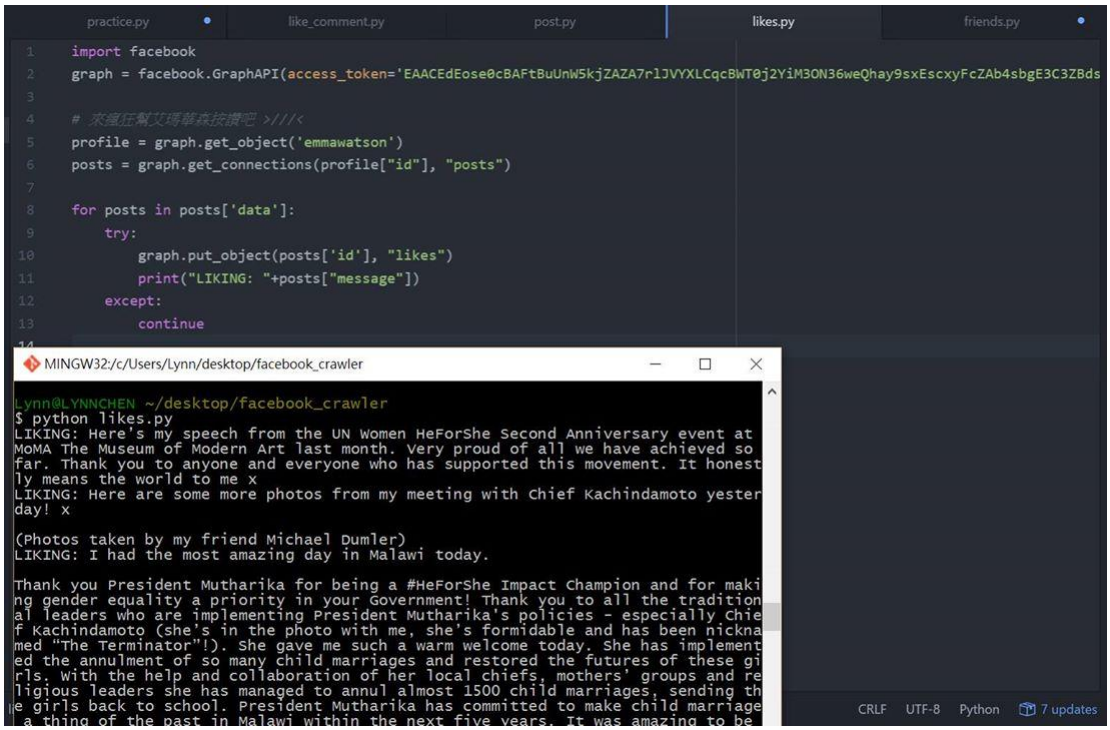
11 graph.put_wall_post(message='我是一个 Graph API 测试，大推 Python 超好玩!', attachment=attachment)



哇~~自动 P0 文出来啦~~~

4. 为心爱的人成为狂赞士吧!


```
1  import facebook
2  graph = facebook.GraphAPI(access_token='your_token', version='2.7')
3
4  # 来疯狂帮埃玛华森按赞吧 >///<
5  profile = graph.get_object('emmawatson')
6  posts = graph.get_connections(profile["id"], "posts")
7
8  for posts in posts['data']:
9      try:
10         graph.put_object(posts['id'], "likes")
11         print("LIKING: "+posts["message"])
12     except:
13         continue
```



成功洗了 N 则埃玛华森 FB 粉专贴文的赞…

今天的 Python 与 FB Graph API 教学先到这边，真正要用到 Dict 来统计出朋友中留言和按赞数最高、最暗恋我们朋友的项目就留待下次再教学吧~~