

# Python安裝與環境建置

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 安裝Anaconda

## □ 安裝Anaconda套件包

<https://www.anaconda.com/products/individual/>

- 包含眾多科學、數學、工程、資料分析的 Python 套件
- 內建spyder 編譯器



Individual Edition

The screenshot shows the download page for the Individual Edition of Anaconda on Windows. At the top, it says "Windows" with the Windows logo. Below that, it says "Python 3.8". Underneath, there are two download links: "64-Bit Graphical Installer (466 MB)" and "32-Bit Graphical Installer (397 MB)". A red underline is drawn under the 64-bit link. Both links are blue and underlined.

Windows

Python 3.8

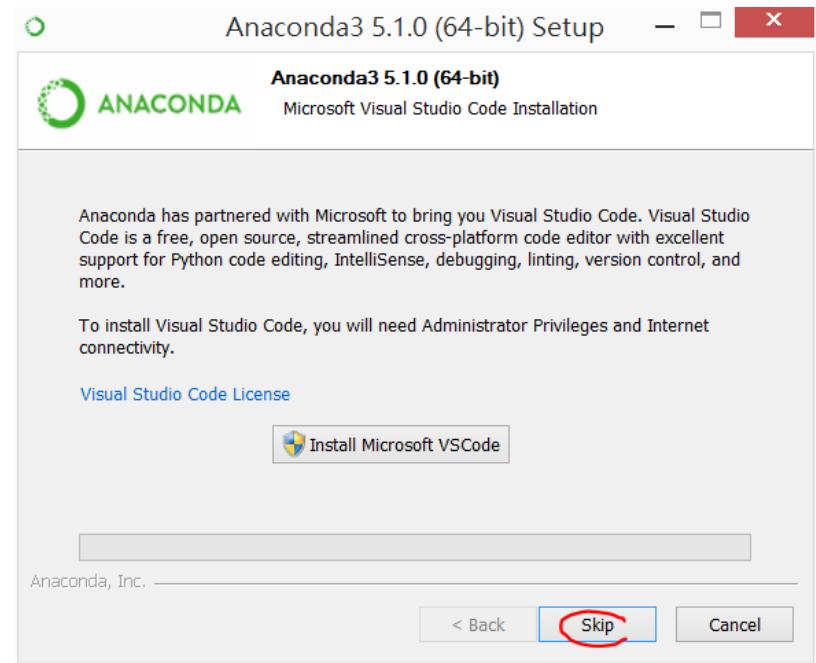
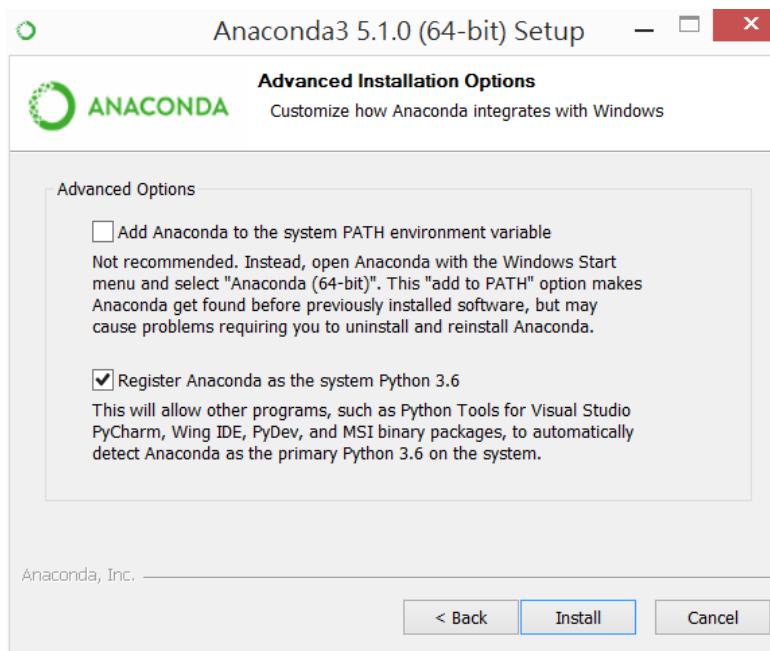
[64-Bit Graphical Installer \(466 MB\)](#)

[32-Bit Graphical Installer \(397 MB\)](#)

# 安裝Anaconda

## □ 執行Anaconda3-xxxx\_64.exe

- 設定路徑， C:\ProgramData\Anaconda3

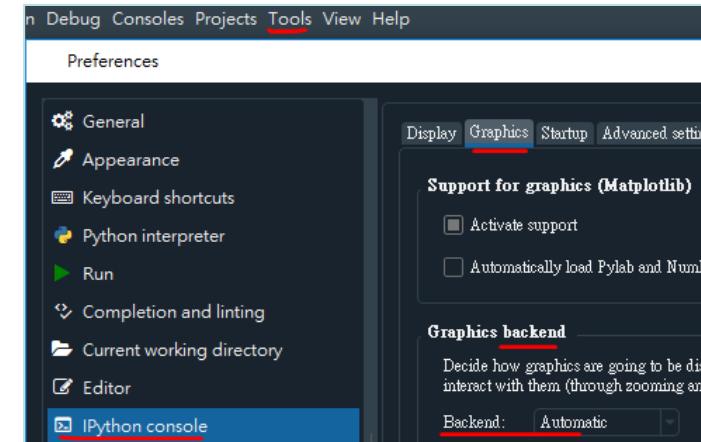


# 使用 Spyder 撰寫

- Jupyter Notebook: 編輯執行器
- Spyder: IDE整合編輯環境
- matplotlib.pyplot要能畫圖，選 Plots
  - Tools > Preferences > iPython console > Graphics > Graphics backend > Automatic

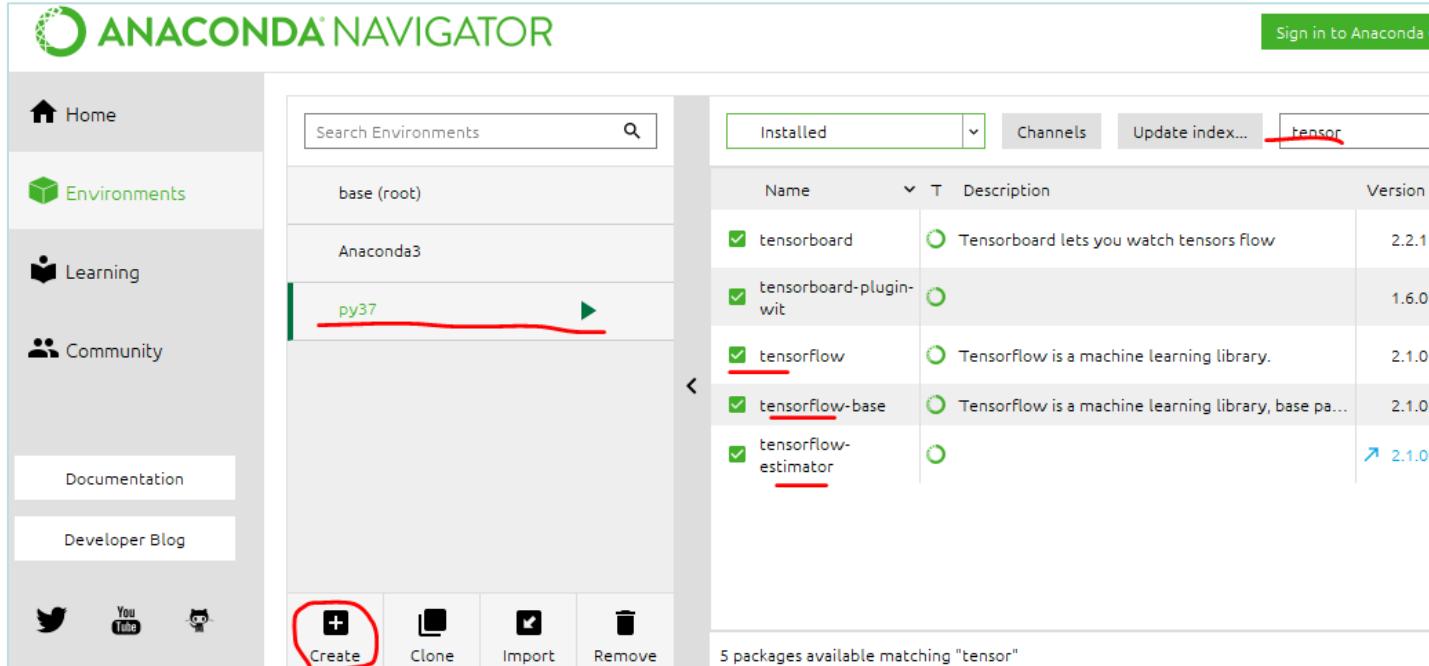
The screenshot shows the Spyder IDE interface. On the left is a code editor with Python code for generating plots. In the center is a figure window titled "Weight Distribution of US Presidents" containing two plots: a scatter plot of weight vs height and a line plot of number vs height. At the bottom, tabs for "Variable explorer", "Help", "Plots" (which is highlighted with a red circle), and "Files" are visible. A status bar at the bottom shows "Console 1/A" and "Python 3.8.1 (default, Mar 2 2020, 13:06:26) [MSC v.1916 64 bit]".

```
import numpy as np
import matplotlib.pyplot as plt
data = np.genfromtxt('president_height_weight.csv')
heights = np.array(data[:,2])
weights = np.random.randint(50, 80, len(heights))
data = np.append(data, weights.reshape(-1, 1), axis=1)
fig = plt.figure()
ax = fig.add_subplot(2,1,1)
ax.scatter(data[:,2:3], data[:,1])
plt.title('Weight Distribution of US Presidents')
plt.xlabel('weight (kg)')
plt.ylabel('Number')
ax = fig.add_subplot(2,1,2)
ax.plot(data[:,2], color='b')
plt.title('Height Distribution of US Presidents')
plt.xlabel('height (cm)')
ax.set_ylabel('Number')
```



# 安裝 tensorflow

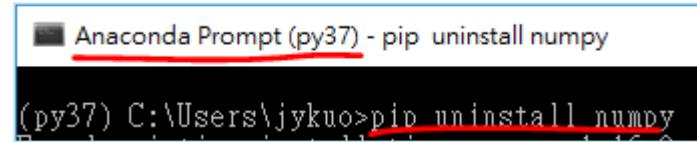
- 開始: Anaconda Prompt 執行 (安裝 python 3.7環境)
  - conda create -n py37 anaconda=2020.02 python=3.7
- UI介面安裝套件 Anaconda Navigator
  - Create 3.7版 (建議使用Prompt會產生捷徑)
  - (Environment – Not installed): tensor - Apply



# 安裝 tensorflow

## □ 先移除numpy：

- pip uninstall numpy
- (pip3 uninstall numpy) (只有 python 3，pip pip3沒分別)



```
Anaconda Prompt (py37) - pip uninstall numpy  
(py37) C:\Users\jykuo>pip uninstall numpy
```

## □ 再安裝1.17以下

- pip install numpy==1.16.0
- (pip3 install numpy==1.16.0)

```
#測試是否安裝 tensorflow
import tensorflow as tf
import numpy as np
print(tf.__version__)
print(np.__version__)
hello = tf.constant('hello tensorflow!')
sess = tf.compat.v1.Session()
print(sess.run(hello))
print(sess.run(hello).decode()) #解成 unicode字串
```

```
1.14.0
1.16.0
b'hello tensorflow!'
hello tensorflow!
```

# Prompt 安裝套件

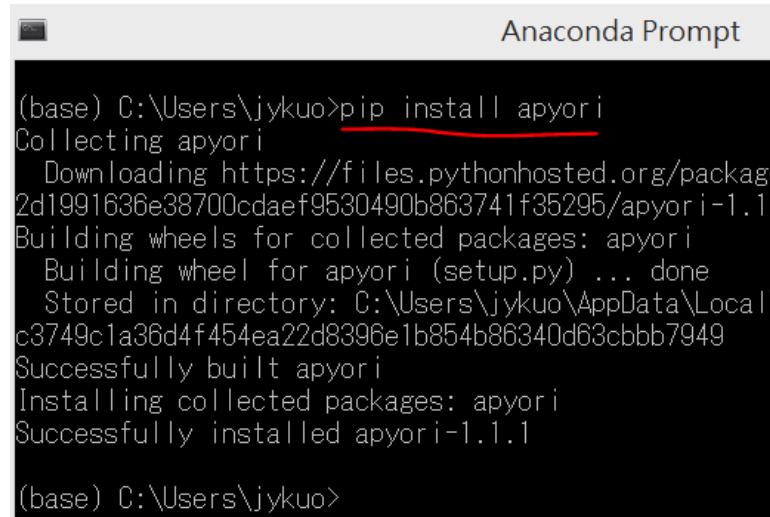
## □ Anaconda Prompt

### ○ 指令執行安裝套件

- conda install "套件名稱"
- conda update "套件名稱"
- conda update –all

### ○ 若為非anaconda repository package

- 安裝association rule套件
- pip install apyori



```
(base) C:\Users\jykuo>pip install apyori
Collecting apyori
  Downloading https://files.pythonhosted.org/packages/2d1991636e38700cdaef9530490b863741f35295/apyori-1.1.1.tar.gz
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Stored in directory: C:\Users\jykuo\AppData\Local\c3749c1a36d4f454ea22d8396e1b854b86340d63cbbe7949
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.1

(base) C:\Users\jykuo>
```

# Prompt 安裝 PyGame 套件

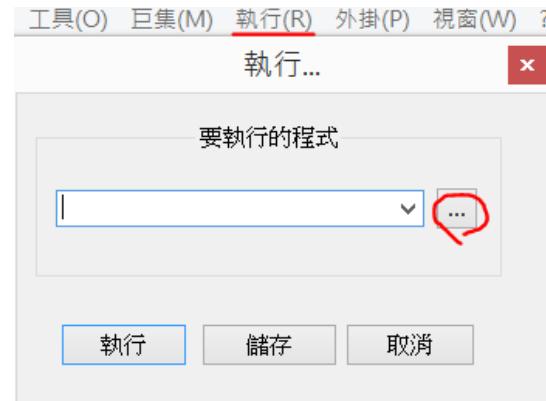
## □ 安裝 pygame

- 1. 開啟 anaconda prompt，檢視 python 版本。
- 2. 下載相對應版本 pygame 安裝包，儲存到 prompt 目錄。網址：<https://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame>
- 3. 安裝指令：python -m pip install --user pygame-1.9.4-cp37-cp37m-win\_amd64.whl

# 使用NotePad++撰寫

- Notepad++ 下載安裝

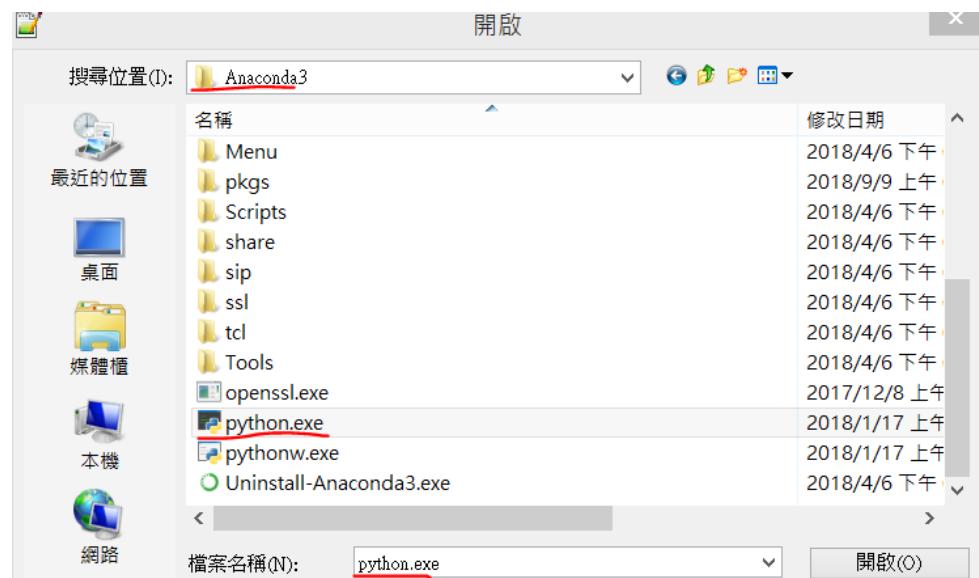
- 選單 - 執行



- 選擇 Anaconda3 目錄

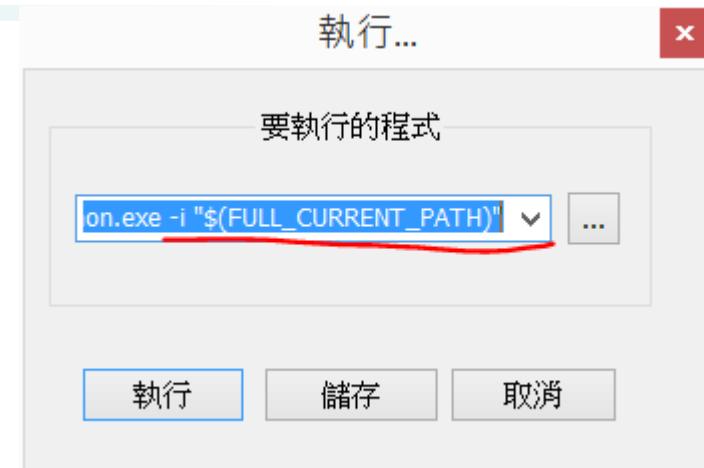
- 在C:\ProgramData\Anaconda3

- 選 python.exe



# 使用NotePad++撰寫

- 選擇 Anaconda3 目錄，
- 在python.exe 空一格填入
  - -i "\$(FULL\_CURRENT\_PATH)"



- 變成
  - C:\ProgramData\Anaconda3\python.exe -i "\$(FULL\_CURRENT\_PATH)"
- 儲存 – 快速鍵設定
  - python, CTRL-F5



# 使用NotePad++撰寫

- 輸入以下程式，儲存檔案，存檔類型->Python file

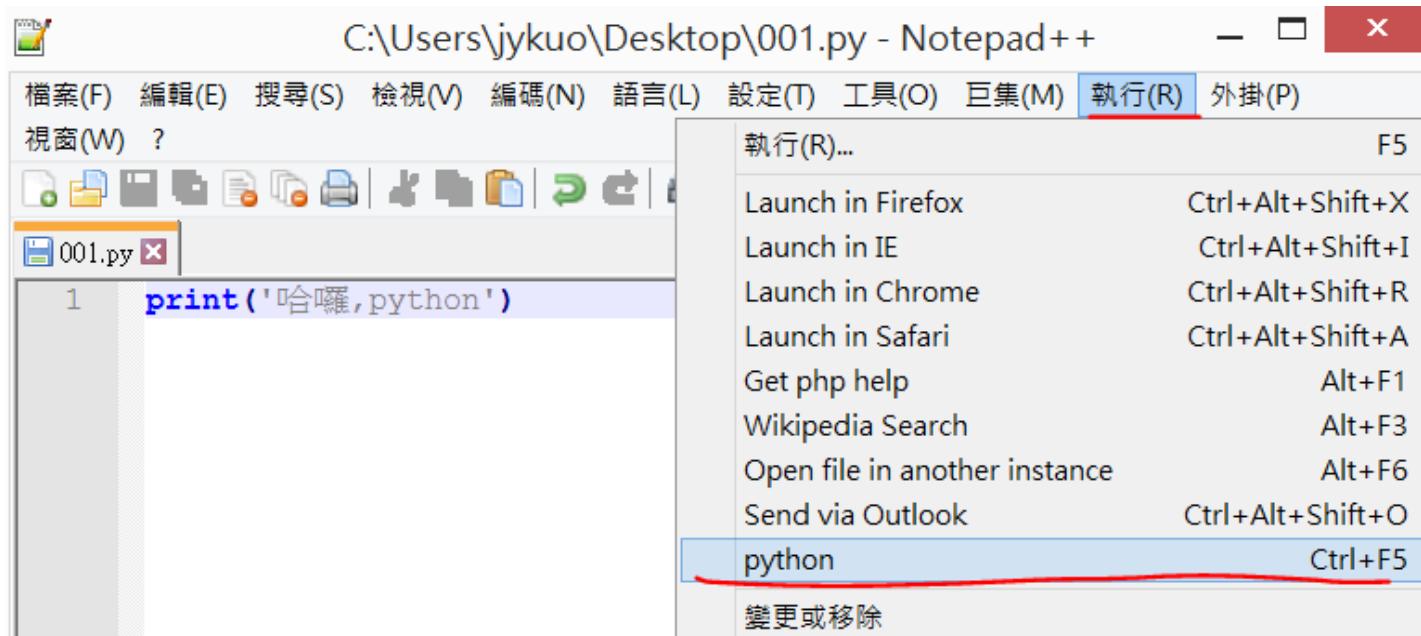
```
print('哈囉,python')
```



# 使用NotePad++撰寫

## □ 執行程式碼

- 選單-執行 – python 或 CTRL+F5



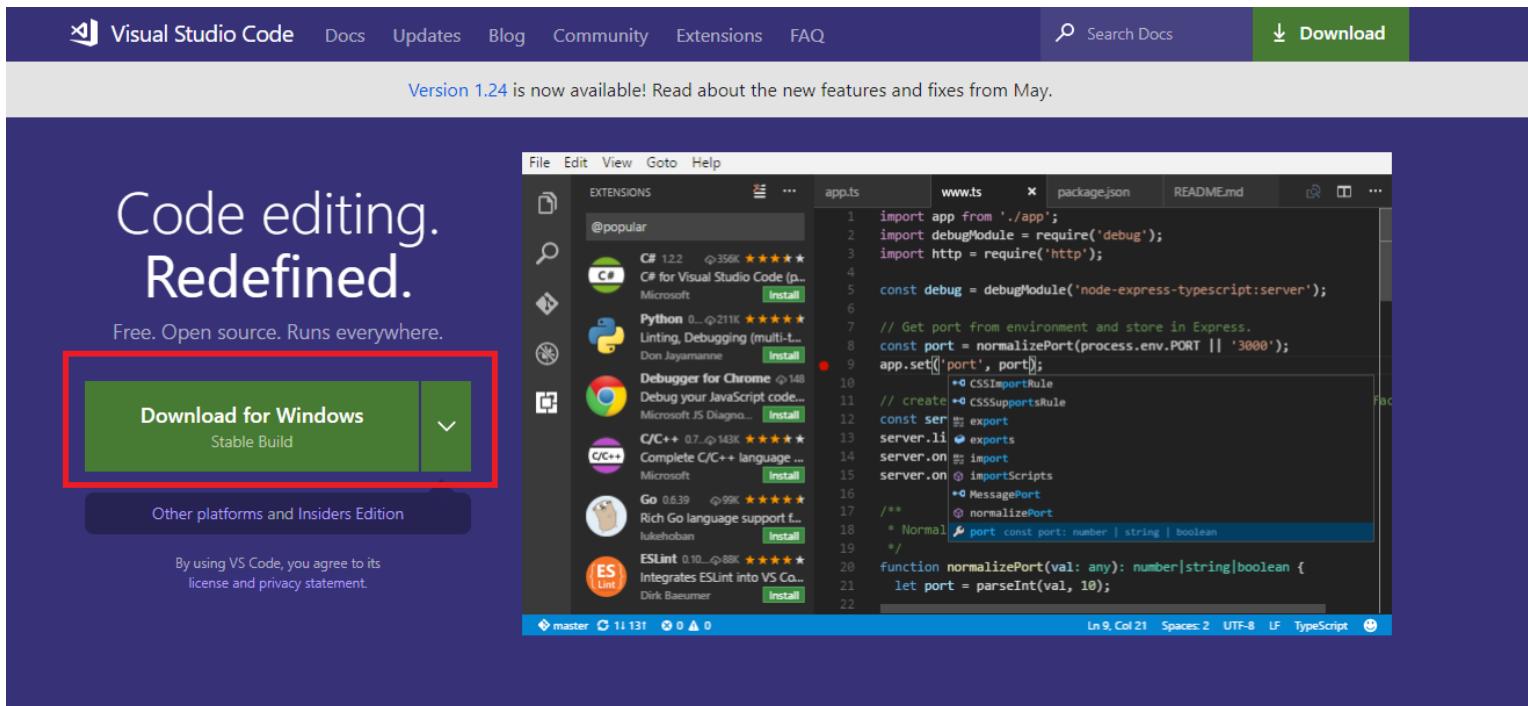
```
C:\ProgramData\Anaconda3\python.exe
哈囉,python
>>>
```

# 使用 VS Code 撰寫

- 安裝VS Code環境
  - 檢視-擴充功能
- 安裝pylint
  - PyLint 是一種廣泛使用的工具，可檢查 Python 程式碼中的錯誤，有助於撰寫良好的 Python 程式碼模式，因此 Visual Studio 已針對 Python 專案整合這項工具。
  - 打開command line > pip install pylint

# 使用 VS Code 撰寫

## □ 前往VS Code官網



# 使用 VS Code 撰寫

## □ 打開檢視 > 擴充功能



## □ 搜尋 python > 點擊安裝



# 安裝pyLint

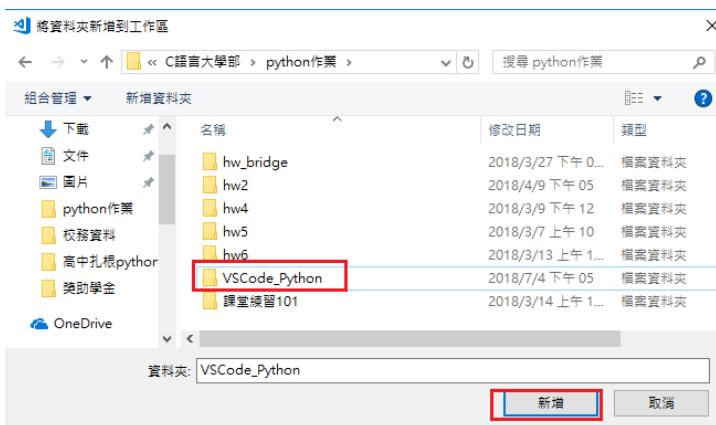
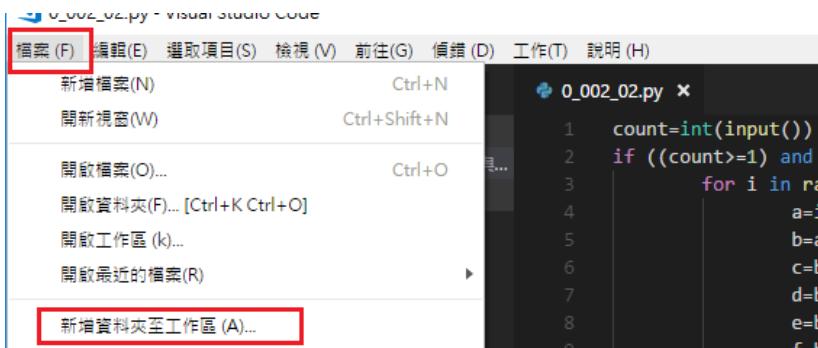
- PyLint 是一種廣泛使用的工具，可檢查 Python 程式碼中的錯誤，有助於撰寫良好的 Python 程式碼模式，因此 Visual Studio 已針對 Python 專案整合這項工具。
- 打開command line > pip install pylint

```
C:\Users\user>pip install pylint
Collecting pylint
  Downloading https://files.pythonhosted.org/packages/f2/95/0ca03c818ba3cd14f2dd4e95df5b7fa232424b7fc6
  /pylint-1.9.2-py2.py3-none-any.whl (690kB)
    100% |████████████████████████████████| 696kB 3.0MB/s
Collecting six (from pylint)
  Downloading https://files.pythonhosted.org/packages/67/4b/141a581104b1f6397bfa78ac9d43d8ad29a7ca43ea
  /six-1.11.0-py2.py3-none-any.whl
Collecting astroid<2.0,>=1.6 (from pylint)
  Downloading https://files.pythonhosted.org/packages/0e/9b/18b08991c8c6aaa827faf394f4468b8fee41db1f73
  /astroid-1.6.5-py2.py3-none-any.whl (293kB)
    100% |████████████████████████████████| 296kB 3.2MB/s
Collecting colorama; sys.platform == "win32" (from pylint)
```

# 安裝pyLint

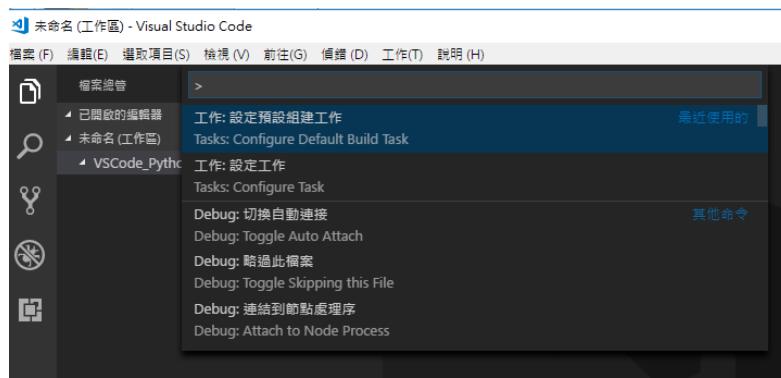
## □ 將資料夾新增到工作區

- Step 1. 點擊 檔案 > 新增資料夾至工作區
- Step 2. 新建一個資料夾，將其選擇為工作區



# 設定Compiler

- Step 1. 按 F1
- Step 2. 點選上方Task: Configure Default Build Task(如下圖)
- Step 3. 從範例建立tasks.json檔案
- Step 4. 點選Others，VS Code會產出一個tasks.json
- Step 5. 將tasks.json內容改為右下圖



```
// See https://go.microsoft.com/fwlink/?LinkId=733558
// for the documentation about the tasks.json file
"version": "2.0.0",
"tasks": [
    {
        "label": "python Compiler",
        "type": "shell",
        "command": "python",
        "args": ["${file}"]
    }
]
```

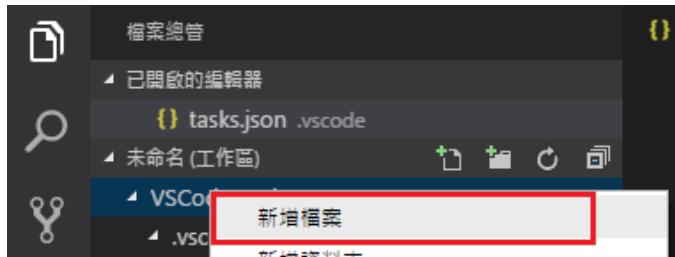
A screenshot of a code editor window titled "tasks.json". The code is a JSON file with the following content:

```
// See https://go.microsoft.com/fwlink/?LinkId=733558
// for the documentation about the tasks.json file
"version": "2.0.0",
"tasks": [
    {
        "label": "python Compiler",
        "type": "shell",
        "command": "python",
        "args": ["${file}"]
    }
]
```

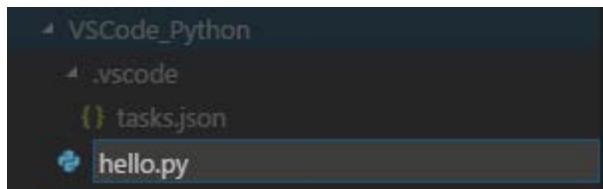
The code is numbered from 1 to 13 on the left.

# 新增python檔

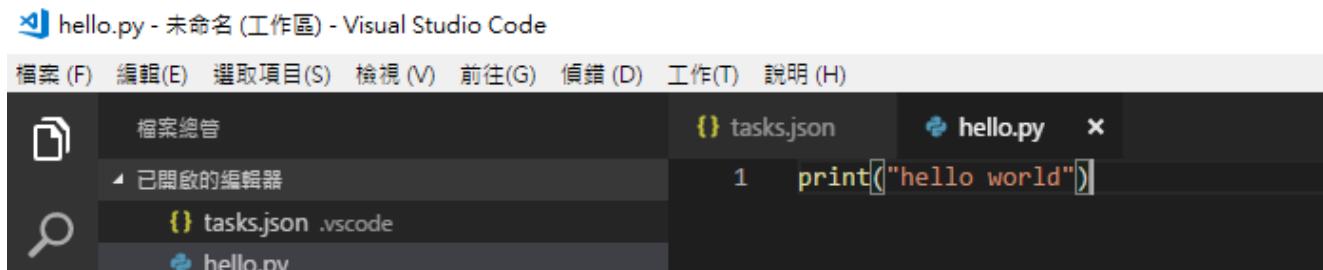
## □ Step 1. 工作區點選右鍵，新增專案



## □ Step 2. 建立python檔(取名為hello，副檔名.py)



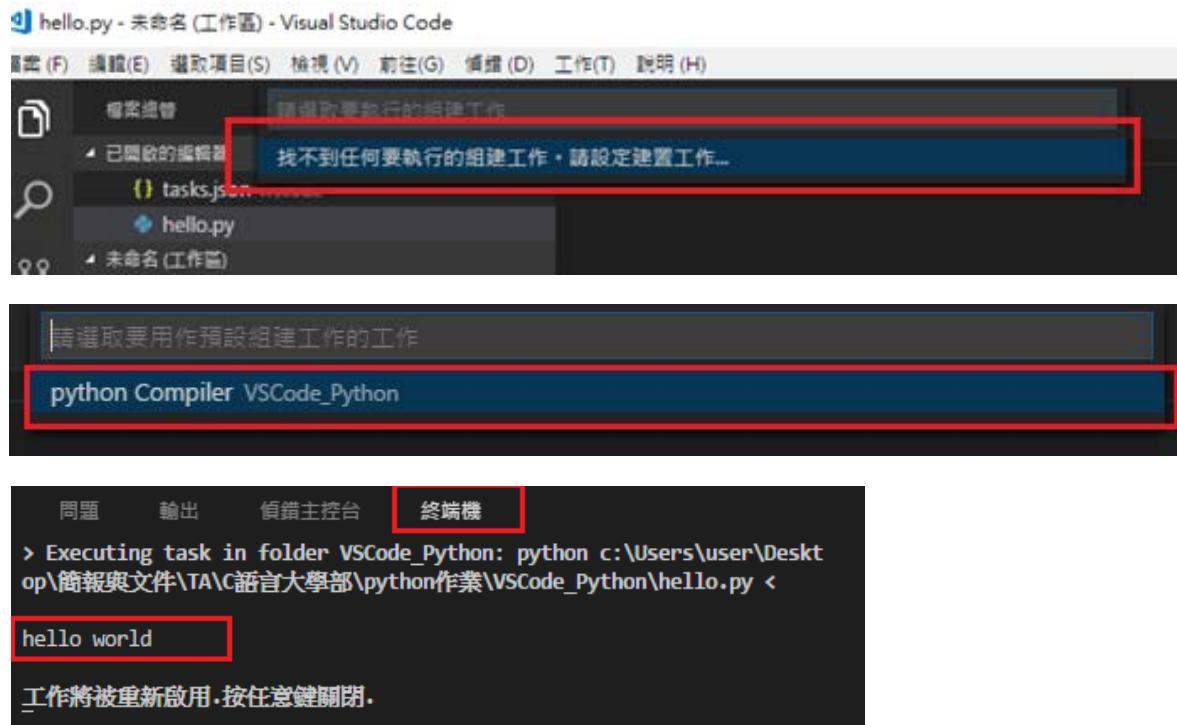
## □ Step 3. 編寫內容，並點選儲存(ctrl+S)



# 執行python

## □ Step 1. 點選Ctrl+ Shift + B (OS X : CMD + SHIFT + B)

- 如果沒有選擇Compiler則會彈出下面圖示，點選紅框部分設定即可



# Linux 安裝 Python

## □ 環境

- 作業系統 : Ubuntu 18.04
- Python 版本 : Python 3.8
- Tensorflow : Tensorflow 1.10.1
- OpenCV : OpenCV 3.4.3

## □ sudo apt-get update

- 若剛安裝ubuntu請輸入以上指令更新。

## □ sudo apt-get install python3-pip

- 若pip3沒有安裝，請輸入以上指令，若有請略過。

## □ pip3 install --upgrade pip

- 若pip3版本不是10.0以上版本，請輸入此指令，否則安裝opencv會有問題，若是請略過。

# Linux 安裝 Python

- 更新後使用pip3指令 若出現 ImportError: cannot import name 'main' 問題
  - 前往路徑cd /usr/bin
  - 輸入指令sudo vi pip3 或 sudo gedit pip3
  - 進行修改，將程式碼改為
    - from pip import \_\_main\_\_
    - if \_\_name\_\_ == '\_\_main\_\_':
    - sys.exit(\_\_main\_\_.main())
  - 儲存後即可正常使用pip3

# Linux 安裝 Python

## □ 安裝python套件

- sudo pip3 install opencv-python
- sudo pip3 install keras
- sudo pip3 install pandas
- sudo pip3 install tensorflow
- sudo pip3 install matplotlib
- sudo apt install python3-tk (此為ub才需要安裝)
- sudo apt install tk-dev (此為ub才需要安裝)

# Python 計算機程式設計

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 計算機程式設計

## □ 資訊系統開發流程

- 需求/問題的取得/理解 – 需求規格書
  - 文字描述、圖形介面呈現
- 系統分析 - 系統規格書
  - 測試案例設計、測試劇本/使用情境設計
- 系統設計
  - 資料結構設計
  - 程式流程設計/問題處理邏輯設計
- 程式碼撰寫
  - 程式語法正確性 (筆試/上機測驗)
  - 程式邏輯正確性 (筆試/上機測驗)
- 系統驗證
  - 靜態分析 – 程式碼檢視 – 人工追蹤程式運作 (筆試)
  - 動態測試 – 測試案例、測試程式執行 (上機測驗)

# 軟體品質問題 1

- 2020：南山人壽境界成就計畫問題。
- 2014：臺灣新戶政系統初上線因相容問題，系統無法製作身分證明。
- 2014：臺灣高速公路收費系統扣款有問題。
- 2007：臺灣高鐵售票系統剛上線，壓力測試不足，系統無法應付龐大湧現購票人潮。
- 2007：臺灣彩卷系統因十億獎金引爆人潮，造成系統頻繁當機。
- 2000：巴拿馬國家癌症中心，病人接受過量放射線照射，5位喪生，15位嚴重併發症。
- 2000：美國海軍飛機因控制軟體缺陷而墜落，4人喪生。
- 1997：韓國航空因雷達控制軟體缺陷，225人喪生。
- 1995：美國航空因導航軟體缺陷，在哥倫比亞撞山造成159人喪生。
- 1994：華航飛機自動駕駛軟體重飛模式和手動控制昇降舵相互對抗，名古屋空難264人喪生。
- 1991：波灣戰爭，愛國者反導彈校時系統缺陷，攔截飛毛腿飛彈失誤，28名士兵喪生。

# 軟體品質問題2

- 1963年美國太空總署，一個FORTRAN 程式迴圈敘述
  - DO 5 I=1,3 ----- (正確)
- 被人為錯誤打成
  - DO 5 I=1.3 ----- (錯誤)
  - FORTRAN編譯器視為
  - DO5I=1.3 ----- (缺陷)
- 導致飛往火星的火箭爆炸，造成一千萬美元的損失。

# Python Basis

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# Python 簡介

## □ 特性

- 容易撰寫，適合初學者
- 功能強大
- 跨平台

## □ 功能

- 資料分析
- 科學計算
- 機器學習
- 網頁設計

## □ 基本語法

- 結尾不須分號，控制指令結尾要使用冒號:
- 冒號所屬指令區塊，需縮排四個空白鍵(不須大括號)

# 標記

- 直譯器利用標記 (token) 解析程式的功能標記有
  - 關鍵字 (keyword)
  - 識別字 (identifier)
  - 字面常數 (literal)
  - 運算子 (operator)

# 關鍵字

- 關鍵字為具有語法功能的保留字 (reserved word) ，Python 的關鍵字

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

# 識別字

- 識別字為寫程式時依需求自行定義的名稱，包括變數 (variable)、函數 (function)、類別 (class) 等
- 除關鍵字外，Python 可用任何 Unicode 編碼的字元當作識別字，**包括中文**。

# 字面常數

- 字面常數是字面上意義，1234代表整數數值一千兩百三十四，是直接寫進 Python 程式原始碼的數值
  - 字串字面常數 (string literal)    'abcd'
  - 字節字面常數 (bytes literal)
  - 整數字面常數 (integer literal)    1234
  - 浮點數字面常數 (floating-point literal)            1.023
  - 複數字面常數 (imaginary literal)                    3 + 2i

# 運算子

## □ 運算子有優先順序

運算子	描述
**	指數
~x	按位翻轉
+x,-x	正負號
*,/,%	乘法、除法與取餘
+, -	加法與減法
<<, >>	移位
&;	按位與
^	按位異或
	按位或
<,<=,>,>=,! =, ==	比較
is, is not	同一性測試
in, not in	成員測試
not x	布林“非”
and	布林“與”
or	布林“或”
lambda	Lambda表示式

# 基本資料型別

- 整數(int)
- 浮點數(float):有小數點的數字
- 複數 complex
- 字串(String):一串文字內容
- 字節(byte):二進位資料
- 序列(Tuple):又稱固定列表，一種有順序、不可變動的資料集合
- 串列(List):又稱可變列表，一種有順序、可變動資料集合
- 集合(Set):無順序的資料集合
- 字典(dictionary):以Key-Value Pair形式存入集合
- 布林(Boolean):表達True(正確、真)與False(錯誤、假)

# 資料型別-整數與浮點數

## □ 數學運算子 (operator) 運算

- +, -, \*, /, \*\* 次方, // 整除, % 餘數

```
print(5 + 9)
print(10 - 7)
print(4 - 10)
print(5 + 3 - 17)
print(6 * 7)
print(6 ** 2)
print(9/5)
print(9//5)
print(9%5)
print(9/0)
```

→ 執行結果?

# 資料型別-複數

## □ 數學運算子 (operator) 運算

- +, -, \*, /, \*\* 次方, // 整除, % 餘數

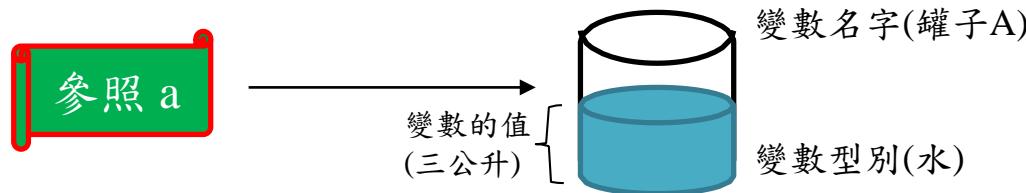
```
a = 1  
b = 2  
c = 3.6  
d = 3.6  
print(int(a + c), int(a +d))  
print(float(a + b))  
print(complex(a+b))  
print(complex(a+b) + complex(2,3))
```

```
4 4  
3.0  
(3+0j)  
(5+3j)
```

# 變數(Variable)與物件(Object)

## □ 變數特性 (不可與關鍵字/保留字相同)

- 名稱自訂，參照/指向一個資料(值)物件
- 資料(值)物件
  - 在電腦記憶體(RAM)中某個位址，好比容器有水三公升



## □ Python 中所有資料(data)都是物件。物件有：

- id() 編號
- type() 資料型別
- value 值

```
a = 3  
print(id(a))  
print(type(a))  
print(a)
```

```
140725290111872  
<class 'int'>  
3
```

# 變數(Variable)與物件(Object)

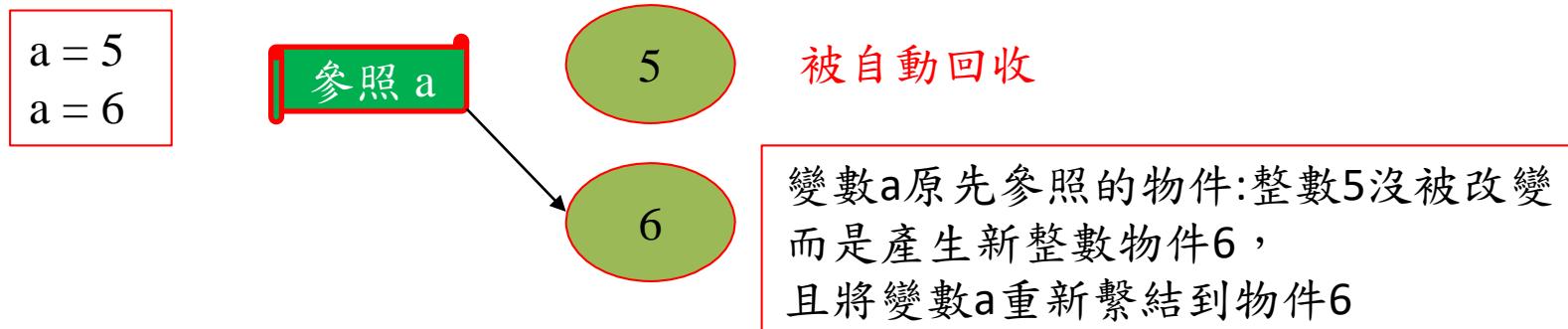
## □ 不可變的 (immutable) 物件資料

- 資料型別 (compound data type) 的元素 (element) 是可以替換，例如序對 (tuple)、整數、浮點數、字串是不可變的

## □ 可變的 (mutable) 物件資料

- 串列 (list) 或字典 (dictionary) 是可變的。

## □ 物件不再使用時，直譯器會自動垃圾收集 (garbage collection)，釋放記憶體空間。

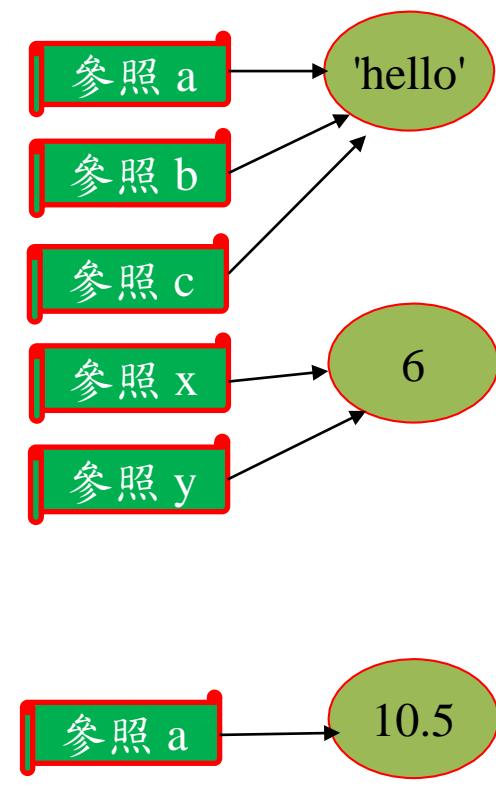


# 不可變物件

- 物件資料不可變的 (immutable)
  - 變數指定時會對應相同的 id

```
a = 'hello'  
b = a  
c = 'hello'  
print (a is b)  
print (a is c)  
print(id(a))  
print(id(b))  
print(id(c))  
a = 10.5  
x = 6  
y = 6  
print(x is y)  
print(x == y)
```

```
True  
True  
2042424477096  
2042424477096  
2042424477096  
True  
True
```



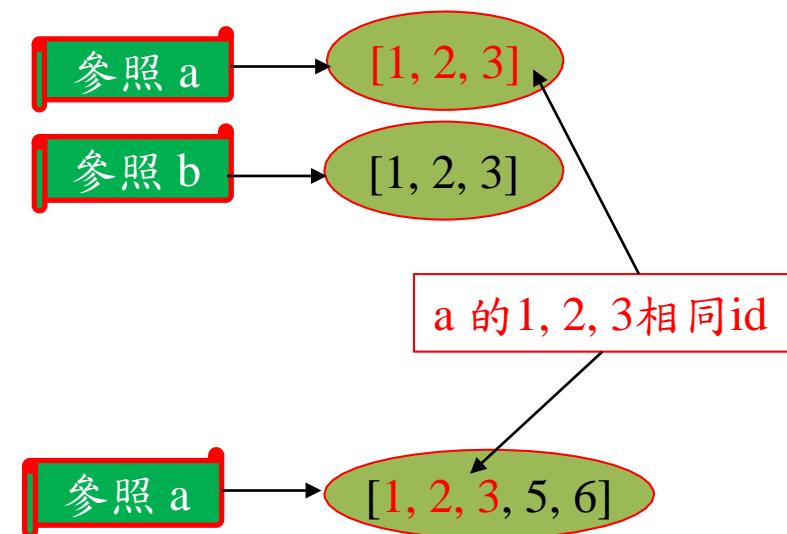
# 可變物件

## 物件資料可變 (mutable)

- 變數指定時會產生新的 id
- 擴充時可變物件變數 a 產生新的id
- 擴充時原本物件資料 a[0], a[1], a[2] id 相同

```
a = [1, 2, 3]
b = [1, 2, 3]
print(id(a))
print(id(b))
print(a is b)
print(id(a[0]))
print(id(a[1]))
a = a + [5, 6]
print(id(a))
print(a)
print(id(a[0]))
print(id(a[1]))
```

```
2042419984392
2042424366664
False
140725290111808
140725290111840
2042424396680
[1, 2, 3, 5, 6]
140725290111808
140725290111840
```



# 表示式(Expressions)

## □ 表示式

```
3 + 5  
3 + (5 * 4)  
3 ** 2  
'Hello' + 'World'  
num + 3
```

# 變數(Variable)和表示式(Expressions)

## □ 賦值運算

- 將右邊運算結果，用 = 賦予(存到/指派)左邊一個變數
- 左邊變數之後可使用其值，其值也可改變。

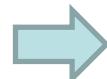
a = 4 + 3

跟數學"相等"意義完全不同



```
a = 4 + 3  
b = a * 4.5  
c = (a+b)/2.5  
a = "Hello World"
```

```
a = 10  
b = a + 5  
a = 10.1  
print(a, b)
```



執行結果?

