主題: **Python** 程式設計簡介

這份講義透過 Python 語言，介紹程式設計概念，並希望幫助初學者，認識基本的資料結構和演算法。我們主要參考的書籍是 treading on Python (by Matt Harrison)，總共有兩冊，內容精簡，對於密集學習 Python 很有幫助。這份講義只是提供入門簡介，希望大家也能充分利用網路上豐富的學習資源，學得更深入、更完整。

網路上的 Python 教材非常多，也有很多精彩的線上課程，例如

Coursera https://zh-tw.coursera.org/course/interactivepython1 (https://zh-tw.coursera.org/course/interactivepython1)

edX https://www.edx.org/course/introduction-computer-science-mitx-6-00-1x-6 (https://www.edx.org/course/introduction-computer-science-mitx-6-00-1x-6)

Udacity https://www.udacity.com/course/programming-foundations-with-python--ud036 (https://www.udacity.com/course/programming-foundations-with-python--ud036) 有了基礎之後，非常推薦大家再繼續看Peter Norvig 的進階課程 https://www.udacity.com/course/design-of-computer-programs--cs212 (https://www.udacity.com/course/design-of-computer-programs--cs212)

安裝 **Python (**版本 **2.7)**

安裝的方式很多，例如Anaconda https://www.continuum.io/downloads (https://www.continuum.io/downloads) 下載安裝之後，可以再依照這個網頁的指令 http://ipython.org/install.html (http://ipython.org/install.html) 安裝 IPython YouTube上面也能找到不少說明影片，例如政大蔡老師的影片https://www.youtube.com/watch?v=WB8hg6UZeY0 (https://www.youtube.com/watch?v=WB8hg6UZeY0) 安裝完成之後，就可以使用 Python 的REPL (Read Evaluate Print Loop)，或是使用 IPython Notebook 環境，開始學習 Python。

用 **print** 輸出資料

常用快捷鍵

- Esc + m 切換成 Markdown 模式
- Esc + y 切換成 Code 模式
- Esc + b 在下面加入一個新的 cell
- Ctrl + Enter 執行目前 cell 中的程式碼
- Ctrl + Alt 執行目前 cell 中的程式碼並且在下面加入一個新的 cell

In [70]:
```
print "Hello, World!"
print ('Hello, World!') # Python 3
print "Hello,", "World!"
print 1, '+', 3, '=', 1+3
```

```
Hello, World!
Hello, World!
Hello, World!
1 + 3 = 4
```

讀取使用者輸入的資料

`raw_input` 讀到的是字串而非數值。如果要把阿拉伯數字轉成對應的數值，要用 `int()` 轉成整數，或用 `float()` 轉成小數。

In [76]:
```
name = raw_input("What's your name: ")
print 'Hello, ' + name + '!'

val = raw_input('How many books? ')
val = int(val) + 1
print val

drink = raw_input("What do you want to drink? ")
print 'Hi ' + name + '. Here is your ' + drink + '!'
```

```
What's your name: Eddie
Hello, Eddie!
How many books? 15
16
What do you want to drink? orange juice
Hi Eddie. Here is your orange juice!
```

變數 **Variables**

寫程式的時候，會需要記錄某些狀態或資訊，我們通常將這些資訊儲存在變數中，之後也可以在從變數中取出我們要的資訊。

In [81]:
```
a_book = "treading on python"
print a_book

another_book = a_book
print another_book

another_book = a_book + ' ' + a_book
print another_book
```

```
treading on python
treading on python
treading on python treading on python
```

基本的資料型態 **Types**

底下介紹的幾種基本的資料型態，包括 string、integer、float、Boolean

In [84]:
```
# string
name = 'Walter'
print name

longer_string = """ I don't know.

    Please tell me.

"""
print longer_string
```

```
Walter
 I don't know.

    Please tell me.
```

In [88]:
```
x = 5     # integer
y = 3.4   # float
print x, y
print x + y
# exercise
print x/2 + y*3
```

```
5 3.4
8.4
12.2
```

In [96]:
```
# Boolean
t = True
f = False
print t and f
print t or f
print not f
print f or (not f)
```

```
False
True
True
True
```

不能用 **Python** 的 **keyword** 當作變數名稱

底下是 Python keyword 列表。 這些 keyword 都不能拿來當作變數名稱

In [100]:
```
import keyword
print keyword.kwlist
```

```
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'el
if', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'i
f', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'rais
e', 'return', 'try', 'while', 'with', 'yield']
```

另外還有一些內建的 function 和 class 的名稱，則是盡量避免，包括 `dict` `file` `id` `list` `open` `str` `sum` `type` `input` `len` `max` `min` `next`

