Python優缺點

- 免費、開源
- 語法非常簡單、優雅，適合剛開始學程式的新手，比起C、JAVA等語言叫簡潔，可謂以一打十。
- 應用廣泛，學會python後可以用在網頁設計(Django)、網路爬蟲(Request、beautifulsoup)、漂亮的視覺化(matplotlib、seaborn、plotly)、地圖(basemap、geoscience)、 圖像處理(PIL、opencv)、機器學習(sklearn)、深度學習(tensorflow、pytorch) …超級多你可以做的XDD
- 可移植性高
- 缺點是執行速度不像C語言、fortran等需編譯的語言來的快

Anaconda 簡介

- Anaconda是一款由Python和C語言開發而成的資料科學集成式開發環境，可安裝在windows、mac、linux系統上
- 開源、免費
- jupyter notebook可使用Pyahon、R、Julia語言
- 可以使用conda來管理模組及python虛擬環境
- 有安裝好jupyter notebook、Spyder等開發環境
- 已經預先安裝好常用到的套件

如何使用**conda**或是**pip**安裝套件

- 如果有下載Anaconda的，通常已經有安裝好conda的套件，windows在cmd打conda install XXX套件，mac在終端機打conda install XXX套件，mac在終端機打conda套件，基本上就可以安裝套件，有時候是conda install -c conda-forge XXX套件，當你要安裝某個套件時先google查清楚如何安裝
- 可以使用conda update XXX套件或是conda remove XXX來管理python中的套件包
- 除了conda，還有pip or pip3來安裝套件，指令很類似，最常用就是pip install

- 大家可以使用conda install -c conda-forge tqdm 來練習安裝tqdm的套件

Jupyter notebook 簡介

Jupyter 常用快捷鍵

功能	Windows	MacOS
換行編輯	Enter	Enter
執行此單元	Ctrl + Enter	Ctrl + Enter
執行此單元後移至下一單元	Shift + Enter	Shift + Enter
執行此單元後新增下一單元	Alt + Enter	Option + Enter
儲存	Ctrl + S	Command + S

使用 ? 表示變數詳細說明 e.g. ?a
使用 ! 表示執行系統指令 e.g. !ls
可使用nb_extension來加強jupyter notebook的功能
迅速將jupyter notebook轉為slides

In [35]:
```
a = 'Hello'
a?
```

Python: Basic syntax

- Python中index都是從0開始
- "#"符號後是註解(comment)，可以在一列的開頭或中間加入
- """
  使用三個"，將要註解的地方包起來，變成多行註解
  """
- 變數取名請避開!
  - and,exec,not,as,finally,or,assert,for,pass,expcept
  - break,from,print,class,global,raise,continue,if,return
  - def,import,try,del,in,while,elif,is,with,else,lambda,yield
- 縮排(Indentation)視為不同的block(在判斷式或迴圈的段落中使用...)
  - 縮排可以用tab或數個空格(至少一個空格)
  - 空格的數量不同，視為不同的block
- python的每個變數視為一個object

In [5]:
```
# 在這邊的是註解
"""
Hello world~~~
在這邊的是多行註解
"""
# 縮排範例
a = 10
if a > 5:
    print("True")
else:
    print("none")
```

```
True
```

變數型態 (Variables)

- 布林(Boolean) e.g. True/False
- 整數(integer) e.g. 24, 100 轉換成數字: int(a)
- 浮點數(float) e.g. 1.2, 3.4 轉換成浮點數: float(a)
- 字串(string)，可用單引號或雙引號括住 e.g. "Hello", 'Hello' 轉換成字串: str(a)
- 確認變數型態或資料型態可以使用type()指令

In [12]:
```
vari = 'Hello world'
print(vari, ' TYPE: ', type(vari))
```

```
Hello world  TYPE:  <class 'str'>
```

字串處理

In [10]:
```
s1 = "Hello world"
s2 = "ML"
print(s1)
print(s1[0:4])
print(s1[:6] + s2)
print(s1.replace('World', 'Kitty'))
```

```
Hello world
Hell
Hello ML
Hello world
```

In [11]:
```
s1 = 'Hello World'
print(s1.split()) # default 是空格
print(s1.upper())
print(len(s1))
# other methods: strip(), splitlines(), etc
```

```
['Hello', 'World']
HELLO WORLD
11
```

coding style: single quotes v.s. double quotes in Python check out here

常見的資料結構 (Data Structure)