# 變數(Variable)和表示式(Expression)

## □ 變數資料型別

- 變數值同時可看出變數型別

Var = 70 (整數)

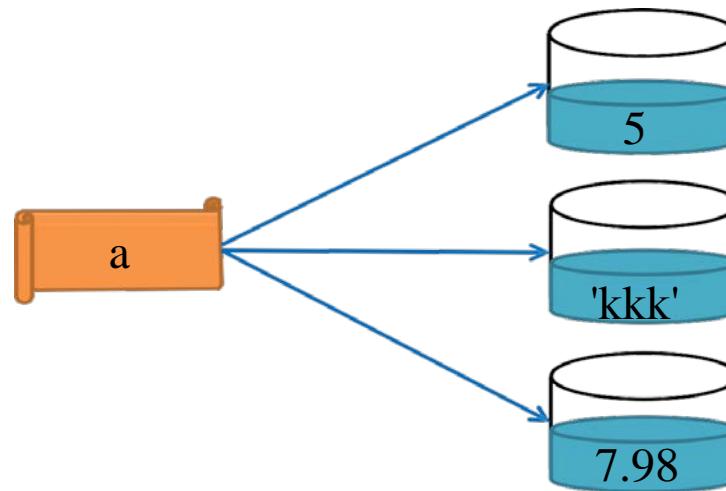
Var = 6.78 (浮點數)

Var = "alcom lab" (字串)

- 變數可以變更資料型別

- 變數 a 代表一個參照 (Reference)

a = 5  
a = 'kkk'  
a = 7.98



# 變數

## □ 變數複合運算

- $+=$ ,  $-=$ ,  $*=$ ,  $/=$
- 多變數一次指派
- 變數內容對調

```
a = 10  
a+=b  
x, y = a, b  
  
x, y = y, x
```

```
a = 10  
a+=b  
x = a  
y = b  
tmp = x  
x = y  
y = tmp
```

# 變數

## □ 變數命名規則

- 允許 a-z A-Z 0-9 和 \_，但是開頭必須是 [\_a-zA-Z] 其中一個
- 變數區分大小寫
- 變數長度沒有限制
- 特殊的變數，開頭和結尾都加兩個下底線，ex: \_\_init\_\_

```
_a = 10  
print(a)  
Bb_ = _a + 5  
2c = 3 + Bb  
a = bb + 5
```

→ 執行結果？

# 資料型別-字串

□ 字串以單引號'或雙引號"包起來，表示文字資料。

□ 字串可存於變數

```
x = 'Hello world'  
print(x)
```

□ "\" 可轉字串某些字元含義，最常見如: \n \t

```
x = 'Hello \n world'  
print(x)
```

□ 字串表示單引號，須以 \ 處理，避免被誤認為字串結束。

```
x = 'I\'m Lucas'  
print(x)
```

# 使用者輸入

## □ `input('提示字')`

- 以字串型別從鍵盤輸入資料
- 等待輸入時，顯示提示字

## □ `int()` 轉換成整數

```
def inOps():
    numX = "123"
    numY = "456"
    num = int(numX) + int(numY)
    print(num)
    name = input('name: ')
    numX = input('First Number: ')
    numY = input('Second Number: ')
    num = int(numX) + int(numY)
    print(name, num)
```

579

name: John

First Number: 80

Second Number: 90

John 170

# Exercise

- 程式讀取兩個整數，將其四則運算印出
  - 其中除法印出整數、與小數
  - 提示:`:input()`回傳是String(字串)，須轉型成int。

# print

- %d 10進位整數輸出
- %f 浮點數10進位輸出
- %e, %E 將浮點數以10進位科學記號輸出
- %o 以8進位整數方式輸出
- %x, %X 將整數以16進位方式輸出
- %s 使用str()將字串輸出
- %c 以字元方式輸出
- %r 使用repr()輸出字串
  - 表達資料型別，例如字串''
- %% 在字串中顯示%

```
>>> text = '%d %.2f %s' % (1, 99.3, 'Justin')
>>> print(text)
1 99.30 Justin
>>> print('%d %.2f %s' % (1, 99.3, 'Justin'))
1 99.30 Justin
>>> 'example: %.2f' % 19.234
'example: 19.23'
>>> "example: %6.2f" % 19.234
'example: 19.23'
```

# Exercise

- 一元二次方程式， $aX^2 + bx + c = 0$ ，輸入a, b, c, 求 方程式的兩個實根。 $X1=(-b+math.sqrt(b*b-4*a*c))$

- Input

- 1

- 2

- 1

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
import math  
x=math.sqrt(34)  
print(x)
```

- Output

- 1.0

- 1.0

# Exercise

- A、B、C三本書價格如下，一顧客欲購買A: x 本、B: y 本、C: z 本（x、y、z 為使用者輸入），請計算需花費多少錢？

定價

A 380

B 1200

C 180



# 資料型別-字串操作

- 使用+運算子結合字串
- 使用\*複製字串
- 使用[]擷取字元

```
def strOp():
    print("Hello" + " World")
    name = "小明"*2
    print(name)
    Hi = "Hi~ 您好!"
    print(Hi[1])
    print(Hi[4])
```

```
Hello World
小明小明
i
您
```



# 資料型別-字串操作

- 櫄取字串 [start:end:step] ,
  - start~(end-1), each step
  - start~(前一個), each step
  - Start, end, -1 表示倒數過來第一個

```
def strGet():
    words = "This is a book, that is a cat~"
    print(words)
    print(words[:])
    print(words[5:])
    print(words[-4:])
    print(words[-14:20:1])
    print(words[-6:5:-1])
    print(words[-4:-10:-2])
```

This is a book, that is a cat~  
This is a book, that is a cat~  
is a book, that is a cat~  
cat~  
that  
a si taht ,koob a s  
cas

# Exercise

□ 輸入兩個英文句子 A, B，

- 將兩個英文句子A, B串聯成 C
- 將C倒著輸出
- C從第五個字到最後第三個字，每隔兩個字輸出

Input

This is a book  
That is a cat



# 資料型別-字串操作

- len()取得長度
- .replace()替換
- .split()切割字串，回傳str list

```
def strOps():
    words = "小明今天去學校，小明遲到。"
    wordLength = len(words)
    print(wordLength)
    sentence = words.replace("小明","湯姆")
    print(words)
    print(sentence)
    wordsSplit = words.split("明")
    print(wordsSplit)
```

13

小明今天去學校，小明遲到。  
湯姆今天去學校，湯姆遲到。  
['小','今天去學校，小','遲到。']

# Exercise

- 輸入兩個英文句子 A, B，兩個英文字 x, y
  - 將兩個英文句子A, B串聯成 C
  - 將C其中的 x 替換成 y，變成 D
  - 輸出C, D 長度的加總
  - 輸出C前三個字，每一個字重複輸出三次。

Input

This is a book

That is a cat

is

was

# string與byte轉換

## □ 字元編碼

- 有ASCII、Unicode(\uXXXX)、GB2312(資訊交換用漢字編碼字元集)、GBK(漢字內碼擴展規範)、GB 18030、Big5等，不同編碼間可以互相轉換。
- UTF-8是Unicode的一種電腦儲存實現方式，即字元編碼格式。UTF-8一般用於網路傳輸。

## □ Python

- 二進制資料由 bytes型別表示
- 文字/字串使用Unicode，由 str型別表示

## □ str 跟 bytes 可互相轉換

- str.encode() 預設編碼 utf-8

str.encode() ⇔ bytes.decode()  
(Unicode) (utf-8)

# string與byte轉換

```
a = bytes([1,2,3,4,5,6,7,8,9])
b = bytes('python', 'ascii')
print(type(a))    # <class 'bytes'>
print(type(b))    # <class 'bytes'>
print(a)          # b'\x01\x02\x03\x04\x05\x06\x07\x08\t'
print(b)          # b'python'
```

# string與byte轉換

```
# bytes轉字符串方式一  
b=b'\xe9\x80\x86\xe7\x81\xab'  
string=str(b,'utf-8')  
print(string)
```

```
# bytes轉字符串方式二  
b=b'\xe9\x80\x86\xe7\x81\xab'  
string=b.decode() # 第一參數預設utf8，第二參數預設strict  
print(string)
```

```
# bytes轉字符串方式三  
b=b'\xe9\x80\x86\xe7\x81haha\xab'  
string=b.decode('utf-8','ignore') # 忽略非法字符，用strict會拋出異常  
print(string)
```

```
# bytes轉字符串方式四  
b=b'\xe9\x80\x86\xe7\x81haha\xab'  
string=b.decode('utf-8','replace') # 用？取代非法字符  
print(string)
```

# string與byte轉換

```
# 字符串轉bytes方式一  
str1='逆火'  
b=bytes(str1, encoding='utf-8')  
print(b)  
  
# 字符串轉bytes方式二  
b=str1.encode('utf-8')  
print(b)
```

```
b'\xe9\x80\x86\xe7\x81\xab'  
b'\xe9\x80\x86\xe7\x81\xab'
```

# I/O

- print預設印一行換行，使用，在每次印出間以空格取代，更自由可使用library中的write函式。

```
import sys
file_in = open('db.txt','r')
file_out = open('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()
```

# 造出 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內容和db.txt一樣

# import

- 用import直接匯入整個python函式庫中所有函式，或用from函式庫import函式，插入特定函式
  - import x.py
    - 會執行 x.py 未空四格的指令
    - 一般函式庫不會有

```
#匯入sys函式庫所有函式，使用write函式前須加sys  
import sys
```

```
#從time函式庫匯入time()函式，使用time()前不需加 time  
from time import time  
sys.stdout.write(str(time())+"\n")
```

Output:

1409796132.99 #當下的time

# import

## □ module

- 一個包含有Python的程式定義及敘述的檔案。
- 檔案名稱是module名稱加上延伸檔名 .py 。
- module裡存在module名字變數，如sys, time，當作全域變數 (global variable)使用。

## □ 常見module

- os模組：封裝不同作業系統的通用api，在不同作業系統下，可使用相同的函數呼叫
- sys模組：作業系統的相關資訊與函數的模組
- built-in模組：內建模組
- time模組：時間相關模組
- re模組：正則表達式

# import

## □ 常見module

- thread模組：執行緒模組
- urllib模組：接受url相關呼叫的模組，接受編碼及解碼
- urllib2模組：接受url相關呼叫的模組
- socket模組：網路通訊模組
- file模組：檔案使用模組

# import

## □ \_\_main\_\_

- A模組/檔案，直接執行時，\_\_main\_\_的值預設是【'\_\_main\_\_'】

```
# A.py
if __name__ == '__main__':
    print('Direct running')
else:
    print('import __name__= ', __name__)
```

Direct running

- A模組/檔案，被import後，\_\_main\_\_的值是，檔案名稱  
➤ 執行 B.py

```
#B.py
import A
print('import A')
```

import \_\_name\_\_ = A  
import A

# format

## □ '字串'.format(參數)

- 字串內以{}代換 format的參數-- 字串資料
- 代換以 0, 1, 2..位置識別，或者以符號識別

```
def format00():
    str='{0} is {1}!!'.format('Justin', 'caterpillar')          #第0個對應後面第0個參數
    print(str)
    str='{real} is {nick}!!'.format(real = 'Justin', nick = 'caterpillar')  #以符號對應
    print(str)
    str='{real} is {nick}!!'.format(nick = 'caterpillar', real = 'Justin')  #符號順序無關
    print(str)
    str='{0} is {nick}!!'.format('Justin', nick = 'caterpillar')
    print(str)                                              #Justin is caterpillar!!
    str='{name} is {age} years old!'.format(name = 'Justin', age = 35)
    print(str)                                              #Justin is 35 years old!
```

# format

```
import math
import sys
def format01():
    str=math.pi
    print(str)                                #3.14159265359'
    str=format(math.pi, '.18f')                 #PI 取小數點18位
    print(str)                                #3.141592653589793116
    str='PI = {0.pi}'.format(math)              #第0個對應後面第0個參數
    print(str)                                #'PI = 3.14159265359'
    str='My platform is {pc.platform}'.format(pc = sys)
    print(str)                                #以符號 pc 對應 sys
    str='My platform is {0.platform}. PI = {1.pi}'.format(sys, math)
    print(str)                                #'My platform is win32'
    #'My platform is win32. PI = 3.14159265359.'
    str='element of index 1 is {0[1]}'.format([20, 10, 5])      #第0個對應後面第0個參數
    print(str)                                #'element of index 1 is 10'
    str='My name is {person[name]}'.format(person = {'name' : 'Justin', 'age' : 35})
    print(str)                                #'My name is Justin'
```

# Exercise

- 某一學生修國文、計算機概論、計算機程式設計三科，使用者輸入名字、學號、三科成績。
  - 計算學生總成績、平均。
  - 印出名字、學號、總成績、平均。
  - 使用 format 印出

Input

K

905067

100

100

100

Output

Name:K

Id:905067

Total:300

Average:100

Output

Name K, Id is 905067, Total score is 300, Average is 100

# Python

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 內建Function

Built-in Functions				
<a href="#">abs()</a>	<a href="#">dict()</a>	<a href="#">help()</a>	<a href="#">min()</a>	<a href="#">setattr()</a>
<a href="#">all()</a>	<a href="#">dir()</a>	<a href="#">hex()</a>	<a href="#">next()</a>	<a href="#">slice()</a>
<a href="#">any()</a>	<a href="#">divmod()</a>	<a href="#">id()</a>	<a href="#">object()</a>	<a href="#">sorted()</a>
<a href="#">ascii()</a>	<a href="#">enumerate()</a>	<a href="#">input()</a>	<a href="#">oct()</a>	<a href="#">staticmethod()</a>
<a href="#">bin()</a>	<a href="#">eval()</a>	<a href="#">int()</a>	<a href="#">open()</a>	<a href="#">str()</a>
<a href="#">bool()</a>	<a href="#">exec()</a>	<a href="#">isinstance()</a>	<a href="#">ord()</a>	<a href="#">sum()</a>
<a href="#">bytearray()</a>	<a href="#">filter()</a>	<a href="#">issubclass()</a>	<a href="#">pow()</a>	<a href="#">super()</a>
<a href="#">bytes()</a>	<a href="#">float()</a>	<a href="#">iter()</a>	<a href="#">print()</a>	<a href="#">tuple()</a>
<a href="#">callable()</a>	<a href="#">format()</a>	<a href="#">len()</a>	<a href="#">property()</a>	<a href="#">type()</a>
<a href="#">chr()</a>	<a href="#">frozenset()</a>	<a href="#">list()</a>	<a href="#">range()</a>	<a href="#">vars()</a>
<a href="#">classmethod()</a>	<a href="#">getattr()</a>	<a href="#">locals()</a>	<a href="#">repr()</a>	<a href="#">zip()</a>
<a href="#">compile()</a>	<a href="#">globals()</a>	<a href="#">map()</a>	<a href="#">reversed()</a>	<a href="#">import</a>
<a href="#">complex()</a>	<a href="#">hasattr()</a>	<a href="#">max()</a>	<a href="#">round()</a>	
<a href="#">delattr()</a>	<a href="#">hash()</a>	<a href="#">memoryview()</a>	<a href="#">set()</a>	

# Function

## □ 使用函式的程式設計好處

- 將大程式切割由多人撰寫，利於團隊分工，縮短程式開發時間。
- 可縮短程式長度，可讀性高，利於測試驗證、除錯、重複使用
- 當再開發類似功能產品，只需稍微修改即可套用。

零件:輪子、儀錶板、引擎



工廠組裝汽車



輸出:汽車



# Function 呼叫

- 呼叫function

- `abs(-5)=?`
  - `print ("Hello, world")`

- 內建函式可以直接使用

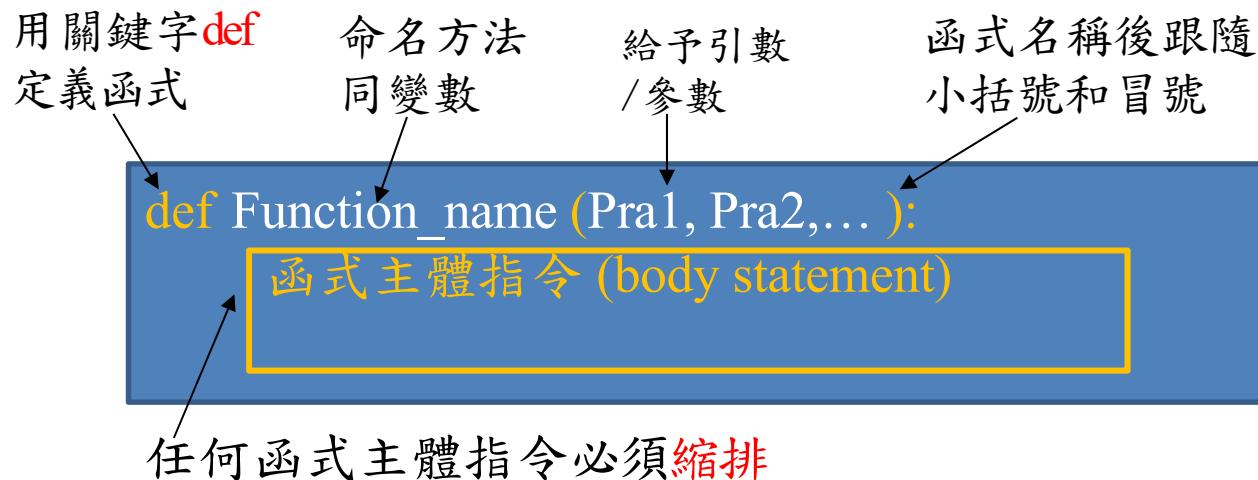
- Length: `len()`
  - Sort: `sorted()` v.s. `object.sort()`

- 內建轉換型態或建立物件

- `int()`
  - `str()`
  - `float()`
  - `list()`
  - `set()`

# Function 定義

- 自行設計與定義函式
  - 必先定義而後使用



```
def CreateCar(wheels, dashboard, engine):  
    car = dashboard + wheels + engine  
    return car
```

```
carA = CreateCar(wheels, dashboard, engine)
```

# 命名規則

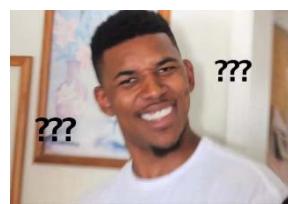
- 函數之命名應有意義，指出函數目的或回傳資料
  - 以駝峰式(Camel-Case)命名、或以小寫或輔以底線命名
  - 避免非慣用縮寫
- 若由多個單字組成，第一個單字要小寫，從第二個單字開始，每個單字的第一個字母大寫
  - studentName, bianryData, dataLength。
- 名稱不使用縮寫，以清楚、明確為原則。
  - 例如 distance 比 d 清楚
  - dataLength 比 length 更明確易理解。
- Boolean 命名用 is開頭
  - isPrime
- 函式表示一種操作，使用動詞作為第一個單字。
  - computeAverage()

# 命名規則

## □ 命名

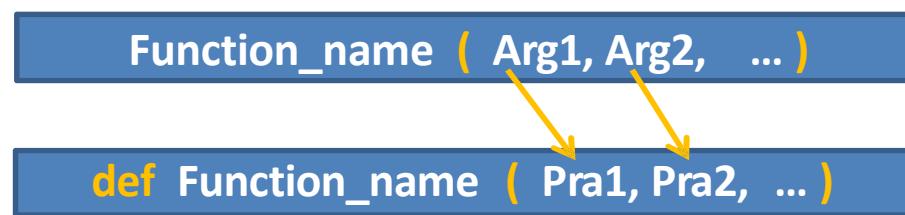
```
def a()
    a=int(input())
    b=int(input())
    c=int(input())
    d=(a+b+c)/3
    return d
```

```
def ComputeAverage()
    englishGrade=int(input())
    mathGrade=int(input())
    chineseGrade=int(input())
    average=(englishGrade+mathGrade+chineseGrade)/3
    return average
```



# Function定義與使用

- 函式建立後不會執行，須在程式中呼叫函式才會執行
  - [ 變數 = ] 函式名稱([ 參數串列 ])
- Parameter / 參數
  - 函式執行時之參考
  - 在函式的宣告，說明被呼叫時必須接收什麼資料
- Argument / 引數
  - 呼叫函式時，放在括號內的變數/值



傳遞的動作為賦值運算， $Pra1 = Arg1$ ,  
 $Pra2 = Arg2$

...

$PraN = ArgN$

# Function定義與使用

## □ 參數預設值

- 建立函式時可為參數設定預設值，呼叫函式時，若沒有傳入該參數時，就會使用預設值。
- 參數設定預設值的方法為「參數 = 值」，例如：

```
def get_area(length, width=4, height=5):  
    return length*width*height
```

```
print(get_area(3))  
print(get_area(3,2))
```

# Function定義與使用

- 函式執行完後可以回傳結果，也可以不回傳
- 當執行到return指令時，會無條件直接離開，結束函式回傳結果
- 回傳值可以為
  - 單一的值或物件
  - 多個值或物件所構成的 tuple
  - 當未使用return指令，預設回傳None

# Function

## □ 神奇算年齡

- 看一下你手機號的最後一位；
- 把這個數字乘上2；
- 再加上5；
- 再乘以50；
- 把得到的數目加上1770
- 最後一個步驟，用這個數目減去你出生西元年。
- 現在你看到一個三位數的數字
- 第一位數字是你手機號的最後一位，接下來就是你的實際年齡
- 如果你看到二位數的數字
- 你手機號的最後一位是0，二位數是你的實際年齡

## □ 寫成一個function

- 要傳入那些參數？

# Function

## □ 神奇算年齡

### ○ 不用 function

```
phoneNum, year = 5, 1969
result = ((phoneNum*2 + 5) *50 + 1770) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
phoneNum, year = 0, 1969
result = ((phoneNum*2 + 5) *50 + 1770) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
phoneNum, year = 3, 1999
result = ((phoneNum*2 + 5) *50 + 1770) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
```

# Function

## □ 神奇算年齡

### ○ 有用 function

```
def computeAge(phoneNum, year):  
    result = ((phoneNum*2 + 5) *50 + 1770) - year  
    newPhoneNum = result//100  
    age = result%100  
    return newPhoneNum, age
```

$$\begin{aligned}(x \times 2 + 5) \times 50 + 1769 - y \\= (2019 - y) + 100x\end{aligned}$$

```
def testAge():  
    print(computeAge(5, 1969))      # 5, 50  
    print(computeAge(0, 1969))      # 0, 50  
    print(computeAge(3, 1999))      # 3, 20
```

```
testAge()
```

# Function

## □ 神奇算年齡

- 明年改公式，若有一百個地方要改，少改一個？

```
phoneNum, year = 5, 1969
result = ((phoneNum*2 + 5) *50 + 1771) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
phoneNum, year = 0, 1969
result = ((phoneNum*2 + 5) *50 + 1771) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
phoneNum, year = 3, 1999
result = ((phoneNum*2 + 5) *50 + 1771) - year
newPhoneNum = result//100
age = result%100
print(newPhoneNum, age)
```

# Function

## □ 神奇算年齡

- 明年改公式，改一個地方

```
def computeAge(phoneNum, year):  
    result = ((phoneNum*2 + 5) *50 + 1770) - year  
    newPhoneNum = result//100  
    age = result%100  
    return newPhoneNum, age
```

```
def testAge():  
    print(computeAge(5, 1969))      # 5, 50  
    print(computeAge(0, 1969))      # 0, 50  
    print(computeAge(3, 1999))      # 3, 20
```

```
testAge()
```

# Function

## □ 定義 function

- def開頭
- 名稱自訂
- 以冒號:結尾
- 所屬內容要縮排

## □ function種類

- 有參數、有回傳值 (傳入資料，處理完後回傳結果)
- 無參數、有回傳值
- 無參數、無回傳值
- 有參數、無回傳值

# Function

## □ 有參數、有回傳值

```
def my_function(x,y):  
    return x+y, x-y
```

```
x, y = my_function(10, 20)  
print (x, y)
```

- 神奇算年齡，輸入參數為何？回傳值為何？
- 算年齡程式

```
from datetime import date
```

```
def computeAgeDay(born):  
    age = date.today()-born  
    return age.days
```

```
def computeAge01(born):  
    today = date.today()  
    return today.year - born.year - ((today.month, today.day) < (born.month, born.day))
```

# Function

- 無參數、有回傳值: 輸入資料

```
def my_input():
    x = int(input())
    return x

s = my_input()
```

- 有參數、無回傳值: 輸出報表、結果

```
def myFunction(name):
    print("Hello~", name)

def main():
    myFunction('John')
    myFunction('Tom')

main()
```

- 無參數、無回傳值

# Exceptions Handling

```
def my_divide():
    try:
        10 / 0 #會讓程式出錯,所以需要特別handle
    except ZeroDivisionError:
        print('不能除以0!!!')
    else:
        print('沒有任何錯誤')
    finally:
        print('無論有沒有例外都會執行這一行')
my_divide()
```

Output:  
不能除以0!!!  
無論有沒有例外都會執行這一行

自己可以製造意外  
自己範圍產生的意外自己可以處理也可以往外丟  
自己或別人製造的意外自己可以處理  
接收別人的意外可以自行處理也可以往外丟  
最後意外沒人處理會讓程式中斷

# Exceptions Handling

## □ Function 有參數、有回傳值，有例外情況

```
def computeAge02(born):
    today = date.today()
    try:
        birthday = born.replace(year=today.year) #算出今年生日
    except ValueError: # raised 當生日 2/29, 但今年沒有 2/29
        birthday = born.replace(year=today.year, month=born.month+1, day=1)
    if birthday > today:
        return today.year - born.year - 1
    else:
        return today.year - born.year

def testAge0():
    print(computeAgeDay(date(1969,10,10)))
    print(computeAge01(date(1969,10,10)))
    print(computeAge02(date(1972,2,29)))

testAge0()
```

# Exceptions Handling

## □ 開啟檔案引發例外情況

```
def computeAge02(born):
    try:
        fp = open("C:\test.txt","r")
        try:
            for index in fp:
                print(index)
        except:
            print("read file error")
        finally:
            fp.close()
    except:
        print("Open error")
```

# Exceptions Handling

## □ 自行引發例外情況

```
def computeAge02(born):
    try:
        raise EOFError #自行引發的例外
    except EOFError:
        print("self error")
    else:
        print("No error")
```

# Exercise

- 輸入兩個整數計算相加、相除，輸出相加與相除結果，如果輸入不是整數，例如字串，輸出 "Input Integer"，如果第二個數是0，輸出 "divide by zero"，最後一定要輸出 "OK"，請使用 try, except, finally。

```
num1 = input()
num2 = input()
try:
    num1 = int(num1)
    num2 = int(num2)
    answer = num1/num2
except ValueError:
    print('Input Integer')
except ZeroDivisionError:
    print('Division zero')
else:
    print(answer)
finally:
    print('OK')
```

# 註解

- 單行註解為#
- 多行註解用 """ 開頭與結尾。

```
"""
Compute Age Equation
As a Function
"""

def getAge(cell):
    age = 3*cell*cell-6*cell+1
    return age
#Compute Total Age
totalAge = getAge(100) + getAge(200) + getAge(300) + getAge(400)
print(totalAge)
```

# function不定個數參數變數

- 建立函式時可讓函式接受沒有預先設定的參數個數，方法是在參數名稱前加星號「\*」
- 參數時使用\*，表示該function的參數接受不定長度引數

```
def getMax(*a):  
    print(type(a))          #印出資料型態  
    tmp = list(a)  
    tmp.sort(reverse=True)  #排序  
    return tmp[0]            #回傳最大的  
  
def test01():  
    result = getMax(5,7,9,8,10,15,22,1,194,9,99,56,1,11)  
    print(result)
```

# function不定個數參數不定變數

- `args` 是不定個數的 tuple，例如 `(1, 2, 3, 4, 5, 6)`，`*args` 是tuple解開到元素，例如上述 `1, 2, 3, 4, 5, 6`
- `kwargs` 是不定個數的dict，`**kwargs`是 dict解開。

```
def print_args(a, b, c, *args):  
    print(a, b, c, args)  
print_args(1, 2, 3, 4, 5)                                # 1 2 3 (4, 5)  
  
def print_kwargs(**kwargs):  
    print(kwargs)  
print_kwargs(foo='bar', hello='world')                      # {'foo': 'bar', 'hello': 'world'}  
  
def print_data(latitude=None, longitude=None):  
    print(latitude, longitude)  
  
data = {'latitude': 0.00, 'longitude': 1.00}  
print_data(**data)                                         # 0.00 1.00  
  
def print_kwargs(lat=None, long=None, **kwargs):  
    print(lat, long, kwargs)  
print_kwargs(1, 2, data='other')                            # 1 2 {'data': 'other'}
```

# 可維護程式

## □ 程式品質

*Any fool can write code that a computer can understand.  
Good programmers write code that humans can understand.*  
- Martin Fowler.

任何傻瓜都能寫出電腦能懂的程式碼。而只有好的程  
式設計師才能寫出人能懂的程式碼

- Martin Fowler.

*Quality is not an act. It is a habit.*

- Aristotle

品質不是動作，是一種習慣

- Aristotle

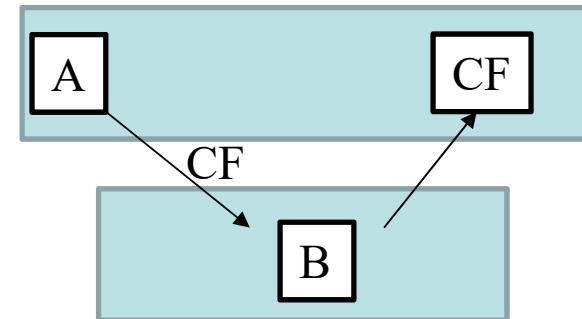
# Exercise

- 程式要切割模組
  - 好的設計
  - MVC架構
  
- 將計算成績練習寫成三個function
  - input
  - compute
  - output

# callback function

## □ callback function(回撥函式) CF

- 函式A 呼叫函式B
- 函式A 將 CF 當參數傳入某函式 B
- 函式 B 執行時會呼叫 CF



```
def B(cf, p):
    print('B:')
    cf(p)

def CF(p):
    print('callback function:', p)

def A():
    print('A:')
    B(CF, 'x')

A()
```

A:  
B:  
callback function: x

# callback function

## □ callback function(回撥函式) CF

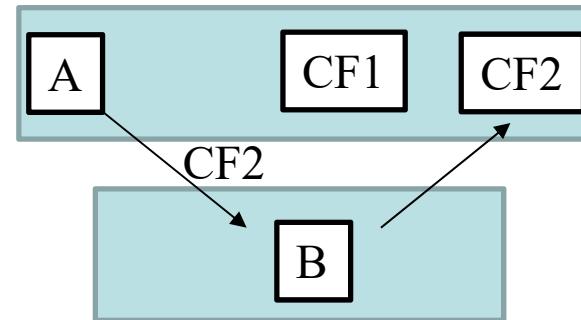
```
def B(cf, p):
    print('wake-up call:')
    cf(p)

def CF1(name):
    print('phone call~ ', name)

def CF2(name):
    print('knock up~ ', name)

def A():
    print('A:')
    B(CF1, 'John')
    B(CF2, 'Tom')

A()
```



A:  
wake-up call:  
phone call~ John  
wake-up call:  
knock up~ Tom

# Exercise

- 請問對於以下程式的敘述，何者錯誤？(1) 程式執行後會立刻中斷程式 (2) 程式會發生編譯錯誤 (3) 程式執行後輸出 A:B: callback function: x (4) 程式執行後輸出 A:B:

```
def B(cf, p):
    print('B:', end="")
    cf(p)
def CF(p):
    print('callback function:', p, end="")
def A():
    print('A:', end="")
    B(CF, 'x')

A()
```

# Exercise

- 寫一個 Call back function，計算不同種類圖形的面積
  - (1) 圓形
  - (2) 正方形

```
import math
def computeArea(cf, p):
    return cf(p)
def square(data):
    return (data*data)
def circle(data):
    return (math.pi*data*data)
print('%d' %computeArea(square, 2))
print('%d' %computeArea(circle, 2))
```

# Python

郭忠義

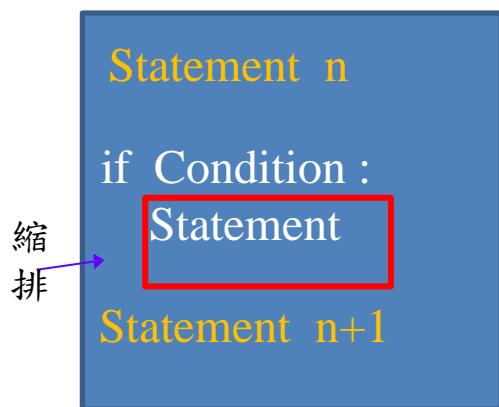
[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

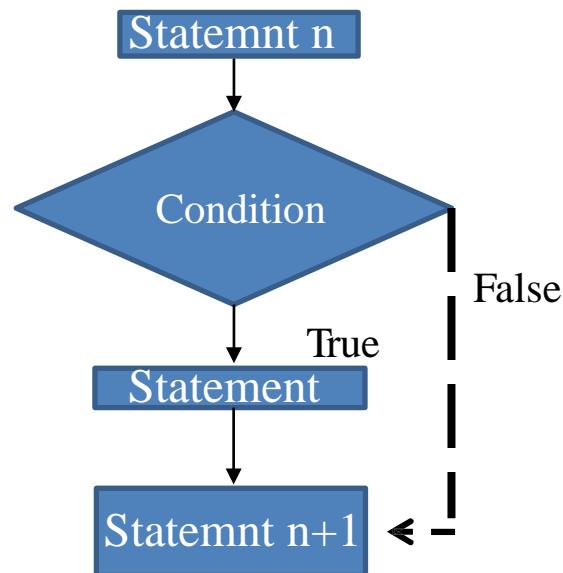
# if條件判斷

## □ 基本語法

- if <condition>，if 後跟隨"條件"(Condition)，結尾冒號
- statements，if 本體區塊 (body block)，縮排一層，若"條件"判斷True，執行此區指令



```
if 2**10 == 1024:  
    print('2^10=1024')
```

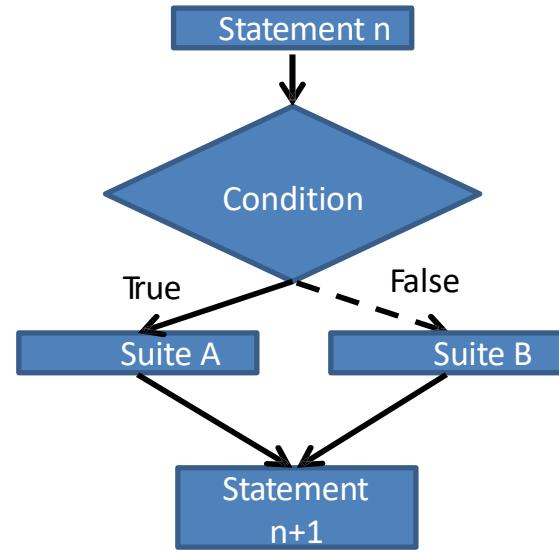


# if條件判斷

## □ 基本語法

- if <condition>，if 後跟隨"條件"(Condition)，結尾冒號
- suite A，if本體區塊 (body block)，縮排一層，若"條件"判斷 True，執行此區指令
- else，結尾冒號
- suite B，else本體區塊，縮排一層，若"條件"判斷為False，執行此區指令

```
Statement n
if Condition :
    suite A
else :
    suite B
Statement n+1
```



# if條件判斷

## □ 基本指令

```
def oe():
    num = int(input('Please input a num:'))
    if num % 2 == 0:
        print(num,'是偶數')
    else:
        print(num,'是奇數')
```

## ○ if else 縮寫

```
if x>y:
    maxValue = x
else:
    maxValue = y
```

```
maxValue = x if x>y else y
```

# If 條件判斷

## □ 條件判斷，求值是 True/False

- 關係比較運算:  $x > 10$ ,  $x < 10$ ,  $x == 10$ ,  $x != 10$

比較運算	語法
相等	$A == B$
不等於	$A != B$
大於	$A > B$
小於	$A < B$
大於等於	$A >= B$
小於等於	$A <= B$

## □ 三種關係比較

```
if 10 < x < 20:  
    print(x, '在 10~20 範圍內')
```

- 其他程式語言：if  $a < x$  and  $x < b$

# If 條件判斷

## □ 輸入分數，判斷是否及格

```
def myFunction():
    print("Hello~")
    score = int(input('輸入分數:'))
    if score>=60:
        print('恭喜你及格')
    else:
        print('不及格，要加油')

def main():
    myFunction()
    myFunction()

main()
```

```
Hello~
輸入分數:61
恭喜你及格
Hello~
輸入分數:59
不及格，要加油
```

## □ 輸入超過100，或負數要如何修正？

# If條件判斷

## □ 邏輯運算子

邏輯	語法
且	A and B
或	A or B
否定	not A

not	A
False	True
True	False

and	A	B
True	True	True
False	True	False
False	False	True
False	False	False

or	A	B
True	True	True
True	True	False
True	False	True
False	False	False

```
2==3 or 3 < 7  
2==3 and 3 < 7  
not 3 < 7
```

# If條件判斷

- 若溫度(temperature)高於30而且沒有風wind=0，或濕度(humidity)大於85，印出'開冷氣'
  - CODE

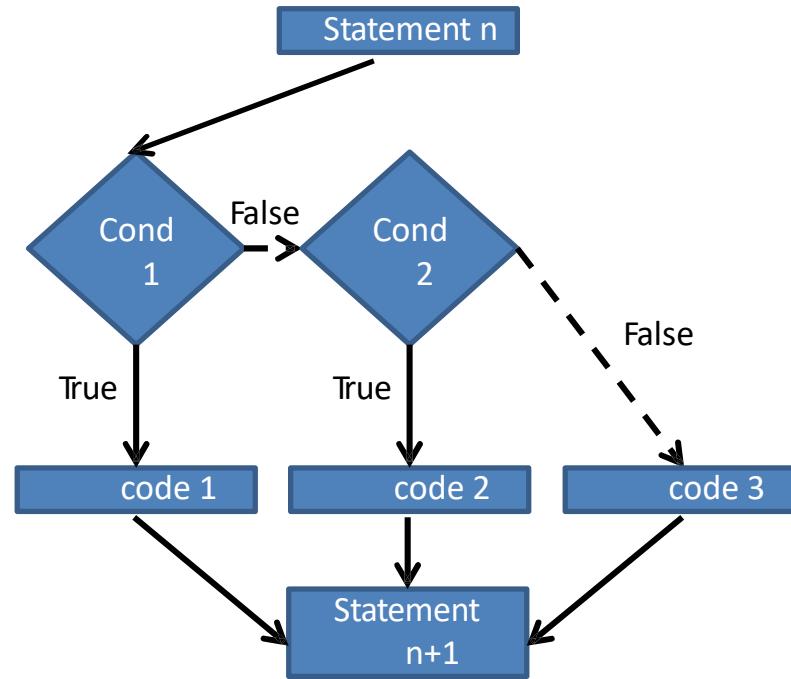
# If條件判斷

## □ if, elif, else (多選1)

Statement n

```
if Condition 1:  
    code 1  
elif Condition 2:  
    code 2  
else:  
    code 3
```

Statement n+1



# If條件判斷

## □ if, elif, else (多選1)範例(比大小)

```
def ops():
    num1 = int(input('Please input a num1:'))
    num2 = int(input('Please input a num2:'))
    if num1 == num2:
        print(num1,'等於',num2)
    elif num1 < num2:
        print(num1,'小於',num2)
    else:
        print(num1,'大於',num2)
```

# If條件判斷

## □ 判斷分數等第A, B, C, D, E, F

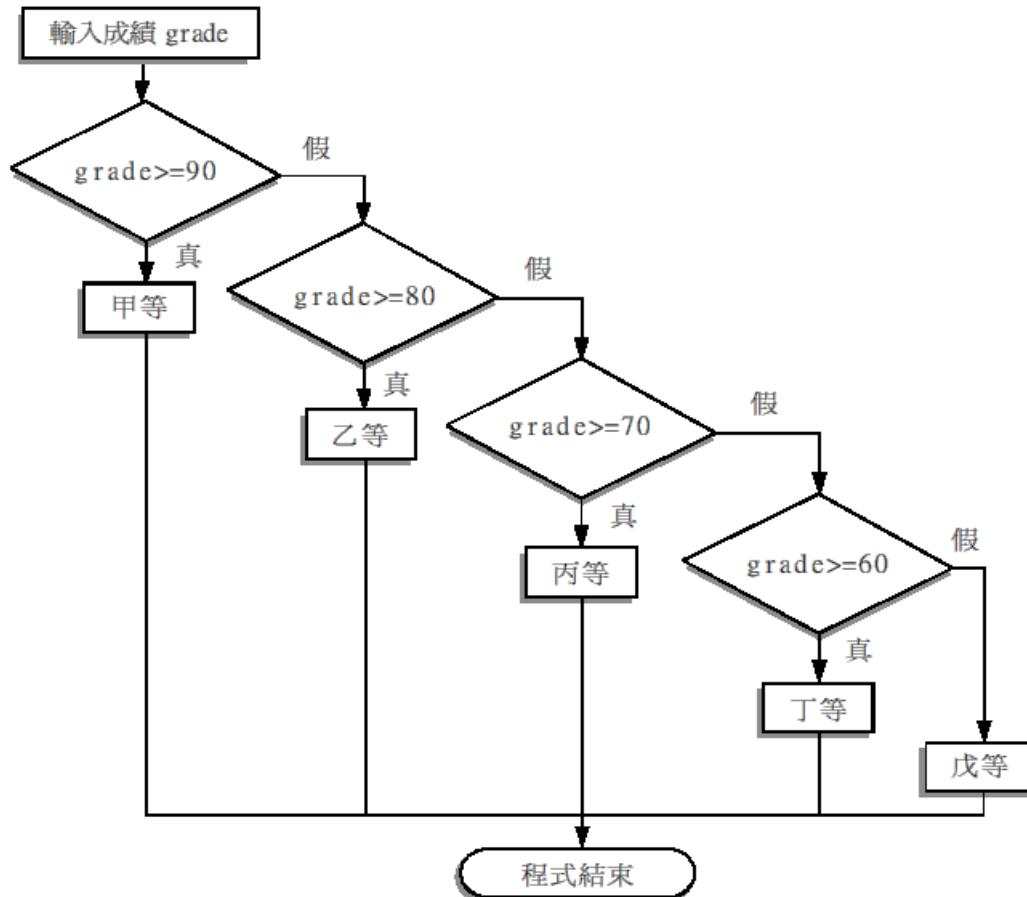
```
def myFunction():
    score = int(input('輸入分數:'))
    if score>=90:
        print('得 A')
    elif score>=80 and score <90:
        print('得 B')
    elif score>=70 and score <80:
        print('得 C')
    elif score>=60 and score <70:
        print('得 D')
    else:
        print('不及格')
def main():
    myFunction()
    myFunction()
    myFunction()
```