In [102]:
```
name = "Walter"
id(name)
```

Out[102]: 62030272L

In [103]:
```
person = name
id(person)
```

Out[103]: 62030272L

In [104]:
```
person is name
```

Out[104]: True

In [107]:
```
name = "Ben"
print id(name)
print id("Ben")
print id(person)
print id("Walter")
person is name
```

```
63033392
63033392
62030272
62030272
```

Out[107]: False

In [109]:
```
type(name)
```

Out[109]: str

In [111]:
```
type(2)
```

Out[111]: int

In [116]:
```
print str(2) + str(3)
print 2 + 3
```

```
23
5
```

In [118]:
```
type(True)
```

Out[118]: bool

In [120]:
```
type((1,2,3))
```

Out[120]: tuple

In [121]:
```
type([1,2,3])
```

Out[121]: list

In [122]:
```
list('abc')
```

Out[122]: ['a', 'b', 'c']

數字運算

```
In [134]: type(3)
Out[134]: int

In [135]: type(2.5)
Out[135]: float

In [137]: type(3 + 2.5)
Out[137]: float

In [138]: coerce(3, 2.5)
Out[138]: (3.0, 2.5)

In [139]: 2.3 + 4 * 5
Out[139]: 22.3

In [141]: type(3/4)
Out[141]: int

In [142]: 3/4
Out[142]: 0

In [143]: 3/4.0
Out[143]: 0.75
```

餘數和次方

```
In [145]: 5 % 3
Out[145]: 2

In [146]: 3 ** 4
Out[146]: 81

In [147]: 3 ** 100
Out[147]: 515377520732011331036461129765621272702107522001L
```

In [149]:
```
type(3**100)
```

Out[149]:
```
long
```

In [151]:
```
import sys
sys.maxint
```

Out[151]:
```
2147483647
```

In [153]:
```
type(sys.maxint + 1)
```

Out[153]:
```
long
```

使用 **format** 設定格式

In [161]:
```
print 'A:{0}  B:{1}  C:{2}'.format(300, 5.6, 'Red')
print 'A:{}  B:{}  C:{}'.format(300, 5.6, 'Red')
print 'A:{2}  B:{1}  C:{0}'.format(300, 5.6, 'Red')

print 'Major {[abc]}'.format({'abc':'Tom'})
print 'Major {abc}'.format(abc='Tom')

# exercise
abc = raw_input()
print '{0}{0}{0}'.format(abc)
```

```
A:300  B:5.6  C:Red
A:300  B:5.6  C:Red
A:Red  B:5.6  C:300
Major Tom
Major Tom
12345
123451234512345
```

In [167]:
```
print 'The binary representation of {0} is {1:b}'.format(9, 9)
print '{0} m = {1:=15.13} ft'.format(2, 2*100/2.54/12)
```

```
The binary representation of 9 is 1001
2 m =  6.561679790026 ft
```

兩個有用的輔助函數 **dir `help**

In [168]:
```
print dir('Tom')
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__e
q__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__get
newargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le__',
'__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__redu
ce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr_
_', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name
_split', '_formatter_parser', 'capitalize', 'center', 'count', 'decod
e', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isa
lnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isuppe
r', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfin
d', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'spli
tlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'uppe
r', 'zfill']
```

In [173]:
```
print 'm' + '     (_ _)       '.strip() + 'm'
print 'm' + '     (_ _)       '.lstrip() + 'm'
print 'm' + '     (_ _)       '.rstrip() + 'm'
print '     (_ _)       '.find('(')
```

```
m(_ _)m
m(_ _)       m
m     (_ _)m
5
```

In [176]:
```
print 'tom'.capitalize()
print 'tom'.upper()
print 'Tom'.lower()
```

```
Tom
TOM
tom
```

In [177]:
```
help('tom'.capitalize)
```

```
Help on built-in function capitalize:

capitalize(...)
    S.capitalize() -> string

    Return a copy of the string S with only its first character
    capitalized.
```

In [178]:
```
help('tom'.find)
```

```
Help on built-in function find:

find(...)
    S.find(sub [,start [,end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.
```

In [181]:
```
print dir(5)
```

```
['__abs__', '__add__', '__and__', '__class__', '__cmp__', '__coerce__',
'__delattr__', '__div__', '__divmod__', '__doc__', '__float__', '__floo
rdiv__', '__format__', '__getattribute__', '__getnewargs__', '__hash_
_', '__hex__', '__index__', '__init__', '__int__', '__invert__', '__lon
g__', '__lshift__', '__mod__', '__mul__', '__neg__', '__new__', '__nonz
ero__', '__oct__', '__or__', '__pos__', '__pow__', '__radd__', '__rand_
_', '__rdiv__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr_
_', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__',
'__rpow__', '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '_
_rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclas
shook__', '__truediv__', '__trunc__', '__xor__', 'bit_length', 'conjuga
te', 'denominator', 'imag', 'numerator', 'real']
```