Tuple

- 不可更動內容、功能少、占空間少
- 創建：使用小括號 () e.g. X=(10, 12)
- 轉換成list e.g. y = list(X)

List - 串列

- 可以放置任何的資料型態
- list沒有element wise的功能！！！
- 創建空List
  - sample_list = []
  - sample_list = list()
- 初始項目
  - sample_list = [10, 'test', 3.14, [10, 12]] (以逗號隔開，資料型態不拘)
  - 串列也可以放進串列
- sub-list或是sub-tuple語法類似取出字串局部: x[0:2] 取出index=0, 1的值

list 的新增

- list在最後增加項目: sample_list.append(10)
- 在串列最後增加串列: sample_list.extend([10, 12])
- 指定位置插入項目: list.insert(index, item)

list的刪除

- 使用del刪除整個list e.g. del sample_list
- 刪除指定位置項目: del e.g. del sample_list[1]
- 移出(回傳)並刪除list項目: pop(index)
- 預設pop()是移出最後一項，若pop(0)則是移出第一項

list的項目位置與排序

- 找出第一次出現的位置: list.index(value)
- 判斷值是否在list中: value in list
- 根據值的內容來排序(sorting): sorted(list)

In [25]:
```
Alist = [0, 1] # 創建list
# append和extend的不同
Alist.append([1, 2, 3])
print(Alist)
Alist.extend([1, 2, 3])
print(Alist)
## 取出list中的list
print(Alist[2][0], Alist[2][1], Alist[2][2])
```

```
[0, 1, [1, 2, 3]]
[0, 1, [1, 2, 3], 1, 2, 3]
1 2 3
```

In [23]:
```
# list 沒有element wise功能
# 兩個list 相加
Alist = [1, 2, 3]
Blist = [1, 2, 3]
Clist = Alist + Blist
print(Clist)
# 一個list乘以某個整數
Clist = Alist * 5
print(Clist)
```

```
[1, 2, 3, 1, 2, 3]
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

In [26]:
```
# list inset, pop用法
Alist = [1, 2, 3]
Alist.insert(2, 'xx')
print(Alist)
Alist.pop(2)
print(Alist)
```

```
[1, 2, 'xx', 3]
[1, 2, 3]
```

In [34]:
```
## list項目位置與排序
Alist = [3, 2, 1]
print(Alist.index(3))
print(2 in Alist)
print('-----------------')
## 排序
print(sorted(Alist))
print(Alist)
print('-----------------')
print('使用list.sort()會讓Alist排序並儲存排序後的樣子')
Alist.sort()
print(Alist)
```

```
0
True
-----------------
[1, 2, 3]
[3, 2, 1]
-----------------
使用list.sort()會讓Alist排序並儲存排序後的樣子
[1, 2, 3]
```

Dictionary - 字典

- 以key-value(鍵-值) pair形式儲存的資料結構
- key必須獨一無二(unique)，若　有重複key會自動覆蓋原key-value pair
- 創建：使用大括號{}，內容可存多種資料型態和結構
- 取值：dictionary[key]

新增與刪除

- 新增字典 e.g. dic.update({'a': b}) or dic['a'] = b
- 刪除項目： del e.g. del dic['a'], or del dic (刪除整個字典)

取值

- 判斷項目(key)是否存在於字典： in e.g. 'a' in dic
- 獲取字典中的資料(皆非list，可用list()轉換)
  - 取得所有的key： dic.keys()
  - 取得所有的value： dic.values()

- 取得所有的key-value pair： dic.items()

In [24]:
```
dicts = {'Jason': 15, 'Hello': 16, 'lists': [0, 1, 2]}
jason = dicts['Jason']
list0 = dicts['lists'][0]
print(jason, list0)
dicts.update({'World': 19})
all_keys = dicts.keys()
print(all_keys)
```

```
15 0
dict_keys(['Jason', 'Hello', 'lists', 'World'])
```

Arithmetic Operators

- 加法 + ， a = a + b → a += b
  字串相加: 形成新的字串
  x = "Hello", y = "World" x+y → "HelloWorld"
- 減法 -, a = a -b → a -= b
- 乘法 *x="Yo!" y=x* 3 → "Yo!Yo!Yo!"
- 除法 /
- 次方 **
- 取餘數 %
- 整除至最近整數 //
- 算數運算符號可以有 +=、-=、*=、/=、%=、**=、//=

Python built-in function

- sqrt(x)
- abs(x)
- max(x1, x2, ...)
- min(x1, x2, ...)
- ceil(x)：無條件進位
- floor(x)：無條件捨去
- round(x, n)：四捨五入到小數點
  下第n位(n可省略)

下面函數必須先引入**math**工具才能使用

import math

- math.exp(x)
- math.log(x)
- math.log10(x)
- math.cos(x), ...
- math.degrees(x): converts angle x from radians to degrees
- math.radians(x): converts angle x from degrees to radians

條件式

