Python - 十分钟入门

http://tech-marsw.logdown.com/blog/2014/09/03/getting-started-with-python-in-ten-minute

Python 是一个强大、快速、简单易读的程序语言

转换到 Python 之后就回不去 C/C++了，尤其在入手 Mac 之后，不用安装任何软件，也不需要编译
对于 Python 的依赖程度越来越重，越是喜爱就越想推广 Python 给身旁的朋友。

What Can Python Do?

- 一些有用的 Python 函式库列表 *(GUI 界面、游戏、网页、服务器、数据库)*
- Python 开发的应用 *有谁在用 Python*

最近找寻了不少资料，许多在线教学都是按部就班，
对于已有程序基础，想要快速入门 Python 还是比较麻烦
因此写了这一篇，给**「已经学过 C/C++, Java 等程序语言」**快速转换 Python 的文件

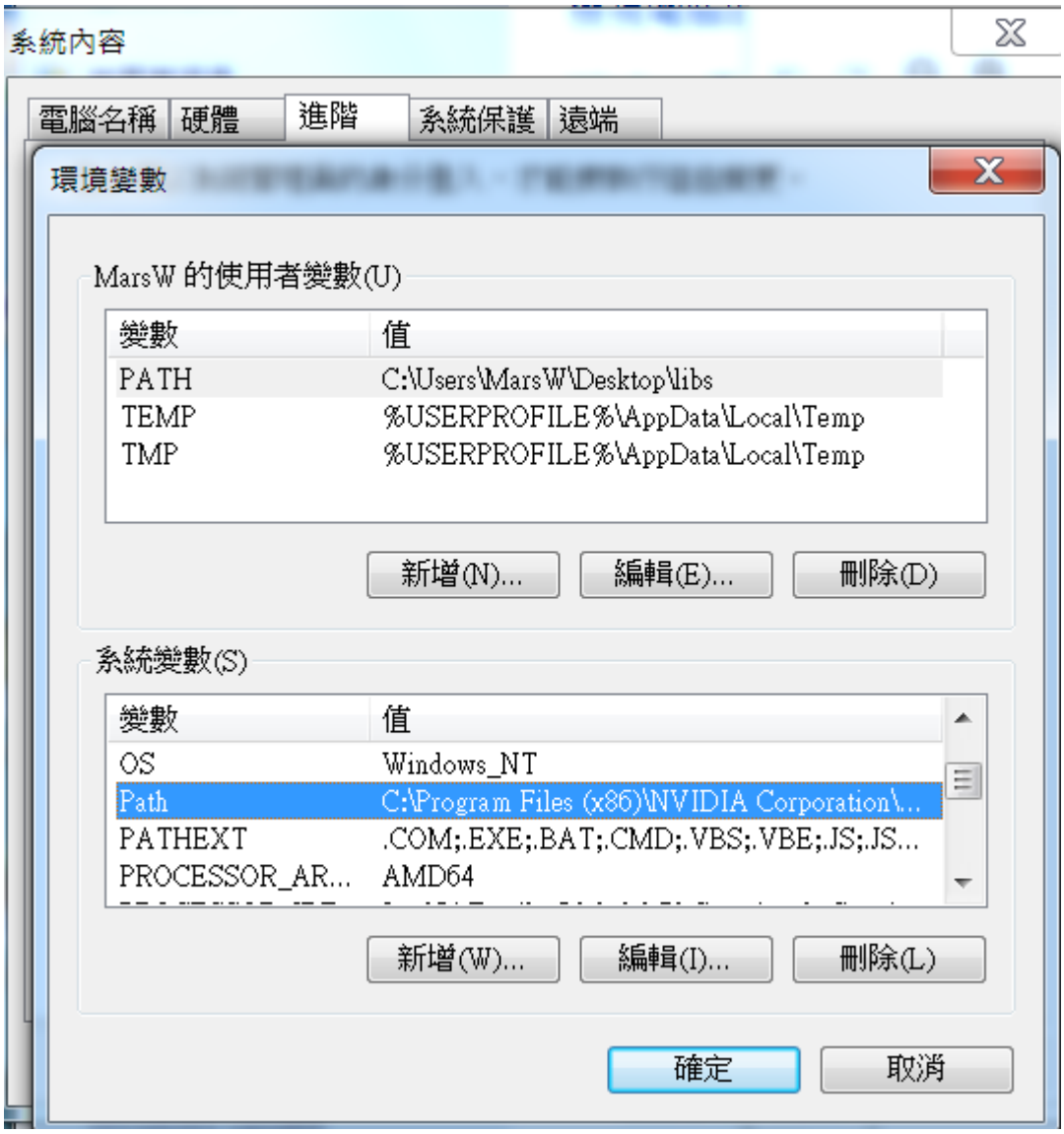
如果是任何程序语言都没有学过，
建议可以从以下资源一步步学习 Python，对于建立起程序设计的概念比较有帮助：