In [184]:
```
(9).bit_length()
```

Out[184]:
```
4
```

更多屬於 **string** 的函數

In [185]:
```
name = 'Major Tom'
print name.endswith('tom')
print name.endswith('Tom')
```

```
False
True
```

In [188]:
```
people = '; '.join(['Alice', 'Bob', 'Carl'])
print people
help('; '.join)
```

```
Alice; Bob; Carl
Help on built-in function join:

join(...)
    S.join(iterable) -> string

    Return a string which is the concatenation of the strings in the
    iterable.  The separator between elements is S.
```

Boolean 以及 None Type

In [192]:
```
names = ['Alice', 'Bob', 'Carl']    # empty list
if names:
    print ', '.join(names)
else:
    print 'No one there'
```

```
Alice, Bob, Carl
```

In [196]:
```
print bool(0), bool(1), bool(''), bool('0'), bool([]), bool([0,1,2])
```

```
False True False True False True
```

In [201]:
```
print bool(None)
x = None
y = None
print id(x), id(y)
print x is y
print x is not y
```

```
False
506046248 506046248
True
False
```

條件判斷

In [206]:
```
print 3 > 5
name = 'Zappa'
print name == 'Zappaa'
print name is None
```

```
False
False
False
```

In [209]:
```
# and or not
num_books = raw_input('How many books? ')
num_books = int(num_books)
if num_books >= 5 and num_books < 10:
    print '10% discount'
    print 'total {0} NTD'.format(num_books*90)
elif num_books > 10:
    print '15% discount'
    print 'total {0} NTD'.format(num_books*85)
else:
    print 'no discount'
    print 'total {0} NTD'.format(num_books*100)
```

```
How many books? 8
10% discount
total 720 NTD
```

`if` 後面有 `:`，接下來屬於 `if` 要做的程式碼，都要縮排。縮排通常是用四個空格

In [215]:
```
name = 'Zappa'
if name.startswith('Z') or name.endswith('a'):
    print "That's a good name."

help(name.startswith)
help(name.endswith)
```

```
That's a good name.
Help on built-in function startswith:

startswith(...)
    S.startswith(prefix[, start[, end]]) -> bool

    Return True if S starts with the specified prefix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    prefix can also be a tuple of strings to try.

Help on built-in function endswith:

endswith(...)
    S.endswith(suffix[, start[, end]]) -> bool

    Return True if S ends with the specified suffix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    suffix can also be a tuple of strings to try.
```

存放更多資料：使用 **list tuple set**

Lists

In [241]:
```
names = []
more_names = ['Alice', 'Bob', 'Carl']
# append
more_names.append('Eddie')
print more_names
print more_names[-4]
# insert
more_names.insert(3, 'Dean')
print more_names
# remove
more_names.remove('Bob')
print more_names
# help(more_names.sort)
more_names.sort(reverse=True)
print more_names
more_names.insert(3, 'Frank')
print more_names
# more_names.sort()
# print more_names
names_sorted = sorted(more_names)
print names_sorted
print more_names
more_names[4] = 'Alex'
print more_names
```

```
['Alice', 'Bob', 'Carl', 'Eddie']
Alice
['Alice', 'Bob', 'Carl', 'Dean', 'Eddie']
['Alice', 'Carl', 'Dean', 'Eddie']
['Eddie', 'Dean', 'Carl', 'Alice']
['Eddie', 'Dean', 'Carl', 'Frank', 'Alice']
['Alice', 'Carl', 'Dean', 'Eddie', 'Frank']
['Eddie', 'Dean', 'Carl', 'Frank', 'Alice']
['Eddie', 'Dean', 'Carl', 'Frank', 'Alex']
```

In [253]:
```
nums1 = range(5)
print nums1
nums2 = range(1,6)  # half open interval [1, 6)
print nums2
nums3 = range(1,10,2)
print nums3
# help(range)
print sum([3,4,5,1,6,7])
print len([3,4,5,2,5,2,6,6])
```

```
[0, 1, 2, 3, 4]
[1, 2, 3, 4, 5]
[1, 3, 5, 7, 9]
26
8
```

Tuples

In [258]:
```
t = (3,5)
emptyt = ()
onet = (3,)
manyt = (1,2,3,4)
print t, emptyt, onet, manyt
# tuples are immutable
# cannot do something like manyt.append(5)
```

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

Sets

In [270]:
```
nums = [1, 2, 3, 5, 2, 3, 4]
num_set = set(nums)
print num_set
num_set2 = set([3, 5])
print 'num_set2: ', num_set2
num_set3 = num_set - num_set2    # diff
print 'num_set3: ', num_set3
print num_set2 | num_set3        # or
print num_set2 & num_set3        # andd
num_set3.add(3)
print 'num_set2: ', num_set2
print 'num_set3: ', num_set3
print num_set2 | num_set3        # or
print num_set2 & num_set3        # add
print num_set2 ^ num_set3        # xor, only in one set but not in anoth
er
```

```
set([1, 2, 3, 4, 5])
num_set2:  set([3, 5])
num_set3:  set([1, 2, 4])
set([1, 2, 3, 4, 5])
set([])
num_set2:  set([3, 5])
num_set3:  set([1, 2, 3, 4])
set([1, 2, 3, 4, 5])
set([3])
set([1, 2, 4, 5])
```