```
if condition1:
    statements
elif condition2:
    statements
else:
    statements
# 一句條件式
a if a > b else b
# 或是
if a > b: print(a)
```

- 邏輯判斷符號 ==, !=, >, >=, <, <=
- 邏輯運算 and, or, not

迴圈

```
# while 迴圈
while condition:
    statements

## 最一般的for迴圈
# range經常用來控制迴圈次數
# range(n) 從0到(n-1)
# range(nmin,nmax,interval) 從nmin到nmax-1

for i in range(10):
    statements
## 或是直接迴圈list
for data in Alists:
    statements
## 或是使用enumerate
for index, data in enumerate(Alists):
    statements
## 有多個list要迴圈時可使用zip
for i, j in zip(a,b):
    statements
## reverse number
for i in reversed(range(5)):
    statements
## 特別的for寫法
listA = [i for i in range(10)]
```

可以使用tqdm模組來出現bar條

In [1]:
```
# tqdm example
from tqdm import tqdm
for i in tqdm(range(100)):
```

```
    pass

Alist = [i for i in range(10)]
print(Alist)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 100/100 [00:00<?, ?it/s]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Exceptions Handling (python中特有的)

- 回溯資訊**(traceback)**就是python藉以告訴你執行期間發生了非預期情況的方式。在python的世界中，執行時期錯誤又稱為例外**(exception)**
- try/except機制，python包含了**try**陳述句，它的存在是為了提供一個方法，讓你得以有系統的處理執行時期的例外和錯誤。

```
try:
    可能會出現錯誤的程式碼
except: #要讓例外具體時，可以寫處理特定類型的錯誤
    錯誤時執行的程式碼(通常可以直些寫pass)
else:
    沒發生錯誤時，執行的程式碼片段
finally:
    不論有無發生錯誤時，都會執行的程式碼片段(當然也可以不加finally)
```

In [27]:
```
x = 1
y = 0
try:
    x / y
except ZeroDivisionError:  # ZeroDivisionError 不一定要寫，寫了是讓exception更清楚
    print('Error: zero division')
else:
    print('Result: ' + str(x/y) + ' (without error)')
finally:
    print('Done')
```

```
Error: zero division
Done
```

定義函式使用**def**

```
def 函式名稱(傳入參數1, 傳入參數2, 傳入參數3=10):
    運算...
    return 回傳值
```

- 傳入參數可以給default值，當沒有變數時傳入時使用default的值

In [1]:
```
def add_n(n1, n2, n3=0):
    sum_ = n1 + n2 + n3
    return sum_

## 使用函式
sum1 = add_n(1, 2)  # 此時使用default值0
print(sum1)

sum2 = add_n(1, 2, 5)
print(sum2)

sum3 = add_n(n1=1, n3=17, n2=10)
print(sum3)
```

```
3
8
28
```

Import module

很重要的指令，因為python非常常引入模組

使用方法:

- import numpy
- import numpy as np，之後就使用縮寫np.cos
- from numpy import cos
- from numpy import * # import所有numpy模組，不建議使用，import需要的模組就好

Numpy

Numpy Reference

NumPy is powerful package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities...
- In this class, we will mainly introduce the NumPy "array" features, plus a few mathematical functions. Before we use the NumPy - array in the python code, we need to first "import" the NumPy toolbox

NumPy是一個針對科學運算開發的python工具庫，提供方便的陣列功能與數學運算工具。 要使用NumPy相關的指令前，要先引入(啟動)NumPy

In [13]:
```
import numpy
# import numpy as np
A=numpy.arange(4)  # 引進numpy，可以開始用numpy相關的指令
print(A)
```

```
[0 1 2 3]
```

NumPy建立陣列的幾種方法

In [15]:
```
import numpy as np
# list to array
Alist = [1, 2, 3]
Alist = np.array(Alist)

#np.array  創造新陣列
A = np.array([1,2,3])   # 1-D陣列
B = np.array([[1,2,3],[4,5,6],[7,8,9]]) # 2-D陣列

print(A)
print(A.shape, B.shape)  # checking the size in each dimension of the arra
y
```