```
哈囉,python
輸入分數:90
得 A
哈囉,python
輸入分數:60
得 D
哈囉,python
輸入分數:59
不及格
```

這個程式邏輯是否有問題?

# If條件判斷

- 判斷分數等第A, B, C, D, E, F，修正判斷邏輯
  - 超過100分或低於0分之處理



# Exercise

□ 計算BMI並輸出分級值， $BMI = \text{體重 (kg)} / \text{身高 (m}^2)$

分 級	身體質量指數
體重過輕	$BMI < 18.5$
正常範圍	$18.5 \leq BMI < 24$
過 重	$24 \leq BMI < 27$
輕度肥胖	$27 \leq BMI < 30$
中度肥胖	$30 \leq BMI < 35$
重度肥胖	$BMI \geq 35$

# Exercise

- A、B、C三本書價格及折扣表如下，一顧客欲購買A: x 本、B: y 本、C: z 本（x、y、z 為使用者輸入），請計算需花費多少錢？

	定價	1~10本	11~20本	21~30本	31本以上
A	380	原價	打9折	打8.5折	打8折
B	1200	原價	打9.5折	打8.5折	打8折
C	180	原價	打8.5折	打8 折	打7折

# Exercise

## □ Code

```
x = int(input('A:'))  
y = int(input('B:'))  
z = int(input('C:'))
```

```
if x>=31:  
    A_discount = 0.8  
elif x>=21:  
    A_discount = 0.85  
elif x>=11:  
    A_discount = 0.9  
elif x>=1:  
    A_discount = 1  
else:  
    A_discount = 0
```

```
if y>=31:  
    B_discount = 0.8  
elif y>=21:  
    B_discount = 0.85  
elif y>=11:  
    B_discount = 0.95  
elif y>=1:  
    B_discount = 1  
else:  
    B_discount = 0
```

```
if z>=31:  
    C_discount = 0.7  
elif z>=21:  
    C_discount = 0.8  
elif z>=11:  
    C_discount = 0.85  
elif z>=1:  
    C_discount = 1  
else:  
    C_discount = 0
```

```
cost= x*380*A_discount + y*1200*B_discount + z*180*C_discount  
print('The total cost is %d' %(cost))
```

# Exercise

## □ 有使用 function 的 Code

```
def getDiscount(x, discounts):
    discount = 0
    if x>=31:
        discount = discounts[0]
    elif x>=21:
        discount = discounts[1]
    elif x>=11:
        discount = discounts[2]
    elif x>=1:
        discount = discounts[3]
    else:
        discount = discounts[4]
    return discount
```

```
x = int(input('A:'))
y = int(input('B:'))
z = int(input('C:'))
```

```
A_discounts=[8, 8.5, 9, 1, 0]
B_discounts=[8, 8.5, 9.5, 1, 0]
C_discounts=[7, 8, 8.5, 1, 0]
```

```
A_discount = getDiscount(x, A_discounts)
B_discount = getDiscount(y, B_discounts)
C_discount = getDiscount(z, C_discounts)
```

```
cost= x*380*A_discount + y*1200*B_discount + z*180*C_discount
print('The total cost is %d' %(cost))
```

# Exercise

- A、B、C三本書價格及折扣表如下，一顧客欲購買A: x本、B: y本、C: z本（x、y、z為使用者輸入），請計算需花費多少錢？
- 幾本區間是否可以輸入，定價表格(區間個數)是否可以輸入？

	定價	1~10本	11~20本	21~30本	31本以上
A	380	原價	打9折	打8.5折	打8折
B	1200	原價	打9.5折	打8.5折	打8折
C	180	原價	打8.5折	打8 折	打7折

# Exercise

## □ 判斷何種三角形

○ 當三個邊長能構成三角形時，再判斷該三角形為鈍角、銳角或是直角三角形，其判別方法如下：

- 1. 直角三角形：其中有兩個邊的平方和等於第三邊的平方。
- 2. 鈍角三角形：其中有兩個邊的平方和小於第三邊的平方。
- 3. 銳角三角形：任兩邊的平方和大於第三邊的平方。

○ 輸入三個整數

○ 輸出：顯示直角三角形(Right Triangle)、鈍角三角形(Obtuse Triangle)、銳角三角形(Acute Triangle)或無法構成三角形(Not Triangle)。

## □ 測試資料：

input  
5 12 13

output

Right Triangle

input  
3 4 5

output

Right Triangle

input  
1 2 3

output

Not Triangle

# Exercise

- 輸入每月網內、網外、市話、通話時間(sec)及網內、網外簡訊則數，求最佳資費。費率如下表：

資費類型	183型	383型	983型
月租費	183元	383元	983元
優惠內容	月租費可抵等額通信費		
語音 網內	0.08	0.07	0.06
(元/秒) 網外	0.1393	0.1304	0.1087
市話(元/秒)	0.1349	0.1217	0.1018
簡訊 網內	1.1287	1.1127	0.9572
(元/則) 網外	1.4803	1.2458	1.1243

- 輸入

- 網內語音(sec)、網外語音(sec)、市話(sec)、網內簡訊數、網內簡訊數測試資料：

input  
500 120 13 2 5  
output

# Exercise

## □ 撲克牌

- A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
- A~10 點數為 1~10，J, K, Q 為 0.5。

## □ X, Y 兩個人各發三張撲克牌，加總點數接近 10.5 則贏。

- 超過 10.5 爆掉分數為 0。

## □ 程式

- 輸入X, Y 兩個人各發的五張撲克牌。
- 輸出兩個人的點數，以及A贏或B贏或平手。

# Exercise

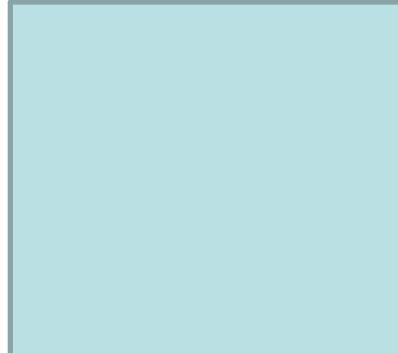
□ 寫出程式流程/步驟

○ 1. 輸入牌面符號 > 2. 轉換點數 > 3. 加總點數(調整點數) > 4. 比大小

□ 設計function和參數傳遞介面

```
def transferPoint(card):
    pork = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
    points =[1,2,3,4,5,6,7,8,9,10,0.5, 0.5,0.5]
    index = pork.index(card)
    return points[index]
```

```
def getSum(x, y, z):
```



return (aPoint)

```
def compare(x, y):
```



# Exercise

- 組合function測試
  - 設計多組資料測試

```
def test():  
    x1, x2, x3 = '10', 'Q', 'A'  
    y1, y2, y3 = '10', 'K', 'J'  
    aPoint = getSum(x1, x2, x3)  
    bPoint = getSum(y1, y2, y3)  
    compare(aPoint, bPoint)  
  
test()
```

# Exercise

## □ 檢查三門課程是否衝堂

- 依序輸入課程編號(數字)、上課小時數(1-3小時)、上課時間(ao  
星期1-5, 第1-9節)

輸入說明

1001 (第二門課課程編號)

3 (3小時)

11 (星期1 第1節課)

59 (星期5 第9節課)

25 (星期2 第5節課)

2020 (第二門課課程編號)

...

2030 (第三門課課程編號)

...

輸出說明

(兩課程編號衝突在哪幾節)

1001 and 2020 conflict on 25

# Python

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# for

## □ for in

- 序列物件(Sequence object)，有順序可數的元素
- 控制變數var又稱迴圈變數/迴圈索引

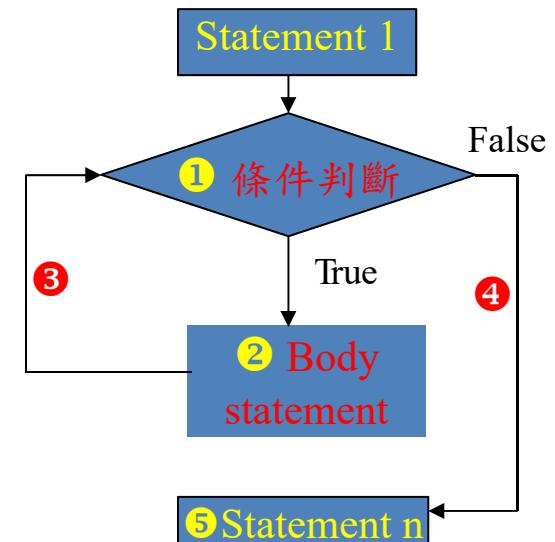
## □ for in 迴圈執行流程

- ① 條件判斷，True
  - 自序列物件中找到下一個元素，取出給var
- ② 執行本體所有指令(Body statement)
- ③ 回到①，
- ④ 條件判斷，若False
  - 找不到下一個元素
- ④ 跳出迴圈， ⑤

## □ 依次取出代入的動作稱為疊代

## □ break 和 continue 可在 for 迴圈中出現

```
Statement 1  
for var in sequence object :  
    Body statement  
Statement n
```



# for - range()

## □ range()函式回傳序列物件 range

使用方法	範例	執行結果
<b>range(終止值)</b> 數字串列到「終止值」的前一個 數字為止，沒有指定起始值，預 設起始值為0，沒有指定遞增值， 預設為遞增1。	for i in range(5): print(i)	0 1 2 3 4
<b>range(起始值, 終止值)</b> 數字串列由「起始值」開始到「終 止值」的前一個數字為止，沒有指 定遞增值，預設為遞增1	for i in range(2,6): print(i)	2 3 4 5
<b>range(起始值, 終止值, 遞增(減) 值)</b> 數字串列由「起始值」開始到 「終止值」的前一個數字為止， 每次遞增或遞減「遞增(減)值」。	for i in range(2,10,2): print(i)  for i in range(100,90,-3): print(i)	2 4 6 8  100 97 94 91

# for - range()

- 序列物件 (Sequence)
  - tuple, String, range(), List
- range(3) , 產生 0, 1, 2

```
def iterOp():
    myRange = range(8)
    print(myRange)
    print(type(myRange))
    print(myRange[2])
    myRange = range(3,8,2)
    print(myRange)
    print(myRange[2])
```

```
range(0, 8)
<class 'range'>
2
range(3, 8, 2)
7
```

# for - range()

□ range(0, 3, 1) , 產生 0, 1, 2

○ ① 條件判斷 , True

    ➤ 找到 0, 1, 2 的第 1 個 , 指定給 i , i=0

○ ② 印出 i , ③ 回到 ①

○ ① 條件判斷 , True

    ➤ 找到 0, 1, 2 的第 2 個 , 指定給 i , i=1

○ ② 印出 i , ③ 回到 ①

○ ① 條件判斷 , True

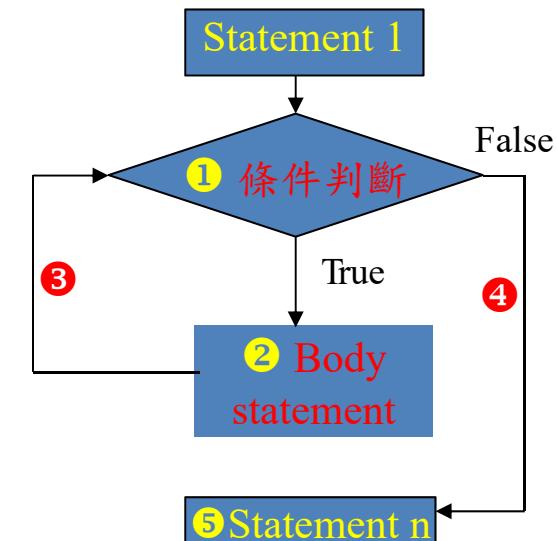
    ➤ 找到 0, 1, 2 的第 3 個 , 指定給 i , i=2

○ ② 印出 i , ③ 回到 ①

○ ① 條件判斷 , False ④

    ➤ 序列物件找不到下一個資料 , 跳出迴圈 ⑤

```
def myPrint():
    for i in range(0, 3, 1):
        print(i)
```

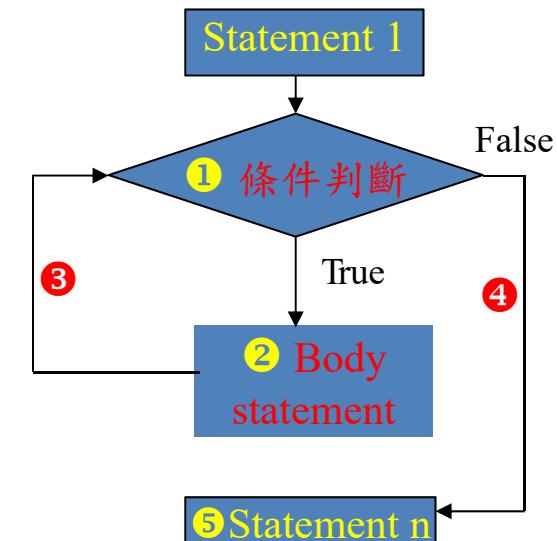


# for - range()

## □ 使用者輸入n，產生 0, 1, 2, .., n

- 程式
- 程式編號
- 程式編號執行順序

```
1. def myPrint(num):  
2.     for i in range(0, num+1):  
3.         print(i)  
4.  
5. n = int(input('number:'))  
6. myPrint(n)
```



# for - range()

□ range(1, 4) , 產生 1, 2, 3

○ ① 條件判斷 , True

➢ 找到 1, 2, 3 的第 1 個 , 指定給 i , i=1

○ ② 印出 i ,  $s = s + i = 0 + 1 = 1$  , ③ 回到 ①

○ ① 條件判斷 , True

➢ 找到 1, 2, 3 的第 2 個 , 指定給 i , i=2

○ ② 印出 i ,  $s = s + i = 1 + 2 = 3$  , ③ 回到 ①

○ ① 條件判斷 , True

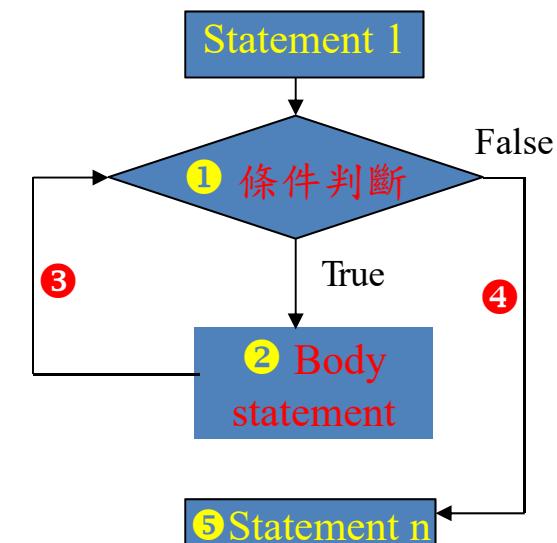
➢ 找到 1, 2, 3 的第 3 個 , 指定給 i , i=3

○ ② 印出 i ,  $s = s + i = 3 + 3 = 6$  , ③ 回到 ①

○ ① 條件判斷 , False ④

➢ 序列物件找不到下一個資料 , 跳出迴圈 ⑤

```
def myPrint():
    s=0
    for i in range(1, 4):
        print(i)
        s = s + i
    print('sum=', s)
```



# for 迴圈

- 輸出  $1 + 2 + 3 + 4 + 5 + \dots + 10$
- 輸出  $1 * 2 * 3 * 4 * 5 * \dots * 10$

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

```
def getSum(num):
    sumValue = 0
    for i in range(num):
        sumValue = sumValue + (i+1)
    return sumValue

def main():
    num = int(input("Input a number:"))
    myPrint(num)
    getSum(num)
    getProduct(num)

main()
```

```
def myPrint(n):
    for i in range(0, n, 1):
        print(i)
```

```
def getProduct(num):
    product = 1
    for i in range(num):
        product = product *i
    return product
```

# Exercise

- 輸入開始值m、結束值n與遞增值step，計算數值加總結果
  - 例如 $3 + 6 + 9 + 12$ ，輸入3為開始值，12為結束值，3為遞增值。

```
def mySum(m, n, step):
```

```
    return sum
```

```
def main():  
    m = int(input("Input a min number:"))  
    n = int(input("Input a max number:"))  
    step = int(input("Input a step number:"))  
    main_sum = mySum(m, n, step)  
    print('sum (%d ~ %d)= %d' %(m, n, main_sum))
```

```
main()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# Exercise

□ 1.若計算m~n其中的偶數，程式碼?

□ 2.輸出兩數

- $m + (m+2) + (m+4) + (m+6) + \dots + n$
- $m * (m+3) * (m+6) * (m+9) * \dots * n$

```
def forOps(m, n):
    for index in range(____):
        print(index)
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# Exercise

## □ 改寫以下程式

- 印出myString中大寫字母
- 計算myString有幾個字元？
- 計算myString有幾個大寫字元？

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

```
def forOps():
    myString = "ATCgATAgcTCGaTCG"
    for index in myString:
        if index.isupper():
            print(index)
```

# for 迴圈

- in 在串列中，一個一個依序取出

```
def forOps():
    myList = ["asm", "C", "C++", "Java", "iOS", "Ruby", "perl", "delphi", "python"]
    for index in myList:
        print(index)
```

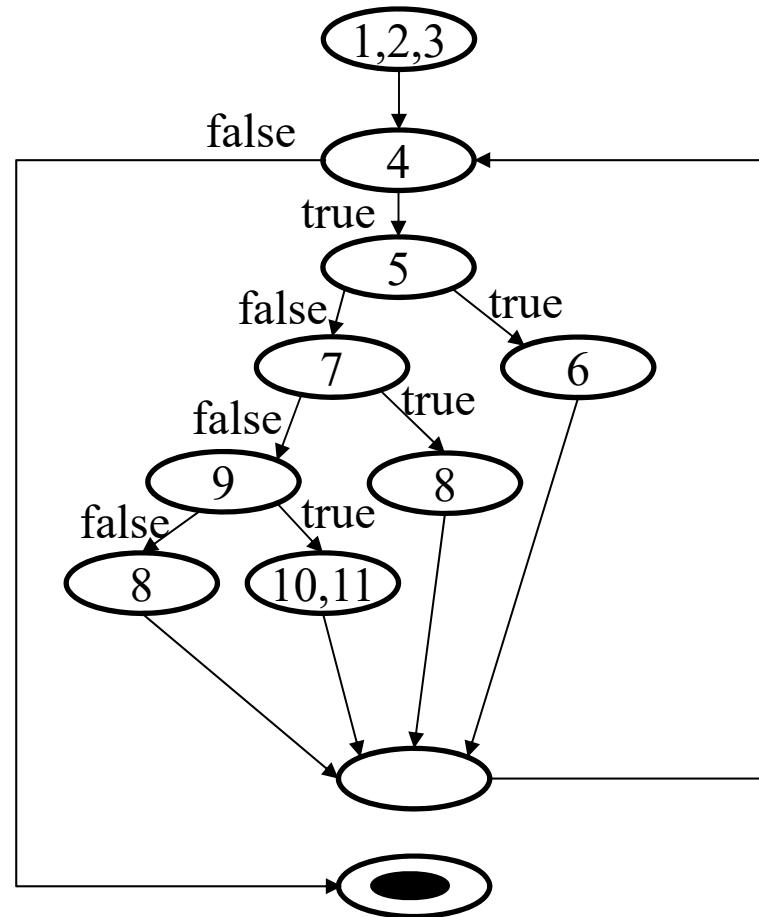
```
1 def forOps():
2     i = 1
3     myList = ["asm", "C", "python", "C++", "Java", "iOS", "Ruby", "perl", "delphi"]
4     for index in myList :
5         if (index == "python"):
6             print(i,index)
7         elif (index == "Java"):
8             print(i, index)
9         elif (i%3!=0):
10            print(i, index)
11            i = i+1
12        else:
13            i=i+1
14 forOps()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# for 迴圈

	i	index	i%3	output
14, 1, 2, 3, 4	1	asm	1	1 asm
5, 7, 9, 10,		C	2	2 C
11, 4	2			
5, 7, 9, 10	3	python	0	3 python
11, 4		C++		
5, 6				
4,		Java		4 Java
12, 13	4		1	
4,		iOS		4 iOS
7, 8				
4,		Ruby	2	5 Ruby
9, 10	5			
11, 4		perl	0	
9, 10				
11, 4	6		1	7 delphi
12, 13	7	delphi		
4				
10,				
11				

# for 迴圈



# for 迴圈

□ 輸入 N 和 N 個整數，輸出其中最大的數。

○ 例如N=5，5個整數 11, 45, 8, 13, 22，

```
def getMax(N):
    num = int(input())
    maxValue = minValue = num;
    for i in range(N-1):
        num = int(input())
        if (num>maxValue):
            maxValue = num
    return maxValue

def main():
    Num = int(input("Input a number:"))
    x = getMax(Num)
    print(x)

main()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# for 迴圈

- 輸入 N 和 N 個整數，輸出其中最大和最小的數。
  - 例如N=5，5個整數 11, 45, 8, 13, 22，

```
def getMaxMin(N):  
    num = int(input())  
    maxValue = minValue = num;  
    for i in range(N-1):  
        num = int(input())  
        if (num>maxValue):  
            maxValue = num  
        if (num<minValue):  
            minValue = num  
    return maxValue, minValue  
  
def main():  
    num = int(input("Input a number:"))  
    x, y = getMaxMin(num)  
    print(x, y)  
  
main()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# for 迴圈

## □ function有迴圈

```
def myPrint01():
    for i in range(0, 10, 1):
        print(i)

def myPrint02(m, n):
    for j in range(m, n, -1):
        print(j, end="")          #不換行

def myPrint03(m):
    for j in range(0, 2*m-1, 1):
        print(j, end="")          #不換行

def main():
    num = 5
    myPrint01()
    myPrint02(num, 8)
    myPrint03(num)
    print()                  #預設換行
    main()
```

```
def myPrint04(listData):
    for i in listData:
        print(i)

def main():
    listData = ['a', 'b', 'c', 'd']
    myPrint04(listData)

main()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# for - range

□ 要印出 1 個'\*'

```
def myPrint():
    for i in range(1):
        print('*', end="")
```

□ 要印出 2 個'\*'

```
def myPrint():
    for i in range(2):
        print('*', end="")
```

□ 要印出 3 個'\*'

寫成程式

```
def myPrint():
    for i in range(3):
        print('*', end="")
```

□ 要印出 n 個'\*'

○ n 是 function 參數

```
def myPrint(n):
    for i in range(n):
        print('*', end="")
```

○ n 從鍵盤輸入

```
n = int(input())
for i in range(n):
    print('*', end="")
```

# for - range

- 要印出 n 個'\*'
  - n 是 function 參數

```
def myPrint(n):  
    for i in range(n):  
        print('*', end='')
```

記住這個

- 要印出
  - 第一行 1 個'\*'
  - 第一行 2 個'\*'
  - 第一行 3 個'\*'

寫成程式

```
def myPrintS():  
    myPrint(1)  
    print()  
    myPrint(2)  
    print()  
    myPrint(3)  
    print()
```

- 寫成LOOP變成這樣

```
def myPrintS():  
    for i in range(1, 4)  
        myPrint(i)  
        print()
```

# for - range

## □ 合起來寫

- 使用兩個LOOP
- 不使用myPrint()

```
def myPrintS():  
    for i in range(1, 4)  
        for j in range(i)  
            print('*', end='')  
    print("")
```

# for - range

- 要印出 n 個'\*'
  - n 是 function 參數

```
def myPrint(n):  
    for i in range(n):  
        print('*', end="")
```

記住這個

- 要印出

```
****  
***  
**  
*
```

寫成程式

```
def myPrintT():  
    myPrint(4)  
    print()  
    myPrint(3)  
    print()  
    myPrint(2)  
    print()  
    myPrint(1)  
    print()
```

- 寫成LOOP變成這樣

```
def myPrintS():  
    for i in range(4, 0, -1)  
        myPrint(i)  
        print()
```

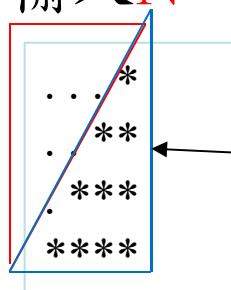
# for - range

- 要印出 n 個自訂符號mark
  - mark可以是'\*', '!'
  - n, mark 是 function 參數

```
def myPrint(n, mark):  
    for i in range(n):  
        print(mark, end="")
```

記住這個

- 輸入  $N = 4$ ，要印出



切割處理

- 第一行印3個.，1個\*
- 第二行印2個.，2個\*
- 第三行印1個.，3個\*
- 第四行印0個.，4個\*

寫成程式

```
def myPrintT(N):  
    if N == 1:  
        print('.', end="")  
        print("*", end="")  
    else:  
        print('.', end="")  
        myPrintT(N-1)  
        print("*", end="")  
    print("")
```

- 寫成LOOP變成？

# for - range

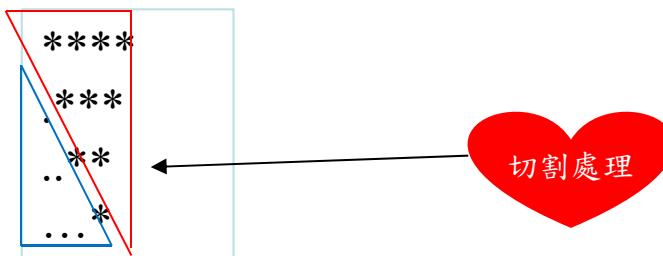
□ 寫成LOOP變成這樣

- 假設N=4
- 請將for 展/拆開，看是否跟上面程式一樣

```
def myPrintT(N):  
    for i in range(1, N+1):  
        myPrint(N-i, '.')  
        myPrint(i, '*')  
        print("")
```

# for - range

□ 輸入  $N = 4$ ，要印出



- 第一行印0個.，4個\*
- 第一行印1個.，3個\*
- 第一行印2個.，2個\*
- 第一行印3個.，1個\*

寫成程式

```
def myPrintT(N):  
    myPrint(0, '.')  
    myPrint(__, '*')  
    print()  
    myPrint(1, '.')  
    myPrint(__, '*')  
    print()  
    myPrint(2, '.')  
    myPrint(__, '*')  
    print()  
    myPrint(3, '.')  
    myPrint(__, '*')  
    print()
```

□ 寫成LOOP變成？

寫成程式

# for - range

□ 要印出 1 個'1'

```
def myPrint():
    for i in range(1,2):
        print(i)
```

□ 要印出 12

```
def myPrint():
    for i in range(1, 3):
        print(i)
```

□ 要印出 123

寫成程式

```
def myPrint():
    for i in range(1, 4):
        print(i)
```

□ 要印出 123,..n

○ n是function參數

```
def myPrint(n):
    for i in range(1, n+1):
        print(i)
```

○ n從鍵盤輸入

```
n = int(input())
for i in range(1, n+1):
    print(i)
```

# for - range

- 要印出 123,..n
  - n 是 function 參數
- 要印出

```
1  
12  
123  
1234
```

寫成程式

```
def myPrint(n):  
    for i in range(1, n+1):  
        print(i)
```

記住這個

```
def myPrintT():  
    myPrint(1)  
    print()  
    myPrint(2)  
    print()  
    myPrint(3)  
    print()  
    myPrint(4)  
    print()
```

```
def myPrintS():  
    for i in range(1, 5)  
        myPrint(i)  
    print()
```

- 寫成 LOOP
- 寫成 兩層 LOOP，不用 function?

# for - range

- 要印出 123,..n
  - n 是 function 參數
- 寫成 Loop

```
1  
12  
123  
1234
```

寫成程式

```
def myPrint(n):  
    for i in range(1, n+1):  
        print(i)
```

記住這個

- 寫成兩層 Loop，不用 function?
  - 合併前兩個程式
  - 兩個 Loop 變數不能用同一個 i
  - 一個用 i, 一個用 j

寫成程式

```
def myPrintS():  
    for i in range(1, 5)  
        myPrint(i)  
        print()
```

```
def myPrintS():  
    for i in range(1, 5)  
        myPrint(i)  
        for j in range(1, i+1):  
            print(j)  
        print()
```

# for - range

- 要印出 123,..n
  - n 是 function 參數
- 要印出

```
1234  
123  
12  
1
```

寫成程式

```
def myPrint(n):  
    for i in range(1, n+1):  
        print(i)
```

記住這個

```
def myPrintT():
```

- 寫成LOOP

```
def myPrintS():  
    for i in range(_____  
        myPrint(i)  
        print()
```

# for - range

- 要印出 1357,..n
  - n 是 function 參數
- 要印出

```
1  
13  
135  
1357
```

寫成程式

```
def myPrint(n):  
    for i in range(1, n+1, 2):  
        print(i)
```

記住這個

```
def myPrintT():  
    myPrint(1)  
    print()  
    myPrint(2)  
    print()  
    myPrint(3)  
    print()  
    myPrint(4)  
    print()
```

```
def myPrintS():  
    for i in range(1, 5)  
        myPrint(i)  
    print()
```

- 寫成LOOP

# for - range

- 要印出 1357,..n
  - n是function參數
- 要印出

```
1357  
135  
13  
1
```

寫成程式

```
def myPrint(n):  
    for i in range(1, n+1, 2):  
        print(i)
```

記住這個

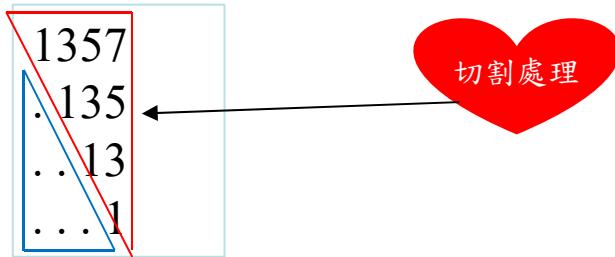
```
def myPrintT():  
    myPrint(4)  
    print()  
    myPrint(3)  
    print()  
    myPrint(2)  
    print()  
    myPrint(1)  
    print()
```

```
def myPrintS():  
    for i in range(4, 0, -1)  
        myPrint(i)  
    print()
```

- 寫成LOOP

# for - range

- 要印出 1357,..n
  - n 是 function 參數
- 要印出



寫成程式

- 寫成LOOP

```
def myPrint(n):  
    for i in range(1, n+1, 2):  
        print(i)
```



```
def myPrintT():
```

```
def myPrintS():  
    for i in range(4, 0, -1)
```

print()

# for 迴圈

## □ function有迴圈

```
def myPrint01():
    for i in range(0, 10, 1):
        print('*')

def myPrint02(m, n):
    for y in range(m, n, -1):
        print('#',end="")

def myPrint03(m):
    for y in range(0, 2*m-1, 1):
        print('*',end="")

def main():
    num = 5
    myPrint01()
    myPrint02(num, 5)
    myPrint03(num)

main()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，變數的值如何改變？程式流程如何變化？

```
def myPrint():
    num = 5
    for x in range(1, num+1):
        for y in range(num, x, -1):
            print(x, y, end="")
        for y in range(0, 2*x-1, 1):
            print(x, y, end="")
    print()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# for 巢狀(nest)迴圈

- 迴圈內還有迴圈，印出金字塔\*

```
def myPrint():
    #num = int(input("Input a number: "))
    num = 5
    for x in range(1, num+1):
        for y in range(num, x, -1):
            print(' ',end="")
        for y in range(0, 2*x-1, 1):
            print('*',end="")
        print()
```

```
*  
***  
*****  
*****  
*****
```

# for 巢狀(nest)迴圈

- 將下面 code 寫成二個 function，如何寫？
- 每一個function只能有一層迴圈其中有一個function，其迴圈內呼叫另一個function

```
def printGold(num):  
    for x in range(1, num+1):  
        for y in range(num, x, -1):  
            print(' ',end="")  
        for y in range(0, 2*x-1, 1):  
            print('*',end="")  
        print()  
  
def main():  
    num = int(input("Input a number:"))  
    printGold(num)
```

# Exercise

- 將Code寫成二個function，每一個function使用一層迴圈

```
1  
12  
123  
1234  
12345  
  
54321  
4321  
321  
21  
1  
  
1  
22  
333  
4444  
55555
```

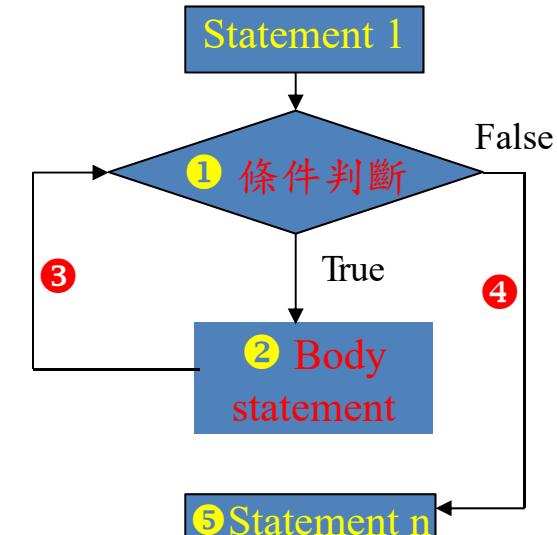
```
1  
121  
12321  
1234321  
123454321  
  
____1____  
____212____  
____32123____  
____4321234____  
  
____4321234____  
____32123____  
____212____  
____1____
```

# while

- 每次判斷條件，為True繼續執行本體指令，False結束迴圈
- 不繼續執行區塊動作稱跳出迴圈或結束迴圈
- 適用於迴圈圈數未知

```
Statement 1  
while Condition :  
    Body Statement  
    Statement n
```

```
def myPrint():  
    i = 0;  
    while (i<10):  
        print(i)  
        i = i+1  
  
def main():  
    myPrint()  
  
main()
```



# while

- 正序印出 hi, python.

```
def myPrint01():
    tmp = "hi, python."
    print(tmp)
    i = 0
    while(i<len(tmp)):
        print(tmp[i])
        i = i + 1
```

```
def myPrint02():
    tmp = "hi, python."
    i = 0
    for c in tmp:
        print(c)
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

#處理特別的字元

```
def myPrint03():
    tmp = "hi, python."
    i = 0
    while i<len(tmp)):
        if tmp[i] == 'h':
            print('###')
        else:
            print(tmp[i])
        i = i + 1
```

# while

```
def mySum(m, n):  
    i = m  
    sumValue = 0  
    while (i <= n):  
        sumValue = sumValue + i  
        i = i + 1  
    print(sumValue)  
    return sumValue
```

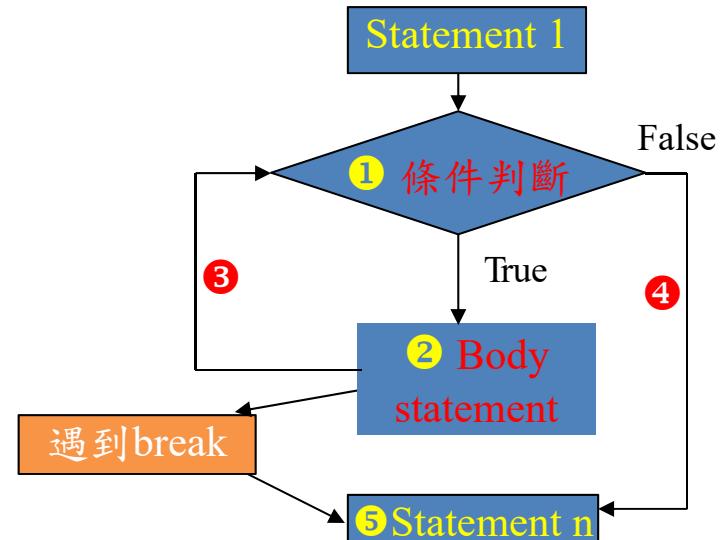
```
def main():  
    minValue = int(input("Input a min number:"))  
    maxValue = int(input("Input a max number:"))  
    main_sum = mySum(minValue, maxValue)  
    print('sum (%d ~ %d)= %d' %(minValue, maxValue, main_sum))  
main()
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

# break

```
#當 i 數到5時就不做
def test01():
    for i in range(1,10):
        number = number +i
        if (i==5):
            break
```

```
def test02():
    sum = 0
    while (True):
        inputOrder =input()
        sum = sum + i
        print(inputOrder, sum)
        if (inputOrder == -1):
            break
```



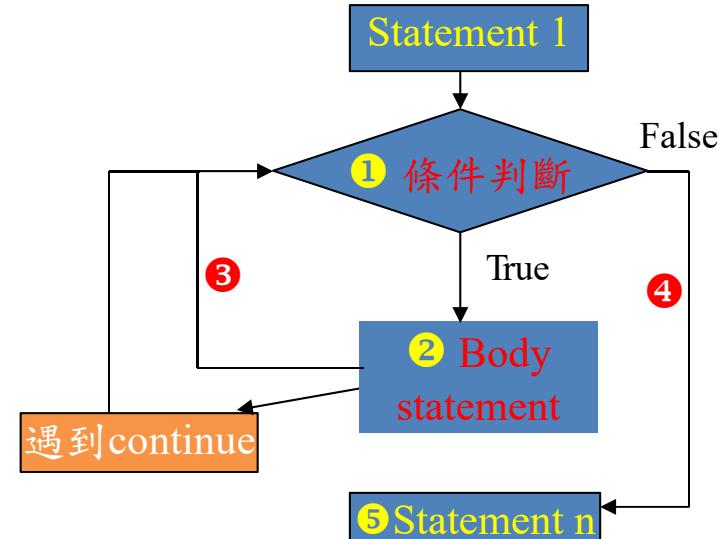
利用 break 在任何時候 跳出迴圈

# continue

```
#當number 沒超過20 不印@，超過印@
def test03():
    number = 0
    for i in range(1,10):
        number = number + i
        if (number<20):
            continue
        print('@', i, number)

    print(i, number)
```

```
def myPrint():
    tmp = "hi, python."
    i = 0
    while(True):
        print(tmp[i])
        i = i + 1
        if (i == (len(tmp) -1)):
            continue
```



利用 continue 在任何時候略過迴圈  
(略過本次迴圈剩餘的運算)

# Exercise

## □ 計算最大公因數

$x = 42$	42	75	1
$y = 75$	33	42	1
	9	33	3
	6	27	3
	3	6	2
	6	0	5
	0		

GCD

- 以較大數 (75) 為被除數，較小數 (42) 為除數， $75 / 42 = 1$  餘 33
- 以前一步驟的除數為被除數，餘數為除數， $42 / 33 = 1$  餘 9
- $33 / 9 = 3$  餘 6
- $9 / 6 = 1$  餘 3
- $6 / 3 = 2$  餘 0，除數 3 為最大公因數

```
def gcd(x, y):
    while (x>0) and (y>0):
        if (x>y):
            x = x%y
        else:
            y = y%x
    return (x if x>y else y)
```

```
print(gcd(18, 24))
print(gcd(90, 36))
```

1	123	321	2	商
	75	246		
1	48	75	1	被除數
	27	48		除數
3	21	27	1	被除數 除數X商
	18	21		餘數
	3	6	2	
		6		
		0		

# Exercise

## □ 計算BMI值的function

- BMI值計算公式:  $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺}^2)$
- 例如：一個52公斤的人，身高是155公分，則BMI為：
- $52(\text{公斤}) / 1.552 (\text{公尺}^2) = 21.6$
- 正常範圍為  $BMI=18.5 \sim 24$
- 輸入身高、體重，輸出BMI值。
- 身高正常範圍 0.5~2.50 公尺，體重正常20~300 公斤，若輸入不在正常範圍，輸出 "Input Error (0.5~2.50)" / "Input Error (20~300)"，請重新輸入。
- 若BMI值太高，輸出 "BMI too hight"，太低輸出 "BMI too low"。
- 可以接受不斷輸入計算，直到輸入-1停止。
- 不會有身高與體重皆不正常之情況。

Sample input :

3 20  
1.55 52  
2.4 299  
-1

Sample output :

Input Error 0.5~2.50  
21.64  
BMI too hight

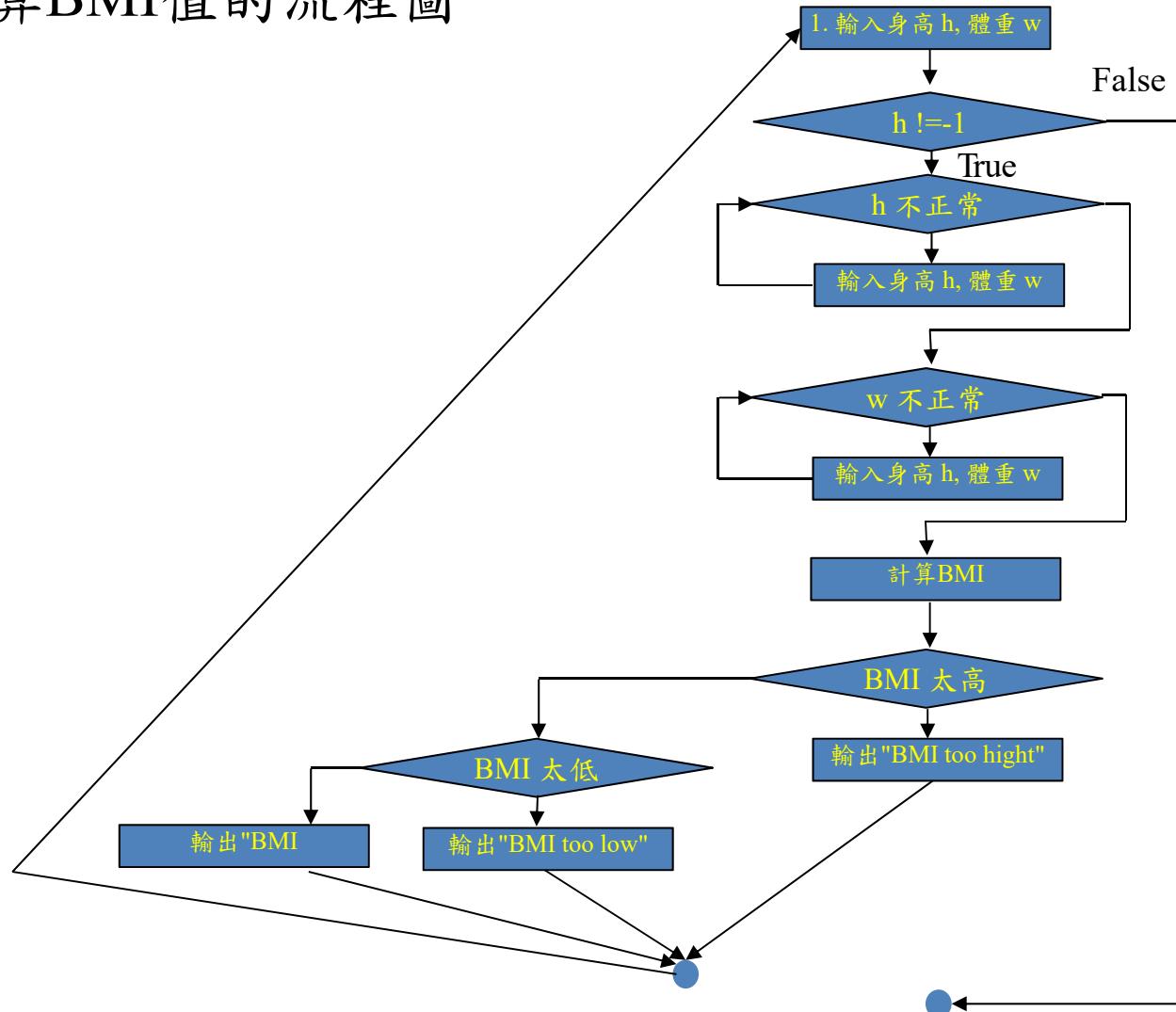
# Exercise

## □ 計算BMI值的流程說明

- 1. 輸入身高 h, 體重 w
- 2. 假如身高  $h == -1$  停止程式
- 3. 假如身高不在正常範圍 0.5~2.50
  - 輸出 "Input Error (0.5~2.50)"
  - 輸入身高 h, 體重 w
- 4. 假如體重不在正常範圍 20~300
  - 輸出 "Input Error (20~300)"
  - 輸入身高 h, 體重 w
- 5. 計算  $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺}^2)$
- 6. 假如 BMI太高，輸出 "BMI too hight"
- 7. 假如 BMI太低，輸出 "BMI too low"。
- 8. 回到步驟 1

# Exercise

## □ 計算BMI值的流程圖



# Exercise

```
def bmi_input():
    x=input()
    x=x.split()
    CM=float(x[0])
    KG=float(x[1])
    CM=float(input())
    KG=float(input())
    BMI=round((KG//CM**2),2)
    print(BMI)
    if BMI > 24:
        print('BMI too high')
    if BMI < 18.5:
        print('BMI too low')
    if BMI >=18.5 and BMI <=24:
        print(BMI)
    while(CM!=-1):
        break
    if CM>=0.5 and CM<=2.5:
        print('Input Error 0.5~2.5')
    if KG >=20 and KG<=300:
        print('Input Error 20~300')
bmi_input()
```

錯在哪裡?

# Exercise