In [278]:
```
# exercise
a = [ [4,6,5], [3,2,1] ]
print a[0]
print a[1]
print a[0][2], a[1][1]
# 4 6 5
# 3 2 1
a[0].sort()
a[1].sort()
a.sort()
print a
```

```
[4, 6, 5]
[3, 2, 1]
5 2
[[1, 2, 3], [4, 5, 6]]
```

In [283]:
```
# exercise
names = ['Alice', 'Carl', 'Bob', 'Carl', 'Tom', 'Alice']
names_set = set(names)
print names_set
names = list(names_set)
names.sort()
print names
```

```
set(['Bob', 'Alice', 'Carl', 'Tom'])
['Alice', 'Bob', 'Carl', 'Tom']
```

迴圈

In [286]:
```
x = 0
for i in [2, 7, 13, 23, 31]:
    if i%3 == 1:
        print i
        x = x + i
print x
```

```
7
13
31
51
```

In [291]:
```
for i in range(10):
    print '$' * i
# exercise
for i in range(9,0,-1):
    print '$' * i
```

```
$
$$
$$$
$$$$
$$$$$
$$$$$$
$$$$$$$
$$$$$$$$
$$$$$$$$
$$$$$$$
$$$$$$
$$$$$
$$$$
$$$
$$
$
```

In [293]:
```
for i in range(1,10):
    for j in range(1,i):
        if (i+j)%7==0:
            print i, j
```

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

In [302]:
```
# list and enumerate
fruits = ['apple', 'banana', 'cherry']
for idx, val in enumerate(fruits):
    print idx, val, fruits[idx]

for idx, _ in enumerate(fruits):
    fruits[idx] = 'nothing'
print fruits
```

```
0 apple apple
1 banana banana
2 cherry cherry
['nothing', 'nothing', 'nothing']
```

break 和 **continue**

In [306]:
```
nums = [1, 5, 3, 7, -3, 9, -5, 11]
total = 0
for elm in nums:
    if elm <= 0:
        break
    total = total + elm
print total

total = 0
for elm in nums:
    if elm > 0:
        total = total + elm
print total
```

```
16
36
```

for 迴圈也能搭配 **else**

如果 for 迴圈不是因為 break 而結束，就會執行 else 所指定的事情

In [309]:
```
nums = [1, 5, 3, 7] # [1, 5, 3, 7, -3, 9, -5, 11]
total = 0
success = False
for elm in nums:
    if elm <= 0:
        break
    total = total + elm
else:
    success = True
print total, success
```

```
16 True
```

In [311]:
```
# exercise
# for val in range(2,100):
# 2, 3, 4, 5, ...
# for i in range(2, val):
#  val % i == 0  --> break
for val in range(2,100):
    for j in range(2,val):
        if val % j == 0:
            break
    else:
        print val
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

找最大值

In [317]:
```
nums = [1, 5, 3, 7, -3, 9, -5, 11, 2, 3]
print max(nums)

maxval = nums[0]
for x in nums:
    if maxval < x:
        maxval = x
        print 'update maxval: ', maxval
print maxval
```

```
11
update maxval:  5
update maxval:  7
update maxval:  9
update maxval:  11
11
```

In [323]:
```
lst1 = [1, 3, 6]
lst2 = [2, 4, 5]
lst3 = []
for i in range(len(lst1)):
    if lst1[i] < lst2[i]:
        lst3.append(lst2[i])
    else:
        lst3.append(lst1[i])
print lst3

print zip(lst1, lst2)
lst3 = []
for p in zip(lst1, lst2):
    if p[0] < p[1]:
        lst3.append(p[1])
    else:
        lst3.append(p[0])
print lst3
```

```
[2, 4, 6]
[(1, 2), (3, 4), (6, 5)]
[2, 4, 6]
```

In [325]:
```
lst1 = [1, 3, 6]
lst2 = [2, 4, 5]
lst3 = []
lst4 = []
for p in zip(lst1, lst2):
    if p[0] < p[1]:
        lst3.append(p[0])
        lst4.append(p[1])
    else:
        lst3.append(p[1])
        lst4.append(p[0])
print lst3
print lst4
```

```
[1, 3, 5]
[2, 4, 6]
```

用 **Dictionary** 儲存資料

In [15]:
```
price = {'apple':10}
price['banana'] = 5
price['cherry'] = 15
price['orange'] = 8
print price['apple']
print price['cherry']
if 'mango' in price:
    print price['mango']
grape_price = price.get('grape', 12)
print grape_price
banana_price = price.get('banana', 3)
print banana_price
```