- CodeCademy *游戏化的学习(英文)*
- Python Tutorial 第一堂（1）揭开序幕 *很完整的中文教学*

Let's Start!

[安装]

- Windows：官网下载 2.7.x 版本。在环境变量加上 C:\Python27;C:\Python27\Scripts
- Mac 内建 2.7.5
- Unix:内建 3.x
- Online：Koding



执行第一个程序 Hello World

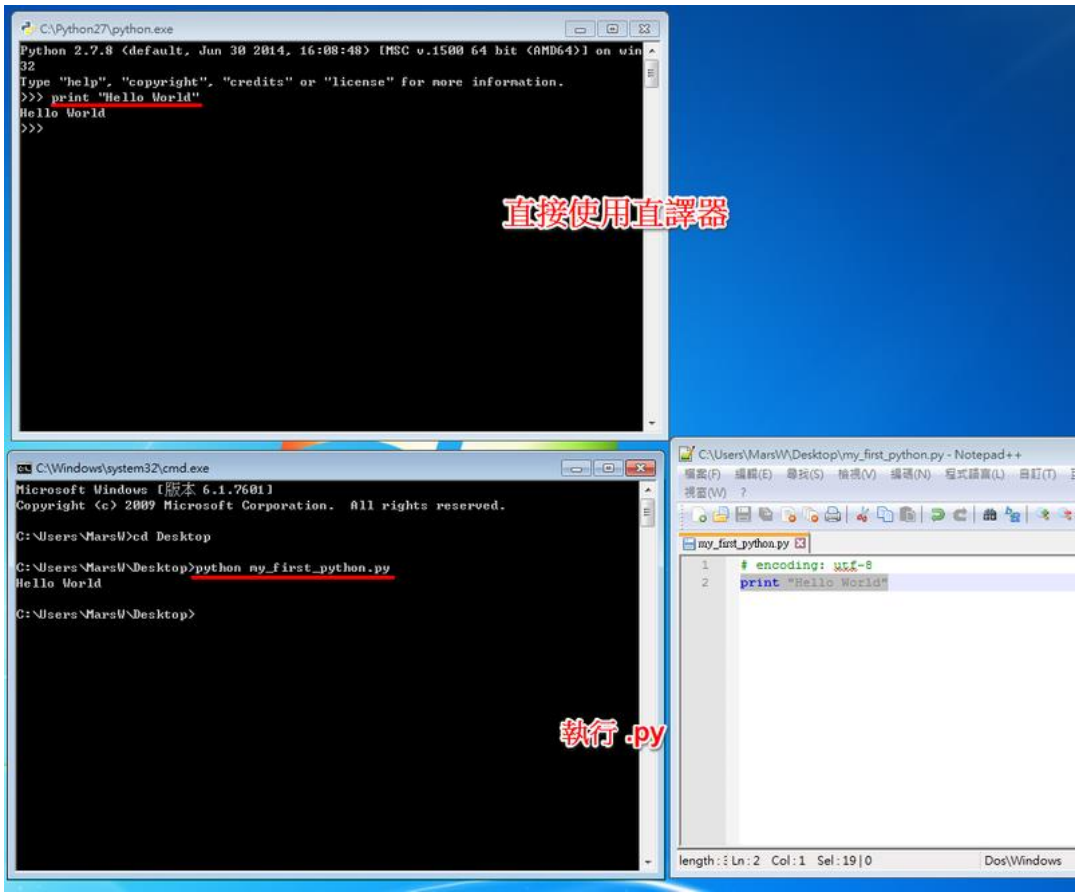
有两种选择：

- 直接开启直译器上输入一行行程序代码，Enter 执行
  - Windows:开启 Python(command line)
  - Mac/Unix:Terminal 输入 python -输入 print "Hello World"
- 跟以往一样写一个.py 的档案，执行 python my_first_python.py