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

In [17]:
```
# show only a sub-set of A
print(A)
print(B[:,2])      # the entire third column of A
print(B[0:2,0:2])  # first two column of the top two rows
```

```
[1 2 3]
[3 6 9]
[[1 2]
 [4 5]]
```

In [18]:
```
c=np.zeros((3,2),dtype=int)  # 創造3x2的陣列，所有元素為整數0
print(c)
```

```
[[0 0]
 [0 0]
 [0 0]]
```

In [ ]:
```
# np.arange(end) or np.arange(start,end,interval) (default start=0, interva
l=1)

A=np.arange(10)     # 只給一個數值的話，代表從0開始，到給定的數值之間、間隔為1的一串
整數
print(A)
B=np.arange(1,10)   # 開始與結束之間、間隔為1的一串整數
print(B)
C=np.arange(1,10,2) # 開始與結束之間、用指定的間隔產生一串整數
print(C)
```

In [19]:
```
# np.linspace(start,end,n) # 在起始與結束的數值之間，均分成n等分的一串數值

A = np.linspace(1,10)    # default n=50
print(A)
B = np.linspace(1, 10 , 10)
```

```
print(B)
```

```
[  1.          1.18367347   1.36734694   1.55102041   1.73469388
   1.91836735   2.10204082   2.28571429   2.46938776   2.65306122
   2.83673469   3.02040816   3.20408163   3.3877551    3.57142857
   3.75510204   3.93877551   4.12244898   4.30612245   4.48979592
   4.67346939   4.85714286   5.04081633   5.2244898    5.40816327
   5.59183673   5.7755102    5.95918367   6.14285714   6.32653061
   6.51020408   6.69387755   6.87755102   7.06122449   7.24489796
   7.42857143   7.6122449    7.79591837   7.97959184   8.16326531
   8.34693878   8.53061224   8.71428571   8.89795918   9.08163265
   9.26530612   9.44897959   9.63265306   9.81632653  10.        ]
[  1.   2.   3.   4.   5.   6.   7.   8.   9.  10.]
```

NumPy 陣列運算

list中不能做element wise運算的問題

陣列的加減乘除（+ - * /）、內建基本數學函數（sine, cosine, exponential, log...）

Note:

- math library的sin, cos, exp...等函數，不能使用在array上，因此要使用Numpy提供的函數
- ' * ' means element-wise multiplication for arrays (相同位置的元素相乘)
- np.dot() is used for matrix multiplication

矩陣乘積--行與列相乘相加

In [ ]:
```
c=np.array([[1, 0], [0, 1]])
print(c)
d=np.array([[4, 1], [2, 2]])
print(d)

B=np.dot(c,d)  # matrix multiplication of two 2D arrays
print(B)
```

In [ ]:
```
# Example of basic math functions for numpy arrays
A=np.linspace(0,np.pi,6)  # generage a 6x1 array between 0 and pi
print(A)

B=np.cos(A)  # cosine (similar for sin, tan...)
print(B)

C=np.exp(A)  # exponential
print(C)

D=np.log(C)  # natural log
print(D)
```

NumPy陣列的基本操作

- np.reshape

np.transpose
- np.tile
- np.hstack, np.vstack
- np.columnstack

In [ ]:
```
#np.reshape  改變"形狀" (將 ixj的陣列變形為mxn,  但必須滿足 ixj = mxn)
A=np.arange(4)
print(A)
B= np.arange(4).reshape((2,2))  #將 1x4 陣列變成 2x2
print(B)
```

In [ ]:
```
#np.transpose 轉置 (維度交換)
C=np.transpose(B)
print(C)
```

In [ ]:
```
# np.tile(array,n) or np.tile(array,(n,m))     # 陣列自我複製
D=np.tile(A,4)      # 陣列自我複製 1xn次
print(D)
E=np.tile(A,(4,2))  # 陣列自我複製 nxm 次 (n是垂直方向重複次數, m是水平方向重複次數)
print(E)
```

In [ ]:
```
# np.hstack  np.vstack     # 不同陣列堆疊(維度大小要相合)
a = np.array([1,2,3])
b = np.array([4,5,6])
c= np.hstack((a,b))
print(c)

d= np.vstack((a,b))
print(d)
```

In [ ]:
```
 # 維度大小不合的陣列無法堆疊
d= np.vstack((a,c))
print(d)
```

In [ ]:
```
np.column_stack # 將數個1D陣列以直行排列成二維 combine multiple 1-D arrays as columns to form a 2-D array
e= np.column_stack((a,b,a,b))
print(e)
```