```
10
15
12
5
```

In [27]:
```
# print dir(price)
# help(price.items)
pv = price.items()
print pv
# help(sorted)
# exercise
print sorted(pv)
for k, v in sorted(pv):
    print k, v
print sorted(pv, key = lambda x: x[1])
```

```
[('orange', 8), ('cherry', 15), ('apple', 10), ('banana', 5)]
[('apple', 10), ('banana', 5), ('cherry', 15), ('orange', 8)]
apple 10
banana 5
cherry 15
orange 8
[('banana', 5), ('orange', 8), ('apple', 10), ('cherry', 15)]
```

collections.Counter

In [44]:
```
import collections
# help(collections.Counter)
c = collections.Counter('abcdeabcdabcaba')
print c
print list('abcdeabcdabcaba')
c.most_common(3)
print list(c.elements())
print ''.join(sorted(c.elements()))
```

```
Counter({'a': 5, 'b': 4, 'c': 3, 'd': 2, 'e': 1})
['a', 'b', 'c', 'd', 'e', 'a', 'b', 'c', 'd', 'a', 'b', 'c', 'a', 'b',
'a']
['a', 'a', 'a', 'a', 'a', 'c', 'c', 'c', 'b', 'b', 'b', 'b', 'e', 'd',
'd']
aaaaabbbbcccdde
```

In [48]:
```
count = collections.Counter(['dog', 'cat', 'dog', 'cat', 'cat', 'pig',
'horse'])
print count
print count['cat']
print count.keys()
print count.values()
print count.items()
# exercise
# cat 3
# dog 2
# horse 1
# pig 1
for k, v in sorted(count.items()):
    print k, v
```

```
Counter({'cat': 3, 'dog': 2, 'horse': 1, 'pig': 1})
3
['horse', 'pig', 'dog', 'cat']
[1, 1, 2, 3]
[('horse', 1), ('pig', 1), ('dog', 2), ('cat', 3)]
cat 3
dog 2
horse 1
pig 1
```

Functions 自定函數

In [53]:
```
def modulo12(x):
    '''
    compute arithmetic modulo 12
    '''
    ret = x % 12
    return ret

#help(modulo12)
print modulo12(23)
print modulo12(34)
```

```
11
10
```

In [60]:
```
def modulo(x, n=2):
    '''
    default modulo 2
    compute x % n
    '''
    if n==0:
        return 0
    else:
        return x%n
print modulo(23, 12)
print modulo(7)
print modulo(8, 0)
```

```
11
1
0
```

In [62]:
```
def gcd(a, b):
    '''
    find the greatest common divisor of a and b
    '''
    if b==0:
        return a
    else:
        return gcd(b, a%b)

print gcd(105,84)
print gcd(105,36)

3
```

```
21
3
```

Out[62]:
```
3
```

In [65]:
```
# '{0:b}'.format(12) # 1*8 + 1*4 + 0*2 + 0*1
def tobinary(x, ans=''):
    '''
    convert a positive decimal number into its binary representation
    '''
    if x==0:
        return ans
    else:
        return tobinary(x/2, str(x%2)+ans)

print tobinary(12)
```

```
1100
```

取出 **list** 元素

In [77]:
```
animals = ['cat', 'dog', 'pig', 'horse']
print animals[0]
print animals[-1]
print animals[0:3]  # [0,3)
print animals[:3]
print animals[0:-1]
print animals[2:]
print animals[:]
print animals[0::2] # stride = 2
print animals[-2::-1] # reverse order
print animals[::-1]
# exercise
print animals[0::2]+animals[1::2]
```

```
cat
horse
['cat', 'dog', 'pig']
['cat', 'dog', 'pig']
['cat', 'dog', 'pig']
['pig', 'horse']
['cat', 'dog', 'pig', 'horse']
['cat', 'pig']
['pig', 'dog', 'cat']
['horse', 'pig', 'dog', 'cat']
['cat', 'pig', 'dog', 'horse']
```

檔案輸入輸出

In [83]:
```
fout = open('a.txt', 'w')
#help(fout)
fout.write('Eddie\n')
fout.close()

with open('a.txt', 'a') as fout:
    fout.write('Jimi\n')

fin = open('a.txt')
#print fin.readline()
for name in fin:
    print name
fin.close()
```

```
Eddie

Jimi
```

自定 **Class**

In [94]:
```
class Fruit(object):
    '''
    Defining fruit
    '''
    def __init__(self, n, c):
        self.name = n
        self.color = c

    def describe(self):
        '''
        Describe the fruit
        '''
        print "It is a {0} {1}.".format(self.color, self.name)

#help(Fruit)
fruit = Fruit("apple", "red")
#dir(fruit)
fruit.describe()
```

```
It is a red apple.
```

Subclass Superclass

In [101]:
```
class TropicalFruit(Fruit):
    '''
    Tropical fruit
    '''
    def describe(self):
        '''
        Describe the fruit
        '''
        super(TropicalFruit, self).describe()
        print "It is sweet."