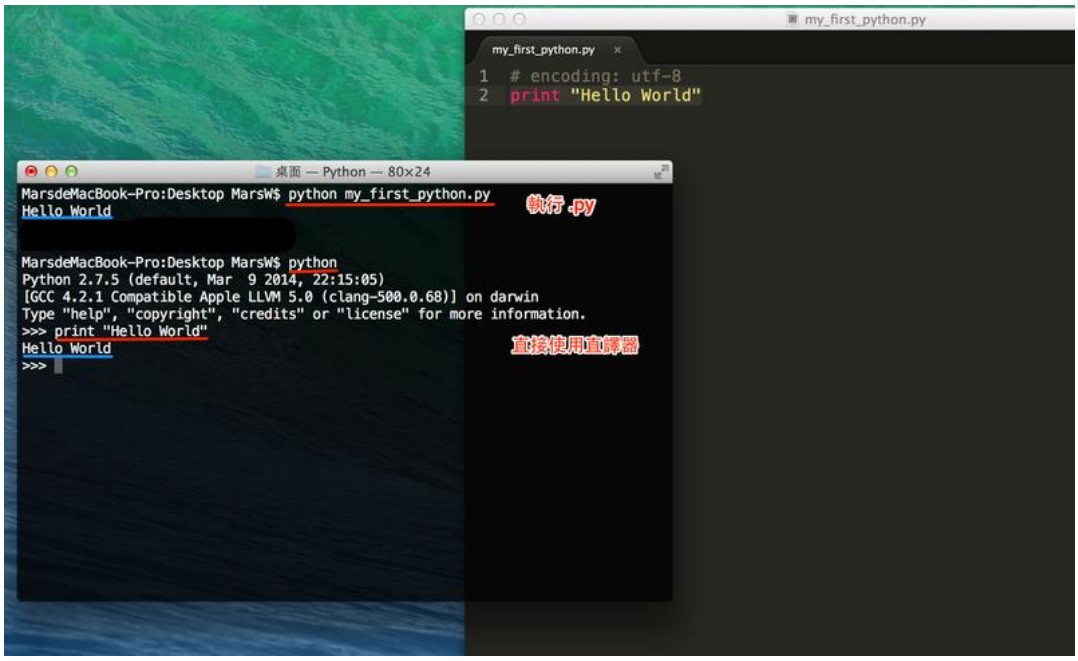
```
my_first_python.py
# encoding: utf-8

print "Hello World"
```