```
def check(CM, KG):
    if CM<0.5 or CM>2.5:
        print('Input Error 0.5~2.5')
        return 0
    if KG <20 or KG>300:
        print('Input Error 20~300')
        return 0
    return 1

def bmi_input():
    while(True):
        x=input()
        x=x.split()
        if x[0]=='-1':
            break
        CM=float(x[0])
        KG=float(x[1])
        if (check(CM, KG)==0):
            continue
        BMI=round((KG/CM**2),2)
        print(BMI)
        if BMI > 24:
            print('BMI too high')
        if BMI < 18.5:
            print('BMI too low')
        if BMI >=18.5 and BMI <=24:
            print(BMI)

bmi_input()
```

這樣對嗎?

# Exercise

```
def inputBMI():
    x=input().split()
    stop=0
    if x[0]=='-1':
        stop = -1
    CM=float(x[0])
    KG=float(x[1])
    return stop, CM, KG
```

```
def check(CM, KG):
    if CM<0.5 or CM>2.5:
        print('Input Error 0.5~2.5')
        return 0
    if KG <20 or KG>300:
        print('Input Error 20~300')
        return 0
    return 1
```

```
def output(CM,KG):
    BMI=round((KG/CM**2),2)
    print(BMI)
    if BMI > 24:
        print('BMI too high')
    if BMI < 18.5:
        print('BMI too low')
    if BMI >=18.5 and BMI <=24:
        print(BMI)
```

```
def computeBMI():
    while(True):
        stop, CM, KG = inputBMI()
        if stop=='-1':
            break
        if (check(CM, KG)==0):
            continue
        output(CM, KG)
```

```
computeBMI()
```

差別在哪裡?

# Exercise撲克牌比大小

## □ 撲克牌

- A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
- A~10 點數為 1~10，J, K, Q 為 0.5。

## □ 電腦與玩家各隨機發撲克牌，加總點數接近 10.5 則贏。

- 超過 10.5 爆掉分數為 0。

## □ 程式

- 電腦隨機發X撲克牌，使用者可選擇要牌或不要牌。
- 電腦隨機發電腦撲克牌，電腦判斷是否停發牌。
- 輸出電腦與玩家的點數，以及電腦贏或玩家贏或平手。

# Exercise撲克牌比大小

## □ 點數

- A~10 點數 1~10，J, K, Q 為 0.5。

## □ 玩法

- 電腦與玩家各隨機發撲克牌，加總點數接近 10.5 則贏。

- 超過 10.5 爆掉分數為 0 且該方不得繼續要牌。

- 任一回合並未要牌的一方，失去要牌權利。

- 程式發一張撲克牌給玩家，玩家可選擇要牌或不要牌。

- 程式發一張撲克牌給電腦，電腦判斷是否停發牌。

## □ 電腦判斷要牌：1. 總點數比玩家小 或 2. 總點數8點以下(含)

## □ 輸出電腦與玩家點數，電腦贏或玩家贏或平手(Tie)輸出：

It's a tie)

# Exercise撲克牌比大小

## □ 輸入範例說明

- A 先發一張給給玩家(玩家獲得A)
- J 再發一張給電腦(電腦獲得J)
- Y 玩家選擇要牌
- 9 發一張給玩家(玩家獲得9)
- 8 電腦牌面0.5點，未超過8點，再發一張給電腦(電腦獲得8)
- N 玩家選擇不要牌
- 5 電腦牌面8.5，低於玩家的10，因此再抽(獲得5)

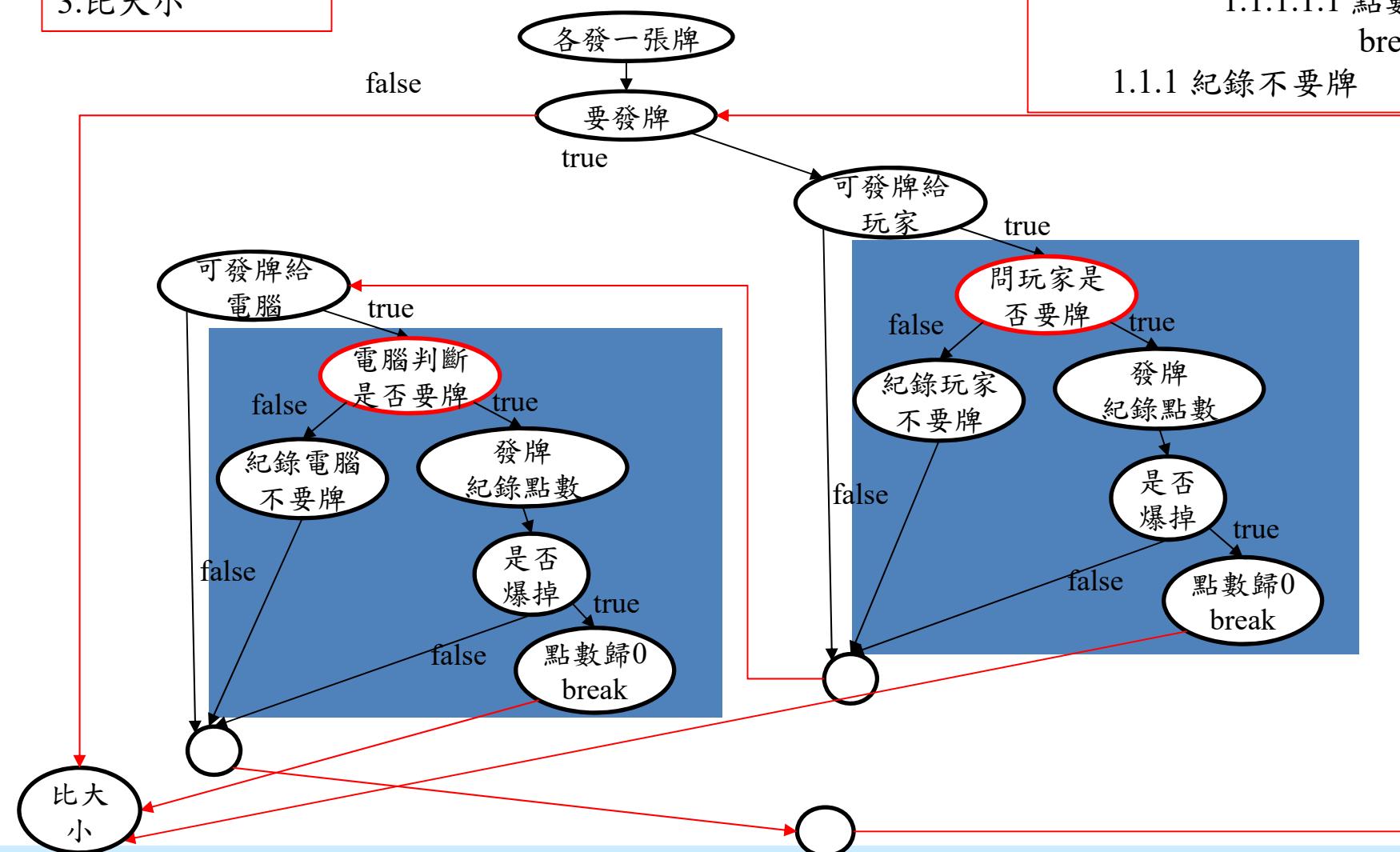
# 撲克牌比大小

流程

1. 各發一張牌
2. 發牌迴圈
  - 2.1. 發牌給玩家
  - 2.2. 發牌給電腦
3. 比大小

發牌給(玩家/電腦)

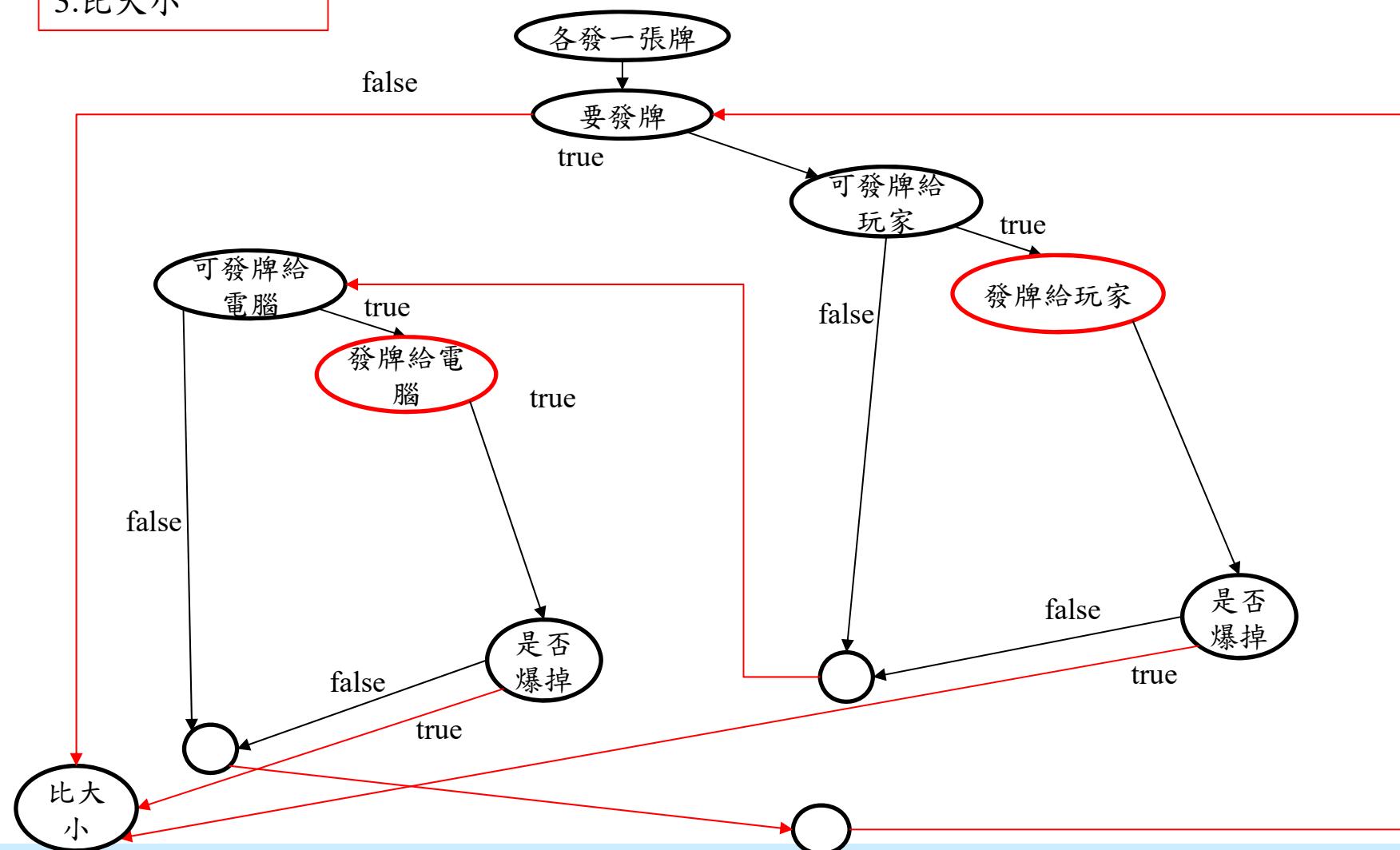
1. 可發牌
  - 1.1 是否發牌
    - 1.1.1. 發牌、紀錄點數
    - 1.1.1.1 是否爆掉
    - 1.1.1.1.1 點數歸0  
break
  - 1.1.1 紀錄不要牌



# 撲克牌比大小

流程

1. 各發一張牌
2. 發牌迴圈
  - 2.1. 發牌給玩家
  - 2.2. 發牌給電腦
3. 比大小



# 撲克牌比大小

流程

1. 各發一張牌
2. 發牌迴圈
  - 2.1. 發牌給玩家
  - 2.2. 判斷是否結束
  - 2.3. 發牌給電腦
  - 2.4. 判斷是否結束
3. 比大小

發牌給玩家 (deal, 參數 - role, myPoint, bPoint, judge\_func, 回傳值 - role, point, over  
1. 可發牌(設定變數 - role)

1.1 是否發牌(輸入變數-wantCard)

變成function判斷是否要牌

1.1.1. 發牌、紀錄點數(變數-myPoint)

1.1.1.1 是否爆掉(function-isExplode)

1.1.1.1.1 點數歸0 (myPoint=0) break (game over)

變成function調整點數

1.1.1 紀錄不要牌(role=false)

# 撲克牌比大小

```
def playerJudge(a, b):
    isWant=input()
    if (isWant=='Y'):
        return True
    else:
        return False
```

```
def computerJudge(a, b):
    if (a<b):
        return True
    elif (a<8):
        return True
    else:
        return False
```

```
def justPoint(point):
    over = False
    if (point>10.5):
        point=0
        over=True
    return over, point
```

```
def deal(myPoint, bPoint, judge):
    over = False
    isWant=judge(myPoint, bPoint)
    if (isWant==True):
        myPoint=myPoint+transferPoint(input())
        over, myPoint=justPoint(myPoint)
    return isWant, myPoint, over
```

```
def game():
    over = False
    computer=player=True
    playerPoint=transferPoint(input())
    computerPoint=transferPoint(input())
    while (player or computer):
        if (player==True):
            player, playerPoint, over=deal(playerPoint, computerPoint, playerJudge)
        if (over==True):
            break
        if (computer==True):
            computer, computerPoint, over=deal(computerPoint, playerPoint, computerJudge)
        if (over==True):
            break
    print(playerPoint, computerPoint)
```

初始化與初始輸入

# Exercise撲克牌比大小

## □ 遊戲規則修改

- 可以有多位玩家，電腦當莊家
- 玩家可以只在某一輪放棄要牌
- 電腦判斷是否要牌，要考慮多位玩家

# Exercise

- 猜數字，隨機(外部輸入)產生一個介於1~10的答案，使用者猜中則停止輸入，根據使用者輸入提示以下訊息：
  - 1.猜太大
  - 2.猜太小
  - 3.猜中了

```
import random
def myFunction():
    ans = random.randint(1,10)
    while True:
        inputData = int(input("Guess 1~10: "))
        if (inputData == ans):
            print("Right")
            break

def main():
    myFunction()
```

# for/while的使用時機

- 需重複進行運算的時候使用迴圈(for/while)
  - 重複次數可清楚計算或當疊代明顯時使用for迴圈
  - 重複次數難以計算，但條件清楚，或有條件的重複時使用
- 重複結構while和for都支援else敘述
  - 迴圈不是因break, return或例外終止時(正常中止)，else\_suite會被執行

```
while Condition :  
    while_body  
else:  
    else_suite
```

```
for var in Condition :  
    for_body  
else:  
    else_suite
```

# Python Unit Test

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

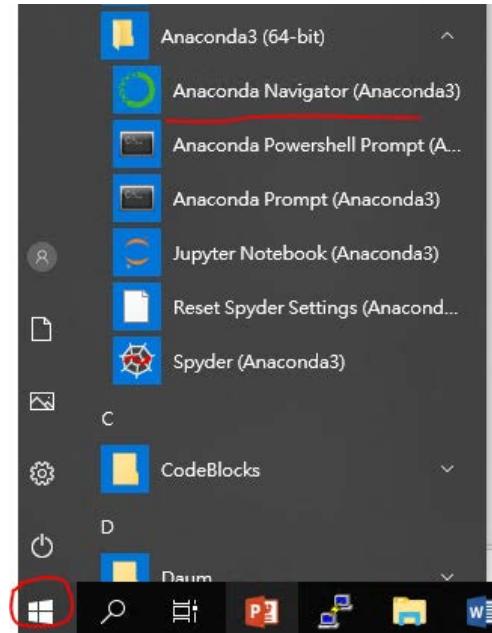
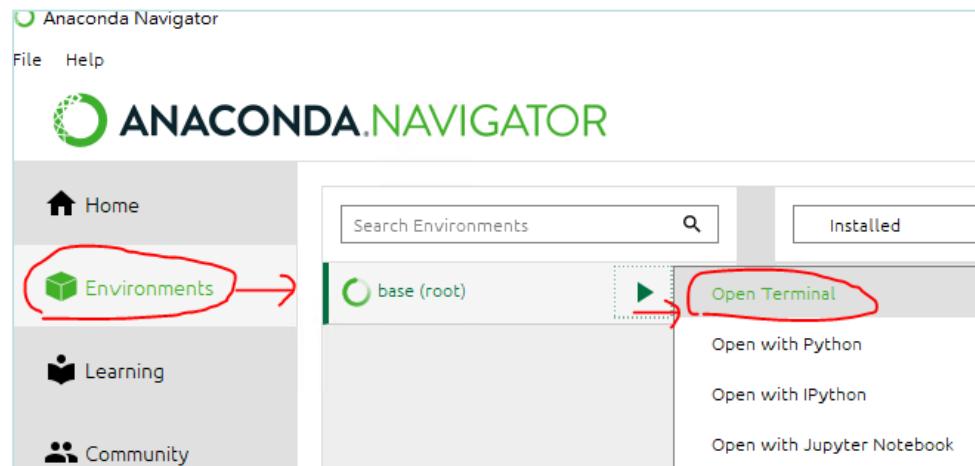
臺北科技大學資訊工程系

# 測試程式

- 測試程式/作業正確性，找出程式錯誤。
  - 設計測試案例(test case)，滿足問題/需求描述的各種條件/情境
  - 撰寫測試驅動程式(test driver)、測試方法(method)
  - 執行測試驅動程式，使的主要程式通過這些測試案例。
- 單元測試(Unit Test)概念
  - 程式設計師，測試自己所寫的程式模組是否有錯誤。
  - 撰寫測試方法(method)，驗證某功能在某測試案例，程式如預期運作。
- 單元測試(Unit Test)重點
  - 設計**足夠多**的測試案例，滿足各種問題條件
  - **正確**：測試主程式讓結果正確
  - **完整**：測試主程式，使涵蓋度達到100%

# 安裝 coverage 套件

## □ 開始:Anaconda:Anaconda Navigator



## □ Environments: Open Terminal

## □ pip install coverage

```
選取 C:\Windows\system32\cmd.exe

(base) C:\Users\jykuo>pip install coverage
Collecting coverage
  Downloading coverage-6.0.2-cp38-cp38-win_amd64.whl (222 kB)
|██████████| 222 kB 2.2 MB/s
Installing collected packages: coverage
Successfully installed coverage-6.0.2
```

# 題目

□ 輸入：體重(公斤), 身高(公分)

○ 合法範圍：體重( $3 \leq \sim \leq 200$ )，身高( $0.50 \leq \sim \leq 2.50$ )。

○ 公式： $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺平方})$

□ 輸出

○ 身高、體重輸入範圍錯誤，輸出-1。

○ 正常輸入範圍：輸出兩位小數，第三位小數後面去尾

○ 【輸出兩位小數，四捨五入】

# 撰寫程式

## □ 撰寫測試BMI程式

- 造出 c:\TEST 目錄，將以下程式存入此目錄
- 程式名稱 mybmi.py

```
def computeBMI(kg, M):  
    if kg>200 and M>2.5:          #錯誤，範圍判斷錯誤  
        return -1  
    if kg<=2 or M<=0.05:  
        return -1  
    BMI = round(kg/(M*M),2)       #錯誤，四捨五入取兩位小數  
    #BMI = ((100*kg/(M*M))//1)/100 #正確，去尾，乘100取整數，再除100取兩位小數  
    return BMI
```

# 設計測試案例

## □ 測試案例分類

- Normal

- 輸入：52公斤，1.55公尺，
- BMI :  $52(\text{公斤})/(1.55 \times 1.55)(\text{公尺}^2) = 21.64412$ (兩位小數 21.64)。

- 錯誤身高

- 錯誤體重

# 撰寫測試驅動程式

## □ 使用unittest (PyUnit) 測試框架

- 將以下程式mybmitest.py放入 c:\TEST

```
01 import unittest          #匯入測試框架套件
02 #import coverage         #匯入涵蓋度紀錄套件
03 import mybmi              #匯入 mybmi.py 內的程式
04 class TestBMI(unittest.TestCase):    #設計測試驅動程式類別
05     def test_computeBMIOK(self):    #設計測試主程式功能的方法，self 是物件本身
06         #assertEqual(執行結果, 期望結果)
07         self.assertEqual(mybmi.computeBMI(52, 1.55), 21.64)
08
09
10
11
12
13
```

# 撰寫涵蓋度報表程式

## □ 使用unittest (PyUnit) 測試框架

### ○ 將以下程式exec\_api.py放入 c:\TEST

```
01 # 使用 API 產生程式碼覆蓋率統計報告 exec_api.py
02 import coverage          # 匯入涵蓋度紀錄套件
03 import unittest          # 匯入測試框架套件
04 # 實體化一個涵蓋度紀錄物件
05 cov = coverage.cov(coverage(branch=True, source=['mybmi']))
06 cov.start() # 啟動測試涵蓋度紀錄
07 suite = unittest.defaultTestLoader.discover("./", "mybmitest.py") # 載入測試套件
08 unittest.TextTestRunner().run(suite) # 執行測試套件組之測試
09 cov.stop()                 # 停止測試涵蓋度紀錄
10 cov.save()                 # 儲存測試涵蓋度資料
11 cov.report()               # 命令列模式展示結果
12 cov.html_report(directory='cov') # 製作測試涵蓋度結果報表，存放在cov子目錄
```

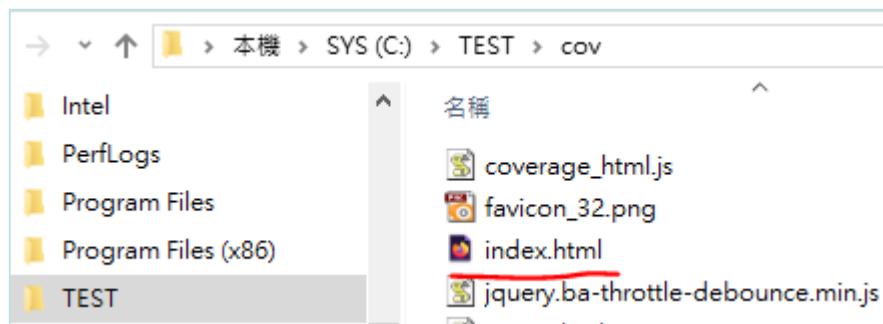
# 執行測試

## □ 測試結果

```
In [1]: runfile('C:/TEST/exec_api.py', wdir='C:/TEST')
          .Name     Stmts  Miss Branch BrPart  Cover
          -----
mybmi.py       7      2      4      2    64%
          -----
TOTAL         7      2      4      2    64%
          -----
Ran 1 test in 0.005s
OK
```

○ 針對一個測試案例，程式測試正確，但涵蓋度64%

## □ 開啟檔案總管，執行(雙擊) index.html



# 執行測試

## ○ 測試涵蓋度(Coverage)/**完整度 64%**

Coverage report: 64%

Module ↑	statements	missing	excluded	branches	partial	coverage
mybmi.py	7	2	0	4	2	64%
<b>Total</b>	<b>7</b>	<b>2</b>	<b>0</b>	<b>4</b>	<b>2</b>	<b>64%</b>

coverage.py v6.0.2, created at 2021-10-25 14:49 +0800

## ○ 點選mybmi.py，查看細節

- 紅色 return沒有執行到
- if 只有執行部分條件判斷(沒有執行到False)

Coverage for **mybmi.py** : 64%

7 statements    5 run    2 missing    0 excluded    2 partial

```
1 | def computeBMI(kg, M):                                #錯誤，範圍判斷錯誤
2 |     if kg>200 and M>2.5:                            #錯誤，範圍判斷錯誤
3 |         return -1
4 |     if kg<=2 or M<=0.05:                            #錯誤，範圍判斷錯誤
5 |         return -1
6 |     BMI = round(kg/(M*M),2)                         #錯誤，四捨五入取兩位小數
7 |     #BMI = ((100*kg/(M*M))//1)/100   #正確，去尾，乘100取整數，再除100取兩位小數
8 |     return BMI
```

# Exercise測試問題

□ 1. 增修驅動程式，使主程式測試更完整(coverage 100%)

○ 測試案例 vs. 主程式

□ 2. 增加測試案例設計，使需求/問題測試更完整

○ 測試案例vs. 需求/題目

# Exercise測試問題(coverage 100%)

## □ 測試案例設計

### ○ 錯誤身高

➤ 輸入：52公斤，0.05公尺

➤ 輸出：-1

### ○ 錯誤體重

➤ 輸入：2公斤，1.55公尺

➤ 輸出：-1

## □ 1.增修驅動程式，使主程式測試更完整(coverage 100%)

### ○ 【mybmitest.py】驅動程式07. 08.加入

➤ self.assertEqual(mybmi.computeBMI(52, 0.05), -1)

➤ self.assertEqual(mybmi.computeBMI(2, 1.55), -1)

### ○ 測試案例 vs. 主程式

# Exercise測試問題(coverage 100%)

## □ 執行測試，未達100%

.Name	Stmts	Miss	Branch	BrPart	Cover
mybmi.py	7	1	4	1	82%
TOTAL	7	1	4	1	82%

Ran 1 test in 0.002s

OK

Coverage for **mybmi.py** : 82%

7 statements    6 run    1 missing    0 excluded    1 partial

```
1 def computeBMI(kg, M):  
2     if kg>200 and M>2.5:                      #錯誤，範圍判斷錯誤  
3         return -1  
4     if kg<=2 or M<=0.05:                        #錯誤，範圍判斷錯誤  
5         return -1  
6     BMI = round(kg/(M*M),2)                      #錯誤，四捨五入取兩位小數  
7     #BMI = ((100*kg/(M*M))//1)/100    #正確，去尾，乘100取整數，再除100取兩位小數  
8     return BMI
```

# Exercise測試問題(coverage 100%)

## □ 測試案例設計

### ○ 錯誤身高、體重

➤ 輸入：500公斤，5公尺

➤ 輸出：-1

## □ 1.增修驅動程式，使主程式測試更完整(coverage 100%)

### ○ 【mybmitest.py】驅動程式09.加入

➤ self.assertEqual(mybmi.computeBMI(500, 5), -1)

Name	Stmts	Miss	Branch	BrPart	Cover
mybmi.py	7	0	4	0	100%
TOTAL	7	0	4	0	100%
<hr/>					
Ran 1 test in 0.001s					
OK					

Coverage for **mybmi.py** : 100%

7 statements 7 run 0 missing 0 excluded 0 partial

```
1 def computeBMI(kg, M):  
2     if kg>200 and M>2.5: #錯誤，範圍判斷錯誤  
3         return -1  
4     if kg<=2 or M<=0.05: #錯誤，範圍判斷錯誤  
5         return -1  
6     BMI = round(kg/(M*M),2) #錯誤，四捨五入取兩位小數  
7     #BMI = ((100*kg/(M*M))//1)/100 #正確，去尾，乘100取整數，再除100取兩位小數  
8     return BMI
```

# Exercise問題

## □ 2. 增加測試案例設計，使需求/問題測試更完整

- 測試案例vs. 需求/題目

## □ 測試案例設計

- 錯誤身高

- 輸入：500公斤，1.55公尺
  - 輸出：-1

- 錯誤體重

- 輸入：200公斤，5公尺
  - 輸出：-1

## □ 2. 增加測試案例設計，使需求/問題測試更完整

- 【mybmitest.py】驅動程式11.12.加入

- `self.assertEqual(mybmi.computeBMI(500, 1.55), -1)`
  - `self.assertEqual(mybmi.computeBMI(200, 5), -1)`

# Exercise問題

- 執行測試，有錯誤，必須修改程式

FName	Stmts	Miss	Branch	BrPart	Cover
-------	-------	------	--------	--------	-------

mybmi.py	7	0	4	0	100%
----------	---	---	---	---	------

TOTAL	7	0	4	0	100%
-------	---	---	---	---	------

---

**FAIL: test\_computeBMIOK (mybmitest.TestBMI)**

---

**Traceback (most recent call last):**

  File "C:\TEST\mybmitest.py", line 11, in test\_computeBMIOK

    self.assertEqual(mybmi.computeBMI(500, 1.55), -1)

**AssertionError: 208.12 != -1**

---

**Ran 1 test in 0.004s**

**FAILED (failures=1)**

# Exercise問題

- 請修改程式達到測試正確、涵蓋度100%
- 上傳ZUVIO，week 07

# Exercise 題目增加需求/條件

□ 輸入：體重(公斤)，身高(公尺)

- 體重( $3 \leq \text{体重} \leq 200$ )，身高( $0.50 \leq \text{身高} \leq 2.5$ )。
- $\text{BMI} = \frac{\text{體重(公斤)}}{\text{身高}^2(\text{公尺平方})}$

□ 輸出

- 身高體重輸入範圍錯誤，輸出"ERROR INPUT"。

Level	BMI	OUTPUT
體重過輕	$\text{BMI} < 18.50$	Underweight
正常範圍	$18.50 \leq \text{BMI} < 24$	Normal
體重過重	$24.00 \leq \text{BMI} < 27$	Overweight
肥胖	$27.00 \leq \text{BMI}$	Obesity

# Exercise 題目增加需求/條件

- 設計一個function，呼叫computeBMI，處理輸出
  - def printResult(weight, height):
- 針對測試驅動程式，設計測試方法testPrintResult(self):
- 設計足夠的測試案例
  - 使主程式測試coverage 100%
  - 修改使主程式正確

# Exercise設計測試案例

## □ 測試案例分類

- 錯誤身高

- 輸入：52公斤，5公分，輸出：ERROR INPUT

- 錯誤體重

- 輸入：2公斤，155公分，輸出：ERROR INPUT

- 輸出：**Underweight**

- 輸出：**Normal**

- 輸入：52公斤，155公分，

- BMI :  $52(\text{公斤}) / (1.55 * 1.55)(\text{公尺平方}) = 21.64412$ (兩位小數 21.64)。

- 輸出：**Overweight**

- 輸出：**Obesity**

# 附錄

# Exercise

- 測試計算平均
- 測試計算年齡
- 測試計算BMI值的function
  - BMI值計算公式:  $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺平方})$
  - 例如：一個52公斤的人，身高是155公分，則BMI為：
  - $52(\text{公斤}) / (1.55 * 1.55)(\text{公尺平方}) = 21.64412$
  - 正常範圍為  $BMI = 18.5 \sim 24$
  - 請設計一個 function，傳入身高、體重，回傳BMI。
  - 請設計一個 function，從鍵盤輸入姓名name、身高、體重。
  - 當BMI太大，輸出 Hi name, Your BMI: xx too HIGHT。
  - 當BMI太小，輸出 Hi name, Your BMI: xxx too LOW。

# Exercise

```
def BMIOutput(name, BMI):
    #print('Hi ', name, end=', Your BMI: %d' %BMI)
    str01 = 'Hi {name}, Your BMI: {BMI}'.format(name=name, BMI=BMI)
    if BMI<=0:
        out = str01 + ' ERROR'
    elif BMI>24:
        out = str01 + ' too HIGH'
    elif BMI<18.5:
        out = str01 + ' too LOW'
    else:
        out = str01 + ' OK'
    return out
```

```
def testBMI():
    name, kg, M = inputData()
    BMI = computeBMI(kg, M)
    print(BMIOutput(name, BMI))
```

```
def computeBMI(kg, M):
    if kg<=0 or M<=0:
        return 0
    #BMI = round(kg/(M*M),2)
    BMI = ((100*kg/(M*M))/1)/100
    return BMI
```

```
def inputData():
    name = input('name: ')
    kg = float(input('KG: '))
    M = float(input('M: '))
    return name, kg, M
```

# Exercise

```
import unittest

class TestBMI(unittest.TestCase):
    def test_computeBMIOK(self):
        self.assertEqual(computeBMI(52, 1.55), 21.64)
        self.assertEqual(computeBMI(50, 1.5), 22.22)
        self.assertAlmostEqual(computeBMI(52, 1.55), 21.63, 1)

    def test_computeBMIOK_Almost(self):
        self.assertAlmostEqual(computeBMI(50, 1.5), 22.21, 1)

    def test_computeBMILow(self):
        self.assertEqual(computeBMI(60, 2), 15)

    def test_computeBMIGHigh(self):
        self.assertEqual(computeBMI(50, 1), 50)

    def test_computeBMINe(self):
        self.assertEqual(computeBMI(50, -1), 0)
        self.assertEqual(computeBMI(-50, 1), 0)

def test_BMIOutput(self):
    out ='Hi John, Your BMI: 50.0 too HIGH'
    self.assertEqual(BMIOutput('John', 50.0), out)

def test_BMIOutputNe(self):
    out ='Hi John, Your BMI: -50.0 ERROR'
    self.assertEqual(BMIOutput('John', -50.0), out)

if __name__ == '__main__':
    unittest.main()
```

# Exercise

- 紿予一組學生名單，包括名字、學號以及其三科成績，計算每位學生的平均分數，並將最高分與最低分的學生姓名分數印出。
- 紿予一組郵遞區號(北北基)的資料，試著寫出兩個函式：
  - area\_to\_zip(area): 傳入值為區域名稱回傳此區域的郵遞區號，  
ex. 呼叫 area\_to\_zip("信義區")，回傳 201
  - zip\_to\_area(zip): 傳入值為郵遞區號回傳區域名稱，呼叫  
area\_to\_zip(106)，回傳 "大安區"。

# 單元測試 (Unit Test)

## □ unittest (PyUnit) 測試框架

- 測試套件(Test suite): 一組 Test Case, TestSuit 或兩者組合。
- Test runner: 負責執行測試並提供測試結果。
- 測試執行器在每個測試執行前執行 setUp 方法，每個測試執行後執行 tearDown 方法。
- 將某些測試方法組成 Test Suit
- 使用 TextTestRunner 執行一組 Test suit

## □ function 傳入多個不定參數 \*

- self.args = (52, 1.55)
- computeBMI(\*self.args)

## □ assertAlmostEqual(執行結果, 期望結果, 比對小數位數)

- self.assertAlmostEqual(computeBMI(50, 1.5), 22.21, 1)

# 單元測試 (Unit Test)

## □ unittest (PyUnit) 測試框架

```
import unittest
class TestBMI(unittest.TestCase):
    def setUp(self):
        self.args = (52, 1.55)
    def tearDown(self):
        self.args = None
    def test_computeBMIOK(self):
        self.assertEqual(computeBMI(*self.args), 21.64)
        #self.assertEqual(computeBMI(50, 1.5), 22.22)
        self.assertAlmostEqual(computeBMI(*self.args), 21.63, 1)
    def suite():
        suite = unittest.TestSuite()
        suite.addTest(TestBMI('test_computeBMIOK'))
        return suite

if __name__ == '__main__':
    runner = unittest.TextTestRunner()
    runner.run(suite())
```

-----  
Ran 1 test in 0.002s

OK

# 單元測試 (Unit Test)

```
import unittest
class TestStringMethods(unittest.TestCase):
    def test_upper(self):
        self.assertTrue('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        #self.assertTrue('Foo'.isupper())
        self.assertTrue('foo'.islower())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        with self.assertRaises(TypeError):
            s.split(2)

if __name__ == '__main__':
    unittest.main()
```

# 單元測試 (Unit Test)

## □ 程式碼涵蓋度 (Code Coverage) – 指令行執行

- 執行指令涵蓋與分支涵蓋

- `coverage run -branch 10802.py`

- 每一條指令是否執行過? 每一個分支判斷 True/False 是否執行過?

- 產生html報表，目錄名稱為 cov

- `coverage html -d cov`

# Python 字串

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 字串存取、長度與複製

- 字串 strings 是 bytes 表示為 unicode 字元陣列.
- 單一字元是長度為一的字串。
- 字串可用雙引號"或用單引號'進行標示
- 使用索引位置[]可以存取元素
- len()取得長度
- 使用 \* 複製字串
- + 串接

```
a = 'Hello \'ho '
b = "World "
c = "Bob said 'hey there.' "
print(a)
print(b+c)
print(a[1], a[4])
print(len(a))
print(b*3)
```

```
def strOp():
    print("Hello" + " World")
    name = "小明"*2
    print(name)
    Hi = "Hi~ 您好!"
    print(Hi[1])
    print(Hi[4])
```

Hello World  
小明小明  
i  
您

Hello 'ho
World Bob said 'hey there.'
e o
10
WorldWorldWorld

# 多行字串

## □ 指定多行字串

- 三個單/雙引號，或

```
a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)
```

```
a = "Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."  
print(a)
```

```
 Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.
```

**With line break**

\ +

```
a = "Lorem ipsum dolor sit amet, \  
consectetur adipiscing elit, \  
sed do eiusmod tempor incididunt \  
ut labore et dolore magna aliqua."  
print(a)
```

```
a = ("Lorem ipsum dolor sit amet," +  
"consectetur adipiscing elit," +  
"sed do eiusmod tempor incididunt" +  
"ut labore et dolore magna aliqua.")  
print(a)
```

```
 Lorem ipsum dolor sit amet, consectetur  
adipiscing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna  
aliqua.
```

**No line break**

# 字串相等

❑ == operator or \_\_eq\_\_()

s1="apple"

```
if (s1=="apple"): print("s1 is apple")
else: print("s1 is NOT apple")
```

s2=" apple"

```
if (s2=="apple"): print("s2 is apple")
else: print("s2 is NOT apple")
```

s3="Apple"

```
if (s3=="apple"): print("s3 is apple")
else: print("s3 is NOT apple")
```

s4="apple"

```
if (s1.__eq__(s4)): print("s1 equals to s4")
else: print("s1 does NOT equal to s4")
```

s1 is apple

s2 is NOT apple

s3 is NOT apple

s1 equals to s4

# 字串資料型別特性

## □ 字串是不可變的

- 無法直接在字串中插入字元。

- >>> name = 'Henny'

- >>> name[0] = 'P' 

## □ 若要改變字串，須用字串函式，如：replace() or slice

- .replace()替換

```
name = 'Hernny'  
data = name.replace('n', 'o')  
print(name)  
print(data)  
full = name[:3]+ 'ing'  
print(full)
```

```
Hernny  
Herooy  
Hering
```

# 字串替換與切割

## □ str.partition(sep)

- 字串切割，只切割第一個符合sep引數的字元，形成3-tuple

```
names='Tom,John,Mary,Bob,Sunny'  
print(names.partition(','))  
print(names.partition(',')[-1])  
print(names.partition(',')[-1])  
print(names.partition(',')[-1])  
print(names.rpartition(','))
```

('Tom', ',',  
'John,Mary,Bob,Sunny')  
John,Mary,Bob,Sunny  
Tom  
,  
('Tom,John,Mary,Bob', ',',  
'Sunny')

## □ str.replace(old, new[, count])

- 替換string中特定字串，old被替換的字串，new要替代的字串，count指定代替的數目

```
str = "It is istring example"  
print (str.replace("is", "was"))  
print (str.replace("is", "was", 1))
```

It was wastring example  
It was istring example

# 字串替換與切割

## □ string.split(sep, maxsplit)

- 由左至右，將string字串的字元以sep字元為分隔進行分割
- 若找不到符合sep的值，回傳整個字串，否則回傳str list

```
num1, num2 = input('enter 2 numbers separated by a space: ').split()  
num1 = int(num1)  
num2 = int(num2) # print(num1, num2)
```

```
def strOps():  
    words = "小明今天去學校，小明遲到。"  
    wordLength = len(words)  
    print(wordLength)  
    sentence = words.replace("小明", "湯姆")  
    print(words)  
    print(sentence)  
    wordsSplit = words.split("明")  
    print(wordsSplit)
```

```
names='Tom,John,Mary,Bob,Sun  
ny'  
print(names.split(','))  
['Tom', 'John', 'Mary', 'Bob', 'Sunny']
```

13

小明今天去學校，小明遲到。  
湯姆今天去學校，湯姆遲到。  
['小', '今天去學校，小', '遲到。']

# 字串替換與切割

## □ str.splitlines(keepends)

- 將字串進行切割
- 以"\n"和"\r"作為分割的區隔字元
- 回傳序列資料型別
- keepends引數預設False，設為True會連同'\n'一併回傳

```
s = "Thank you for the music\nWelcome to the  
jungle\nMa"  
print(s.splitlines())  
print(s.splitlines(True))
```

```
[('Thank you for the music', 'Welcome to the  
jungle', 'Ma')  
[('Thank you for the music\n', 'Welcome to the  
jungle\n', 'Ma')]
```

# Exercise

## □ 計算字串有多少字

```
s='Given a string and count how many words in the string'  
words=s.split()  
print(len(  ))
```

## □ 計算字串有多少非重複字，可使用集合

```
s='Given a string and count how many words in the string'  
words=s.split()  
nonDuplicate=  )  
print(len(nonDuplicate))
```

# 字串檢查

## □ in or not in

```
txt = "The rain in Spain stays mainly in the plain"  
x = "ain" in txt  
print(x)
```

True

```
txt = "The rain in Spain stays mainly in the plain"  
x = "ain" not in txt  
print(x)
```

False

# Exercise

- 檢查 'e' 是否在 'Umbrella'
- 檢查輸入是否為 'python'

```
s = 'Umbrella'  
if 'e' █ s:  
    print("True")  
else:  
    print("False")
```

```
s = input('enter a string: ')  
if s █ 'python':  
    print("True")  
else:  
    print("False")
```

# 字串判斷

- str.isalpha(): 判斷是否為a~z, A~Z
- str.isnumeric(): 判斷是否為數字
- str.isdigit(): 判斷是否數字
- str.decimal(): 判斷是否數字
- str.islower(): 判斷是否全小寫
- str.isspace(): 判斷是否全空白

```
s='505'  
s1='2com'  
s2= "\u00B2" #unicode 2  
print(s.isdigit())  
print(s2.isnumeric())  
print(s.isdecimal())  
print(s1.isalnum())  
print(s1.isalpha())
```

True  
True  
True  
True  
False

```
s = "50800"  
print(s.isdigit())  
s = "abcd"  
print(s.islower())  
s = " "  
print(s.isspace())
```

True  
True  
True

# 字串判斷

## ❑ `isdigit()`

- True: Unicode 數字，byte 數字(單字節)，全形數字(雙字節)
- False: 漢字數字
- Error: 無

## ❑ `isdecimal()`

- True: Unicode 數字，，全形數字(雙字節)
- False: 漢字數字
- Error: byte 數字(單字節)

## ❑ `isnumeric()`

- True: Unicode 數字，全形數字(雙字節)，漢字數字
- False: 無
- Error: byte 數字(單字節)

# 字串判斷

```
num = "1" #unicode
print(num.isdigit()) # True
print(num.isdecimal()) # True
print(num.isnumeric()) # True
num = "1" # 全形
print(num.isdigit()) # True
print(num.isdecimal()) # True
print(num.isnumeric()) # True
num = "四" # 漢字
print(num.isdigit()) # False
print(num.isdecimal()) # False
print(num.isnumeric()) # True
num = b"1" # byte
print(num.isdigit()) # True
print(num.isdecimal()) # AttributeError 'bytes' object has no attribute 'isdecimal'
print(num.isnumeric()) # AttributeError 'bytes' object has no attribute 'isnumeric'
```

# 字串轉換

## □ string.lower()

- 將string內的字元從大寫字母轉換小寫

## □ string.upper()

- 將string內的字母從小寫轉為大寫

## □ string.title()

- 將字串內所有[a-z]單字第一個字元轉換成大寫

## □ string.swapcase()

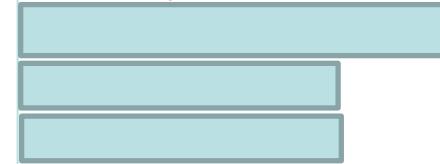
- 將string字串裡的字母大小寫互轉

```
s = "Hello my FRIENDS"  
print(s.swapcase())
```

hELLO MY friends

```
s = "Hello my FRIENDS"  
print(s.lower())  
print(s.upper())  
print(s.capitalize())  
print(s.title())
```

hello my friends



# Exercise

- 使用者輸入任意整數 n，當輸入的n不為整數，提示使用者輸入資料型別錯誤，且不斷重新讓使用者輸入，若輸入值為整數，將其print至螢幕上，例如 n=100

```
while(True):
    n=input('請輸入一個整數：')
    if n.isint()():
        print(n)
    else:
        print('型態錯誤，請輸入整數')
```

# Exercise

## □ 計算字串大小寫數字與特殊字元

- input\_str = "P@#yn26at^&i5ve"
- Expected Outcome: Chars = 8 Digits = 3 Symbol = 4

```
def findDigitsCharsSymbols(inputString):  
    charCount = 0  
    digitCount = 0  
    symbolCount = 0  
    for char in inputString:  
        if char.islower() or char.isupper():  
            charCount+=1  
        elif char.isdigit():  
            digitCount+=1  
        else:  
            symbolCount+=1  
    print("Chars =", charCount, "Digits =", digitCount, "Symbol =",  
          symbolCount)
```

```
findDigitsCharsSymbols("P@#yn26at^&i5ve")
```

# 字串切片

- 撷取字串 [start:end:step] , start~(end-1), each step
  - -1 表示倒數過來第一個

```
def strGet():
    words = "This is a book, that is a cat~"
    print(words)
    print(words[:])
    print(words[5:])
    print(words[-4:])
    print(words[-14:20:1])
    print(words[-6:5:-1])
    print(words[-4:-10:-2])
```

This is a book, that is a cat~  
This is a book, that is a cat~  
is a book, that is a cat~  
cat~  
that  
a si taht ,koob a s  
cas

# 字串切片

## □ 切片 Slicing

```
b = "Hello, World!"  
print(b[2:5])
```

llo

```
b = "Hello, World!"  
print(b[-5:-2])
```

orl

## □ string.split()將字串依指定的字元(字串)切割，，回傳str list

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

```
a= ("how", "are", "you")  
x = "-".join(a)  
print(x)
```

```
myTuple = ("John", "Peter", "Vicky")  
x = "#".join(myTuple)  
print(x)
```

John#Peter#Vicky

# Exercise

□ 輸入兩個英文句子 A, B，兩個英文字 x, y

- 將兩個英文句子A, B串聯成 C
- 將C其中的 x 替換成 y，變成 D
- 輸出C, D 長度的加總
- 輸出C前三個字，每一個字重複輸出三次。
- C從第五個字到最後第三個字，每隔兩個字輸出
- 將C倒著輸出

```
def f(A, B, x, y):
```

```
    C = A + B
```

Input

This is a book

That is a cat

is

was

# Exercise

## □ output?