# help(TropicalFruit)
fruit = TropicalFruit("pineapple", "yellow")
#dir(fruit)
fruit.describe()
```

```
It is a yellow pineapple.
It is sweet.
```

Importing libraries

In [5]:
```
from math import exp
exp(1)
```

Out[5]: 2.718281828459045

In [6]:
```
import math
math.log(1)
```

Out[6]: 0.0

In [8]:
```
from math import exp, log, log10
log10(3)
```

Out[8]: 0.47712125471966244

In [16]:
```
import numpy as np
a = np.array([(1,2,3), (4,5,6)])
b = np.ones((2,3), dtype=np.int32)

'''
a =
1 2 3
4 5 6

b =
1 1 1
1 1 1

a.T =
1 4
2 5
3 6

1 4   1 1 1
2 5   1 1 1
3 6

5 5 5
7 7 7
9 9 9

1 2 3   1 1
4 5 6   1 1
        1 1

6  6
15 15
'''

print a
print b
print type(a)

print np.dot(a.T, b)
print np.dot(a, b.T)
```

```
[[1 2 3]
 [4 5 6]]
[[1 1 1]
 [1 1 1]]
<type 'numpy.ndarray'>
[[5 5 5]
 [7 7 7]
 [9 9 9]]
[[ 6  6]
 [15 15]]
```

In [39]:
```
import requests
# JSON: JavaScript Object Notation
r = requests.get('http://api.openweathermap.org/data/2.5/forecast/city?id=1675107&APPID=4688d730b381f8c77c6384835886fa88 (http://api.openweathermap.org/data/2.5/forecast/city?id=1675107&APPID=4688d730b381f8c77c6384835886fa88)')
cc = r.json()
# print cc['city']
print cc['list'][0]
```

```
{u'clouds': {u'all': 20}, u'rain': {u'3h': 0.41}, u'sys': {u'pod':
u'd'}, u'dt_txt': u'2016-04-20 06:00:00', u'weather': [{u'main': u'Rai
n', u'id': 500, u'icon': u'10d', u'description': u'light rain'}], u'd
t': 1461132000, u'main': {u'temp_kf': 3.95, u'temp': 301.79, u'grnd_lev
el': 903.81, u'temp_max': 301.79, u'sea_level': 1027.79, u'humidity': 7
0, u'pressure': 903.81, u'temp_min': 297.838}, u'wind': {u'speed': 0.7
6, u'deg': 3.50085}}
```

In [ ]:

產生無止境的數列

In [46]:
```
def fib():
    n, i, j = 1, 0, 1
    while True:
        yield n, j
        n, i, j = n+1, j, i+j

for i, x in fib():
    if i > 20:
        break
    print i, x
```

```
1 1
2 1
3 2
4 3
5 5
6 8
7 13
8 21
9 34
10 55
11 89
12 144
13 233
14 377
15 610
16 987
17 1597
18 2584
19 4181
20 6765
```

list : map reduce

In [50]:
```
num_list = [0,1,2]
#['0','1','2']
str_num_list = map(str, num_list)
print str_num_list

# exercise  + 1
# lambda x: x+1
num_list_1 = map(lambda x: x+1, num_list)
print num_list_1

# exercise
# [0,1,2]
# [(0,'0'),(1,'1'),(2,'2')]
num_list_p = map(lambda x: (x,str(x)), num_list)
print num_list_p
```

```
['0', '1', '2']
[1, 2, 3]
[(0, '0'), (1, '1'), (2, '2')]
```

In [54]:
```
num_list = [4,5,6,7]
print reduce(lambda a,x: a+x, num_list, 0)
# exercise *
print reduce(lambda a,x: a*x, num_list, 1)
# exercise (22, '4567')
print reduce(lambda a,x: (a[0]+x , a[1]+str(x) ) , num_list, (0, '') )
```