!!!注意中文的档案要加上# encoding: utf-8

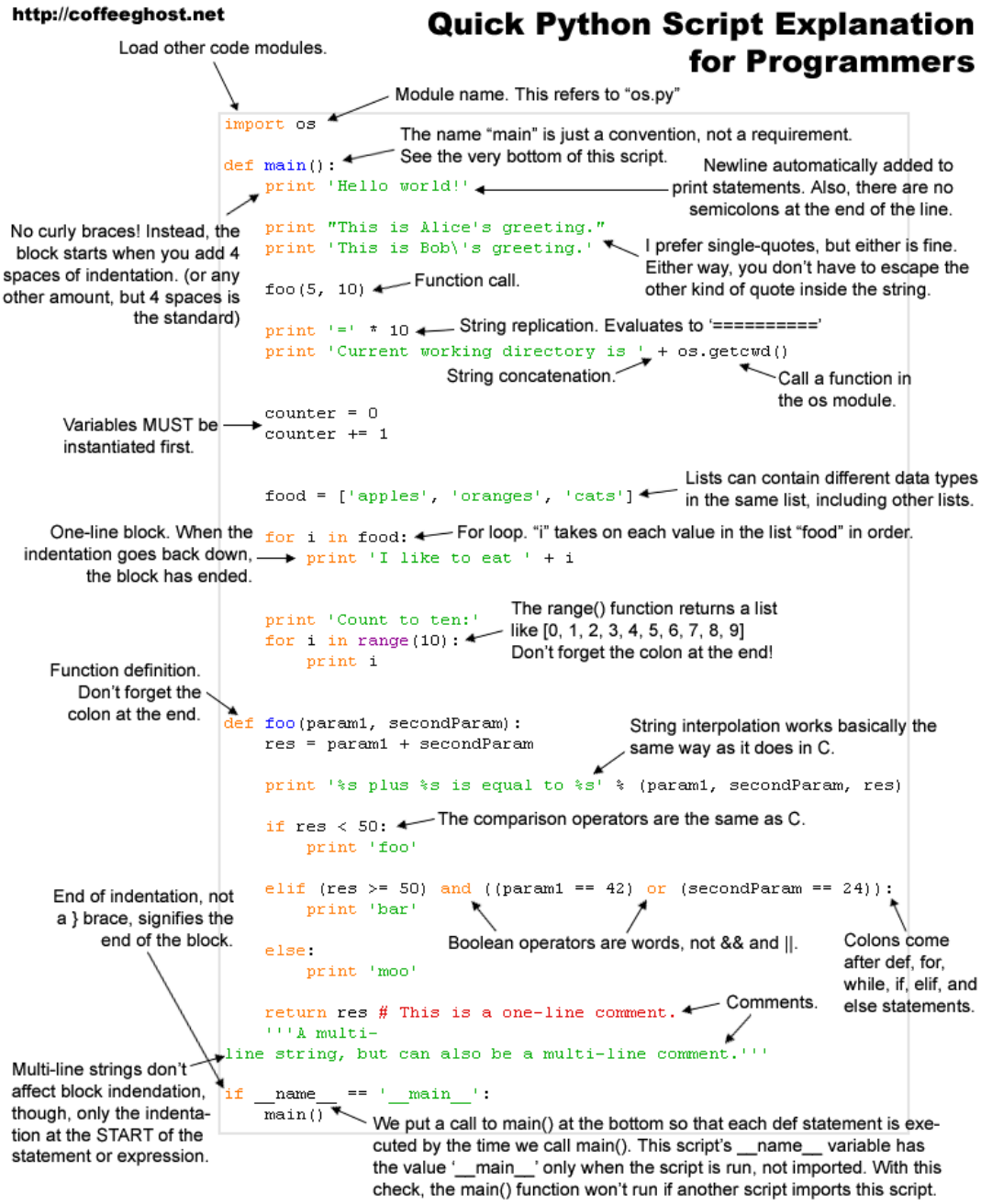
Windows



Mac/Unix



//2014-10-20 补充
网络上看到一张图片大约 40 行的 code 也可以很清楚的供有程序基础的人快速了解 Python [简中对照版]



之后的部分，会更详细介绍 Python 好用的功能，将以.py 的档案形式展示程序代码。

[语法]

python 不需要任何代表结尾的符号(ex;)
python 不需要先指定变量的型态，之后也可以任意转换型态

python 可以透过=,+=,-=直接赋值，也可同时给多个变量赋值
单行批注为#,多行批注则用"""开头与结尾，多运用批注可以帮助你或其他人看懂你的程序代码。

```
# encoding: utf-8

"""
这是一个简单的 python 程序
介绍基本的语法
"""
x = 3
x += 2
x -= 1
print x
x,y = 99.99,5
print x,y

Output:
4
99.99 5
```

[数据类型]

list 是可以随意更动大小的数组，可透过 append 增加。
python 的 list 提供很多好用函式：

- len()可以算 list 长度
- sum()可以计算 list 中所有数值的加总(但 list 中的元素都需为数值,不可与字符串混合)
- count 则是可以计算 list 中某个元素出现次数

```
# encoding: utf-8

my_list = []
my_list.append(1)
my_list.append(2)
my_list2 = [55.55,"Hi",3,99,222,222]
my_list2[0]=333.333

print len(my_list),sum(my_list),my_list2.count(222)
print my_list2[0],my_list2[-1],my_list2[1:3],my_list2[2:]
```