```
def f(data, s):
    length = len(data)
    for i in range(length):
        newData=data[0:i]+ s + data[i+1:length]
        print(newData)

f('abcd','X')
```

# Exercise

- 輸入 s1, s2, 在 s1 中間加入 s2 後輸出

```
def appendMiddle(s1, s2):  
    middleIndex = int(len(s1) / 2)  
    print("Original Strings are", s1, s2)  
    middleThree = s1[: ]+ s2 +s1[middleIndex:]  
    print("After appending new string in middle", middleThree)  
  
appendMiddle("Chrisdem", "IamNewString")
```

# Python List

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 內建資料型別

## □ 序列資料型別 (sequence types) 共有六種

- str: 字串 (string) , 不可變 (immutable)
- bytes: 字節 (byte) , 不可變 (immutable)
- bytearray: 字節陣列 (byte array) , 可變 (mutable)
- list: 串列 , 元素有順序、可修改、可重複
- tuple: 序對 , 元素有順序、不可修改、可重複
- range: 內建函數 range() 回傳的物件 (object) , 常用於for 迴圈  
(for loop)

## □ Collection 資料型別

- list, tuple
- set: 元素無順序、可修改、無重複、無索引 indexed
- dictionary (dict): 元素無順序、可修改、無重複、有索引 indexed

# 序對(Tuple)

## □ Tuple的操作

- 元素有順序關係，可以有不同資料型別的元素
- 元素允許是tuple
- 使用小括號()，索引位置可以存取元素
- 唯讀不可變更的資料結構，不可取代tuple中任一元素

```
f = (2,'a',4,5)
g = ()
h = (2, [3,4], (10,11,12))
x = f[1]
y = f[1:3]
z = h[2][1]
print(f)
print(g)
print(x)
print(y)
print(z)
```

```
(2, 'a', 4, 5)
()
a
('a', 4)
11
```

# Tuple建構

## □ 造出

```
def test01():
    thistuple = ("apple",)
    print(type(thistuple))
#NOT a tuple
    thistuple = ("apple")
    print(type(thistuple))

test01()
```

you have add  
a comma after  
the item,

<class 'tuple'>  
<class 'str'>

## □ Join Tuples (use + operator)

```
def test02():
    tuple1 = ("a", "b", "c")
    tuple2 = (1, 2, 3)
    tuple3 = tuple1 + tuple2
    print(tuple3)
```

test02()

('a', 'b', 'c', 1, 2, 3)

## □ tuple() Constructor

```
def test02():
    thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
    print(thistuple)
```

test02()

('apple', 'banana', 'cherry')

# Tuple建構

- Tuples 是 unchangeable，不能移除元素，能刪除整個 tuple

```
thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange"      # This will raise an error
```

```
def test03():
    fruit = ('apple','banana', 'cherry')
    del fruit
    print(fruit)      #沒有這個變數
```

```
test03()
```

- 判斷是否存在

```
def test02():
    thistuple = ("apple", "banana", "cherry")
    if "apple" in thistuple:
        print("Yes, 'apple' is in the fruits tuple")
```

```
test02()
```

Yes, 'apple' is in the fruits tuple

# Tuple 基本存取

## □ 存取一個元素

```
def test01():
    thistuple = ("apple", "banana", "cherry")
    print(thistuple[1])
```

test01()      banana

## □ Range of indexes (-1 最後一個, -2 最後第二個)

```
def test02():
    thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
    print(thistuple[2:5])
```

test02()      ('cherry', 'orange', 'kiwi')

# Tuple基本方法

## ○ len() method

```
def test02():
    thistuple = ("apple", "banana", "cherry")
    print(len(thistuple))
```

3

## ○ count() – 回傳特定值的個數

```
def test02():
    thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
    x = thistuple.count(5)
    print(x)
```

2

## ○ index() – 搜尋定值回傳索引

```
def test02():
    thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
    x = thistuple.index(8)
    print(x)
```

3

# List 基本運算

□ List定義使用"中括號"，內不限定放數字文字

- `list1 = [1,2,3,4,5,'e','w','f']`

□ List有順序

- `print(my_list[0])` # 0為位置，第一項

- `print(my_list[-1])` # 倒數第1項

□ `len()`計算list長度

□ `sum()`計算list中所有數值加總(但list中元素都須數值)

□ `count` 計算list中某個元素出現次數

□ `sort()`：將串列進行排序

- `reverse = True` 意思為 反轉排列

- 不同型別排序會出現錯誤

# 索引值範圍

## □ 回傳 2 to 5 (excluded)

```
def test01():
    thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
    print(thislist[2:5])
```

```
test01()
```

```
['cherry', 'orange', 'kiwi']
```

## □ 回傳 -4 (included) 到 -1 (excluded)

```
def test02():
    thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
    print(thislist[-4:-1])
```

```
test02()
```

```
['orange', 'kiwi', 'melon']
```

# list 基本運算

## □ List操作

```
def testList01():
    list1 = ['cat', 'tiger', 'dog']
    list2 = [2, 3, 2]
    print(list1[0])
    print(list2[-1])
    print(len(list1))
    print(sum(list2))
    print(list1.count('cat'))
    print(list2.count(2))
    list1[0] = 'bigCat'      #改變某一個元素值
    print(list1)
```

```
cat
2
3
7
1
2
['bigCat', 'tiger', 'dog']
```

```
list1 =[59, 33, 77, 62, 101, 243, 189, 5]
list1.sort()
list1.sort(reverse=True)
print(list1)
```

# list 操作 - 新增

- + : 兩個List加在一起
- insert(object) : 在指定位置插入新物件元素
- append(object) : 在最後加入物件元素
- extend(object) : 逐一取出object內元素，加在最後
  
- nums = (10,5,7,1,6,2)
  - tuple不提供排序的方法，若要排序(由小到大)該怎辦？
  - 轉成 list

```
f = (5, 7, 11, 23, 45, 2)
g = list(f)
print(g)
g.sort()
print(g)
```

```
[5, 7, 11, 23, 45, 2]
[2, 5, 7, 11, 23, 45]
```

# list 操作 - 新增

## □ 新增

```
def testList02():
    list1 = ['cat']
    list2 = [2, 3, 2]
    list1.append('dog')
    print(list1)
    list1.insert(1, 'ant')
    print(list1)
    list1.extend('ti')
    print(list1)
    list3 = list1 + list2
    list4 = [list1, list2]
    print(list3)
    print(list4)
    list2.append(list1) #list 的 list
    print(list2)
```

```
['cat', 'dog']
['cat', 'ant', 'dog']
['cat', 'ant', 'dog', 't', 'i']
['cat', 'ant', 'dog', 't', 'i', 2, 3, 2]
[['cat', 'ant', 'dog', 't', 'i'], [2, 3, 2]]
[2, 3, 2, ['cat', 'ant', 'dog', 't', 'i']]
```

# list 操作 - 新增

```
def myList():
    list = ['a', 'b', 'c']
    for x in list:
        print(x)

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:
    print (i, end=' ')
def main():
    myList()
```

a  
b  
c  
2 state  
range(i,j) is i~j-1  
1 2 3 4 5 6 7 8 9 10

# list 操作 - 刪除

- `list.pop()` # 刪除末尾
- `list.pop(X)` # 刪除指定位置的內容
- `remove(item)` : 刪除指定項

```
def testList():
    list1 = ['ant', 'cat', 'dog', 'tiger', [1, 2, 'mouse']]
    print(list1[2])
    print(list1[4])
    print(list1[4][2])
    list1.remove('dog')
    print(list1)
    list1.pop(0)
    print(list1)
    list1.pop(1)
    print(list1)
```

```
testList()
```

```
dog
[1, 2, 'mouse']
mouse
['ant', 'cat', 'tiger', [1, 2, 'mouse']]
['cat', 'tiger', [1, 2, 'mouse']]
['cat', [1, 2, 'mouse']]
```

# list 操作 – 切片

## □ 切割(slice) List

- my\_list[1:3] # print 1~2 項
- my\_list[:-1] # print 0~倒數第2項

```
list = ['a', 'b', 'c']
print(list[0:2])
print(list[::-2])
print(list[::-2])
print(list[::-1])
```

```
['a', 'b']
['a', 'c']
['c', 'a']
['c', 'b', 'a']
```

## □ 將串列複製到一個獨立串列：copy()函式

## □ 轉換函式：list()

```
a = [1, 2, 3, 4]
b = a.copy()
c = list(a)
d = a[:]
```

```
['1', '2', '3', '4']
['1', '2', '3', '4']
['1', '2', '3', '4']
```

# list 操作 – 切片

<pre>def myList():     my_list = []     my_list.append(1)     my_list.append(2)     my_list2 = [5.5,22,'Hi',3,99,22,66]     my_list2[0]=3.3     print(len(my_list),sum(my_list),my_list2.count(222))     print(my_list2[0])     print(my_list2[2])     print(my_list2[-1])     print(my_list2[:-1])     print(my_list2[1:4])     print(my_list2[2:3])     print(my_list2[2:])     list2 = my_list + my_list2     list3 = my_list*2     print(list2, list3)     print(list2==list2[:])</pre>	<pre>2 3 0 3.3 Hi 66 [3.3, 22, 'Hi', 3, 99, 22] [22, 'Hi', 3] ['Hi'] ['Hi', 3, 99, 22, 66] [1, 2, 3.3, 22, 'Hi', 3, 99, 22, 66] [1, 2, 1, 2] True</pre>
---	---

myList()

# list 基本操作 – 搜尋

## □ 搜尋

- 判斷值是否存在該串列

- "bb" in data

- True

- "aa" in data

- True

- 尋找某個項目的index

```
student = ['tom', 'job', 'jay', 'jeff']
print(student.index('jeff'))
```

```
def listTest():
    data = ['aa', 'bb', 'cc', 'dd']
    t = 'bb' in data
    print(t)
    print('aa' in data)
    if 'cc' in data:
        print('cc in')
```

```
listTest()
```

True  
True  
cc in

# list產生器

- 結合迴圈與條件測試式，減少繁瑣語法。

```
def listTest():
    inputData = input("Enter a list of number: ").split(" ")
    number_list = []
    for number in inputData:
        number_list.append(int(number))

    print(number_list)
```



[運算式 for 項目 in 可疊代項目]

```
inputData = input("Enter a list of number: ").split(" ")
number_list = [int(number) for number in inputData]
print(number_list)
number_list = [int(number)*2 for number in inputData]
print(number_list)
```

# list產生器

□ 運算式 for 項目 in 可疊代項目 if 條件式。

```
number_list = [number for number in range(1,6) if number%2==1]  
print(number_list)
```

```
def listTest():  
    inputData = input("Enter a list of number: ").split(" ")  
    number_list = [int(number) for number in inputData if int(number)%2==1]  
    print(number_list)
```

```
listTest()
```

```
for row in range(1,4):  
    for col in range(1,3):  
        print([row, col])
```



```
data = [[row, col] for row in range(1,4) for col  
       in range(1,3)]  
print(data)
```

# Exercise

- $a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]$ . 輸出所有偶數元數的 list

```
a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
b = [number for  ]
print(b)
```

- 寫 `isPrime(x)`，用 list generator 產生某串列  $a$  中是質數者

# Exercise

- 使用random.randint()產生1~30整數10個放進list，將最前面和最後面放進新list印出。

```
def list_ends(a_list):
    return [a_list[0], a_list[-1]]

# generate a list using random integers
import random
a=[]
for i in range(10):
    a.append(random.randint(1, 30))
print(a)

print(list_ends(a))
```

# Python 排序

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

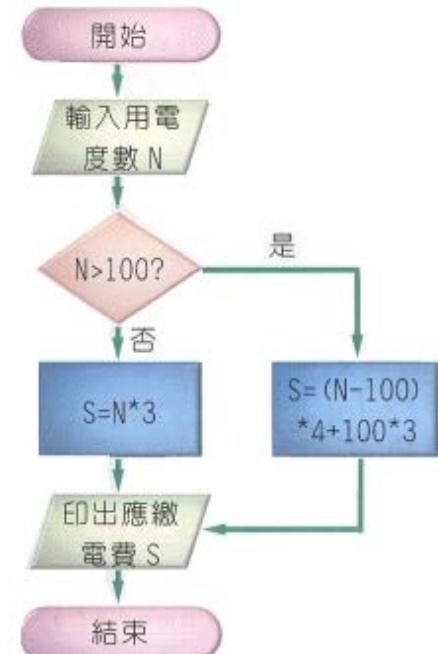
# 演算法 (Algorithm)

## □ 處理步驟

- 1) 若用電在100度以內，每度3元。
- 2) 100度以上，超過的部份每度4元。

## □ 時間複雜度

print(a[0])	O(1)
for i in range(len(a)): print(a[i])	O(n)
for i in range(len(a)): for j in range(len(a)): print(a[j])	O(n <sup>2</sup> )



# 選擇排序法 Selection Sort

□ 時間複雜度:  $O(n^2)$

○ Best Case :  $O(n^2)$

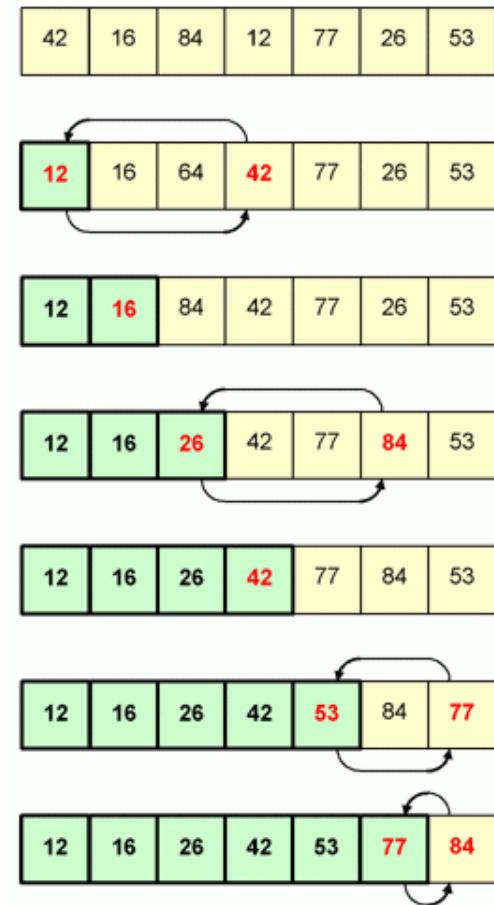
○ Worst Case :  $O(n^2)$

○ 無論資料順序如何，都會執行兩個迴圈

```
def maxIndex(data, i):
    m = i
    for j in range(i, len(data)):
        if data[j] < data[m]:
            m = j
    return m

def s_sort(data):
    for i in range(len(data)):
        m = maxIndex(data, i)
        if m != i:
            data[i], data[m] = data[m], data[i]

data = [42, 16, 84, 12, 77, 26, 53]
print(s_sort(data))
```



# 選擇排序法

## □ 演算步驟

- 每次在剩下的資料中找出最小的資料，將該資料丟到當前的正確位置。
  - 從  $i = 1$  到  $n-1$ ， $n-1$  回合
  - 每回合自第  $i$  筆到第  $n$  筆中排出最小值，與第  $i$  筆資料交換

```
def s_sort(data):
    for i in range(len(data)):
        m = i
        for j in range(i,len(data)):
            if data[j]< data[m]:
                m = j
        if m!=i:
            data[i], data[m] = data[m], data[i]
```

```
data = [42, 16, 84, 12, 77, 26, 53]
print(s_sort(data))
```

# Bubble Sort

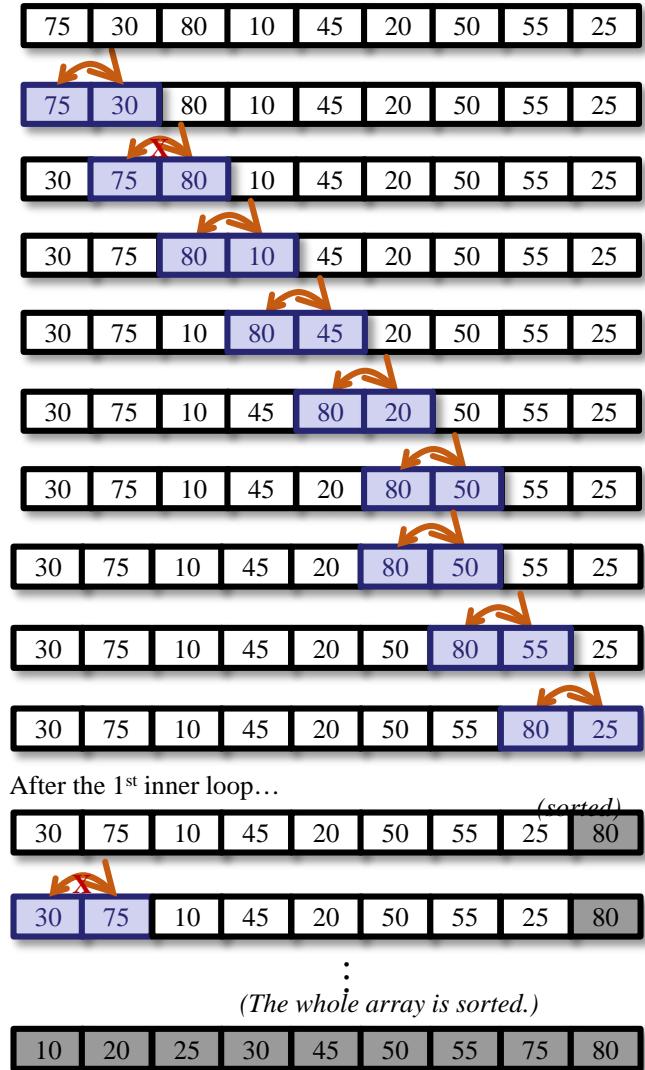
□ 如果元素大於下一個元素則交換

○ Best Case :  $O(n)$

○ Worst Case :  $O(n^2)$

```
def b_sort(data):
    for i in range(len(data)):
        for j in range(len(data)-1):
            if (data[j]>data[j+1]): #若大於下一個
                data[j], data[j+1] = data[j+1], data[j]

data = [56, 23, 41, 12, 67, 84, 36]
b_sort(data)
print(data)
```



# 插入排序法Insertion Sort

□ 複雜度: Best case -  $O(n)$ , Worst case -  $O(n^2)$

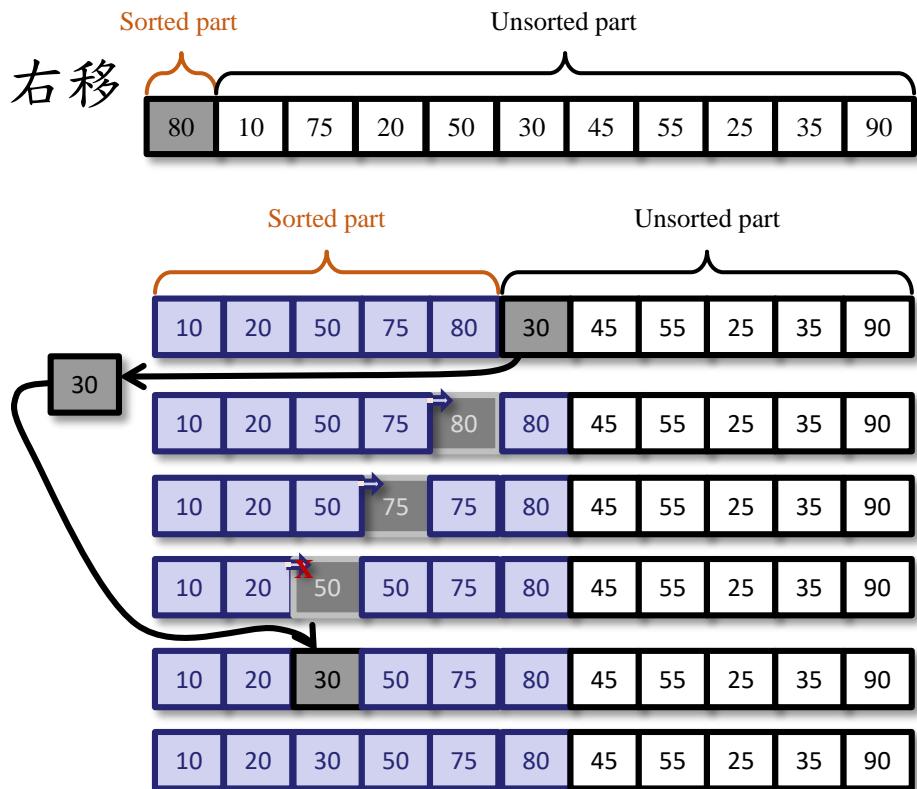
□ 演算步驟

○ 一開始，只有第一個元素在sorted part，其他在unsorted part

○ 設定排序的值為key

○ 若key值左邊元素大於key則右移

○ 把key設定為右移的最左邊



# 插入排序法Insertion Sort

## □ Code

```
def i_sort(data):
    for i in range(len(data)):
        n=data[i]           #還沒排序值
        j=i-1              #還沒排序的值的左邊一格
        while(j>=0 and data[j]>n):
            data[j+1]=data[j]  #右移
            j=j-1
        data[j+1]=n         #把key值帶進去

data = [56, 23, 41, 12, 67, 84, 36]
i_sort(data)
print(data)
```

# Python Recursive

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 遞迴函式(Recursive Function)

□ 函式執行中不斷自己呼叫自己，範例

○ 階層計算(factorial)

- $n! = n \times (n-1)!$
- $5! = 5 \times 4! = 5 \times 4 \times 3! = 5 \times 4 \times 3 \times 2! = 5 \times 4 \times 3 \times 2 \times 1! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

```
def factorial (num):  
    if (num == 1):  
        return num  
    else:  
        return num * factorial(num - 1)  
  
print(factorial(5))                      #120
```

# 遞迴函式(Recursive Function)

## □ 設計遞迴函式兩個重點

- (1) (**base condition**)，結束呼叫的終止條件。

➤ 為避免函式永無止盡自我呼叫 (self-calling)，需設計一個明確**終止條件**

- (2)(**general condition**)，遞迴自我呼叫的方式

```
def factorial_loop (n):  
    factor = 1  
    for i in range(1,n+1):  
        factor *= i  
    return factor
```

```
print(factorial_loop(5))      # 120
```

```
def factorial (num):  
    if (num == 1):  
        return num  
    else:  
        return num * factorial(num - 1)
```

終止條件

自我呼叫

```
print(factorial(5))          #120
```

# Exercise

## □ 費氏數列 Fibonacci sequence

- 一個數列，每一項都等於其前兩項的和，
- 第 n 項等於第 n-1 項以及第 n-2 項的和

$$f(n) = \begin{cases} 1 & \text{if } n = 1 \text{ or } 2 \\ f(n - 1) + f(n - 2) & \text{if } n \geq 3 \end{cases}$$

編號每一行程式

劃出流程圖

寫下執行編號順序

寫下輸出內容

$$f(0) = 0, f(1) = 1$$

$$f(n) = f(n-1) + f(n-2), n >= 2$$

```
def fibonacci(n):
    if n == 0:
        return [ ]
    elif n == 1:
        return [ ]
    else:
        return [ ]
```

```
print(fibonacci(0))
print(fibonacci(1))
print(fibonacci(2))
print(fibonacci(3))
print(fibonacci(4))
print(fibonacci(5))
```

# 遞迴函式特性

## □ 遞迴優點

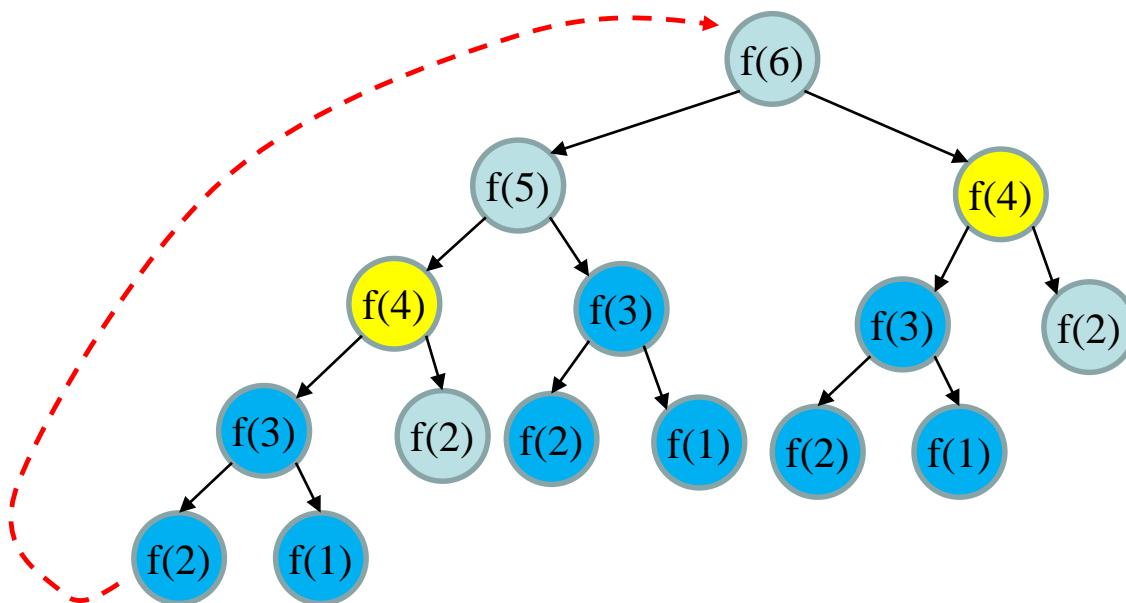
- 容易理解，
- 縮短程式碼長度

## □ 增加時間複雜度 (time complexity) 與空間複雜度 (space complexity)

- 要計算  $\text{fibonacci}(n-1) + \text{fibonacci}(n-2)$  時，要先計算並記住  $\text{fibonacci}(n-1)$  及  $\text{fibonacci}(n-2)$ ，這個「記住」需記憶體
- 計算第 n 項的值需計算第 n-1 項及第 n-2 項的值，而第 n-1 項的值又來自於 n-2 及 n-3 項，計算第 n-2 項又要用到第 n-3 項及第 n-4 項...
- 函式回傳值存在記憶體直到同一層函式被執行完為止，因此佔用大量空間，且曾計算過的又重新計算，浪費時間空間，造成程式效率不佳(inefficiency)

# Exercise 改善遞迴效率

- 費式數列 1, 1, 2, 3, 5, 8, ..
- 使用迴圈
- 使用遞迴 + 串列，避免不必要的重複計算



# Exercise 改善遞迴效率

- 費式數列 1, 1, 2, 3, 5, 8, .. (以空間list換取時間)

- 遞迴版本演化

```
def h1(n):
    if n==0 or n==1:
        return 1
    else:
        f1 = h1(n-1)
        f2 = h1(n-2)
        return f1+f2

def h2(n):
    if n==0 or n==1:
        return 1
    else:
        return h2(n-1)+h2(n-2)

def test():
    n=7
    print(h1(n))
    print(h2(n))

test()
```

# Exercise 改善遞迴效率

□ 費式數列 1, 1, 2, 3, 5, 8, .. (以空間list換取時間)

○ 遞迴版本演化

```
def h3(data,n):
    if n==0 or n==1:
        return 1
    else:
        if data[n-1]>0:
            f1=data[n-1]
        else:
            f1 = h3(data, n-1)
        f2 = h3(data, n-2)
        fvalue = f1+f2
        data[n]= fvalue
    return fvalue

def test():
    n=7
    data = [1, 1] + [0 for i in range(n+1)]
    print(h3(data, 7))
```

```
def h4(data, n):
    if data[n]==0:
        data[n] = h4(data, n-1)+h4(data, n-2)
    return data[n]

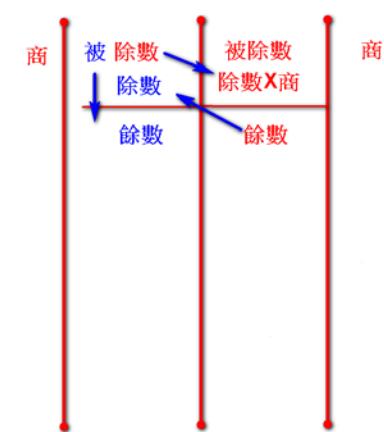
def test():
    n=7
    data = [1, 1] + [0 for i in range(n+1)]
    print(h4(data, n))
```

# Exercise

## □ 計算最大公因數 GCD

```
def gcd(x, y):
    while (x>0) and (y>0):
        if (x>y):
            x = x%y
        else:
            y = y%x
    return (x if x>y else y)
```

1	123	321	2
1	75	246	
1	48	75	1
3	27	48	
3	21	27	1
	18	21	
	3	6	2
		6	
		0	



## □ 使用遞迴函式的方式計算最大公因數 GCD

```
def gcd(m, n):
    if n == 0:
        return  
    else:
        return gcd(n,  )
```

```
print(gcd(18, 24))
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

自我呼叫

# Exercise

- 一個數列 K 的前兩項是 0、1，之後每一項為  $K_n = 2*K_{n-1} + 3*K_{n-2}$ ，使用遞迴函式計算  $K_n$ ，輸入一個正整數 N，印出該數列的第 N 項。
  
- 輸入一個字串，計算字串裡面數字的個數 (`noOfDigits(n)`)
- 輸入一個整數，計算裡面數字的個數 (`noOfDigits(n)`)
- 輸入一個整數，計算數字的加總 (`digitSum(n)`)
- 輸入一字串，判斷是否 palindrome 回文  
(`checkPalindrome(wordPal, index)`)
- 輸入字串，輸出字串反轉
- 撰寫 binary search

# Exercise

```
def f(z):
    if (len(z)<=1):
        if z in [str(i) for i in range(10)]:
            return 1
        else:
            return 0
    if z[0] in [str(i) for i in range(10)]:
        return 1+f(z[1:])
    return f(z[1:])
```

```
def main():
    print(f('1_9f8276xe5432r!1^'))
    print(f('9'))
```

# Exercise Solution

```
def digitSum(n):
    if(n == 0):
        return 0
    return ((n%10) + digitSum(      )) #呼叫 DigitSum 自己
print(digitSum(678))
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

```
def check(x):
    strLen = len(x)
    if strLen==0 or strLen==1: return True
    if x[0]!=x[strLen-1]: return False
    return check(x[1:      ]) #呼叫自己
```

```
print(check('abcba'))
print(check('abba'))
print(check('abcdedcba'))
print(check('amobma'))
```

```
def noOfDigits(n):
    if (n!=0):
        return 1 + noOfDigits(      )
    else:
        return 0
print(noOfDigits(12301))
```

# Exercise 1/2

□ permutation 稱為 "arrangement number"

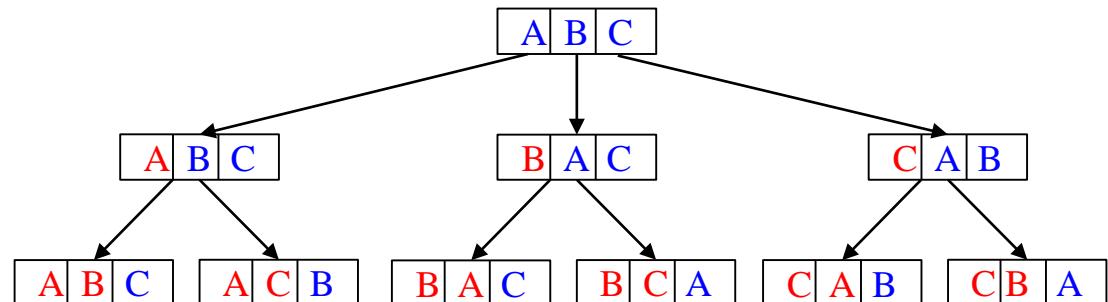
- 'AB' 排列 'AB', 'BA'

$$P(AB) = AP(B) | BP(A)$$

$$P(B) = ? , P(A) = ?$$

- $f('ABC')$

- ' $A$ ' +  $f('BC')$
- ' $B$ ' +  $f('AC')$
- ' $C$ ' +  $f('AB')$



# Exercise 1/2

□ 'abc'

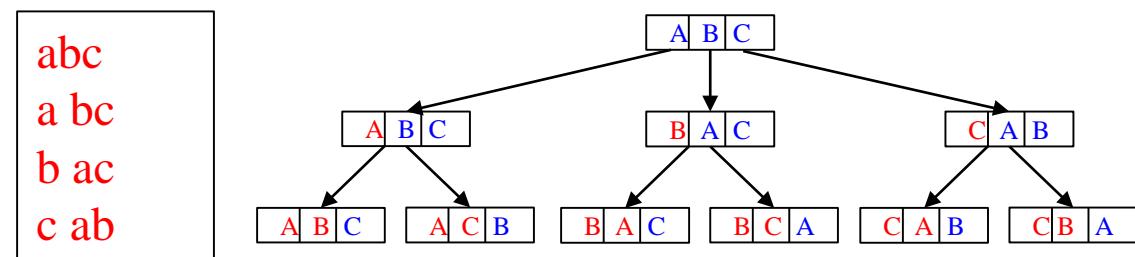
- 「'a' + 'bc'」 、 「'b'+'ac'」 、 「'c'+'ab'」 、

```
data = 'abc'  
#data = list(data)  
print(data)  
  
length = len(data)  
x=data[0]  
y=data[0:0]+data[0+1:length]  
print(x,y)
```

```
x=data[1]  
y=data[0:1]+data[1+1:length]  
print(x,y)
```

```
x=data[2]  
y=data[0:2]+data[2+1:length]  
print(x,y)
```

```
data = 'abc'  
print(data)  
length = len(data)  
for i in range(length):  
    x = data[i]  
    y=data[:i]+data[i+1:length]  
    print(x,y)
```



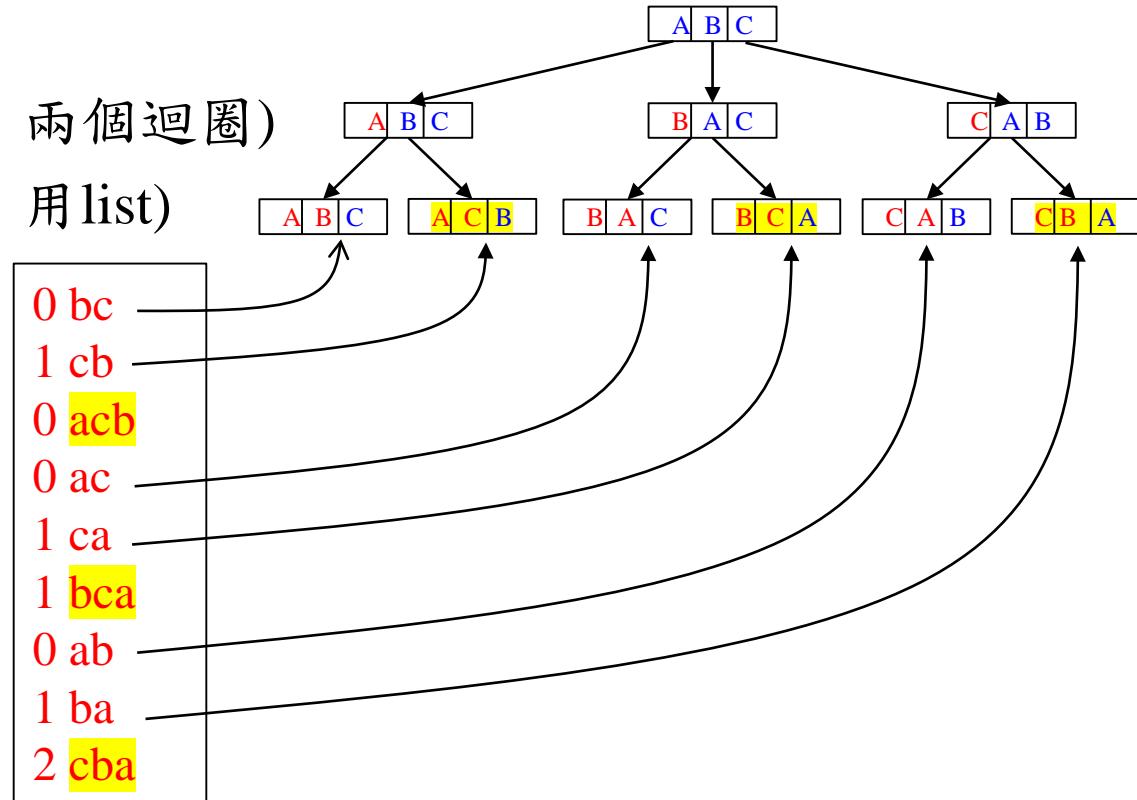
# Exercise 1/2

## □ 遞迴第一版

- 使用一個迴圈(不夠，兩個迴圈)
- 回傳一個字串(不夠，用 list)

```
def f(data):  
    n = len(data)  
    if n<=1:  
        return data  
    for i in range(n):  
        x=data[i]  
        y=data[0:i]+data[i+1:n]  
        r = x + f(y)  
        print(i, r)  
    return r
```

```
f('abc')
```

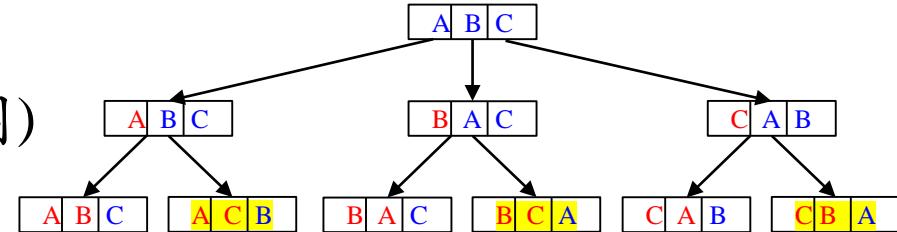


# Exercise 1/2

## □ 遞迴第一版

- 使用一個迴圈(不夠，兩個迴圈)
- 回傳一個字串(不夠，用 list)

```
def f(data):
    n = len(data)
    if n<=1:
        return [data]
    r = []
    for i in range(len(data)):
        x = data[i]
        y = data[:i]+data[i+1:]
        z = f(y)
        for sub in z:
            r = r + [x+ sub]
    return r
```



```
print(f('ABC'))
```

# Exercise 2/2

□ 輸入字串，輸出字串排列

```
def f(s):
    if len(s)==1:
        return [s]
    result = []
    for i in range(len(s)):
        result += [s[i]+ sub for sub in f(s[:i]+s[i+1:])]
    return result
```

```
print(f('ABC'))
print(f('ABCD'))
```

編號每一行程式  
劃出流程圖  
寫下執行編號順序  
寫下輸出內容

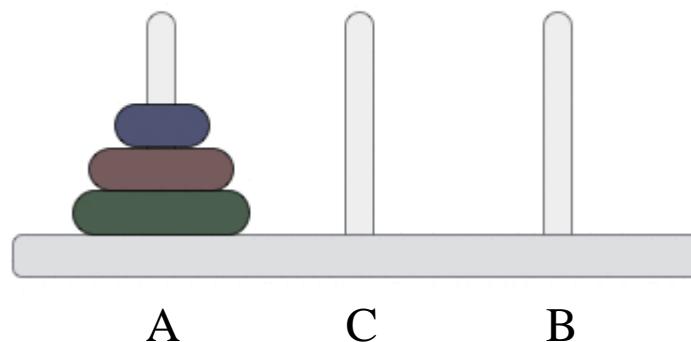
```
['ABC', 'ACB', 'BAC', 'BCA', 'CAB', 'CBA']
['ABCD', 'ABDC', 'ACBD', 'ACDB', 'ADBC', 'ADCB', 'BACD', 'BADC', 'BCAD',
 'BCDA', 'BDAC', 'BDCA', 'CABD', 'CADB', 'CBAD', 'CBDA', 'CDAB', 'CDBA',
 'DABC', 'DACP', 'DBAC', 'DBCA', 'DCAB', 'DCBA']
```

# 河內塔

## 問題

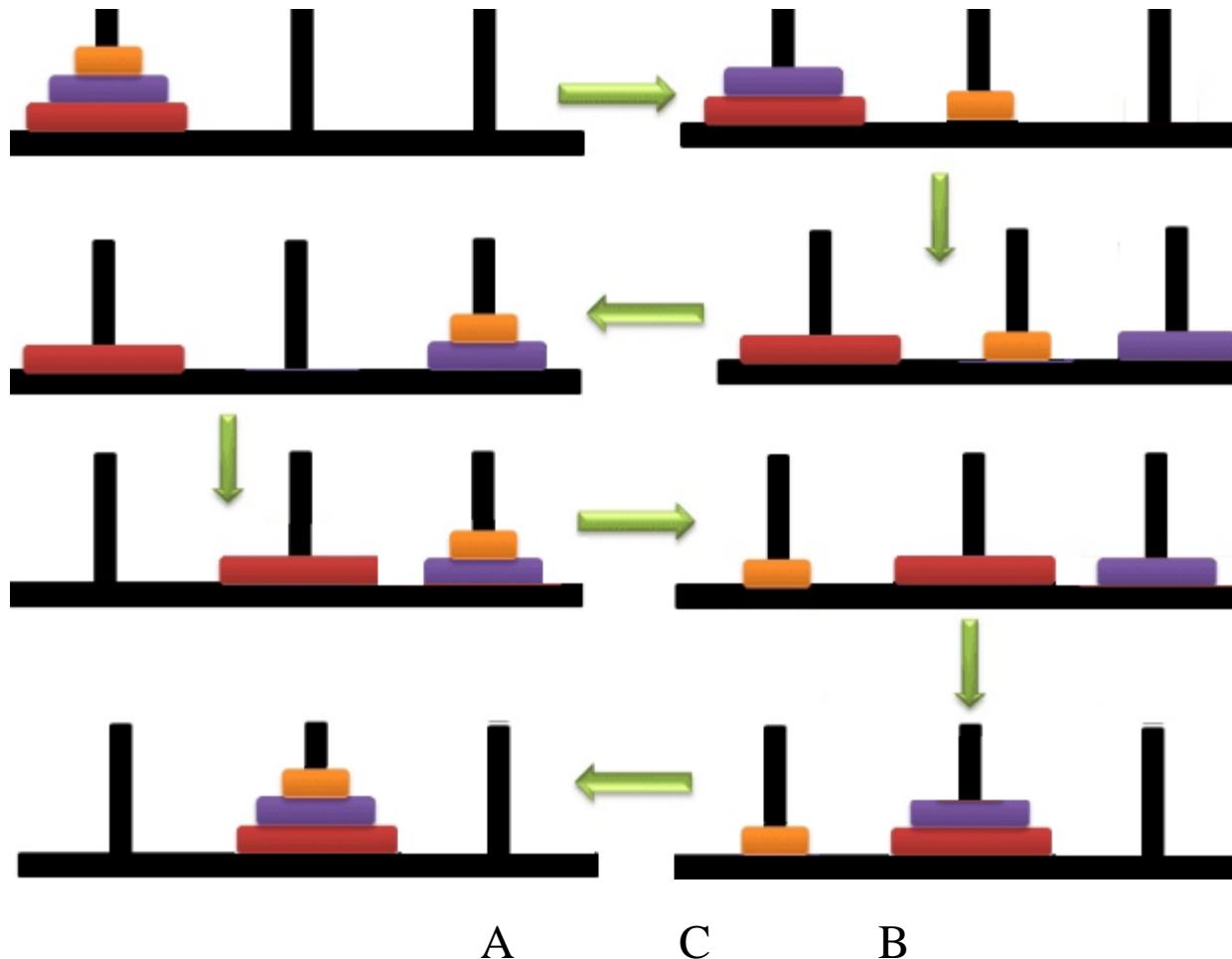
- 有三根竿子A，B，C。A竿上有N個( $N > 1$ )穿孔圓盤，尺寸由下到上依次變小。按下列規則將所有圓盤移至C杆：
  - 每次只能移動一個圓盤。
  - 大盤不能疊在小盤上面。
  - 圓盤可以在任意一個竿子上。
- 可將圓盤臨時置於B竿，也可將從A竿移出的圓盤重新移回A竿，但都須遵循上述規則。

Step: 0



# 河內塔

## 問題



# Exercise 河內塔

```
def move(disk, source, destination):
    print(disk,':',source,'->',destination)

def f(n, S, D, T):
    if n==1:
        move(n, S, D)
    elif n==2:
        move(n-1, S, T)
        move(n, S, D)
        move(n-1, T,D)
    else:
        f(n-1, S, T, D)
        move(n, S, D)
        f(n-1, T, D, S)

f(3,'A','C','B')
```

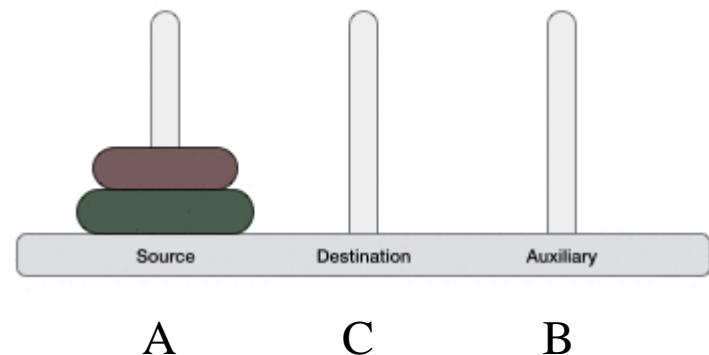
# Exercise 河內塔

## □ 解決

- 將較小的(頂部)圓盤移動到輔助竿子B。
- 將較大的(底部)圓盤移動到目標竿子C。
- 將較小的圓盤從輔助竿子B移動到目標竿子C。
- 請寫出執行流程與編號

