資料科學的工具使用（1）

參考資料:Standford CS231http://cs231n.github.io/python-numpy-tutorial/

複習一下quicksort

```
def quicksort(arr):

# recursion都要有一個終點
  if len(arr) <= 1:
    return arr

# 抓中間
  pivot = arr[len(arr) // 2]

# 把list依照大小分成三部分
  left = [ x for x in arr if x > pivot]
  middle = [ x for x in arr if x == pivot]
  right = [x for x in arr if x < pivot]

# 核心概念：
# 然後利用遞迴，不斷的一直分成有序三部分，越分越小，
# function call會一直stack up，最後全部都分好後，再不斷的組合起來，
  return quicksort(left) + middle + quicksort(right)

test_arr = [2,3,45,5,333,22,1,2,3,34,4]

quicksort(test_arr)
```

基本操作

```
# 確認python版本
!python --version


# assign variable
x = 3

print(type(x))
print(x)
x += 1
x *= 6
# python doesn't have X++ as Javascript
```

AND, OR, NOT, XOR

```python
t = True
f = False

print( t and f)
print( t or f)
print( not t)

# XOR
print( t != f)
```

Basic string op

```
s = "hello"
# s.capitalize()
# s.upper()
# s.replace('l', 'X')

list(range(5))
```

numpy baisc

```python
import numpy as np

list(range(5)) # [0, 1, 2, 3, 4]

a = np.array([1,2,3])
type(a)

# see row,col pair
a.shape

# access element
a[2]
```

cont.

```
# 2 by 3 matrix
b = np.array([[1,2,3],
             [4,5,6]])

# check shape
b.shape

# access element
b[1,2]

# create diagonal matrix
c = np.eye(8)
c
```

Subarry from original array

```
# get subarray from ori-array,
# modify the subarry also change the ori-array

a = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [9,10,11,12]])
a

b = a[0:2, 1:3]  # row 0 to 2, w/o 2  and col 1 to 3, w/o 3
b
```

Integer array indexing

另一個access elt 的操作方法

```
# Integer array indexing

# create a matrix first
a = np.array([[1,2],
              [3, 4],
              [5, 6]])

# like, row-col pair (0,0), (1,1), (2,0) to access elt
a[[0,1,2],
  [0,1,0]]

# equvalient to this
np.array([a[0, 0], a[1, 1], a[2, 0]])

# a[[0,1,2],
#   [1,0,1]] # 2,3 6
```

Integer array indexing example

```
# arrange is another way to create list
# np.arange(4)
# np.arange(1,3)
# np.arange(1,6,2)
# np.arange(1,2,0.1)

a = np.array([[1,2,3],
              [4,5,6],
              [7,8,9],
              [10, 11, 12]])

b = np.array([0,2,0,1])

# np.arange is just another way for creating list
a[np.arange(4),b]

# [0,1,2,3]
# [0,2,0,1]

# you can also change the element
a[np.arange(4),b] += 100
a
```

Boolean index in matrix

```
# so you can create a treu/false matrix with given condition
a = np.array([[1,2],
              [3, 4],
              [5, 6]])
b = a > 2
b
# you can also feed this matrix into a and ouput
# only the true value list / or rank 1 array

a[b]

# or more succceinetly
a[a>2]
```

dot prodcut and elt-wise multiplication

you can define data type with `dtype=np.float64`
another one is `dtype=np.int64`

```
x = np.array([[1,2],
              [3,4]], dtype=np.float64)

y = np.array([[5,6],
              [7,8]], dtype=np.float64)

v = np.array([9,10])
w = np.array([11, 12])

print(x.dot(y))
print(x * y)
```

sum up array by row/col

```
x = np.array([[1,2],
              [3,4]])

print(np.sum(x))

# add up col and ouput np rank1 array
print(np.sum(x, axis=0))
type(np.sum(x, axis=0))

# add up row
print(np.sum(x, axis=1))
```

transpose

```
# transpose

x = np.array([[1,2],
              [3,4]])
x.T
```

reshape

we use reshape to match elt-wise and conduct outer product

```
v = np.array([1,2,3])

# both way are fine
v1 = np.reshape(v, (3,1))  # v.reshape(3,1)

w = np.array([4,5])
print(v1)
print(w)

# outer product
v1 * w
```

amazing broadcasting

```
x = np.array([[1,2,3],
              [4,5,6]])

v = np.array([1,2,3])

# broadcasting
# it will auto help me make two matricesthe same rank
# and compute
x + v
```

broadcast example

```
x = np.array([[1,2,3],
              [4,5,6]])

w = np.array([4,5])

# x shape is 2,3
# x transpose will be 3,2 and we can broadcast against w
# then we transpose back

# (x.T + w).T

# or you can reshape w to 2, 1 and brocast afainst x
# np.reshape(w,(2,1)) + x


# nyumpy also do array scalar multiplication via boardcasting

x * 2
```