```
22
840
(22, '4567')
```

look-and-say sequence

11 21 1211 111221 312211 13112221 1113213211

In [ ]:
```
# run-length encoding
# '11122233' =>  [(3,'1'), (3,'2'), (2,'3')]
# [(3,'1'), (3,'2'), (2,'3')] => '11122233'
```

In [70]:
```
def rld(lst):
    '''
    '''
    return reduce(lambda a,x: a+x[0]*x[1] , lst, '')

rld([(3,'1'), (3,'2'), (2,'3')])

def f(a, x):
    '''
    '''
    if not a:
        a.append((1,x))
        return a
    else:
        p = a[-1]
        if x==p[1]:
            a[-1] = (p[0]+1,x)
            return a
        else:
            a.append((1,x))
            return a

def rle(s):
    return reduce(f, list(s), [])

rle('11122233')

def looksay(s):
    while True:
        r = rle(s)
        s = reduce(lambda a,x: a+str(x[0])+x[1] , r, '')
        yield s

for x in looksay('11'):
    if len(x)>20:
        break
    print x
```

```
21
1211
111221
312211
13112221
1113213211
31131211131221
13211311123113112211
```

In [58]:
```
11 => [(2,'1')] => '2'+'1'  '21' => [(1,'2'),(1,'1')]   '1'+'2'+'1'+'1'
'1211'
'1211' => [(1,'1'), (1,'2'), (2,'1')]  '11'+'12'+'21'  '111221'
21 1211 111221 312211 13112221 1113213211
```

Out[58]:
```
['1', '1', '1', '2', '2', '2', '3', '3']
```

In [75]:
```
def selection_sort(lst):
    outlst = []
    while lst:
        # v = min(lst)
        v = reduce(lambda a,x: x if x<a else a, lst, lst[0])
        lst.remove(v)
        outlst.append(v)
    return outlst

print selection_sort([4,3,8,7,5,2,4,1,6])
print sorted([4,3,8,7,5,2,4,1,6])
```

```
[1, 2, 3, 4, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 4, 5, 6, 7, 8]
```

In [78]:
```
def insertion_sort(lst):
    outlst = []
    for i in lst:
        for idx, val in enumerate(outlst):
            if i < val:
                outlst.insert(idx,i)
                break
        else:
            outlst.append(i)
    return outlst

print insertion_sort([4,9,8,7,5,2,4,1,6])
print sorted([4,9,8,7,5,2,4,1,6])
```

```
[1, 2, 4, 4, 5, 6, 7, 8, 9]
[1, 2, 4, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```

```

In [95]:
```
def f(a,x):
    i, lst = a
    for idx, val in enumerate(lst[i:]):
        if x<val:
            lst.insert(i+idx, x)
            out = (i+idx, lst)
            break
    else:
        lst.append(x)
        out = (len(lst)-1, lst)
    return out

def merge(u, v):
    '''
    u = [1,4,5]
    v = [2,3,7]
    reurn [1,2,3,4,5,7]
    '''
    _, r = reduce(f, v, (0, u))
    return r

def merge_sort(lst):
    if len(lst)<=1:
        return lst
    lst1 = merge_sort(lst[:len(lst)/2])
    lst2 = merge_sort(lst[len(lst)/2:])
    return merge(lst1,lst2)

print merge_sort([4,15,9,14,8,7,5,2,4,1,6,10,12,11])
```

```
[1, 2, 4, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15]
```

In [92]:
```
def f(a,x):
    i, lst = a
    for idx, val in enumerate(lst[i:]):
        if x<val:
            lst.insert(i+idx, x)
            out = (i+idx, lst)
            break
    else:
        lst.append(x)
        out = (len(lst)-1, lst)
    return out

def merge(u, v):
    '''
    u = [1,4,5]
    v = [2,3,7]
    reurn [1,2,3,4,5,7]
    '''
    _, r = reduce(f, v, (0, u))
    return r

print merge([1,3,8,9],[2,4,5,6,7])
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```
def merge(u, v):
    '''
    u = [1,4,5]
    v = [2,3,7]
    reurn [1,2,3,4,5,7]
    '''
    _, r = reduce(f, v, (0, u))
    return r
```

In [101]:
```
def quicksort(lst):
    if not lst:
        return []
    a = lst[0]
    lst1 = filter(lambda x:x<a, lst[1:])
    lst2 = filter(lambda x:x>=a, lst[1:])
    return quicksort(lst1) + [a] + quicksort(lst2)

print quicksort([4,2,1,8,3,6,5,7])
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

In [98]:
```
print filter(lambda x: x>4, [4,2,1,8,3,6,5,7])
```

```
[8, 6, 5, 7]
```

估計圓周率

In [136]:
```
import random
import time
# [0, 1)
np = 3*10**7
s = time.time()
print 4.0*sum([1 for _ in range(np) if random.random()**2+random.random
()**2<1])/np
e = time.time()
print e-s
```

```
3.14180573333
28.256000042
```