```
1 def move(i, x, y):  
2     print(i , x , y)  
3  
4 def hanoi(i, A, C, B):  
5     if(i ==1): move(i , A , C)  
6     else:  
7         hanoi(i-1, A, B, C)  
8         move(i , A , C)  
9         hanoi(i-1 , B , C , A)  
10  
11 hanoi(3, 'A', 'C', 'B')
```

Step: 0



# 實作搜尋(Search)串列

## □ 二元搜尋(Binary Search)

- 資料須要排序
- 每次都從範圍(left~right)的中間點 $mid=(left+right)/2$ 找
- 中間點太大，往左找，中間點太小，往右找

假設找9

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
1	2	5	7	9	14	23	26
left=0			mid = 3 7<9 left = 3				right=7
					mid = 5 (7+3)/2 14>9 right= 5		
				mid = 4 (3+5)/2			

# Exercise二元搜尋

- 請寫出執行流程與編號
- 請寫出遞迴程式

```
01 def binarySearch(data, left, right, key):  
02     while left<right:  
03         mid = (left+right)//2  
04         if data[mid]==key: return mid  
05         elif data[mid]>key: right = mid  
06         else: left = mid  
07     return -1  
08  
09 data=[1, 2, 5, 7, 9, 14, 23, 26]  
10 print(binarySearch(data, 0, 7, 9))  
11 data=[11, 23, 49, 57, 66, 78, 84, 91]  
12 print(binarySearch(data, 0, 7, 84))
```

```
01 def binarySearch(data, left, right, key):  
02     mid = (left+right)//2  
03     if data[mid]==key: [ ]  
04     if left==right: [ ]  
05     if data[mid]>key: right = mid-1  
06     else: left = mid+1  
07     return binarySearch(data, [ ])  
08  
09 data=[1, 2, 5, 7, 9, 14, 23, 26]  
10 print(binarySearch(data, 0, 7, 9))  
11 data=[11, 23, 49, 57, 66, 78, 84, 91]  
12 print(binarySearch(data, 0, 7, 84))
```

# Exercise 數位電路模擬

□ 模擬一個數位電路。

- 輸入  $n$  是二進位 8 位元，輸出是二進位 4 位元。
- 輸入範圍從 00000000 到 11111111 (十進位 0~255).

○ 此數位電路內具有回饋迴路，其功能如下：

- $C(m) = m \quad \text{if } m = 0 \text{ or } m = 1$
- $C(m) = C(m/2) \quad \text{if } m \text{ is even 偶數}$
- $C(m) = C((m+1)/2) \quad \text{if } m \text{ is odd 奇數}$

○ 此電路有一個紀錄器，會記錄跑過幾次回饋迴路，最後輸出為回饋電路跑過的次數。

- 例如  $m=00001010$ (十進位 10)，則電路內部運算回饋電路輸入依序為十進位 5, 3, 2, 1 。
- $C(10)=C(5)=C(3)=C(2)=C(1)=1$
- 共跑過 4 次。則此電路輸出為 0100 (十進位 4)。

# Exercise 數位電路模擬I

□ 模擬一個數位IC，內有回饋電路與紀錄器電路。

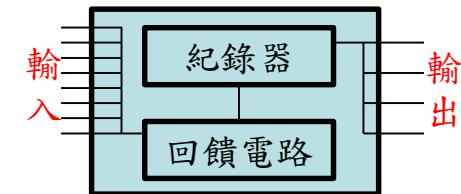
- 輸入 $m$  是二進位 8 位元，輸出是二進位 4 位元。
- 輸入範圍從 00000000 到 11111111 (十進位 0~255).

○ 數位IC內有一個回饋電路，回饋方式：

- $C(m) = m$  if  $m = 0$  or  $m = 1$  (十進位)
- $C(m) = C(m/2)$  if  $m$  偶數(十進位)
- $C(m) = C((m+1)/2)$  if  $m$  奇數(十進位)
- 例如  $m=00001010$ (十進位 10)，則電路回饋依序為十進位 5, 3, 2, 1 。
- $C(10)=C(5)=C(3)=C(2)=C(1)=1$ ，共回饋 4 次。

○ 數位IC內有一個紀錄器，會記錄回饋電路的回饋次數。

- $R(m) = [C(m)$  的回饋次數]，例如  $R(10) = 4$  。



○ 數位IC的輸出為紀錄器所記錄之回饋電路的回饋次數。

- 若數位IC的輸入為 $m=00001010$ (十進位 10)，因回饋電路的回饋次數為4，則此數位IC輸出為 0100(十進位 4)。

# Exercise 數位電路模擬I

□ 模擬一個數位IC，內有回饋電路與紀錄器電路。

輸入說明:

二進位 8 bit 位元

第一行是第一個測試案例資料

接著是一行 0 分隔測試資料

第三行是第二個測試案例資料

....

最後 -1 結束

輸出說明:

二進位 4 bit 位元

每一行是一個測試案例資料的結果

Sample Input:

00000000

0

11111111

0

00000001

0

10000000

0

00111111

-1

Sample Output:

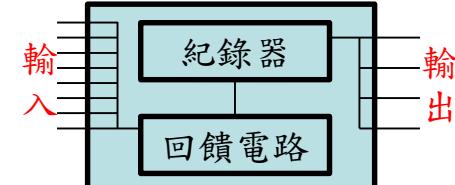
0000

1000

0000

0111

0110



# 快速排序法(Quick sort)

## □ 概念

- 選取某個元素做為基準值，令此基準值為target。
- 將所有比target小的資料，都放在target左邊；
- 所有不比target小的資料都放在target右邊。

## □ 步驟

- 選取第0個元素 69為基準
- 從最右邊往左找比基準69還小的元素32
- 從最左邊往右找比基準69還大的元素81
- 兩個元素交換



索引	0	1	2	3	4	5	6	7	8	9
數值	69	81	30	38	9	2	47	61	32	79

# 快速排序法(Quick sort)

```
01 def QuickSort (data, left, right):
02     if (left>=right): return data
03     i = left
04     j = right
05     target = data[left]
06     while i!=j:
07         while (data[j]>target) and (i<j):
08             j = j-1 #從右邊開始找
09         while (data[i]<=target) and (i<j):
10             i = i+1 #從左邊開始找左邊開始找比基準點大，
11                 #如果有找到又沒與從右邊的相遇
12                 #表示 data[i] 一定可以換到比較小的
13                 #否則 data[i] 一定是小的最邊緣，可以跟中間值交換
14     if(i<j):
15         data[i], data[j] =data[j], data[i] #左右沒相遇則可交換
16         print(data)
17     data[left], data[i] = data[i], data[left] #i是中間值
18     data= QuickSort(data, left, i-1) #處理左半邊
19     data = QuickSort(data, i+1, right) #處理右半邊
20     return data
21
22 print(QuickSort([34, 23, 52], 0, 2))
```

# 快速排序法(Quick sort)

- 須從右邊right開始往左找 j--，找比中間點小的準備交換
  - 之後才可以左邊left開始往右找 i++，找比中間點大的交換
  - 若先left往右找，必停在比[0]大地方，接著right往左找，必停在right==left，那就會把比[0]大，換到[0]地方

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
6	9 3	7	0	1	3 9

i=1

j=5

6	3	7 1	0	4 7	9
---	---	-----	---	-----	---

i=2

j=4

6	3	1	0	7	9
---	---	---	---	---	---

i=2

j=3

j=3,  
i=3

先由右往左找 j--，再由左往右找 i++， $a[3]<6$ ,  $i=2+1=3$   
此時  $a[i=3]$  與  $a[left=0]$  交換，正確

6 0	3	1	0 6	7	9
-----	---	---	-----	---	---

6	3	1	0	7	9
---	---	---	---	---	---

j=4

i=2      i=3      i=4

先由左往右找 i++， $i=3, j=4$ 時還可再 i++， $i=4$ ，  
再由右往左找 i==j，仍然  $j=4$   
此時  $a[i=4]$  與  $a[left=0]$  交換將發生錯誤

6 7	3	1	0	7 6	9
-----	---	---	---	-----	---

# Exercise

- 輸入一串數字(1~9)，及一正整數N，輸出是否有N個總和相等的子集(需包含所有元素)\*\*請用遞迴\*\*

Input	Output
4 3 2 3 5 2 1 4	True $(1+4 = 3+2 = 3+2 = 5, \text{共}4\text{組})$
4 3 2 3 5 2 1 2	True $1+4+3+2 = 3+2+5, \text{共}2\text{組}$
4 3 2 3 5 2 1 6	False

## □ 基本概念

- 假設每一個總和value，有N個value，所有輸入數字加總sum
  - N個總和相等，  $N \times value = sum$
- 若輸入  $data = [4, 3, 2, 3, 5, 2, 1], N=4$ ，有4組，每組加總是5
  - $value = sum(data)/N, (4+3+2+3+5+2+1)/4 = 5$
- 4組，都有加總value=5，一次檢查一組，再把這組數字從data移除

# Exercise

- 演算法/演算步驟，若  $\text{data} = [4, 3, 2, 3, 5, 2, 1]$ ,  $N = 4$ 
  - 迴圈檢查四組，都要找到一組數字加總  $\text{value}=5$ ，放進bag
    - 一次檢查一組，找出放入bag，再把這一組數字(bag)從data移除
      - 第一次
        - »  $\text{data} = [4, 3, 2, 3, 5, 2, 1]$ ,  $N = 4$ ,  $\text{value} = 5$ ,  $\text{bag} = [4, 1]$
        - »  $\text{data}$ 移除 bag， $\text{data} = [3, 2, 3, 5, 2]$
      - 第二次
        - »  $\text{data} = [3, 2, 3, 5, 2]$ ,  $N = 4$ ,  $\text{value} = 5$ ,  $\text{bag} = [3, 2]$
        - »  $\text{data}$ 移除 bag， $\text{data} = [3, 5, 2]$
      - 第三次
        - »  $\text{data} = [3, 5, 2]$ ,  $N = 4$ ,  $\text{value} = 5$ ,  $\text{bag} = [3, 2]$
        - »  $\text{data}$ 移除 con， $\text{data} = [5]$
      - 第四次
        - »  $\text{data} = [5]$ ,  $N = 4$ ,  $\text{value} = 5$ ,  $\text{bag} = [5]$

# Exercise

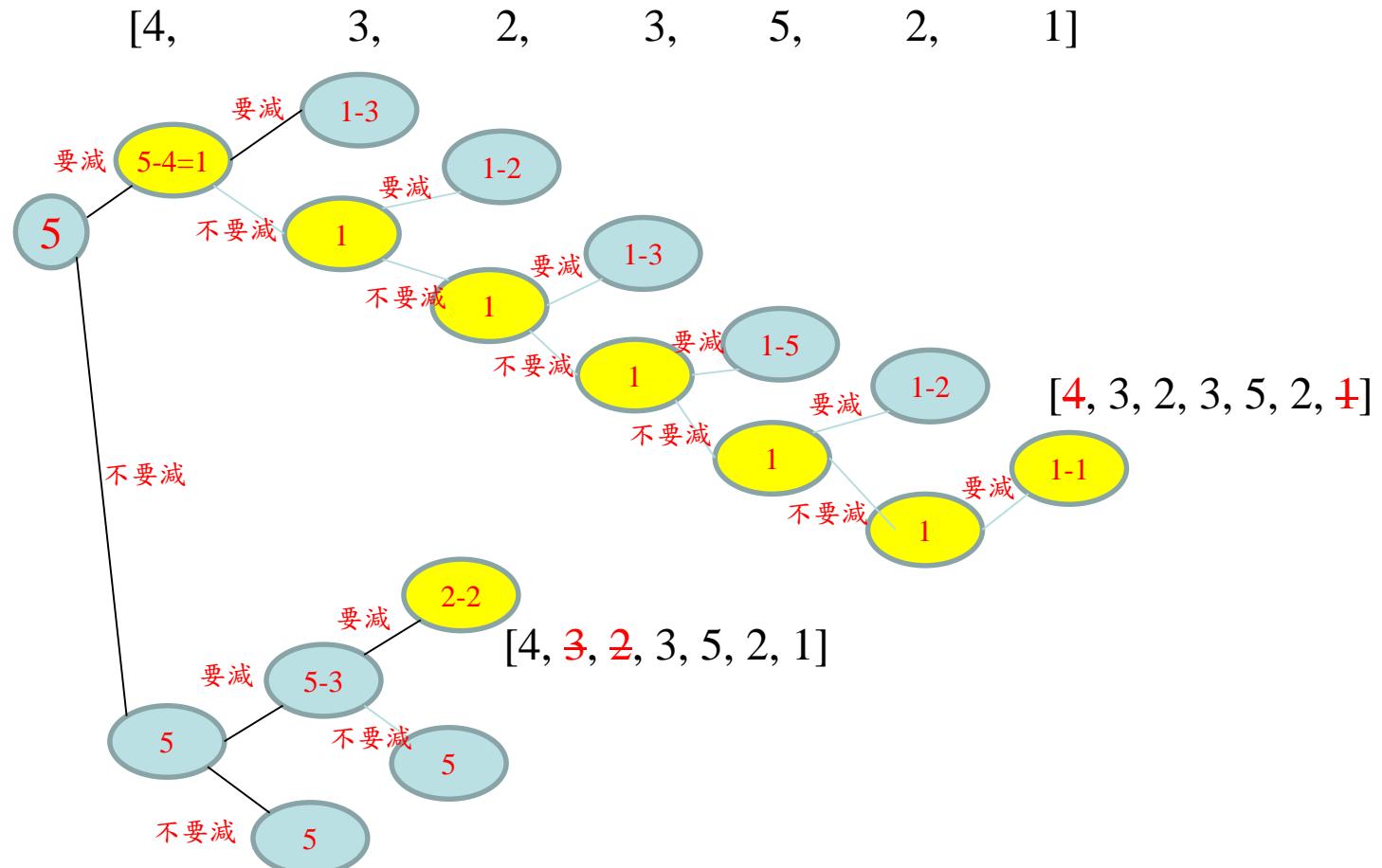
## □ 每次檢查的演算法/演算步驟 findBag

- $\text{data} = [4, 3, 2, 3, 5, 2, 1]$ ,  $N = 4$ ,  $\text{value} = 5$ ,  $\text{bag} = [4, 1]$

- index從0開始，value每次試著減掉 $\text{data}[\text{index}]$ ，最後剛好減完
  - 第一次 $\text{index}=0$ :  $\text{value}-4$  ( $\text{data}[0]$ ) =  $5 - 4 = 1$
  - 第二次:  $3, 2, 3, 5, 2$ 都不是，遇到1 ( $\text{index}=6$ )，完成
  - $\text{data}$ 移除 bag後， $\text{data} = [3, 2, 3, 5, 2]$

# Exercise

## □ 遍迴的演算法/演算步驟



# Exercise

## □ **findBag**遞迴的演算法/演算步驟

- 基本/結束條件1: data全找了(index=0~6)找不到，**False**
- 基本/結束條件2: data[index]==剩下的value，完全找到，把 data[index]加入bag，**True**
- 一般情況
  - 目前這個要，value-data[index]，遞迴呼叫，往下找(index+1)
    - 把data[index]加入bag
  - 目前這個不要，value不減data[index]，遞迴呼叫，往下找(index+1)
- 輸入參數
  - data
  - value
  - index

# Exercise – 遞迴版1

□ compute遞迴演算步驟是？

- 
- 

```
def remove(data, bag):
    for i in bag:
        data.remove(i)

def findBag(data, index, value, bag):
    if index>=len(data) or value<data[index]:
        return False
    elif data[index]==value:
        bag.append(data[index])
        return True
    elif findBag(data, index+1, value, bag)==True:
        return True
    elif findBag(data, index+1, value-data[index], con)==True:
        bag.append(data[index])
        return True
    else: return False
```

```
def compute(N, data, value):
    if N==0: return True
    bag = []
    if findBag(data, 0, value, bag)==False:
        return False
    print(bag)
    remove(data, bag)
    return compute(N-1, data, value)

def f(N, data):
    print(compute(N, data, sum(data)/N), '-')

f(4, [4, 3, 2, 3, 5, 2, 1])
f(2, [4, 3, 2, 3, 5, 2, 1])
f(2, [1, 2, 3, 5])
f(2, [1, 5, 11, 5])
```

# Exercise – 遞迴+迴圈版

## □ findBag遞迴演算步驟是?有何不同?

```
def remove(data, bag):
    for e in bag:
        data.remove(e)
```

```
def findBag(data, num, bag):
    if data[0]==num:
        bag.append(data[0])
        return True
    elif len(data)==0 or num<0:
        return False
    if findBag(data[1:], num-data[0], bag)==True:
        bag.append(data[0])
        return True
    if findBag(data[1:], num, bag)==True:
        return True
    return False
#每一次檢查第0個
```

```
def compute(data, N):
    num = sum(data)//N
    if num*N != sum(data):
        return False
    for i in range(N):
        bag = []
        if findBag(data, num, bag)==False:
            return False
        print('bag=', bag)
        remove(data, bag)
    return True
```

```
compute([4, 3, 2, 3, 5, 2, 1], 4)
compute([4, 3, 2, 3, 5, 2, 1], 2)
compute([1, 2, 3, 5], 2)
compute([1, 5, 11, 5], 2)
```

# Exercise – 遞迴版2

## □ compute遞迴演算步驟是?有何不同?

```
def remove(data, bag):
    for e in bag:
        data.remove(e)

def findBag(data, num, bag):
    if data[0]==num:
        bag.append(data[0])
        return True
    elif len(data)==0 or num<0:
        return False
    if findBag(data[1:], num-data[0], bag)==True:
        bag.append(data[0])
        return True
    if findBag(data[1:], num, bag)==True:
        return True
    return False
#每一次檢查第0個
```

```
def compute(data, N, num):
    if N==0: return True
    bag = []
    if findBag(data, num, bag)==False:
        return False
    print('r=',bag)
    remove(data, bag)
    return compute(data, N-1, num)

def process(data, N):
    num = sum(data)/N
    if num*N != sum(data):
        return False
    return compute(data, N, num)
```

```
process([4, 3, 2, 3, 5, 2, 1], 4)
process([4, 3, 2, 3, 5, 2, 1], 2)
process([1, 2, 3, 5], 2)
process([1, 5, 11, 5], 2)
```

# Exercise – 迴圈版

## □ 演算法/演算步驟?

- 找出所有可能組合，測試哪一種組合，總和是value

```
def remove(data, bag):
    for e in bag:
        data.remove(e)

def findBag(data, code, value):
    N = len(data)
    bag = []
    for i in range(N):
        index = code%2
        code = code//2
        if index==1:
            value = value-data[i]
            bag.append(data[i])
        if value!=0: bag=[]
    return bag
```

```
def compute(data, value):
    for i in range(2**len(data)):
        bag = findBag(data,i, value)
        if (len(bag)>0):
            print(bag)
            remove(data, bag)
```

```
compute([4, 3, 2, 3, 5, 2, 1], 4)
compute([4, 3, 2, 3, 5, 2, 1], 2)
compute([1, 2, 3, 5], 2)
compute([1, 5, 11, 5], 2)
```

# Python List 應用

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# Built-in methods for List

Method	Description
<a href="#"><u>append()</u></a>	Adds an element at the end of the list
<a href="#"><u>clear()</u></a>	Removes all the elements from the list
<a href="#"><u>copy()</u></a>	Returns a copy of the list
<a href="#"><u>count()</u></a>	Returns the number of elements with the specified value
<a href="#"><u>extend()</u></a>	Add the elements of a list (or any iterable), to the end of the current list
<a href="#"><u>index()</u></a>	Returns the index of the first element with the specified value
<a href="#"><u>insert()</u></a>	Adds an element at the specified position
<a href="#"><u>pop()</u></a>	Removes the element at the specified position
<a href="#"><u>remove()</u></a>	Removes the item with the specified value
<a href="#"><u>reverse()</u></a>	Reverses the order of the list
<a href="#"><u>sort()</u></a>	Sorts the list
<a href="#"><u>list()</u></a>	It is also possible to use the list() constructor to make a new list

# Exercise

- 輸入 list，回傳所有不重複的元素 list

```
a = [1,2,3,4,3,2,1]  
print dedupe_v1(a)
```

```
# this one uses a for loop  
def dedupe_v1(x):  
    y = []  
    for i in x:  
        if i not in y:  
            y.append(i)  
    return y
```

```
#this one uses sets  
def dedupe_v2(x):  
    return list(set(x))
```

# Exercise

## □ 輸入兩個list

- $a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]$
- $b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1]$
- 輸出兩個都有的元素【每個元素不重複/交集】
- 以下程式可能有錯，如何修正？

```
def ex01():
    a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
    b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
    c = []
    for i in b:
        if i in a:
            c.append(i)
    print(c)
ex01()
```

[1, 2, 3, 5, 8, 13]

if (i in a) and (i in b):

# Exercise

□ 輸入兩個 list，回傳任一個有的元素一次【每個元素不重複/聯集】

```
def ex01():
    a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
    b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
    c = []
    for i in a:          #加入不重複的 a 元素
        if i not in c:
            c.append(i)
    for i in b:          #加入不重複的b元素
        if i not in c:
            c.append(i)
    print(c)
```

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
c = set(a)&set(b)      #使用 set & 交集
print(c)
```

```
c = set(a)|set(b)      #使用 set | 聯集
print(c)
```

# Exercise

- 輸入字串，判斷是否為迴文
- Example of palindrome: madam

```
wrd=input("Please enter a word: ")  
wrd=str(wrd)  
rvs=wrd[::-1]  
print(rvs)  
if wrd == rvs:  
    print("This word is a palindrome")  
else:  
    print("This word is not a palindrome")
```

```
def reverse(word):  
    x = ""  
    for i in range(len(word)):  
        x += word[len(word)-1-i]  
    return x  
  
word = input('give me a word:\n')  
x = reverse(word)  
if x == word:  
    print('This is a Palindrome')  
else:  
    print('This is NOT a Palindrome')
```

# join()

## □ string.join(iterable)

```
myList = ["name", "John", "country", "Norway"]  
mySeparator='TEST'
```

```
x=mySeparator.join(myList)  
  
print(x)
```

```
myTuple = ("John", "Peter", "Vicky")  
  
x = "#".join(myTuple)  
  
print(x)
```

John#Peter#Vicky

# Exercise

## □ 程式

- 輸入 My name is Michele
- 輸出 Michele is name My

## □ 程式

- 給定一串數字，將這一連串數字分為奇數在左，偶數在右，奇數按照由小到大排，偶數按照由大到小排。
- 輸入說明:輸入整數n代表有n個數，接著輸入n行整數
- 輸出說明:將奇數放左(小到大排)，偶數在放右(大到小排)

```
numList=[int(input()) for i in range(int(input()))]
right = sorted([x for x in numList if ], reverse=True)
left  = sorted([x for x in numList if ])
print(left + right)
```

# Exercise

## □ 程式

- 輸入一行n個整數，將這一連串數字分為被3整除在左，被3除餘1在右，被3除餘2在中間，中間按照由小到大排，左右按照由大到小排。

```
data = input().split()
```

```
numList=[int(x) for x in data]
```

```
right = sorted([x for x in numList if x % 3 == 0], reverse=True)
```

```
left = sorted([x for x in numList if x % 3 == 1], reverse=True)
```

```
middle = sorted([x for x in numList if x % 3 == 2])
```

```
print(left + middle + right)
```

# Exercise階梯數字

- 輸入一個整數N，輸出小於N的所有階梯數字個數
  - 檢查K是否階梯數字
    - 分析一串長數字是否單調上升
    - '11111111112222334455667778999' True
    - '11111111112222334457667778999' False
    - 傳入一個數字字串，回傳在第幾個字元沒有單調上升

演算法第一版，N是一般整數  
1. 輸入N，K=1，Count=0  
2. While (檢查 K<N)  
    2.1. 檢查 K 是否階梯數字  
        Count+1  
    2.2. K 加1

#檢查 K 是否階梯數字  
`def check(x):  
 for i in range(len(x)-1,-1,-1):  
 if x[i]<x[i-1]:  
 return i-1  
 return -1`

# Exercise 階梯數字

- 輸入一個整數N，輸出小於N的所有階梯數字個數
  - N是長整數，Python整數資料型別無法表示，須以字串表示

演算法第二版，主要流程(長整數數字字串N)

1. K='1'，Count=0

2. While (檢查  $K < N$ ) (cmp(K, N)函式)

    2.1. 檢查 K 是否階梯數字 (check(K)函式)

        Count+1

    2.2. K 加1 (addOne(K)函式)

cmp(K, N)函式

//若K長度較N短 <，回傳True，

//否則從最K高位，往下一個數字一個數字比 (Loop)

    //若有<則回傳 True

    //迴圈結束未返回，則回傳 False

check(K)函式

//i 從最高位往下檢查，(Loop)

    // i>i-1 則為第 i 個數字非遞增，回傳 i

    //若都沒有發生，則為遞增，回傳 -1

# 階梯數字

## addOne函式(K)

若K 個位數<'9'，加 1 回傳 return

carry = 1，K個位數9，加1要進位

K[i] 從 K 十位數開始，一直到最高位數，迴圈

K[i]若為 9，又有進位carry==1，K[i]變成 0 且下一位進1，下一個carry=1

K[i]若<9，且carry=0，中斷迴圈

最高位有進位，多一位數

```
def addOne(x):
    length = len(x)
    if x[length-1]<'9':
        return x[:length-1]+str(int(x[length-1])+1)
    carry = 1
    x = x[:length-1]+'0'
    for i in range(len(x)-2,-1,-1):
        if int(x[i])+carry==10:
            x = x[:i]+'0'+x[i+1:]
            carry = 1
        else:
            s= str(int(x[i])+carry) #x[i]=
            return x[:i]+s+x[i+1:]
    if (carry==1):
        x = '1' + x
    return x
```

```
x1 = '1111111111111111112349'
x2 = '1111111111111111112350'
x3 = '111111111111111111112355'
x4 = '11111111111111111111999999'
x5 = '11111111111111111111222222'
# 從最後往前找到非9，在前一個加一，
# 後面全變成此數字
y = addOne(x2)
```

# 階梯數字

第三版，直接找下一個階梯數字，主流程(數字字串N)

1. K='11' , Count=0
2. While (檢查 K<N) (cmp函式)
  - 2.1. 找出下一個階梯數字 (getNext函式)

Count+1

(cmp函式)

```
//若長度較短為 <，回傳True，  
//否則從最高位，往下一個數字一個數字比 (Loop)  
//若有<則回傳 True  
//迴圈結束未返回，則回傳 False
```

getNext函式

```
x1 = '1111111111111111112349'  
x2 = '1111111111111111112359'  
x3 = '1111111111111111112355'  
x4 = '1111111111111111112355'  
x5 = '111111111111111111999999'  
x6 = '11111111111111112222222'
```

# 從最後往前找到非9，在前一個加1，後面全變成此數字

11	97
12	98
13	99
14	111
15	112
16	113

# 階梯數字

getNext函式(K)

```
//index 從最低位開始找不是 9，Loop  
    //若找到第 index 位不是9  
//將最低位~第 index 位，都設為: 第 index位 數字 +1  
//若每一位數都是 9，  
    //多一位數，且每一位數都設為 1
```

```
def getNext(x):  
    index = len(x)-1  
    for i in range(len(x)-1,-1,-1):  
        if x[i]!='9':  
            index = █  
            break  
    s = str(int(x[index])+1)  
    r = x[:index]  
    for i in range(index,len(x),1):  
        r = r+█  
    return r
```

# 階梯數字

長度 \ 開頭	9	8	7	6	5	4	3	2	1
1	1	1	1	1	1	1	1	1	1
2	1	2	3	4	5	6	7	8	.....
3	1	3	6	10	15	21	28	.....	
4	1	4	10	20	35	56	.....		
5	1	5	15	35	70	.....			
6	1	6	21	56	.....				
7	1	7	28	.....					
8	1	8	.....						
.....	.....	.....							

# 階梯數字

- 以9為開頭最高位且長度為1、以8為開頭且長度為1,...的階梯數字之數量。是一個經典的巴斯卡三角形。
  - 長度L的階梯數字之數量把9開頭、8開頭等長度皆為L加總。
- N為2232，累積數量
  - 長度1+長度2+長度3=219 個數字
  - 長度4，開頭2，所以開頭為1要計入：
    - $219 + \text{開頭為1且長度為4} = 384$
  - 長度3，(2XXX)
    - 接著數字為2，沒有遞增所以沒有其他新的階梯數字。
  - 長度2，(22XX)，後是3，遞增，以222為開頭的階梯數字加上
    - 以2開頭且長度為2的階梯數字，有8個， $384 + 8 = 392$
  - 長度1，(223X)，後是2，數字變小，接下來遞增也不會有新的
  - 因此， $\leq 2232$ 的階梯數字有392個。

# 階梯數字

```
def gen(n):
    d0 = [1 for i in range(9)]
    data=[d0]
    for i in range(n):
        d=[1]
        for j in range(1,9):
            c = d[j-1]+data[i][j]
            d.append(c)
        data.append(d)
    return data

N = str(2232)
count=0
length = len(N)
data = gen(length)
```

```
for i in range(1, length):
    count = count + sum(data[i-1])
print(count) #長度1+長度2+長度3=219 個數字
#length

for i in range(0, int(N[0])-1):
    count = count + data[length-1][8-i]
print(count) #2232長度4，開頭2，開頭為1要計入
#length-1

for i in range(int(N[0])-1,int(N[1])-1):
    count = count + data[length-2][8-i]
print(count) #2232接著的數字為2，沒有遞增
#length-2

for i in range(int(N[1])-1,int(N[2])-1):
    count = count + data[length-3][8-i]
print(count) #2232再來3，因遞增，2開頭長度2階梯數字
#length-3

for i in range(int(N[1])-1,int(N[2])-1):
    count = count + data[length-3][8-i]
print(count) #2232再來2，數字變小，接著不會有階梯數字
```

# 階梯數字

## □ N為2479，累積數量

- 長度1+長度2+長度3=219 個數字
- 長度4，開頭2，開頭1要計入， $219 + \text{開頭為1且長度為4} = 384$
- 長度3，(2XXX)，後是4，遞增，以22, 23開頭的階梯數字
  - 以2開頭且長度為3的階梯數字，有36個， $384 + 36 = 420$
  - 以3開頭且長度為3的階梯數字，有28個， $420 + 28 = 448$
- 長度2，(24XX)，後是7，遞增，以244, 245, 246開頭階梯數字
  - 以4開頭且長度為2的階梯數字，有6個， $448 + 6 = 454$
  - 以5開頭且長度為2的階梯數字，有5個， $454 + 5 = 459$
  - 以6開頭且長度為2的階梯數字，有4個， $459 + 4 = 463$
- 長度1，(227X)，後是9，遞增，以2277, 2278開頭的階梯數字
  - 以7開頭且長度為1的階梯數字，有1個， $463 + 1 = 464$
  - 以8開頭且長度為1的階梯數字，有1個， $464 + 7 = 465$
  - 長度是1，若有計入，要加1， $465 + 1 = 466$

# 階梯數字

```
def gen(n):
    d0 = [1 for i in range(9)]
    data=[d0]
    for i in range(n):
        d=[1]
        for j in range(1,9):
            c = d[j-1]+data[i][j]
            d.append(c)
        data.append(d)
    return data
```

```
def countData(data, index, start, end):
    count=0
    for i in range(start-1, end-1):
        count = count + data[index-1][8-i]
    return(count)
```

```
def compute(N):
    count=0
    length = len(N)
    data = gen(length)
    for i in range(1, length):
        count = count + countData(data, i, int(N[length-i-1]), int(N[length-i]))
    if (i==1):
        count = count + countData(data, 1, int(N[0]), int(N[1]))
    #print('=>',count)
    print('----', count,'----')
```

```
compute('54321')      # 1875
compute('23456')       # 1365
compute('88888888')   # 24301
compute('525')         # 184
compute('25')          # 22
compute('1234567891234567891234') # 18239779
compute('12345678912345678912345678912345') # 332267361
compute('2479')        # 466
```

# 線段覆蓋

- 紿定一些線段，求這些線段所覆蓋的長度
- 輸入說明
  - 第一列輸入一個正整數  $n$ : 代表共有  $n$  組測試案例。
  - 接著的  $n$  列每一列是一個線段的兩端點，每一個端點是一個整數介於  $0 \sim 60000$  之間，端點之間以一個空格區隔，線段個數不超過  $5000$ 。
  - 每一線段小的數字先輸入。
- 輸出說明
  - 輸出一個正整數，為這些線段所覆蓋的總長度，重疊的部分只能算一次。

# 線段覆蓋

- 全部使用list非常耗時  $60000 * 5000$

```
def getLength(m):
    sumList = []
    for x in range(m):
        inputList = [int(num) for num in input().split()]
        if 0 <= inputList[0] <= 60000 and 0 <= inputList[1] <= 60000:
            rangeList = [i for i in range(inputList[0], inputList[1])]
            for i in rangeList:
                if i not in sumList:
                    sumList.append(i)
                else:
                    print("error")
    return len(sumList)
```

# 線段覆蓋

```
def getInput(m):
    inputList=[]
    for i in range(m):
        inputData = input().split()
        inputList.append([int(inputData[0]), int(inputData[1])])
    inputList.sort()          #取得所有輸入線段後排序
    return inputList

m = int(input())
inputList = getInput(m)
count = 0                      #線段覆蓋加總
last = 0                        #最後統計線段覆蓋的點
for i in range(m):
    if inputList[i][0]>last:      # 分開的線段
        count = count + (inputList[i][1]-inputList[i][0])
        last = inputList[i][1]
    elif inputList[i][1]>last:     # 重複的線段
        count = count + (inputList[i][1]-last)
        last = inputList[i][1]      #最後統計的點
print(inputList)
print(count)
```

# 撲克牌花色判斷

- 1. 四種花色黑桃、紅心、磚塊、梅花，定義 S, H, D, C。
- 2. 牌面符號 A, 2, …, J, Q, K，點數 2~10 為 2~10, A 為 14, J 為 11, Q 為 12, K 為 13，共有 52 張牌。
- 3. 花色大小：黑桃 > 紅心 > 方塊 > 梅花。
- 4. 輸入編碼規則為 **花色 + 牌面符號**，例如 S10 表黑桃 10、D7 表磚塊 7，CQ 表梅花 Q。
- 5. 牌型由小到大如下：
  - 散牌：單一張牌單張，沒有任何花色牌型，編號 0。
  - 一對：兩張數字一樣為 Pair，編號 1。
  - 兩對：2 組 Pair 的牌為 Two pair，編號 2。
  - 三條：三張一樣數字的為 Three of a Kind，編號 3。
  - 順子：數字連續的 5 張牌為 Straight，包括 [2, 3, 4, 5, 6], …, [11, 12, 13, 14, 2], [12, 13, 14, 2, 3], [13, 14, 2, 3, 4], [14, 2, 3, 4, 5]，編號 4。

# 撲克牌花色判斷

- 同花：五張同一花色的牌為 Flush，編號 5。
- 葫蘆：Three of a Kind 加一個 Pair 為 Full House，編號 6。
- 四條：四張一樣數字為 Four of a Kind，編號 7。
- 同花順：數字連續的 5 張且花色一樣為 Straight Flush，編號 8。

## ○ 輸入說明：

- 1. 輸入 5 張撲克牌，判斷那一類型的牌形編號(0~8)。
- 2. 檢查是否錯誤輸入，若錯誤，輸出 "Error input"。
- 3. 要檢查是否重複發牌，若重複，輸出 "Duplicate deal"。
- 4. 第一列輸入 5 個編碼由空格分開，表示 5 張撲克牌。

## ○ 輸出說明：

- 1. 輸出一個 0~8 整數，代表牌型編號；以「最大牌型輸出」。
- 2. 數字連續定義為：K(13) 和 A(14) 有相連，A(14) 和 2 有相連，依此類推。

# 撲克牌花色判斷

```
def cardPoint(playerCardPoint):
    pork=['A','2','3','4','5','6','7','8','9','10','J','Q','K']
    points=['14','2','3','4','5','6','7','8','9','10','11','12','13']
    index=pork.index(playerCardPoint)
    return int(points[index])

def straight(card):          #11, 12, 13, 14, 2
    card.sort()               #排序
    if(max(card)==14 and min(card)!=10): #有14沒有10表示會跨連續
        for i in range(5):      #跨連續小於10均加 13後可連續
            if card[i]<10:
                card[i]=card[i]+13
    card.sort()               #排序
    for i in range(4):        #判斷連續數字
        if card[i]!=card[i+1]-1:
            return 0
    return 1
```

# 撲克牌花色判斷

```
def getGrade(card, flow):
    #紀錄幾張卡花色相同
    f = [0, 0, 0, 0, 0, 0]
    #紀錄幾張卡點數相同
    p = [0, 0, 0, 0, 0, 0]
    #檢查幾張卡花色相同
    for c in ['S', 'H', 'D', 'C']:
        f[flow.count(c)]=f[( )]+1
    #檢查幾張卡點數相同
    for i in range(2, 15):
        p[card.count(i)]=p[( )]+1
```

```
if straight(card)==1 and f[5]==1: #同花順
    return 8
if p[4]==1: #四條
    return 7
if p[3]==1 and p[2]==1: #葫蘆
    return 6
if (f[5]==1): #同花
    return 5
if straight(card)==1: #順
    return 4
if p[3]==1: #三條
    return 3
if p[2]==2: #兩對
    return 2
if p[2]==1: #一對
    return 1
return 0 #散牌
```

# 撲克牌花色判斷

```
def compute(cards):
    p = [cardPoint(c[1:]) for c in cards]
    f = [c[0] for c in cards]
    print(getGrade(p,f))

def test():
    print(getGrade([13, 2, 6, 4, 14], ['S','H','S','D','C'])) #0
    print(getGrade([ 8, 8, 6, 4, 2], ['S','S','S','D','H'])) #1
    print(getGrade([13, 13, 6, 14, 14], ['S','S','S','D','C'])) #2
    print(getGrade([13, 13, 6, 14, 13], ['S','S','S','D','C'])) #3
    print(getGrade([ 7, 8, 6, 4, 5], ['S','S','S','D','C'])) #4
    print(getGrade([14, 8, 6, 4, 5], ['S','S','S','S','S'])) #5
    print(getGrade([13, 13, 14, 14, 13], ['D','D','D','D','D'])) #6
    print(getGrade([13, 13, 13, 14, 13], ['D','D','D','D','D'])) #7
    print(getGrade([12, 2, 13, 3, 14], ['S','S','S','S','S'])) #8

compute(['HQ', 'DK', 'CA', 'D2 ', 'S3'])
compute(['H5', 'D5', 'C5', 'D10', 'S5'])
```

# 反階梯數字

```
def cmp(x, y, xLen, yLen):  
    if (xLen>yLen):  
        return -1  
    for i in range(xLen-1,-1,-1):  
        if (x[i]>y[i]):  
            return -1;  
        elif ((x[i]<y[i])):  
            return 1  
    return 0
```

```
def add(d, xLen):  
    i=0  
    index=1  
    if d[1]>d[0]:  
        d[0] = d[0] + 1  
    return d, xLen  
    for index in range(2, xLen, 1):  
        if (d[index]>d[index-1]):  
            d[index-1]=d[index-1]+1  
            for i in range(0,index-1):  
                d[i]=0  
            return d, xLen  
        if d[xLen-1]==9:  
            d.append(1)  
            for i in range(0, xLen):  
                d[i] = 0  
            xLen = xLen +1  
            return d, xLen  
        d[xLen-1]=d[xLen-1]+1;  
        for i in range(0,xLen-1):  
            d[i] = 0  
        return d, xLen
```

# 反階梯數字

```
def test(inputX, A):
    inputX = [int(i) for i in inputX]
    A = [int(i) for i in A]
    i = count = 0
    xLen = yLen = 0
    y = [1,0]
    inputR = inputX[::-1]
    AR = A[::-1]
    target = sum(AR[1::2])
    xLen = len(inputR)
    yLen = len(y)
    while (cmp(inputR, y, xLen, yLen)<0):
        y, yLen = add(y, yLen)
        x = y[::-1]
        count = sum(x[::2])
        if (count == target):
            print(x)
        i = i + 1
```

```
def main():
    test('987654321','493584437776333')
    #test("9876543219876", "1234567891234");
main()
```

# 數字穿插

- 紿定一串數字，將數字重新排序，數字的左右兩側不能是自己。
- 輸出
  - 符合上述條件中，最小的數字。
  - 例如1 2 2 3，因左右兩側不能是自己，2的左右不能是2，符合條件:1232, 2132, 2123, 3212，
  - 其中最小數字，是1232。

# 數字穿插

```
def check(x, y):
    x = str(x)
    y = str(y)
    xLen = len(x)
    for i in range(xLen-1):
        if x[i]==x[i+1]:
            return 0
    for i in range(10):
        if x.count(str(i))!=y.count(str(i)):
            return 0
    return 1
```

```
def f(n, m):
    for i in range(n, m+1):
        if check(i, m)==1:
            print(i)
            break
```

```
#f(1122,2211)
f(11112233,33221111)
```

# 黃金分配

- $N$ 塊黃金( $N < 15$ )，每一塊黃金重 $1 \sim X$ 公克，平均分給A, B, C三個人，每人最多不得拿 $M$ 塊( $M < N$ )，有幾種分法。
  - $N=5$ ，重= $[1, 2, 3, 4, 5]$ ，每人最多拿2塊，6種分法
    - $[A, B, C] = \{([1, 4], [2, 3], [5]), ([1, 4], [5], [2, 3]), ([2, 3], [1, 4], [5]), ([2, 3], [5], [1, 4]), ([5], [2, 3], [1, 4]), ([5], [1, 4], [2, 3])\}$

# 黃金分配 - 迴圈版

```
def check(x, value, M):
    return (sum(x[0])==value and sum(x[1]) ==value and sum(x[2])==value
        and len(x[0])<=M and len(x[1])<=M and len(x[2])<=M)

def process(num, a, N, value, M):
    answer = [[],[],[]]
    for i in range(N):
        index = num%3
        num=num//3
        answer[index].append(a[i])
    if check(answer, value, M)==True:
        return answer
    else:
        return None

def push(result, answer):
    d=(tuple(sorted(answer[0])),tuple(sorted(answer[1])),tuple(sorted(answer[2])))
    result.add(d)
```

# 黃金分配- 迴圈版

```
def dist(a, M):
    result = set()
    N = len(a)
    value = sum(a)//3
    count = 1
    for i in range(N):
        count = count*3
    for i in range(count):
        answer=process(i, a, N, value, M)
        if answer!=None:
            push(result, answer)
    print('-'*30, '\nn=', len(result))
    for i in result: print(i)
```

```
#dist([1, 2, 3, 4, 5], 3)
#dist([1, 2, 4, 6, 8], 2)
#dist([2, 4, 6, 8, 10, 12], 2)
#dist([2, 4, 6, 8, 10, 12, 15], 3)
dist([1, 2, 3, 4, 5, 6, 7, 8, 9],4)
dist([9, 1, 2, 3, 4, 5, 6, 7, 8, 9],5)
```

# 黃金分配- 遞迴版

```
def dist(result, a, x, y, z, value, M):
    n = len(a)
    if n==0:
        if check(x, y, z, value, M):
            push(result, x, y, z)
        return True
    t = a[1:]
    dist(result, t, x+a[:1], y, z, value, M)
    dist(result, t, x, y+a[:1], z, value, M)
    dist(result, t, x, y, z+a[:1], value, M)
    return True

def test(a, M):
    result = set()
    x, y, z = [[], [], []]
    value = sum(a)//3
    if value*3 == sum(a): dist(result, a, x, y, z, value, M)
    print('-'*30, '\nn=', len(result))
    for i in result: print(i)
```

# 賓果遊戲 V1

## □ N人賓果遊戲

- 每位玩家各自輸入一個九宮格，九個數字 $n_1, n_2, \dots, n_9$ ，且 $n_i \neq n_j$  當  $i \neq j$ ，而  $1 \leq n_i, n_j \leq 9$ ， $1 \leq i, j \leq 9$ 。
- 電腦從1~9整數中選三個數字 $C_1, C_2, C_3$ ，且 $C_i \neq C_j$  當  $i \neq j$ ，而  $1 \leq C_i, C_j \leq 9$ ，且  $1 \leq i, j \leq 3$ 。
  - 這三個數字，只在一位玩家九宮格中，成一條水平線，如圖 $n_1, n_2, n_3$ 、垂直線如圖 $n_1, n_4, n_7$ ，或對角線如圖 $n_1, n_5, n_9$ ，

## □ 獲勝情況

- 第一位玩家贏，第二位玩家贏、和平手。

$n_1$	$n_2$	$n_3$
$n_4$	$n_5$	$n_6$
$n_7$	$n_8$	$n_9$

# 賓果遊戲V1

□ 將輸入資料轉成九宮格，對應到【0/1檢測矩陣】

- data 使用者設定的九宮格資料[n1, ..., n9]
- selected 挑選的數字
- mData 【0/1檢測矩陣】

0	0	1
1	1	1
1	0	1

```
def mapData(data, selected, N):
    mData = [[0 for i in range(N)] for j in range(N)]
    for x in selected:
        index = data.index(x)
        mData[index//N][index%N] = 1
    print(mData)
    return mData
```

# 初始化陣列  
# 對應挑選數字  
# 找出九宮格相對位置  
# 轉出設定對應0/1檢測矩陣  
# 回傳0/1檢測矩陣

# 賓果遊戲V1

## □ 判斷0/1檢測矩陣連線

- 0/1檢測矩陣，成一條水平線，如圖n1, n2, n3、垂直線如圖n1, n4, n7，或對角線如圖n1, n5, n9，

```
def checkBingo(data, N):    #0/1矩陣  
    count = 0  
    for i in range(N):        #對角線1  
        if data[i][i]==1:  
            count = count + 1  
    if count==N: return True  
  
    for i in range(N):        #對角線2  
        if data[i][N-i-1]==1:  
            count = count + 1  
    if count==N: return True
```

```
for i in range(N):          #3條水平線  
    count = 0  
    for j in range(N):  
        if data[i][j]==1:  
            count = count + 1  
    if count==N: return True  
  
for i in range(N):          #3條垂直線  
    count = 0  
    for j in range(N):  
        if data[j][i]==1:  
            count = count + 1  
    if count==N: return True  
return False
```

N1	N2	N3
N4	N5	N6
N7	N8	N9

0	0	1
1	1	1
1	0	1

# 賓果遊戲V1

## □ 測試資料

```
def test01():
    mData = [[1,0,1],
              [1,1,1],
              [0,1,0]]
    print(checkBingo(mData, 3))          #測試【判斷0/1檢測矩陣連線】
    data =[1, 3, 5, 7, 9, 2, 4, 6, 8]   #九宮格輸入
    mData = mapData(data, [1, 2, 3], 3)  #轉成【0/1檢測矩陣】
    print(checkBingo(mData, 3))          #【判斷0/1檢測矩陣連線】
```

1	0	1
1	1	1
1	0	1

# 賓果遊戲V2

- 檢測八條連線是否完成，**store** = [0, 0, 0, 0, 0, 0, 0, 0]
- 編號0, 1, 2 (0~N-1) row 水平線
- 編號3, 4, 5 (N~2\*N-1) column 垂直線
- 編號6, 7 (2\*N, 2\*N+1) 左上右下、右上左下
- 針對每一個選中的數字**num**，是否有在八條連線中，+1

```
def checkBingo2(data, store, num, N):  
    row, column = data.index(num)//N, data.index(num)%N #算出x, y 座標位置  
    store[row] += 1  
    store[column+N] += 1  
    if row == column: #符合左上右下連線  
        store[N*2] += 1  
    if row==(N-1-column): #符合右上左下連線  
        store[N*2+1] += 1
```

# 賓果遊戲V2

- 逐一檢測八條連線是否完成，
  - **store** = [0, 0, 0, 0, 0, 0, 0, 0]，紀錄八條連線狀態，任一數字 =N(3)，擇有連線成功
  - **data** 九宮格設定數字，N=3
  - **selected** 挑選的數字

```
def compute(data, selected, N):  
    store = [0 for i in range(2*N+2)]          # 初始化八條連線紀錄  
    for num in selected:  
        checkBingo2(data, store, num, N)         # 逐一檢測八條連線是否完成  
        print(store)  
        if N in store: return 'Bingo!'           # 任一數字=N(3)，擇有連線成功  
    return 'Sorry~'  
  
def test02():  
    data =[1, 3, 5, 7, 9, 2, 4, 6, 8]  
    print(compute(data, [1, 2, 3, 5], 3))
```

# 賓果遊戲V2

## □ AB兩人遊戲

```
def compute(dataA, dataB, selected, N, M):
    storeA = [0 for i in range(2*N+2)]
    storeB = [0 for i in range(2*N+2)]
    for num in selected:
        checkBingo(dataA, storeA, num, N)
        checkBingo(dataB, storeB, num, N)
        print(storeA, storeB)
    if (N in storeA) and (N in storeB):
        return       
    elif (N in storeA):
        return 'A Win'
    elif (N in storeB):
        return 'B Win'
    return 'Tie'
```

```
def test():
    inputData = input().split()
    N, M = int(inputData[0]), int(inputData[1])
    dataA = input().split()
    dataB = input().split()
    selected = input().split()
    print(compute(dataA,       , selected, N, M))
```

# 會議安排

## □ 問題

- 某公司有數間會議室 A, B, C, D..., 24小時開放各單位登記舉辦會議，每個會議有不同的使用時間。由於會議眾多，可能發生時間衝突，總務部門需從這些活動中，選取時間上不衝突的活動讓他們如願使用會議室。公司希望所有會議室能做最有效的利用：
  - 登記排出最多的加總使用時數。
  - 登記排出最多會議的使用數量。

## □ 演算步驟

- 計算所有活動組合，1, 2, 3, 12, 13, 23, 123。
- 計算任一組合活動中，是否可以安排進2間會議室。
- 計算任一組活動中，使用會議室的總時數。
- 計算任一組活動中，總活動數。
- 輸出所有活動組合，找出最大的指標(總時數或總活動數)

# 會議安排

## □ 輸入說明

- 第一行輸入兩個整數M, N，M是會議室間數，N是申請舉辦會議個數。
- 其後N行，每一行有三個整數，代表每一個申請會議的編號、開始時間、結束時間。開始與結束時間以8~18代表早上八點到下午六點。

## □ 輸出說明

- 從所有的申請會議中，計算「最長使用總時數」—輸出所有可使用會議室的會議編號，讓所有會議室的總使用時數最長。
- 從所有的申請會議中，計算「最多會議使用數。如果有多組解，依編號，由小至大依序列出所有解。

# 會議安排

## □ 測試資料

輸入範例	輸出範例
1	Selective Activities:room: 3, 7, 11 Total Utilization Hours:12
1,1,4	Selective Activities:room: 3, 9, 11 Total Utilization Hours:12
2,3,5	Selective Activities:room: 5, 9, 10, 11 Total Utilization Hours:12
3,0,6	Solutions with most utilized activities: Selective Activities:room: 2, 4, 9, 10, 11 Total Utilization Hours:11
4,5,7	Selective Activities:room: 2, 4, 8, 10, 11 Total Utilization Hours:10
5,3,8	
6,5,9	
7,6,10	
8,8,11	
9,8,12	
10,2,3	
11,12,14	
-1	
2	
1,1,4	
2,3,5	
3,0,6	
4,5,7	
5,3,8	
6,5,9	
7,6,10	
8,8,11	
9,8,12	
10,2,3	
11,12,14	
-1	

# 字元組合

□ 紿一字符串  $s$ ，求  $n$  個字元組合，輸入 'abcde', 2, ['ab', 'ac', ...]