Output:
2 3 2
333.333 222 ['Hi', 3] [3, 99, 222, 222]

dictionary 像是 hash-table 一样有一个 key 对应一个变量

```
# encoding: utf-8

passwd={'Mars':00000,'Mark':56680}
passwd['Happy']=9999
passwd['Smile']=123456

del passwd['Mars']
passwd['Mark']=passwd['Mark']+1

print passwd
print passwd.keys()
print passwd.get('Tony')
```

Output:
{'Happy': 9999, 'Smile': 123456, 'Mark': 56681}
['Happy', 'Smile', 'Mark']
None

set 则是集合，可以进行联集、交集、差集等运算

```
# encoding: utf-8

admins = set()
users = {'Smile', 'Tony','Happy','Sherry','Allen','Andy', 'Mars'}
admins.add('ihc')
admins.add('Mars')

print admins & users
print admins | users
print admins ^ users
print admins - users
print users - admins
```

Output:
set(['Mars'])
set(['Allen','Andy','Smile','Mars','Tony','ihc','Happy','Sherry'])
set(['Andy', 'Allen', 'Tony', 'Smile', 'Happy', 'ihc', 'Sherry'])
set(['ihc'])
set(['Sherry', 'Andy', 'Allen', 'Tony', 'Smile', 'Happy'])

[字符串]

字符串可用双引号"或用单引号'来进行标示

```
# encoding: utf-8

s = "Hello"
s += 'World'
s1 = "HelloWorld".replace("ll","1")
s2 = "Hello"[0]+"i"
print s,s1,s2,len(s)

Output:
HelloWorld HeloWorld Hi 10
```

其中 python 字符串内建的分割函式 string.split()很好用，可以将字符串依指定的字符(字符串)切割

```
s3 = "This is a sentence."
s3_split=s3.split(' ')
print s3_split

Output:
['This', 'is', 'a', 'sentence.']
```

而中文的处理，我们可以透过 unicode 的编解碼来处理

```
# encoding: utf-8

s="台湾"
u = s.decode('utf8')

print '台',s[0],u[0]
print u[0]==u'台'

Output:
台 ? 台 #没有译码过的 s 是显示不出来每一个"中文字"的
True
```

[型别转换、基本运算符]

要注意的是，由于 python 不用特别宣告变量，要注意不能让不同型别的变量同时运算
ex: x+s 会出错，因为整数型态与字符串型态，两者是不能同时运算的

```
# encoding: utf-8

x=2**3
y=3/2
s="3"
print ord('a'),ord('c'),chr(ord('a')+2)
print y,int(s)/2,float(s)/2,3%2
print str(x+y),str(x)+str(y)

Output:
97 99 c
1 1 1.5 1
9 81
```

[Flow Control：判断式、循环]

再来就进入到程序设计中很重要的部分：流程控制
也可以从这部分开始看到 Python 之所以容易阅读的原因，
每个流程的结尾是用冒号:
属于该流程底下的执行动作不需要任何括号，而是使用缩排
缩排可以使用 Tab 或四格 Space，但不可混用，建议是把编辑器设定成 Tab 对应四格空白
缩排可以让程序代码更容易阅读，了解整个程序的架构&逻辑，也是写程序重要的习惯之一。

在 python 的判断式中，and,or,not 是逻辑运算符。
python 提供一个很好的函式 range，范围是左边数字到右边数字-1，在撰写循环时可以更加快速。
另外 in 函式可以用来判断某个型别中是否有某个元素，非常的方便!

```
# encoding: utf-8

my_list=[]
for i in range(0,10):  """//for(i=0;i<10;i++)"""
    my_list.append(i+1)
if my_list[0]==1 and len(my_list)<10:
    my_list[0]+=1
    print "1 state"
elif (10 in my_list) or not(len(my_list)==10):
    print "2 state"
```

```
    print "range(i,j) is i~j-1"
else:
    print "3 state"
    print "none of above"

for i in my_list:       """//for(i=0;i<my_list.length();i++)"""
    print i,           """//cout<<my_list[i]"""
print
```

```
Output:
2 state
range(i,j) is i~j-1
1 2 3 4 5 6 7 8 9 10
```