電腦猜數字

In [49]:
```
import random, re
from functools import partial
def f(na, nb, x, y):
    '''
    x ~ y
    na = 1
    nb = 2
    x = (3,5,7,9)
    y = (9,5,6,7)
    '''
    for i, v in enumerate(x):
        if v == y[i]:
            na = na - 1
        for j, w in enumerate(y):
            if i<>j and v==w:
                nb = nb -1
    if na==0 and nb==0:
        return True
    else:
        return False

nums = [(x,y,z,w) for x in range(1,10) for y in range(0,10) for z in ran
ge(0,10) for w in range (0,10) if len(set([x,y,z,w]))==4]
n = 8
while n>0:
    guess = random.choice(nums)
    print ''.join(map(str, guess))
    res = map(int, filter(lambda x: x is not '', re.split('[, AB]', raw_
input('?A?B'))))
    print res[0], res[1]
    nums = filter(partial(f,res[0],res[1],guess), nums)  # f()  [(1,2,3,
4),(5,6,7,8)]
    if len(nums)<=50 :
        print nums
        break
    n = n-1
```

```
6075
?A?B1A1B
1 1
7018
?A?B0 2
0 2
2671
?A?B0 2
0 2
[(1, 2, 0, 5), (1, 7, 3, 5), (1, 7, 4, 5), (1, 7, 9, 5), (1, 8, 6, 5),
(6, 1, 0, 3), (6, 1, 0, 4), (6, 1, 0, 9), (6, 1, 3, 0), (6, 1, 4, 0),
(6, 1, 9, 0), (6, 2, 8, 0), (6, 3, 8, 7), (6, 4, 8, 7), (6, 7, 8, 3),
(6, 7, 8, 4), (6, 7, 8, 9), (6, 8, 0, 2), (6, 8, 2, 0), (6, 8, 3, 7),
(6, 8, 4, 7), (6, 8, 9, 7), (6, 9, 8, 7), (8, 1, 6, 5), (8, 7, 2, 5)]
```

In [47]:

```
from functools import partial
def g(a,b):
    return 2*a+3*b
print g(3,5)
g3 = partial(g,3)
# g3 = g(3, ?)
# g3(5) = g(3, 5)
print g3(5)
```

```
21
21
```

In [44]:

```
def f(na, nb, x, y):
    '''
    x ~ y
    na = 1
    nb = 2
    x = (3,5,7,9)
    y = (9,5,6,7)
    '''
    for i, v in enumerate(x):
        if v == y[i]:
            na = na - 1
        for j, w in enumerate(y):
            if i<>j and v==w:
                nb = nb -1
    if na==0 and nb==0:
        return True
    else:
        return False

na, nb = 1, 1
x = (3,5,7,9)
y = (4,5,6,7)
f(na, nb, x, y)
```

Out[44]:

```
True
```

Graphs

找路徑 Depth First Search

In [5]:
```
dgraph = {'A':{'B','C'}, 'B':{'D'}, 'C':{'D','E'}, 'D':{'A','E'}, 'E':
{'B'} }
def haspath(g, src, dst, path=[], best_pt=None):
    if src == dst:
        path = path + [dst]
        if best_pt==None:
            return path
        elif len(path)<len(best_pt):
            return path
        else:
            return best_pt

    if src not in g.keys():
        return None
    path = path + [src]
    # path.append(src)
    for n in g[src]-set(path):
        # if n not in path:
        best_pt = haspath(g, n, dst, path, best_pt)
    return best_pt

print haspath(dgraph, 'A', 'E')
print haspath(dgraph, 'D', 'B')
print haspath(dgraph, 'E', 'A')
#print dgraph['A']
#print dgraph['D']
# path =['D','A'] 'C'
```

```
['A', 'C', 'E']
['D', 'A', 'B']
['E', 'B', 'D', 'A']
```

In [59]:
```
print dgraph['D']
print dgraph['A']
```

```
set(['A', 'E'])
set(['C', 'B'])
```

Graphs

找路徑 Breadth First Search

In [73]:
```
dgraph = {'A':{'B','C'}, 'B':{'D'}, 'C':{'D','E'}, 'D':{'A','E'}, 'E':
{'B'} }
def bfs(g, src, dst):
    queue = [(src,[src])] #[('A',['A'])]
    while queue:
        c, path = queue.pop(0) # c='B', path=['A','B'], queue=[('C',
['A','C'])]
        for n in g[c] - set(path):  # g[c]=g['B']={'D'}
            if n==dst:
                yield path+[dst]  # ['A','B','D']
            else:
                queue.append( (n, path+[n]) )  #


for p in bfs(dgraph, 'D', 'B'):
    print p

print map(len, bfs(dgraph, 'D', 'B'))
print sorted(bfs(dgraph,'D','B'), key=len)
```

```
['D', 'A', 'B']
['D', 'E', 'B']
['D', 'A', 'C', 'E', 'B']
[3, 3, 5]
[['D', 'A', 'B'], ['D', 'E', 'B'], ['D', 'A', 'C', 'E', 'B']]
```

In [68]:
```
queue = ['A']
queue.append('B')
print queue
print queue[0]
print queue
# help(queue.pop)
print queue.pop(0)
print queue
stack = ['A']
stack.append('B')
print stack
print stack.pop()
print stack
#
```

```
['A', 'B']
A
['A', 'B']
A
['B']
['A', 'B']
B
['A']
```