○ 假設  $f(s, n)$  可以解問題，則問題可以分解成

○  $['a' + f('bcde', n-1)] + [" + f('bcde', n)]$

□ 遞迴  $f(s, n)$

○ 結束條件，回傳 list(因為會有許多組合)

➤  $n == 0$ ，回傳空字串(list)

➤  $n == \text{len}(s)$ ，回傳  $s$ (list)

○ 一般條件

➤  $s[0]$  要取，回傳： $s[0] + f(s[1:], n-1)$ ，

– 這一層已經取一個字元， $n-1$ ，字串少第一個字元  $s[1:]$

– 要把取的字元  $s[0]$  加進答案

➤  $s[0]$  不要取，回傳： $f(s[1:], n)$ ，

– 這一層不取任一個字元， $n$ ，字串少第一個字元  $s[1:]$

# 字元組合

## □ Code

```
def findCombi(data, n):
    if n==len(data): return [data]
    elif n==0: return []
    s0 = findCombi(data[1:], n)
    s1 = [data[0]+s for s in findCombi(data[1:], n-1)]
    return sorted(s0+s1)

print('sol=',findCombi('abcde', 5))
print('sol=',findCombi('abcde', 4))
print('sol=',findCombi('abcde', 3))
print('sol=',findCombi('abcde', 2))
print('sol=',findCombi('abcde', 1))
```

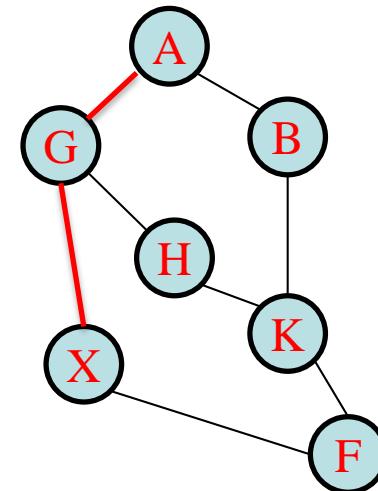
```
sol= ['abcde']
sol= ['abcd', 'abce', 'abde', 'acde', 'bcde']
sol= ['abc', 'abd', 'abe', 'acd', 'ace', 'ade', 'bcd', 'bce', 'bde', 'cde']
sol= ['ab', 'ac', 'ad', 'ae', 'bc', 'bd', 'be', 'cd', 'ce', 'de']
sol= ['a', 'b', 'c', 'd', 'e']
```

# 數字組合

- 紿定n個0~9數字，任取其中m個數，求可以組合出幾個整數，其中，那些數，每一位數字加總可以被3整除
  - 2, 3, 5, 7, 8, 9，任取4數

# 部落旅遊

- 小明要到沙哈拉沙漠中的各部落旅遊，某些部落之間有安全通道可以走。
  - 紿定N個部落間有通道資料，請問要從A部落到X部落，最少需經過幾個通道
    - 通到資料，A B, G H, A G, H K, K F, F X, G X, B K
- 建立地圖(dict, 點:相鄰點集合)
  - {A:[B, G], B:[K], G:[A, H, X], H:[K], K:[B, H, F], F:[K, X], X:[F, G]}
- 搜尋最少通道數A-X
  - A-B-K-F-X =>4
  - A-G-H-K-F-X=>5
  - A-G-X =>2
- 找出路徑 A-G-X



# 部落旅遊

## □ 建立地圖(dict, {點:[相鄰點集合/鄰居], ...})

```
def addNeighbour(data, x, y):
    neighbour = data[x] if x in data.keys() else []
    # 若x 節點存在則取出x的鄰居，否則[]
    if y not in neighbour: neighbour.append(y)
    # 加入 x 鄰居 y
    data[x]=neighbour
    # 加入 x 節點，和新的鄰居

def addPair(data, pair):
    addNeighbour(data, pair[0], pair[1])
    # 加入 0 節點 1 鄰居
    addNeighbour(data, pair[1], pair[0])
    # 加入 1 節點 0 鄰居

def test01():
    data=dict()
    x = [['A','B'],['G','H'],['A','G'],['H','K'],['K','F'],['F','X'],['G','X'],['B','K']]
    for pair in x: addPair(data, pair)
    print(data)
```

```
{'A': ['B', 'G'], 'B': ['A', 'K'], 'G': ['H', 'A', 'X'], 'H': ['G', 'K'], 'K': ['H', 'F', 'B'], 'F': ['K', 'X'], 'X': ['F', 'G']}
```

# 部落旅遊

## □ 廣度搜尋，最少通道數A-X

```
def findNeighbour(data, source, target):
    passedNodes = dict()                                # 拜訪過的節點，{節點:level}
    stack = [[source,0]]                                # 待拜訪節點，[開頭點:levle=0]
    while True:
        if len(stack)==0: return -1
        currentNode = stack.pop(0)
        currentNodeName = currentNode[0]
        level = currentNode[1]
        if currentNodeName==target:                      #待拜訪節點，找不到
            #path = findPath(data, passedNodes, currentNodeName, level-1)   #找路徑
            return level
        passedNodes[currentNodeName]=level               #取出一個待拜訪節點
        for node in data[currentNodeName]:               #取出該節點名稱
            if node not in passedNodes.keys():           #取出該節點level
                stack.append([node, level+1])             #找到最終節點
                #path.append(node)
                #回傳經過幾層
                #不是最終節點，存入(拜訪過節點)
                #針對每一個相鄰節點，存入(待拜訪節點)
                #只存(未拜訪過節點)

def test01():
    data=dict()
    x = [['A','B'],['G','H'],['A','G'],['H','K'],['K','F'],['F','X'],['G','X'],['B','K']]
    for pair in x: addPair(data, pair)
    print(findNeighbour(data, 'A', 'X'))               # 2
    print(findNeighbour(data, 'F', 'A'))               # 3
```

# 部落旅遊findPath(v1)

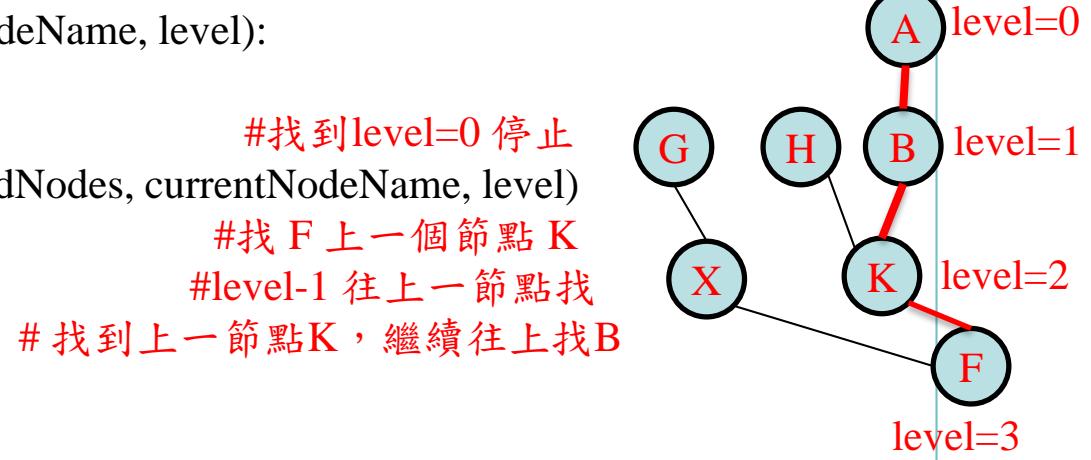
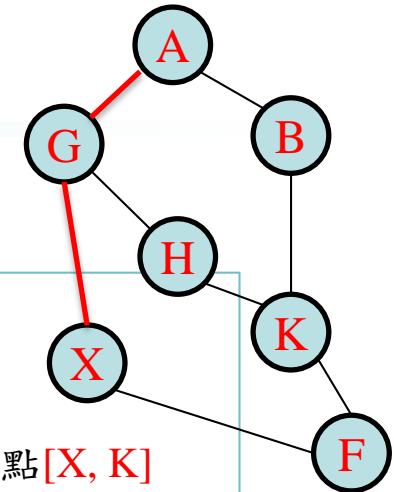
□ 從目標節點倒回去搜尋，F(0)-K(1)-B(2)-A(3)

```

def findPrevNode(data, passedNodes: dict, targetnodeName, level):#找上一層
    nodeCandidate = []
    for nodeName, nodeLevel in passedNodes.items():
        if nodeLevel==level: nodeCandidate.append(nodeName)
    for nodeName in nodeCandidate:
        if targetnodeName in data[nodeName]: return nodeName # K 相鄰節點是 F，回傳K

def findPath(data, passedNodes, currentNodeName, level):
    path = []
    while level>=0: #找到level=0 停止
        nodeName=findPrevNode(data, passedNodes, currentNodeName, level) #找 F 上一個節點 K
        path.append(nodeName) #level-1 往上一節點找
        currentNodeName = nodeName # 找到上一節點K，繼續往上找B
    print(path)
    return path

```



# 部落旅遊findPath(v2)

□ 從目標節點倒回去搜尋， F(0)-K(1)-B(2)-A(3)

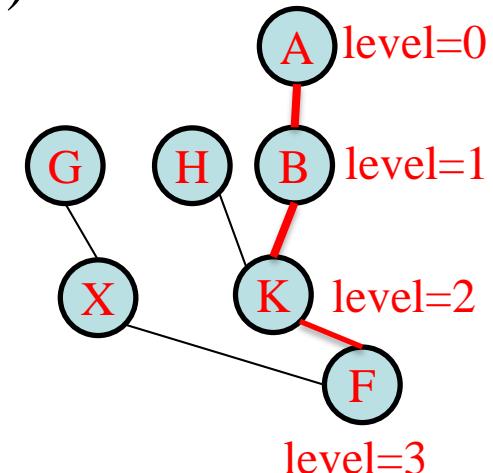
○ 04，F的相鄰點list()，`data[currentNodeName]`

○ 和走過的節點list()，`list(passedNodes.keys())`

○ 變成集合set()

○ 交集&

○ 轉成串列list()，只有一個元素[0]



```
01 def findPath(data, passedNodes, currentNodeName, source):  
02     path = []  
03     while currentNodeName!=source:                      # 從 F 一直找到原始點 A 停止  
04         currentNodeName=list(set(data[currentNodeName])&set(list(passedNodes.keys())))[0]  
05         path.append(currentNodeName)  
06     print(path)  
07     return path
```

□ 假設中途一定要到某部落旅遊，則最少要如何走

# Python 字串操作

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 字串格式化

- `format()`，使用 {} 放參數

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
myorder = "I want to pay {2} dollars for {0} pieces of item {1}." 
print(myorder.format(quantity, itemno, price))
```

I want 3 pieces of item 567 for 49.95 dollars.

I want to pay 49.95 dollars for 3 pieces of item 567.

# 字串格式化

## □ str.ljust(length, [char])

- 將字串向靠左對齊

- []可選的參數

- char是預設空格，補齊字元

## □ str.rjust(length, [char])

- 將字串向靠右對齊

## □ str.center(length, [char])

- 將字串向中間對齊

```
s="banana"
```

```
s1=s.rjust(20)
```

```
s2=s.ljust(20)
```

```
s3=s.center(20)
```

```
print(s1, "is my favorite fruit.")
```

```
print(s2, "is my favorite fruit.")
```

```
print(s3, "is my favorite fruit.")
```

```
print(s.rjust(20, "O"))
```

```
print(s.ljust(20, "O"))
```

```
print(s.center(20, "O"))
```

banana is my favorite fruit.

banana is my favorite fruit.

banana is my favorite fruit.

oooooooooooooooObanana

bananaoooooooooooooo

ooooooooObananaooooooo

# 字串格式化

## □ 三種格式化 (%-formatting, str.format, f-string)

- '%s'，後面接 tuple 資料型別，轉成字串資料型別

```
a, b, d = 2, 3, 'ans'
```

```
s = d + ':' + str(a) + '/' + str(b) + '=' + str(round(a/b, 3))
```

```
print(s)
```

```
x = '%s: %d/%d =%.3f' % (d, a, b, a/b)
```

```
print(x)
```

```
y = '{d1}:{a1}/{b1}={c1:.3f}'.format(a1=a, b1=b, c1=a/b, d1=d)
```

```
print(y)
```

```
z = f'{d}:{a}/{b}={a/b:.3f}'
```

```
print(z)
```

```
ans: 2/3 =0.667
```

```
ans: 2/3 =0.667
```

```
ans: 2/3 = 0.667
```

```
ans: 2/3 = 0.667
```

```
a = list('Apple')
```

```
print(a)
```

```
content = '%s'*len(a) % tuple(a)
```

```
# len(a) =5
```

```
# '%s'*5  '%s %s %s %s %s'
```

```
# '%s %s %s %s %s' % ('A', 'p','p', 'T', 'e')
```

```
print(content)
```

```
['A', 'p', 'p', 'T', 'e']
```

```
Apple
```

# 字串格式化

○^ (居中)、<(左對齊)、>(向右對齊)、{:,}(分隔數字)調整輸出

```
d = 'ans'  
e = 'US$'  
a = 980000  
x = '{d1:^10}={a1:<10,d}-{e1:>10}'.format(a1=a, d1=d, e1=e)  
print(x)
```

```
ans =980,000 - US$
```

# 字串重整

## □ string.strip([chars])

- 將string字串變數裡的左右兩邊空白字元刪除掉
- chars參數可決定要刪除的字元

```
t1 = "      aaaaaa      bbbbbbb  aaa      ccccc "
t2 = "aaaaaabbbbbbb  aaa      ccccc aaaaaa"
print(t1.strip())
print(t2.strip('a'))
```

```
aaaaaa  bbbbbbb  aaa  ccccc
bbbbbb  aaa  ccccc
```

## □ string.lstrip([chars]): 左邊空白字元刪除

## □ string.rstrip([chars]): 右邊空白字元刪除

- string.rstrip('c'): 右邊'c'字元去掉，直到空白

```
s1 = "      aaaaaa      bbbbbbb  aaa ccccc cc"
s2 = "      aaaaaa      bbbbbbb  aaa ccccc "
print(s1.lstrip())
print(s2.rstrip())
print(s1.rstrip('c'))
```

```
aaaaa  bbbbbbb  aaa ccccc cc
aaaaa  bbbbbbb  aaa ccccc
aaaaa  bbbbbbb  aaa ccccc
```

# 字串重整

## □ string.zfill(width)

- 將string變數內字串前補0，直到string變數的長度等於width參數設定的長度

```
s = "50"  
print(s.zfill(3))  
print(s.zfill(10))  
print(s.zfill(0))
```

```
050  
0000000050  
50
```

# 字串搜尋

## □ string.count(sub[, start[, end]])

- 回傳此字串裡有多少個sub字元

```
text='abbggccdeefgggijklggimo'  
print(text.count('g'))  
print(text.count('g',4,-4))
```

7  
5

```
images="xbox.gif, iphone.jpg"  
print(images.startswith("xbox"))  
print(images.startswith(".gif"))  
print(images.startswith("iphone",10, 20))
```

True  
False  
True

## □ str.startswith(prefix[, start[, end]])

- 判斷傳入的prefix字串字元是否為開始字元

## □ str.endswith(suffix[, start[, end]])

- 判斷字串內是否有符合suffix引數的值

```
images="xbox.gif, iphone.jpg"  
print(images.endswith(".jpg"))  
print(images.endswith(".gif"))  
print(images.endswith(".gif",0, 8))
```

True  
False  
True

# Exercise

## □ 計算字串有多少字元和空白

```
s='Given a string and count how many characters and spaces in the  
string'  
print(len(s))  
words=s.split(' ')  
print(words)
```

# Exercise

## □ 計算字串的字元數

- Sample String : 'google.com'
- Expected Result : {'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}
- Sample string: 'thequickbrownfoxjumpsoverthelazydog'
- Expected output :

- o 4
- e 3
- u 2
- h 2
- r 2
- t 2

```
inputStr = "google.com"
countDict = { }
for char in inputStr:
    countDict[char]=inputStr.[ ](char)
print(countDict)
```

# 字串搜尋

## □ sring.rfind(sub[,start[, end]])

- 從右到左尋找，sub預計要搜尋的字元

## □ string.find(sub[, start[, end]])

- 從左到右尋找

- 搜尋字串變數裡符合sub的字元位置

- 找不到回傳-1.

## □ string.rindex(sub[, start[, end]])

- 由右至左搜尋，s字串變數搜尋不到sub字串將會回傳  
ValueError錯誤訊息

```
s = "Mi casa, su casa."  
print(s.rindex("casa"))  
print(s.rindex("casa",2,10))  
#print(s.rindex("haha"))
```

12  
3

```
s = "Mi casa, su casa."  
print(s.find("casa"))  
print(s.find("haha"))  
print(s.rfind("casa"))  
print(s.find("casa",5))
```

3

```
text='abcdefgabcdefg'  
print(text.find('a'))  
print(text.find('a',1))  
print(text.rfind('e'))
```

0  
7  
11

# Exercise

- 找出字串中 “USA” 個數，大小寫視為相同
  - input\_str = "Welcome to USA. usa awesome, isn't it?"
  - Expected outcome: The USA count is: 2

```
inputString = "Welcome to USA. usa awesome, isn't it?"  
substring = "USA"  
tempString = inputString.lower()  
count = tempString.count(substring.lower())  
print("The USA count is:", count)
```

- 找到 "Hello, World"字串中第一個和最後一個'o' 和 ','位置

```
s = "Hello, World"  
print(s.f("o"))  
print(s.r("o"))  
print(s.f(","))  
print(s.r(","))
```

4  
8  
5  
5

# 字串轉換

## □ string.translate(map)

○ 將 string 中的字元以 map 中配對的字元轉換，

➤ 搭配map=str.maketrans(from, to)

```
th3s 3s str3ng 2x1mpl2....4!!!  
th3s 3s str3ng 2x1mpl2....w4w!!!
```

```
fromStr = "aeiou"  
toStr = "12345"  
str = "this is string example....wow!!!"  
map=str.maketrans(fromStr, toStr)  
#map=str.maketrans(fromStr, toStr,'w')  
print(str.translate(map))
```

```
dict= {"a": "123", "b": "456", "c": "789"}  
string = "abcdef"  
map=string.maketrans(dict)  
print(string.translate(map))
```

```
123456789def
```

# Exercise

- 找字串中最大和最小長度的word
- 反轉字串中的 words
- 輸入逗號隔開的一串 word，輸出排序好不重複的 word
  - Sample Words : red, white, black, red, green, black
  - Expected Result : black, green, red, white

```
s = input('enter a string: ')
print(s[0:-1])
words=s[0:-1].split()
for word in words[0:-1]:
    print(word, end=' ')
```

# Exercise

## □ 找基因序列

- 若DNA序列由A, C, G, T四個字元組成的字串(string) ,
- 基因(gene)是隱藏於DNA序列中的部分片段(子字串)。
- 紿定一DNA序列(長度小於50) , 找出在裡面的基因；規則：
  - 1.前面是ATG，後面接TAG、TAA、或TGA；  
– 例如ATGTTTTAA。
  - 2.長度為3的倍數，其中未含有ATG、TAG、TAA或TGA。
- 例如，給定DNA序列，其中包含兩個基因，  
**CCATGTTTTAACCACTGCCTAAATGGGGCGTTAGTT**：
  - 基因TTT前面為ATG，後面接著TAA，長度3，其中未含ATG、TAG、TAA或TGA。
  - 基因GGGCGT前面為ATG，後面接著TAG，長度6，其中未含ATG、TAG、TAA或TGA。
- 撰寫程式輸入DNA序列，找出該序列中所有基因。

# Exercise

```
def check(gen):
    if len(gen)==0:                      #空的基因
        return False
    for tag in ['ATG', 'TAG','TAA','TGA']: #不能含有這些tag
        if gen.find(tag)>-1:
            return False
    if len(gen)%3==0:                     #長度3倍數
        return True
    return False
```

# Exercise

```
def findGen(dna):
    startTag, endTags = 'ATG', ['TAG','TAA','TGA']
    length = len(dna)
    startIndex=dna.find(startTag)+3          #基因前面是 startTag 'ATG'
    endIndex=length+1                        #初始化結束點
    for tag in endTags:                     #從三個 end tag 找出最小符合基因字串
        endTemp=dna.find(tag, startIndex)
        if endTemp!=-1 and endTemp<endIndex: #找最小符合的 end tag 的點
            endIndex = endTemp
    if endIndex==length+1 or startIndex<3: #找不到前後tag
        return 0, 0, 'None'
    gen = dna[startIndex:endIndex]           #找到基因
    if check(gen):                         #驗證基因
        return 1, endIndex+3, gen
    return 2, startIndex+3, 'None'
```

# Exercise

```
def finAllGen(dna):
    i, count = 0, 0
    print('==>')
    while (True):
        dna=dna[i:]          #往後找基因
        b, i, gen = findGen(dna) #b=1找到, i往後找索引
        if (b==1):
            print(gen)
            count=count+1      #找到幾個基因
        elif (b==0):           #找到最後沒找到
            break
        if count==0:            #完全沒找到
            print('沒有基因')
```

```
finAllGen('CCATTTTAACCATGCCTAAATGGGGCGTTAGTT')
```

```
finAllGen('TAAGATGAATGA')
```

```
finAllGen('ATGAAATGA')
```

```
finAllGen('ATGTGAATGAAATGA')
```

```
finAllGen('TTATGTTAAAAGGATGTTAATGTAAGGGCGTTAGTT')
```

```
finAllGen('AATAGATGTTAACGTGATATGGGGATGTCATAGATGCCCTCACCTAA') 18
```

```
==>
TTT
GGCGT
==>
沒有基因
==>
AAA
==>
AAA
==>
沒有基因
==>
TCA
CCCTTCACC
```

# 附錄

# 字串函數 1/3

Method	Description
<a href="#"><u>capitalize()</u></a>	Converts the first character to upper case
<a href="#"><u>casefold()</u></a>	Converts string into lower case
<a href="#"><u>center()</u></a>	Returns a centered string
<a href="#"><u>count()</u></a>	Returns the number of times a specified value occurs in a string
<a href="#"><u>encode()</u></a>	Returns an encoded version of the string
<a href="#"><u>endswith()</u></a>	Returns true if the string ends with the specified value
<a href="#"><u>expandtabs()</u></a>	Sets the tab size of the string
<a href="#"><u>find()</u></a>	Searches the string for a specified value and returns the position of where it was found
<a href="#"><u>format()</u></a>	Formats specified values in a string
<a href="#"><u>format_map()</u></a>	Formats specified values in a string
<a href="#"><u>index()</u></a>	Searches the string for a specified value and returns the position of where it was found
<a href="#"><u>isalnum()</u></a>	Returns True if all characters in the string are alphanumeric
<a href="#"><u>isalpha()</u></a>	Returns True if all characters in the string are in the alphabet
<a href="#"><u>isdecimal()</u></a>	Returns True if all characters in the string are decimals

# 字串函數 2/3

<a href="#"><u>isdigit()</u></a>	Returns True if all characters in the string are digits
<a href="#"><u>isidentifier()</u></a>	Returns True if the string is an identifier
<a href="#"><u>islower()</u></a>	Returns True if all characters in the string are lower case
<a href="#"><u>isnumeric()</u></a>	Returns True if all characters in the string are numeric
<a href="#"><u>isprintable()</u></a>	Returns True if all characters in the string are printable
<a href="#"><u>isspace()</u></a>	Returns True if all characters in the string are whitespaces
<a href="#"><u>istitle()</u></a>	Returns True if the string follows the rules of a title (i.e., Check if each word start with an upper case letter)
<a href="#"><u>isupper()</u></a>	Returns True if all characters in the string are upper case
<a href="#"><u>join()</u></a>	Joins the elements of an iterable to the end of the string
<a href="#"><u>ljust()</u></a>	Returns a left justified version of the string
<a href="#"><u>lower()</u></a>	Converts a string into lower case
<a href="#"><u>lstrip()</u></a>	Returns a <b>left trim</b> version of the string
<a href="#"><u>maketrans()</u></a>	Returns a translation table to be used in translations
<a href="#"><u>partition()</u></a>	Returns a tuple where the string is parted into <b>three parts</b>
<a href="#"><u>replace()</u></a>	Returns a string where a specified value is replaced with a specified value

# 字串函數 3/3

<u>rfind()</u>	Searches the string for a specified value and returns the last position of where it was found
<u>rindex()</u>	Searches the string for a specified value and returns the last position of where it was found
<u>rjust()</u>	Returns a right justified version of the string
<u>rpartition(sep)</u>	以 sep 從最右端分割 str 為三個部份，結果回傳具有三個子字串的序對
<u>rsplit([sep[,maxsplit]])</u>	將 str 從最右端以 sep 分割成子字串，回傳儲存子字串的串列 maxsplit 為子字串最多的數量
<u>rstrip([chars])</u>	從 str 的最右端中移除 chars 字元，預設為空白字元
<u>split([sep[, maxsplit]])</u>	將 str 以 sep 分割成子字串，回傳儲存子字串的串列， maxsplit 為子字串最多的數量
<u>splitlines([keepends])</u>	將 str 以新行符號分割成子字串，回傳儲存子字串的串列
<u>startswith(prefix[, start[, end]])</u>	判斷 str 是否以 prefix 開頭
<u>strip([chars])</u>	從 str 中移除 chars 字元，預設為空白字元
<u>swapcase()</u>	將 str 中的英文字母進行大小寫轉換
<u>title()</u>	Converts the first character of each word to upper case
<u>translate()</u>	Returns a translated string
<u>upper()</u>	將 str 的英文字母都改成大寫
<u>zfill(width)</u>	回傳以 0 填滿 width 的新字串

# 跳脫字元

□ |

Code	Result	Example	Result
\'	Single Quote	txt = 'It's alright.' print(txt)	It's alright.
\\"	Backslash	txt = "This will insert one \\ (backslash)." print(txt)	This will insert one \ (backslash).
\n	New Line	txt = "Hello\nWorld!" print(txt)	Hello World!
\r	Carriage Return	txt = "Hello\rWorld!" print(txt)	Hello World!
\t	Tab	txt = "Hello\tWorld!" print(txt)	Hello World!
\b	Backspace	txt = "Hello \bWorld!" print(txt)	HelloWorld!
\f	Form Feed		forces the printer to eject the current page and to continue printing at the top of another
\ooo	Octal value	txt = "\110\145\154\154\157" print(txt)	Hello
\xhh	Hex value	txt = "\x48\x65\x6c\x6c\x6f" print(txt)	Hello

python2中，字串分作  
**unicode**及 **str**兩種物件，  
分別對應到 **Unicode**以  
及**編碼狀態**

# string與byte轉換

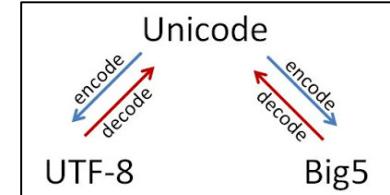
- Python3 中，字串分 **str** 及 **byte** 兩種物件，分別對應 **Unicode** 及 **編碼狀態**。

- 建立字串實例如 "中文" 時，是 Unicode 文字。

- 在字串前加一個英文字母 u，如 u"中文"，直接建立 unicode 物件實例

```
### input ###
print(type("中文"))
print(type("中文".encode("utf-8")))
print(type(u"中文"))
print(len("中文"))
```

```
<class 'str'>
<class 'bytes'>
<class 'str'>
2
```

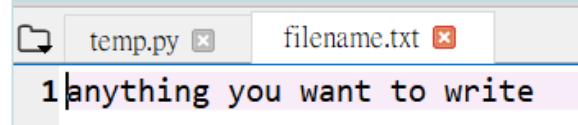


- 讀檔寫檔時，建立 I/O 實例透過參數encoding 編碼。

python2的 I/O 物件並沒有支持 encoding 這個參數

```
### input ###
with open("filename.txt",'w',encoding='utf-8') as outfile:
    outfile.write("anything you want to write")
with open("filename.txt",'r',encoding='utf-8') as infile:
    text = infile.read()
print(type(text))
```

```
<class 'str'>
```



# string與byte轉換

## □ 字元編碼

- 有ASCII、Unicode(\uXXXX)、GB2312(資訊交換用漢字編碼字元集)、GBK(漢字內碼擴展規範)、GB 18030、Big5等，不同編碼間可以互相轉換。
- UTF-8是Unicode的一種電腦儲存實現方式，即字元編碼格式。UTF-8一般用於網路傳輸。

## □ Python

- 二進制資料由 bytes型別表示
- 文字/字串使用Unicode，由 str型別表示

## □ str 跟 bytes 可互相轉換

- str.encode() 預設編碼 utf-8

str.encode() ⇔ bytes.decode()  
(Unicode) (utf-8)

# string與byte轉換

```
a = bytes([1,2,3,4,5,6,7,8,9])
b = bytes('python', 'ascii')
print(type(a)) # <class 'bytes'>
print(type(b)) # <class 'bytes'>
print(a)      # b'\x01\x02\x03\x04\x05\x06\x07\x08\t'
print(b)      # b'python'
```

```
s = u'\u4eba\u751f\u82e6\u77ed\uff0cpy\u662f\u5cb8'
print(type(s)) # <class 'str'>
print(s)      #人生苦短，py是岸
s_utf8 = s.encode(encoding='utf-8')
print(s_utf8)
#b'\xe4\xba\xba\xba\xe7\x94\x9f\xe8\x8b\xab\xe7\x9f\xad\xef\xbc\x8cpy\xe
6\x98\xaf\xe5\xb2\xb8'
```

# string與byte轉換

```
# bytes轉字符串方式一  
b=b'\xe9\x80\x86\xe7\x81\xab'  
string=str(b,'utf-8')  
print(string)
```

```
# bytes轉字符串方式二  
b=b'\xe9\x80\x86\xe7\x81\xab'  
string=b.decode() # 第一參數預設utf8，第二參數預設strict  
print(string)
```

```
# bytes轉字符串方式三  
b=b'\xe9\x80\x86\xe7\x81haha\xab'  
string=b.decode('utf-8','ignore') # 忽略非法字符，用strict會拋出異常  
print(string)
```

```
# bytes轉字符串方式四  
b=b'\xe9\x80\x86\xe7\x81haha\xab'  
string=b.decode('utf-8','replace') # 用？取代非法字符  
print(string)
```

# string與byte轉換

```
# 字符串轉bytes方式一  
str1='逆火'  
b=bytes(str1, encoding='utf-8')  
print(b)
```

```
# 字符串轉bytes方式二  
b=str1.encode('utf-8')  
print(b)
```

```
b'\xe9\x80\x86\xe7\x81\xab'  
b'\xe9\x80\x86\xe7\x81\xab'
```

# Python File (Simple)

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 開啟檔案

□ <file> = open(<filename>, <mode>)

- <file> : 開啟檔案物件
- <filename> : 檔名
- <mode> : 開檔模式
  - r: 唯讀
  - w (write): 覆蓋寫入內容
  - a (append): 在後加入新內容

```
infile = open('hello.txt', 'r') # 以讀取模式開檔  
outfile = open('hello.txt', 'w') # 以寫入模式開檔
```

<b>r</b>	開啟檔案只讀取
<b>rb</b>	開啟檔案只讀取格式為binary
<b>r+</b>	開啟檔案可讀取寫入
<b>rb+</b>	開啟檔案可讀取寫入binary
<b>w</b>	開啟檔案只寫入
<b>w+</b>	可讀取可寫入模式
<b>wb</b>	開啟檔案只寫入二進位格式
<b>a</b>	只允許新增
<b>a+</b>	允許新增與讀取

# 關閉檔案

- <file>.close() #將緩衝區內容寫入檔案，關閉檔案
  - 檔案讀寫可能產生IOError，而不會呼叫fp.close()造成關檔錯誤。
  - 為保證是否出錯都正確關閉檔案，可使用try ... finally

```
try:  
    fp = open('hello.txt', 'r')  
    print(fp.read())  
    fp.write('test')  
except FileNotFoundError:  
    print("file not found")  
except:  
    print("Something wrong")  
finally:  
    fp.close()
```

- 用with開檔，若錯誤，檔案仍會自動關閉，
  - 執行return, continue, break跳出with指令區塊
  - 發生例外(Exception)

```
with open('hello.txt', 'r') as fp:
```

# 讀取檔案

- <file>.read()
  - 讀取全部或剩餘資料，回傳長字串
- <file>.readline()
  - 讀取下一行資料，回傳字串
  - 利用迴圈一次讀一行資料

```
with open('filename.txt', 'r') as infile:  
    while True:  
        data = infile.readline()      # 一次讀一行資料  
        print(type(data))  
        print(data)  
        if not data:                # 所有資料讀取完畢  
            break  
        print(line, end="")         # end="": 不要自動加斷行
```

# 讀取檔案

## □ <file>.readlines()

- 讀取全部或剩餘資料，
- 回傳 **串列**，每個元素都是一行資料

```
with open('filename.txt', 'r') as fp:  
    data2 = fp.readlines()  
    print(type(data2))  
    print(data2)
```

```
with open('filename.txt', 'r') as infile:  
    for line in infile.readlines(): # 一次讀取所有資料，再一行一行處理  
        print(line, end="")
```

```
# Python 讀檔將每行資料存到串列中的元素，上述程式可簡化  
with open('filename.txt', 'r') as infile:  
    for line in infile:  
        print(line, end="")
```

# 寫檔 4 個方法

## □ write

```
# 開啟檔案  
fp = open("filename.txt", "a")  
# 寫入 This is a testing! 到檔案緩衝區  
fp.write("This is a testing!")  
# 將緩衝區寫入檔案，關閉檔案  
fp.close()
```

## □ writelines

```
# 開啟檔案  
fp = open("filename.txt", "w")  
# 將 lines 所有內容寫入到緩衝區  
lines = ["One\n", "Two\n", "Three\n"]  
fp.writelines(lines)  
# 將緩衝區寫入檔案，關閉檔案  
fp.close()
```

## □ print

```
# 開啟檔案  
fp = open("filename.txt", "a")  
# 寫入 This is a testing! 到檔案緩衝區  
print("This is a testing!\n", file=fp)  
# 將緩衝區寫入檔案，關閉檔案  
fp.close()
```

## □ with open

```
with open(filename, 'r') as in_file:  
    with open("filename.txt", 'a') as out_file:  
        for line in in_file:  
            out_file.write(line)
```

# Exercise

- 利用迴圈一次讀一行資料，將偶數行資料印出

```
with open('filename.txt', 'r') as infile:  
    line_num=0  
    for line in i  
        line_num+=1  
        if line_num%  
            print(line, end="")
```

- 一次讀取、印出多行資料，將全部資料的第一個字與最後一個字印出

```
fp = open('hello.txt', 'w')  
fp.write("First line\n#Second line\n#Third")  
fp.close()  
with open('hello.txt', 'r') as infile:  
    data = infile.r  
    print(data)  
    print(len(data))  
    print(data[0], c)
```

# 讀寫檔案

- 顯示檔案所有行，忽略以#開頭的行

```
with open("./hello.txt") as f:  
    for line in f:  
        if line.strip()[0] != "#":  
            print(line)
```

- 把/passwd檔案中'root'字串用'west'替換，另存tmp檔案

```
with open("passwd.txt") as f1:  
    # 遍歷檔案的每一行內容；  
    for line in f1:  
        # 字串替換  
        bline = line.replace("root", "west")  
        with open("tmp", "a+") as f2:  
            # 寫入新檔案  
            f2.write(bline)
```

root	word
user	pass

west	word
user	pass

# Exercise

- 把./passwd檔案中 xi 字串用 yi 替換，另存tmp檔案
  - $X = ['x1', 'x2', 'x3']$ ,  $Y = ['y1', 'y2', 'y3']$

```
import copy
x=["x1","x2","x3"]
y=["y1","y2","y3"]

with open('passwd') as f1:
    # 遍歷檔案的每一行內容 ;
    for line in f1:
        bline=copy.copy(line)
        # 字串替換
        for i in range(1,3):
            print(x[i],y[i])
            bline = bline.replace("x"+str(i),"y"+str(i))
    with open("tmp", "a+") as f2:
        # 寫入新檔案
        f2.write(bline)
```

# 讀取CSV檔案

## □ 一列一列的讀取出csv資料

```
import csv  
#f= open('data.csv', encoding='utf-8')  
f= open('data.csv')  
readFile = csv.reader(f)  
for row in readFile:  
    print(row)  
f.close()
```

## □ 使用with開啟csv檔案

- 加上 newline=""，為讓資料中包含的換行字元可正確解析

```
import csv  
with open('data.csv', newline='') as csvfile:  
    readFile = csv.reader(csvfile)  
    for row in readFile:  
        print(row)
```

# 讀取CSV檔案

## □ 指定分隔字元

- 資料欄位分隔字元非使用預設逗號，而是其他字元，讀取時要指定分隔字元

```
import csv
with open('data.csv', newline='') as csvfile:
    readfile = csv.reader(csvfile, delimiter=':')
    for row in readfile:
        print(row)
```

## □ 讀取成 Dictionary

- 讀取csv 檔案內容後，轉為dictionary 格式
- **csv.DictReader()**自動把第一列(row)當作欄位名稱，第二列後的每一列轉為 dictionary，如此可使用欄位名稱存取資料

```
import csv
with open('D:\\Courses\\Python\\data.csv', newline='') as csvfile:
    readfile = csv.DictReader(csvfile)
    for row in readfile:
        print(row['班級'], row['學號'], row['期中考成績'])
```

# Exercise

## □ 讀取檔案中 xi 字串被 yi 替換，另存tmp檔案

- $X = [xi]$ ,  $Y = [yi]$  , X是英文，Y是中文翻譯
- X, Y 分別存在 data.csv 檔案的第一 row 和第三 row，第二和第四 row 是備註

```
import csv
def getHeader(): #讀取檔案第0行和第2行
    i=0
    f= open('data.csv')
    readFile = csv.【】
    for row in readFile:
        if i==0:
            eng=row
        if i==2:
            chi=row
        i=i+1
    f.【】
    print(eng)
    print(chi)
    return eng, chi
getHeader()
```

# Exercise Solution

```
def convert(aFile, bFile, eng, chi):
    f1 = open(aFile)
    f2 = open(bFile, 'w', encoding='utf-8-sig')
    data = f1.read()
    print(type(data))
    print(data)
    bline = data
    #zip 可將eng, chi對應的元素打包成一個個tuple，回傳 tuples 組成的list
    for e, c in zip([REDACTED]):
        bline = bline.[REDACTED]
    #    if e in data:
    #        bline = bline.replace(e, c)
    print(bline, '#####')
    f2.write(bline)
    f1.close()
    f2.close()

eng, chi = getHeader()
convert('d:\Courses\Python\data.csv', 'data.csv', eng, chi)
```

# 寫入CSV檔案

## □ 一次寫入二維表格

- 若資料是已整理好二維表格，可一次把整張表格寫進 csv 檔案

```
import csv
# 二維表格
table = [['班級', '學號', '成績'],
          ['資工一', '109590001', 90],
          ['資工一', '109590002', 85]]
with open('output.csv', 'w', newline="") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerows(table) # 寫入二維表格
```

## □ 寫入 Dictionary

- 資料格式是 dictionary，可使用 `csv.DictWriter()` 寫入 csv 檔案中

```
import csv
with open('output.csv', 'w', newline="") as csvfile:
    columns = ['班級', '學號', '成績']
    # 將 dictionary 寫入 CSV 檔
    writer = csv.DictWriter(csvfile, fieldnames=columns, delimiter=':')
    writer.writeheader() # 寫入第一列的欄位名稱
    writer.writerow({'班級': '資工一', '學號': '109590003', '成績': 95}) # 寫入資料
    writer.writerow({'班級': '資工一', '學號': '109590004', '成績': 88}) # 寫入資料
```

delimiter=':' is optional

# Exercise

- 一行一行讀檔案 score.txt
  - 計算平均，將平均寫到最下面

Input file: score.txt

班級,學號,期中考成績,  
資工一,109590001,88,  
資工一,109590002,90,  
資工一,109590003,92,  
資工一,109590004,85,  
資工一,109590005,87,  
資工一,109590006,95,  
資工一,109590007,80,  
資工一,109590008,84,  
資工一,109590009,86,  
資工一,109590010,83,

Output file: avg\_score.txt

Class,Student ID,Score,  
資工一,109590001,88,  
資工一,109590002,90,  
資工一,109590003,92,  
資工一,109590004,85,  
資工一,109590005,87,  
資工一,109590006,95,  
資工一,109590007,80,  
資工一,109590008,84,  
資工一,109590009,86,  
資工一,109590010,83,  
平均,,87.0,

# Exercise

## □ 製作一個csv檔 score.csv

- 一行一行讀檔案 score.csv，製作成字典
- 計算每位學生平均，寫在學生資料最後，計算全班平均，寫到最下面

Input file: score.csv