[自定义函式 Function]

python 定义函式用 def 开头，同样以冒号:结尾，还有缩排。

```
def my_function(x,y):
    return x-10,y+10
x,y = my_function(10,20)
print x,y
```

```
Output:
0 30
```

[类别 Class]

Class 的初始化函式是由两条底线包含着 init 做宣告。

```
# encoding: utf-8

class Student:
    def __init__(self, name, grade, age):
        self.name = name
        self.grade = grade
        self.age = age
    def set_name(self, name):
        self.name = name

student_objects=[]
student_objects.append( Student('john', 'B', 15) )
student_objects.append( Student('dave', 'A', 12) )
```

```
student_objects.append( Student('jane', 'A', 10) )
student_objects[0].set_name('John')

for i in student_objects:
    print i.name,i.grade,i.age

Output:
John B 15
dave A 12
jane A 10
```

[导入外部资源 import]

可以用 import 直接导入整个 python 檔中所有的函式
或是用 from 档案 import 函式，插入特定的函式

```
# encoding: utf-8

import sys """插入 sys 档案中所有函式，使用 sys 檔中的 write 函式前须加档名"""
from time import time """从 time 档案插入 time()函式，使用 time()前不需要加檔名"""
sys.stdout.write( str(time()) + "\n" )

Output:
1409796132.99 #当下的 time
```

[I/O]

前例中已经看到很多使用 print 当 Output 的案例
但 print 预设是印每一行就会加一个换行，要使用,才会在每次印出之间以空格取代
要更自由一点，可以使用 library 中的 write 函式。

```
# encoding: utf-8

import sys
file_in = file('db.txt','r')
file_out = file('copy.txt','w')
for line in file_in:
    for i in range(0,len(line)):
        if line[i]!="\n":
            sys.stdout.write(line[i]+',')
```

```
        else:
            sys.stdout.write(line[i])
        file_out.write(line[i])

sys.stdout.write("\n")
file_in.close()
file_out.close()
"""
# db.txt
1111
2222
ssss
wwww
5555
"""
```

Output:
1,1,1,1,
2,2,2,2,
s,s,s,s,
w,w,w,w,
5,5,5,5,
copy.txt 内容和 db.txt 一样。

[例外处理 Exceptions Handling]

```
# encoding: utf-8

def my_divide():
    try:
        10 / 0 #会让程序出错,所以需要特别 handle

    except ZeroDivisionError:
        print "不能除以 0!!!"
    else:
        print "没有任何错误"
    finally:
        print "无论有没有例外都会执行这一行"
my_divide()
```

Output:
不能除以 0!!!
无论有没有例外都会执行这一行

[排序]

排序是用程序处理数据中最常用到功能，python 提供了很方便的 sort 函式
lambda 是简易型函式，只能回传一个值，因此如果需要两个值以上的排列顺序，
会用 attrgetter

```
# encoding: utf-8

class Student:
    def __init__(self, name, grade, age):
        self.name = name
        self.grade = grade
        self.age = age
    def set_name(self, name):
        self.name = name

student_objects=[]
student_objects.append( Student('john', 'B', 15) )
student_objects.append( Student('dave', 'A', 12) )
student_objects.append( Student('jane', 'A', 10) )

student_objects.sort(key=lambda i: i.grade)
for i in student_objects:
    print i.name,i.grade,i.age
print

from operator import attrgetter

student_objects.sort(key=attrgetter('grade', 'age'),reverse=True)
for i in student_objects:
    print i.name,i.grade,i.age
print

Output:
dave A 12
jane A 10
john B 15

john B 15
dave A 12
jane A 10
```

以上介绍包含了一些最常用到的 python 功能，给想快速转换 python 的人一个入门文章。