```
班級,學號,國文,數學,英文  
資工一,109590001,80,80,80  
資工一,109590002,90,90,90  
資工一,109590003,70,70,70  
資工一,109590004,60,60,60,
```

Output file: output.csv

```
Class,Student ID,average,  
資工一,109590001,80,  
資工一,109590002,90,  
資工一,109590003,70,  
資工一,109590004,60,  
75,75,75,75
```

# Exercise

- 製作一個csv檔 score.csv
  - 一行一行讀檔案 score.csv，製作成字典
  - 計算每位學生平均，寫在學生資料最後，計算全班平均，寫到最下面
- 輸出成 output.csv

score.csv

班級,學號,國文,數學,英文  
資工一,109590001,80,80,80  
資工一,109590002,90,90,90  
資工一,109590003,70,70,70  
資工一,109590004,60,60,60,

Output.csv

Class,Student ID,average,  
資工一,109590001,80,  
資工一,109590002,90,  
資工一,109590003,70,  
資工一,109590004,60,  
75,75,75,75

# Exercise

```
import csv
def trans(row):
    data = {}
    score = 0
    subject = ['國文','英文','數學']
    for key, value in row.items():
        print('=>', key, value)
        if key in subject:
            score = score + int(value)
    for key, value in row.items():
        [REDACTED]
    data['average'] = score//3
    return data
```

```
with open('D:\\x.csv', newline="") as csvfile:
    readFile = csv.DictReader(csvfile)
    #print(readFile)
    inData = []
    for row in readFile:
        print(row)
        inData [REDACTED]
print(inData)
```

```
with open('y.csv', 'w', newline="") as csvfile:
    #columns = ['班級','學號','國文','數學','英文']
    columns = ['Class', 'Student Id','average']
    # 將 dictionary 寫入 CSV 檔
    writer = csv.DictWriter(csvfile, fieldnames=columns, delimiter=',')
    writer.writeheader() # 寫入第一列的欄位名稱
    for data in inData:
        [REDACTED] # 寫入資料
```

# Python Set

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# set基本操作

- 集合型態的字面常數使用大括弧{ }
- 屬於複合資料型態 (compound data type) ，包含多個元素。
- 無序、元素不重複
- 使用in測試元素是否在集合中
- 使用is測試兩集和是否一樣

```
numX = {1,1,1,2,2,3,4,5}  
numY = {5, 4, 2, 1, 3}  
print(numX)  
print(numY)  
print(4 in numY)  
print(numX==numY)  
print(numX is numY)  
print(id(numX))  
print(id(numY))
```

```
{1, 2, 3, 4, 5}  
{1, 2, 3, 4, 5}  
True  
True  
False  
2042423848296  
2042423847400
```

# set基本操作

- mySet = set() : 使用 set() 造出空集合物件
- mySet = {} 建立字典，非空集合
- .add() : 加入一個元素 (.update() : 加入多個元素)

```
def setOp():
    Language = set()
    print(Language)
    mySet = {}
    print(type(mySet))
    Language = { "asm", "C", "C++", "Java", "iOS", "Ruby", "perl", "delphi" }
    Language.add("python")
    print(Language)
```

```
set()
<class 'dict'>
{'asm', 'python', 'C++', 'perl', 'Java', 'iOS', 'C', 'Ruby', 'delphi'}
```

# set基本操作

- 使用in測試元素是否在集合中
- & 交集運算，| 聯集運算，- 差集運算
- ^ 排除相同元素(XOR)運算
- > 測試左集合是否為右集合的父集
- < 測試左集合是否為右集合的子集
  
- `nums = [1,1,1,2,2,3,4,5]`，若需將重複元素去除怎麼辦

```
nums = [1,1,1,2,2,3,4,5]
num = set(nums)
print(num)
```

```
{1, 2, 3, 4, 5}
```

# set基本操作

## □ & | - ^ > < 等元素性质

```
def setOp():
    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)
```

```
{'Mars'}
{'Mars', 'Happy', 'Smile', 'ihc', 'Tony', 'Andy', 'Sherry', 'Allen'}
{'Happy', 'ihc', 'Tony', 'Smile', 'Andy', 'Sherry', 'Allen'}
{'ihc'}
{'Happy', 'Smile', 'Tony', 'Andy', 'Sherry', 'Allen'}
```

# set基本操作

```
setx = set(["apple", "mango"])
sety = set(["mango", "orange"])
setz = set(["mango"])
my_issubset = setx <= sety
print(my_issubset)
my_issuperset = setx >= sety
print(my_issuperset)
my_issubset = setz <= sety
print(my_issubset)
my_issuperset = sety >= setz
print(my_issuperset)
print()
```

```
zz=setx.issubset(sety)
print(zz)
yy=setx.isdisjoint(sety)
print(yy)
```

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

# Set函式

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another specified set
discard()	Remove the specified item ( <b>won't raise error if item doesn't exist</b> )
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element ( <b>raise error if item doesn't exist</b> )
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

# Set函式

```
my_set={"apple", "banana", "cherry"};
my_set.update(["orange", "mango", "grapes"])
print(my_set)
print(len(my_set))
my_set.remove("banana")
print(my_set)
#my_set.remove("kiwi")
my_set.discard("kiwi")
print(my_set)
#my_set.difference_update("apple")
#print(my_set)
```

編號每一行程式  
寫下輸出內容

```
{'orange', 'mango', 'apple', 'cherry', 'banana', 'grapes'}
6
{'orange', 'mango', 'apple', 'cherry', 'grapes'}
{'orange', 'mango', 'apple', 'cherry', 'grapes'}
```

# Set函式

編號每一行程式  
寫下輸出內容

```
setx = set(["apple", "mango"])
sety = set(["mango", "orange"])
setw = setx.difference(sety)
print(setw)
print(setx)
setx.difference_update(sety)
print(setx)
```

{'apple'}
{'mango', 'apple'}
{'apple'}

```
setx = set(["apple", "mango"])
sety = set(["mango", "orange"])
setw = setx.intersection(sety)
print(setw)
print(setx)
setx.intersection_update(sety)
print(setx)
```

{'mango'}
{'mango', 'apple'}
{'mango'}

```
setx = set(["apple", "mango"])
sety = set(["mango", "orange"])
setw = setx.symmetric_difference(sety)
print(setw)
print(setx)
setx.symmetric_difference_update(sety)
print(setx)
```

{'orange', 'apple'}
{'mango', 'apple'}
{'orange', 'apple'}

# Exercise

- 寫一個程式輸入兩個字串，輸出兩個字串均有的部分

```
s1=input("Enter first string:")  
s2=input("Enter second string:")  
a=list(set(s1)|set(s2))  
print("The letters are:")  
for i in a:  
    print(i)
```

編號每一行程式  
寫下執行編號順序  
輸入字串1='banana'  
輸入字串2='orange'  
寫下輸出內容

- 寫一個程式求出集合中最大與最小值

```
seta = set([5, 10, 3, 15, 2, 20])  
#Find maximum value  
print(max(seta))  
#Find minimum value  
print(min(seta))
```

# Exercise

- 使用集合寫一個程式，輸入一個字串，若字串都是單一字元，回傳 **True**，有重複字元，回傳 **False**.

## sample input

just => True (has unique characters)  
Alexander => False (has duplicates)

編號每一行程式  
寫下執行編號順序  
輸入= ‘alexander’  
寫下輸出內容

```
def is_unique(given_string):
    #creating an empty set
    chars_set = set()
    for char in given_string:
        #if char already in set
        #it is duplicate
        if char in []:
            return False
        else:
            #char not in set, add it to chars_set
            chars_set.add()
    #if no duplicates
    return True
print(is_unique("just"))
```

# Exercise

- 使用集合寫程式，顯示字元沒有出現在第二個字串，但出現在第一個字串。
- 使用集合寫程式，輸入兩個字串，輸出均有的字元。
  
- 輸入兩個集合，檢查是否一個集合是另一個集合的子集，若是則刪除所有子集的元素

```
firstSet = {27, 43, 34}
```

```
secondSet = {34, 93, 22, 27, 43, 53, 48}
```

- 輸出

```
First set is subset of second set - True
```

```
Second set is subset of First set - False
```

```
After deleting the First set, the Second set is {93, 22 , 53, 48}
```

# Exercise

## □ 理想大學環境

- 每一大學可用下列七種屬性表示：

- BC(Big Campus)：代表有大校園。
- NC(Next to City)：代表鄰近有大城市。
- CT(Convenient Transportation)：代表交通方便。
- NS(Next to Sea)：代表靠海。
- NM(Next to Mountain)：代表依山。
- HL(Has Lake)：代表校園有湖。
- NL(Near Landscape)：代表附近有風景區。

- 輸入說明：

- 1. 輸入理想大學條件，用 + 號區格條件是 "或" 的關係，沒有 + 區隔是 "且" 的關係，屬性間和 + 間以空白間隔。例如： BC NS + CT HL 是找有大校園且靠海，或交通方便且校園有湖的所有大學。

# Exercise

- 2. 第一行一正整數，代表大學個數  $n$  ( $n \leq 10$ )。
  - 3. 其後  $n$  行，每一行為大學名稱，接著大學具備屬性。大學名稱最多 10 個字母，屬性 2 個字母，均為英文字母，大學名稱及屬性間以一個空白分隔。
  - 4. 接下來一行正整數  $m$ ，為查詢個數， $m \leq 10$ 。
  - 5. 其後  $m$  行，每一行有一個查詢。條件為校園屬性。
- 輸出說明：(1, 2輸出各得 1/2分數)
- 1.  $m$  行，第  $i$  列印出第  $i$  個查詢中，所有符合之大學名稱。
    - (1) 若有多個大學符合一個查詢，大學間以空白分隔。
    - (2) 每行查詢輸出順序，根據先後查詢條件符合的大學順序。
  - 2. 如果都沒有完全符合，則輸出最多符合的大學。

# Exercise

Sample Input	Sample Output
<b>5</b> <b>NSYSU NC CT NS NM</b> <b>NTU BC NC CT NS</b> <b>NCCU BC NL HL</b> <b>Providence BC NC</b> <b>NTHU BC NS</b> <b>2</b> <b>BC NS + CT HL</b> <b>NM + BC NL + BC NC</b>	<b>NTU NTHU</b> <b>NSYSU NCCU NTU Providence</b>
<b>5</b> <b>NSYSU NC CT NS NM</b> <b>NTU BC NC CT NS</b> <b>NCCU BC NL HL</b> <b>Providence BC NC</b> <b>NTHU BC NS</b> <b>1</b> <b>BC NS NL + CT HL</b>	<b>NTU NTHU</b>

# Exercise

```
def getData():
    university = {}
    n = int(input())
    for i in range(n):
        item = input().split()
        university[item[0]] = set(item[1:])
    return university
```

```
def match(con, feature):
    conds = con.split(' + ')
    maxNum = 0
    for i in range(len(conds)):
        if feature.issuperset(set(conds[i].split()))):
            return -1
        k = len(feature & set(conds[i].split()))
        if k > maxNum: maxNum = k
    return maxNum
```

```
def compute(con, university):
    data = {}
    for name, feature in university.items():
        data[name] = match(con, feature)
        if data[name] == -1:
            print(name, end=' ')
    if -1 not in data.values():
        value = max(data.values())
        for key, v in data.items():
            if v == value: print(key, end=' ')
    print()
```

```
def main():
    university = getData()
    n = int(input())
    for i in range(n):
        con = input()
        compute(con, university)
```

# Python

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# lambda

- lambda 參數1, 參數2, ...: 運算式A if 關係運算式 else 運算式B
  - (inline anonymous function)
  - 提供簡易功能，只處理一個運算式，回傳一個值。

```
lambda param1, param2, ... : expression
```

```
def fun(param1, param2, ...):  
    return expression
```

- 其中expression不能放assignment，=等號。

```
def func(x, y, z):  
    return x + y + z  
func2 = lambda x,y,z : x+y+z  
print(func(1, 2, 3))  
print(func2(1, 2, 3))
```

Output  
6  
6

# Lambda vs Function

- Lambda不需定義名稱，一般函式(Function)需定義名稱。
- Lambda函式只能有一行運算式，一般函式(Function)可以有多行運算式。
- Lambda每一次運算完會自動回傳結果，一般函式(Function)要回傳結果須加上 return 關鍵字。

```
max = lambda m, n: m if m > n else n  
print(max(10, 3)) # 顯示 10
```

```
def max(m, n):  
    return m if m > n else n  
  
print(max(10, 3)) # 顯示 10
```

- 若 function 不好重複使用，可傳遞參數給lambda 而非一般function
- lambda式典型用法是搭配filter()和map()內建函式

# Exercise

## □ Output

```
score = int(input('請輸入分數 : '))
level = score // 10
{
    10 : lambda: print('Perfect'),
    9 : lambda: print('A'),
    8 : lambda: print('B'),
    7 : lambda: print('C'),
    6 : lambda: print('D')
}.get(level, lambda: print('E'))()
```

# Sorted函式

## □ 基本排序

```
x = [4, 2, 5, 3, 1]
y = sorted(x)
z = sorted(x, reverse = True) #反向
print(y)
print(z)
x.sort()
print(x)
```

## □ 反向排序

## □ 自訂排序鍵值函數

- 根據第三個數字元素排序：

```
scores = [
    ('Jane', 'B', 12),
    ('John', 'A', 15),
    ('Dave', 'B', 11)]
print(sorted(scores, key = lambda s: s[2]))
```

```
[('Dave', 'B', 11), ('Jane', 'B', 12), ('John', 'A', 15)]
```

# Exercise

## □ 修改程式

- 排序用 lambda 指定 key為avg，請將該敘述改以英文成績eng排序

```
def dataProcess(scores):
    grades=[]
    for stu in scores:
        name = stu[0]
        eng, math, phy = int(stu[1]), int(stu[2]), int(stu[3])
        avg = round((eng + math + phy)/3, 2)
        grades.append([name, [eng, math, phy], avg])
    sortedGrade = sorted(grades, key=lambda x: x[-1], reverse= True)
    print (sortedGrade)

dataProcess([('John', 90, 80, 90), ('Mary', 100, 90, 85), ('Tom', 80, 90, 70)])
```

```
sortedGrade = sorted(grades, key=lambda [ ] , reverse= True)
```

# map

## □ map(function, iterable1 [, iterable2])

- 輸入 function 和不定個數的 iterable (list, tuple, ...)
- 將 iterable 當參數傳入 function，一一回傳 function 執行結果
  - 對傳入的 iterable 每一個元素進行對映，回傳新對映後 iterable
- 使用 map() 可縮短程式碼和加速執行
- map 無法處理
  - iterable 長度不一致
  - 對應位置運算元型別不一致

```
a=list(map(lambda x , y : x ** y, [2,4,6],[3,2,1]))  
print(a)
```

```
scores = [50, 52, 54, 56, 58, 60]  
new_scores = map(lambda x: 60 if 55 <= x < 60 else x, scores)  
print(list(new_scores))  
# 顯示處理後成績：[50, 52, 54, 60, 60, 60]
```

```
def multiply2(x, y):  
    return x * y
```

```
a=list(map(multiply2, [2, 4, 6], [3, 2, 1]))  
print(a)      # Output [6, 8, 6]
```

# Exercise

## □ 利用 map() 函式，輸出？

```
def adder(x,y,z):
    return x+y+z

def multiple2(x):
    return x*x

list1 = [1,3,5,7,9]
for x in map(multiple2,list1):
    print(x, end=' ')
print([x for x in map(multiple2,list1)])
print([multiple2(x) for x in list1])
list1 = [1,3,5,7,9]
list2 = [2,4,6,8,10]
list3 = [100,100,100,100,100]
print([x for x in map(add, list1, list2, list3)])
my_list = [1, 2, 3]
a=list(map( lambda i: i * i, my_list ))
print(a)
```

Output  
1 9 25 49 81



# Exercise

- 利用 map()，將 'asDfA13' 字元從大寫轉為小寫，小寫轉為大寫

```
def u2l_and_l2u (s):
    return s.upper() if [ ]() else s.[ ]()

a = list(map(u2l_and_l2u,'asDfA13'))
ms=""
b=ms.join(a)
print(b)
c = ms.join([str(elem) for elem in a])
print(c)
for c in a:
    ms=ms+c
print(ms)
```

ASdFa13  
ASdFa13  
ASdFa13

# filter

## □ filter(function, sequence)

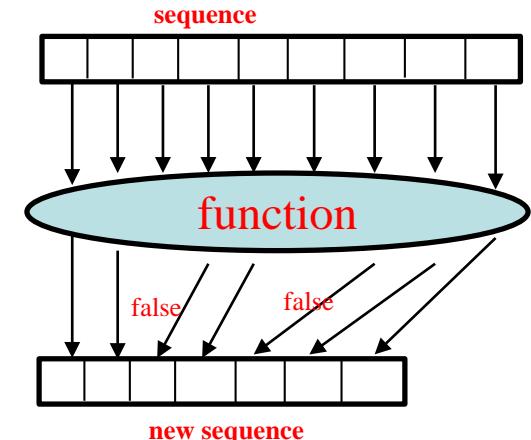
- sequence 的每個元素當參數傳給 function 進行 True/False 判斷
- 過濾 sequence 中的元素，回傳符合條件之元素組成的新 sequence (將回傳值為 True 的項目組成一個 iterator)

```
# a = list(filter(lambda x: x % 2 == 1, range(10)))  
def is_odd(n):  
    return n % 2 == 1  
a = list(filter(is_odd, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]))  
print(a)
```

[1, 3, 5, 7, 9]

```
import math  
def is_sqr(x):  
    return math.sqrt(x) % 1 == 0  
  
a = list(filter(is_sqr, range(1, 101)))  
print(a)
```

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]



```
scores = [90, 50, 80, 40, 100]  
fail_scores = filter(lambda x: True if x < 60 else False, scores)  
# 回傳 <filter object at 0x7f17c658a630>  
print(list(fail_scores)) #[50, 40]
```

# Exercise

- scores=[['John', 90, 80, 90],['Bob', 50, 70, 40], ['Mary', 100, 90, 85],['Tom', 80, 90, 70]]
  - 使用 filter, lambda, 輸出平均及格的名字、平均

```
scores=[['John', 90, 80, 90],['Bob', 50, 70, 40], ['Mary', 100, 90, 85],['Tom', 80, 90, 70]]
```

```
data = filter(lambda x: x[1]+x[2]+x[3]>=150, scores)
k = list(data)
for x in k:
    print(x[0], (x[1]+x[2]+x[3])/3)
print(k)
```

```
data_s = sorted(k, key = lambda x: (x[1]+x[2]+x[3])/3)
for x in data_s:
    print(x[0], (x[1]+x[2]+x[3])/3)
```

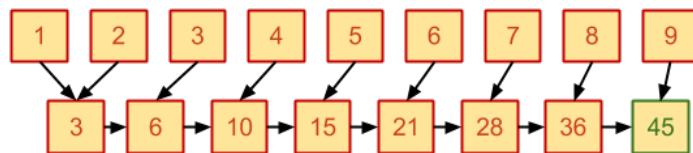
# reduce()

## □ reduce (f, seq[, init()])

○ 每次迭代都將上次迭代結果與下一個seq元素一同傳入**二元**(binary)函式(具有兩個參數的函式)中執行

○ init是optional，若指定，則當第一次迭代第一個參數，若無則取seq第一個元素

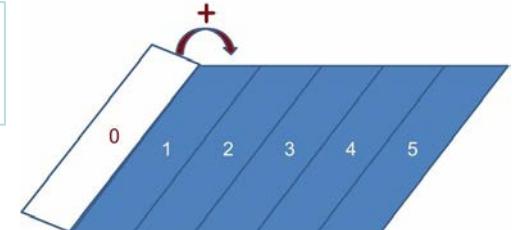
○ 對list的每個元素反覆呼叫函式f，回傳最終結果值



```
import functools as ft
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
def fn(x, y):
    return x+y
print(ft.reduce(fn, a)) #45
```

```
import functools as ft
a=ft.reduce(lambda sum, elem: sum + elem, [1, 2, 3, 4, 5], 0)
print(a)
print (ft.reduce(lambda x, y: x * y, range(1, 6),2))
```

15  
240



# Nest function or Inner function

- "nested function" or "**inner function**"

- 定義 function裡面的function。

```
def make_repeater(n):
    return lambda s: s*n
twice = make_repeater(2) #n=2
print(twice('word')) #s=word
print(twice(5))      #s=5
```

Output  
wordword  
10

```
def function1(): # outer function
    print ("Hello from outer function")
    def function2(): # inner function
        print ("Hello from inner function")
    function2()
function1()
```

Hello from outer function  
Hello from inner function

# 使用 lambda 實作 inner function

```
def num1(x):
    print('x=',x)
    def num2(y):
        print('y=',y)
        return x * y
    return num2
res = num1(10)      #outer
print(res(5))       # inner
print(num1(10)(5))
```

x= 10  
y= 5  
50

# use lambda for inner function

```
def num1(x):
    print('x=',x)
    return lambda y: x * y

res = num1(10)      #outer
print(res(5))
print(num1(10)(5))
```

x= 10  
50

## □ 定義 inner function 原因: Encapsulation

```
def outer_function(x):
    # Hidden from the outer code
    def inner_increment(x):
        return x + 2
    y = inner_increment(x)
    print(x, y)
```

```
inner_increment(5)
outer_function(5)
```

NameError: name 'inner\_increment' is not defined

# Python Dictionary

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# Dictionary字典

## □ 建立字典

- 變數可利用大括弧{ }，以key : value 為配對的資料項目

```
d1 = {}  
d2 = dict()  
print(d1)
```

english = {'book': '書本', 'food': '食物'}

key

value

## □ 字典元素無順序性

- 與list, tuple不同，不以索引存取元素，以「鍵」**key**存取元素
  - key 須是不可變的 (immutable) 資料型別，如數字、字串(string)
  - key不必從「0」開始
- value 沒有限制
- 字典不能切片。
- 使用key取得value

```
book = english['book']  
print(book)
```

# Dictionary字典

## □ 字典動態結構

### ○ 增加新的鍵值對

➤ `data['2']='B'`

### ○ 修改字典中的值

➤ `data['2']='C'`

### ○ del刪除鍵值對

➤ `del data['2']`

### ○ 利用in和not in運算子，可檢查字典中是否存在某個鍵或值。

```
english = {'game':'遊戲', 'food':'食物'}
```

```
english['cat'] = '貓'
```

```
english['game'] = '比賽'
```

```
del english['game']
```

```
d3 = { 'name':'John', 'age':20 }
```

```
if 'address' in d3:
```

```
    print(d3['address'])
```

```
else:
```

```
    print('None')
```

# Exercise

## □ 輸出

```
def f():
    data = {'name':'Tom', 'age':18, 'phone':'0933123001'}
    print(data)
    del data['name']                      #刪除
    print(data)
    studentAge = data['age']               #以 key 取值
    print(studentAge)
    if 'age' in data:                     #以 in 檢驗 key 是否存在
        print(data['age'])
```

```
x = ('key1', 'key2', 'key3')
y = 0
thisdict = dict.fromkeys(x, y)
print(thisdict)
```

```
{'key1': 0, 'key2': 0, 'key3': 0}
```

# Dictionary字典

## □ 字典keys()、values()和items()方法

○ 回傳字典的鍵，值或鍵和值的列表值。

○ 資料型別分別是dict\_keys、dict\_values和dict\_items，可用於for。

```
d3 = { 'name':'John', 'age':20 }
```

```
for key in d3:  
    print(key)
```

```
for key in d3.keys():  
    print(key)
```

```
for value in d3.values():  
    print(value)
```

```
for key, value in d3.items():  
    print(key, value)
```

```
name  
age  
name  
age  
John  
20  
name John  
age 20
```

# Exercise

## □ 輸出

```
passwd={'Mars':00,'Mark':56}  
passwd['Happy']=99  
passwd['Smile']=12  
del passwd['Mars']  
passwd['Mark']=passwd['Mark']+1  
print (passwd)  
print (passwd.keys())  
print (passwd.get('Tony'))
```

```
{  
[  
N
```

```
car = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
x = car.items()  
print(x)
```

```
dict_items([  
])
```

# Exercise

## □ 輸出

```
car = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
x = car.values()  
print(x)  
y = car.keys()  
print(y)  
for key in car.keys():  
    print(car[key])
```

dict\_values([ ...])

dict\_keys([ ...])

# Exercise

## □ 輸出？

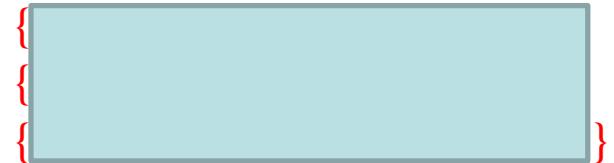
```
def tr(index):
    table = [chr(w) for w in range(ord('A'), ord('Z') + 1)]
    if int(index)>26: return '-1'
    return table[int(index)-1]

def check(x, data):
    if len(x)==0: return True
    if len(x)==1:
        data[x]= tr(x)
        return True
    if len(x)>=2:
        if tr(x[0:2])!=-1':
            data[x[0:2]]=tr(x[0:2])
            check(x[2:], data)
        check(x[:1], data)
        check(x[1:], data)
    return True
```

```
data ={ }
check('52', data)
print(data)
```

```
data ={ }
check('25', data)
print(data)
```

```
data ={ }
check('625', data)
print(data)
```



# Exercise

## □ 刪除

```
car = { 'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

```
car.popitem()
```

```
print(car)
```

```
car.pop('model')
```

```
print(car)
```

```
x = car.setdefault('color', 'White')
```

```
print(x)
```

```
print(car)
```

```
{ 'brand': 'Ford', 'model': 'Mustang'}
```

```
{ 'brand': 'Ford'}
```

```
White
```

```
{ 'brand': 'Ford', 'color': 'White'}
```

# 計算出現字元次數

## □ get

- 回傳key的value
- 如果沒有值，回傳第二個參數的值

```
s = 'I_am_a_programmer_and_I_have_no_life'  
my_dict = {}  
for letter in s:  
    my_dict[letter] = my_dict.get(letter, 0) + 1  
print(my_dict)
```

```
{'I': 2, '_': 8, 'a': 5, 'm': 3, 'p': 1, 'r': 3, 'o': 2, 'g': 1, 'e': 3, 'n': 2, 'd': 1, 'h': 1, 'v': 1, 'T': 1, 'i': 1, 'f': 1}
```

# Dictionary轉型與合併

## □ dict()轉型與合併 update()

```
studentA = [['name','Tom'],['age',19]]  
w=dict(studentA)  
print(w)  
studentB = ('name','John'),('age',18)  
x=dict(studentB)  
print(x)  
studentC = ('name','Mary'),('age',17)  
y=dict(studentC)  
print(y)  
studentD = ('English',80),('Chinese',85))  
z=dict(studentD)  
print(z)  
y.update(z)  
print(y)  
y.clear()  
print(y)
```

```
{'name': 'Tom', 'age': 19}  
{'name': 'John', 'age': 18}  
{'name': 'Mary', 'age': 17}  
{'English': 80, 'Chinese': 85}  
{'name': 'Mary', 'age': 17, 'English': 80, 'Chinese': 85}  
{}
```

# Exercise

- 寫一個程式輸入一個學生兩筆資料(id, name), (age, address)進入兩個list，再轉成兩個字典，之後合併兩個字典，輸出合併的字典。

```
def ex00():
    sid = input('id:')
    name = input('name:')
```

```
d1.update(d2)
print(d1)
```

```
id:101
name:John
age:18
address:Taipei
{'101': 'John', '18': 'Taipei'}
```

```
def ex01():
    studentA, studentB = {}, {}
    sid = input('id:')
    studentA['id'] = sid
    name = input('name:')
    studentA['name']=name
    age = input('age:')
    studentA['age'] = age
    address = input('address:')
    studentA['address']=address
    studentA.update(studentB)
    print(studentA)
```

# Exercise

- 寫一個程式輸入三個學生姓名與成績，以字典儲存
  - 計算輸出所有學生成績的平均。
  - 輸入一個名字當key，從字典移除這個學生成績，把剩下的名字與成績輸出。

```
def ex01():
    studentA, studentB = {}, {}
    sid = input('id:')
    studentA['id'] = sid
```

```
studentA.update(studentB)
print(studentA)
```

# Exercise

□ 輸入N個學生姓名與國英數三科成績，轉成字典

{name:[score1, score2, score3]}

○ 輸出國文及格學生姓名、成績，以姓名排序

○ 輸出所有學生成績，以國文成績排序，由大到小

Input

4

Tom 60 80 70

Mary 90 70 60

John 70 80 60

Kan 50 100 80

-----  
output

```
def ex03(N):
    student = {}
    for i in range(N):
        score = [0, 0, 0]
        x = input().split()
        name = x[0]
```

```
for x in report2:
    print(x[0], x[1])
```

# 最大值

## 找出字典最大值與其Key

```
data = {'Tom': 90, 'John': 85, 'Mary': 75, 'Kevin': 90}  
name = max(data, key = data.get)  
print(name)  
maxValue = max(data.values())  
print(maxValue)  
for keys, values in data.items():  
    if values == maxValue:  
        print(keys, values)
```

```
Tom  
90  
Tom 90  
Kevin 90
```

# Dictionary函式

Method	Description
clear()	清空
copy()	回傳副本
fromkeys()	Returns a <b>dictionary</b> with the specified keys and values
get()	Returns the <b>value</b> of the specified key
items()	Returns a <b>list</b> containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the <b>element</b> with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

# Exercise

- 取得 key, value 和個數

```
num = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
print("key value count")
count=0
for key, value in num.items():
    count=count+1
    print(key,' ',value,' ', count)
```

key	value	count
1	10	1
2	20	2
3	30	3
4	40	4
5	50	5
6	60	6

- 產生字典並輸出，{x: x\*x}, x=1~n)

```
n=int(input("Input a number "))
d = dict()
for x in range(1,n+1):
    d[x]=x*x
print(d)
```

Input a number 5  
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

# Dictionary排序

## □ 排序

- 字典dict的keys()是iterable物件
- num雖然可以放在 iter()內，但其變成 dict.keys()

```
num = {'n2': [2, 3, 1], 'n3': [5, 1, 2], 'n1': [3, 2, 4]}
it = iter(num)
data = next(it)
print(data)          # n2
sorted_num = sorted(num)
print(sorted_num)    # ['n1', 'n2', 'n3']
```

- 字典dict的values()是iterable物件，可依此排序 values

```
num = {'n1': [2, 3, 1], 'n2': [5, 1, 2], 'n3': [3, 2, 4]}
sorted_dict = {x: sorted(y) for x, y in num.items()}
print(sorted_dict)
```

{'n2': [1, 2, 3], 'n3': [1, 2, 5], 'n1': [2, 3, 4]}

# Dictionary排序

□ 排序 – 字典dict的keys(), values(), items()是iterable物件，

```
num = {'n3': [2, 3, 1], 'n1': [5, 1, 2], 'n4': [3, 2, 4], 'n2': [6, 1, 1]}
sorted_dict = sorted(num.keys(), reverse=True)
print(sorted_dict)
sorted_dict = sorted(num) #字典排序，只取出key排序
print(sorted_dict)
```

['n4', 'n3', 'n2', 'n1']  
['n1', 'n2', 'n3', 'n4']

#d.items() 將 d轉換為可疊代物件 ('n3', [2, 3, 1]), (...), (...)

# item[1] 表示 value, 根據 value 排序

```
sorted_dict = sorted(num.items(), key=lambda item:item[1])
print(sorted_dict)
```

# item[0] 表示 keys，根據 key 排序

```
sorted_dict = sorted(num.items(), key=lambda item:item[0])
print(sorted_dict)
```

[('n3', [2, 3, 1]), ('n4', [3, 2, 4]), ('n1', [5, 1, 2]), ('n2', [6, 1, 1])]  
[('n1', [5, 1, 2]), ('n2', [6, 1, 1]), ('n3', [2, 3, 1]), ('n4', [3, 2, 4])]

# Exercise

## □ 理想大學環境

- 每一大學可用下列七種屬性表示：

- BC(Big Campus)：代表有大校園。
- NC(Next to City)：代表鄰近有大城市。
- CT(Convenient Transportation)：代表交通方便。
- NS(Next to Sea)：代表靠海。
- NM(Next to Mountain)：代表依山。
- HL(Has Lake)：代表校園有湖。
- NL(Near Landscape)：代表附近有風景區。

- 輸入說明：

- 1. 輸入理想大學條件，用 + 號區格條件是 "或" 的關係，沒有 + 區隔是 "且" 的關係，屬性間和 + 間以空白間隔。例如： BC NS + CT HL 是找有大校園且靠海，或交通方便且校園有湖的所有大學。

# Exercise

- 2. 第一行一正整數，代表大學個數  $n$  ( $n \leq 10$ )。
  - 3. 其後  $n$  行，每一行為大學名稱，接著大學具備屬性。大學名稱最多 10 個字母，屬性 2 個字母，均為英文字母，大學名稱及屬性間以一個空白分隔。
  - 4. 接下來一行正整數  $m$ ，為查詢個數， $m \leq 10$ 。
  - 5. 其後  $m$  行，每一行有一個查詢。條件為校園屬性。
- 輸出說明：(1, 2輸出各得 1/2分數)
- 1.  $m$  行，第  $i$  列印出第  $i$  個查詢中，所有符合之大學名稱。
    - (1) 若有多個大學符合一個查詢，大學間以空白分隔。
    - (2) 每行查詢輸出順序，根據先後查詢條件符合的大學順序。
  - 2. 如果都沒有完全符合，則輸出最多符合的大學。

# Exercise

Sample Input	Sample Output
<b>5</b> <b>NSYSU NC CT NS NM</b> <b>NTU BC NC CT NS</b> <b>NCCU BC NL HL</b> <b>Providence BC NC</b> <b>NTHU BC NS</b> <b>2</b> <b>BC NS + CT HL</b> <b>NM + BC NL + BC NC</b>	<b>NTU NTHU</b> <b>NSYSU NCCU NTU Providence</b>
<b>5</b> <b>NSYSU NC CT NS NM</b> <b>NTU BC NC CT NS</b> <b>NCCU BC NL HL</b> <b>Providence BC NC</b> <b>NTHU BC NS</b> <b>1</b> <b>BC NS NL + CT HL</b>	<b>NTU NTHU</b>

# Exercise

```
def getData():
    university = {}
    n = int(input())
    for i in range(n):
        [REDACTED]
    return university
```

```
def match(con, feature):
    conds = con.split(' + ')
    maxNum = 0
    for i in range(len(conds)):
        [REDACTED] :
    return maxNum
```

```
def compute(con, university):
    data = {}
    for name, feature in university.items():
        [REDACTED]
    if -1 not in data.values():
        value = max(data.values())
        for key, v in data.items():
            if v==value: print(key, end=' ')
        print()
```

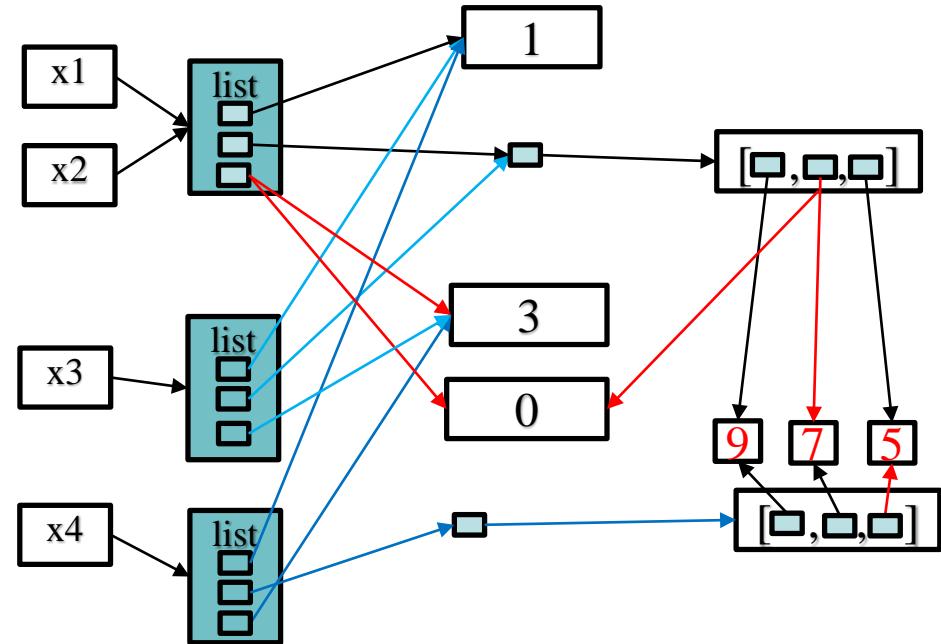
```
def main():
    university = getData()
    n = int(input())
    for i in range(n):
        con = input()
        compute(con, university)
```

# Shallow Copy vs. Deep Copy

- Assignment (=) 造一個新變數reference指向原始字典物件
- 二種方式造字典：
  - Shallow copy(淺拷貝):複製一層 references
  - Deep copy(深拷貝):遞迴 (recursively) 複製 (copies) 所有層 references

```
import copy
x1 = [1, [9, 7, 5], 3]
x2 = x1
x3 = copy.copy(x1)
x4 = copy.deepcopy(x1)

x1[0] = 0
print(x1, x2)
print(x3, x4)
x1[1][1]=0
print(x1, x2)
print(x3, x4)
```

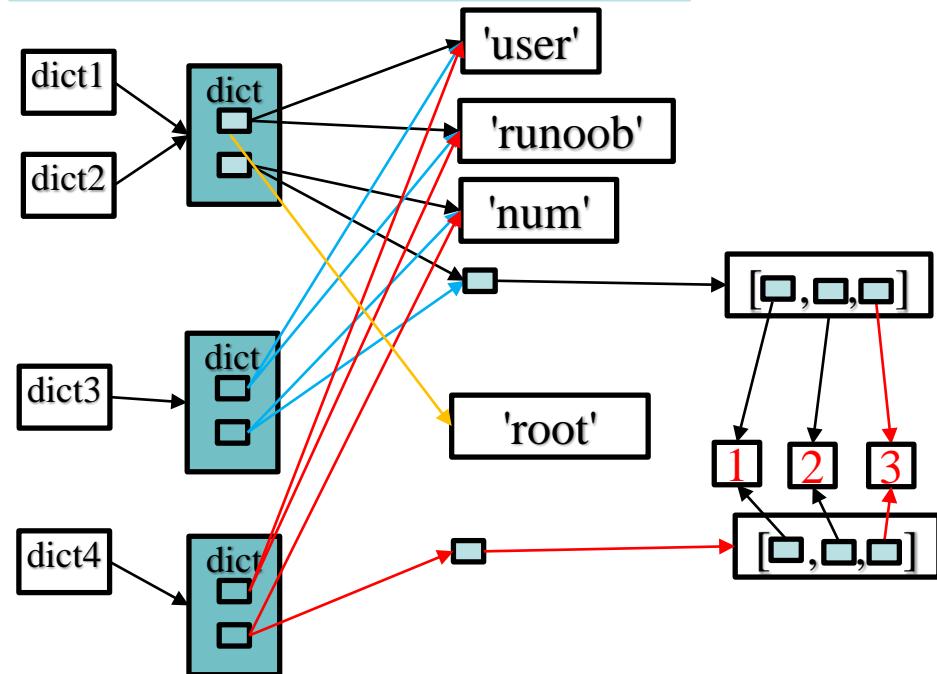


```
[0, [9, 7, 5], 3] [0, [9, 7, 5], 3]
[1, [9, 7, 5], 3] [1, [9, 7, 5], 3]
[0, [9, 0, 5], 3] [0, [9, 0, 5], 3]
[1, [9, 0, 5], 3] [1, [9, 7, 5], 3]
```

# Shallow Copy vs. Deep Copy

```
import copy
dict1={'user':'runoob','num':[1,2,3]}
dict2=dict1
dict3=dict1.copy()
dict4=copy.deepcopy(dict1)
dict1['user']='root'
dict1['num'].remove(1)
print(dict1)
print(dict2)
print(dict3)
print(dict4)
del dict1['user']
print(dict1)
print(dict2)
print(dict3)
print(dict4)
```

```
{'user': 'root', 'num': [2, 3]}
{'user': 'root', 'num': [2, 3]}
{'user': 'runoob', 'num': [2, 3]}
{'user': 'runoob', 'num': [1, 2, 3]}
{'num': [2, 3]}
{'num': [2, 3]}
{'user': 'runoob', 'num': [2, 3]}
{'user': 'runoob', 'num': [1, 2, 3]}
```



# 巢狀字典

## □ 字典內包含字典稱巢狀字典(**nested dictionaries**)

```
myfamily = {  
    "child1" : {  
        "name" : "Emil",  
        "year" : 2004  
    },  
    "child2" : {  
        "name" : "Tobias",  
        "year" : 2007  
    },  
    "child3" : {  
        "name" : "Linus",  
        "year" : 2011  
    }  
}
```

```
child1 = {  
    "name" : "Emil",  
    "year" : 2004  
}  
child2 = {  
    "name" : "Tobias",  
    "year" : 2007  
}  
child3 = {  
    "name" : "Linus",  
    "year" : 2011  
}  
  
myfamily = {  
    "child1" : child1,  
    "child2" : child2,  
    "child3" : child3  
}
```

# 巢狀字典

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},  
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
```

```
print(people[1]['name'])  
print(people[1]['age'])  
print(people[1]['sex'])
```

```
people[3] = {}  
people[3]['name'] = 'Luna'  
people[3]['age'] = '24'  
people[3]['sex'] = 'Female'  
people[3]['married'] = 'No'  
print(people[3])  
people[4] = {'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'}  
print(people[4])
```

```
del people[3]['married']  
del people[4]['married']  
print(people[3])  
print(people[4])  
print()  
del people[3], people[4]  
print(people)
```

John 27 Male {'name': 'Luna', 'age': '24', 'sex': 'Female', 'married': 'No'} {'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'} {'name': 'Luna', 'age': '24', 'sex': 'Female'} {'name': 'Peter', 'age': '29', 'sex': 'Male'}  {1: {'name': 'John', 'age': '27', 'sex': 'Male'}, 2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}}	
---	--

# 巢狀字典的尋訪

```
people = {1: {'Name': 'John', 'Age': '27', 'Sex': 'Male'},  
          2: {'Name': 'Marie', 'Age': '22', 'Sex': 'Female'}}
```

```
for p_id, p_info in people.items():  
    print("\nPerson ID:", p_id)
```

```
    for key in p_info:  
        print(key + ':', p_info[key])
```

Person ID: 1

Name: John

Age: 27

Sex: Male

Person ID: 2

Name: Marie

Age: 22

Sex: Female

# 字典Comprehension

## □ 造字典，{數字:平方}

```
D = {}  
D[0] = 0  
D[1] = 1  
D[2] = 4  
D[3] = 9  
D[4] = 16  
print(D) # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

## □ 使用迴圈

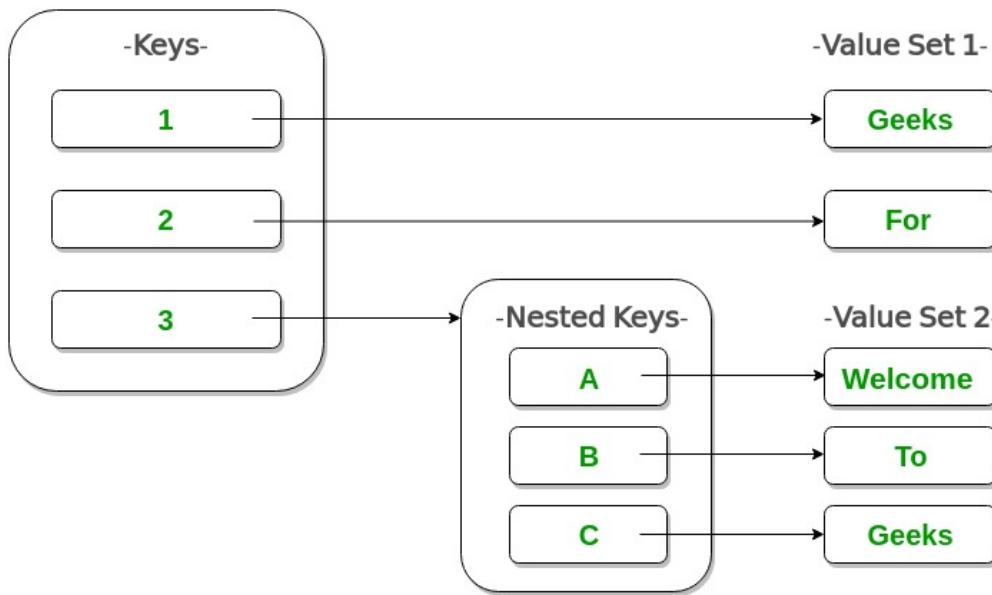
```
D = {}  
for x in range(5):  
    D[x] = x**2  
print(D) # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

## □ 使用comprehension

```
D = {x: x**2 for x in range(5)}  
print(D) # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

# Exercise

## 造巢狀字典



Letter Grade	Number Grade
A+	97
A	95
A-	92
B+	87
B	85
B-	82
C+	77
C	75
C-	72
D+	67
D	65
D-	62
F	50

```
Dict = {1: 'Geeks', 2: 'For', 3: {  : '} }  
print(Dict)
```

# Python Array

郭忠義

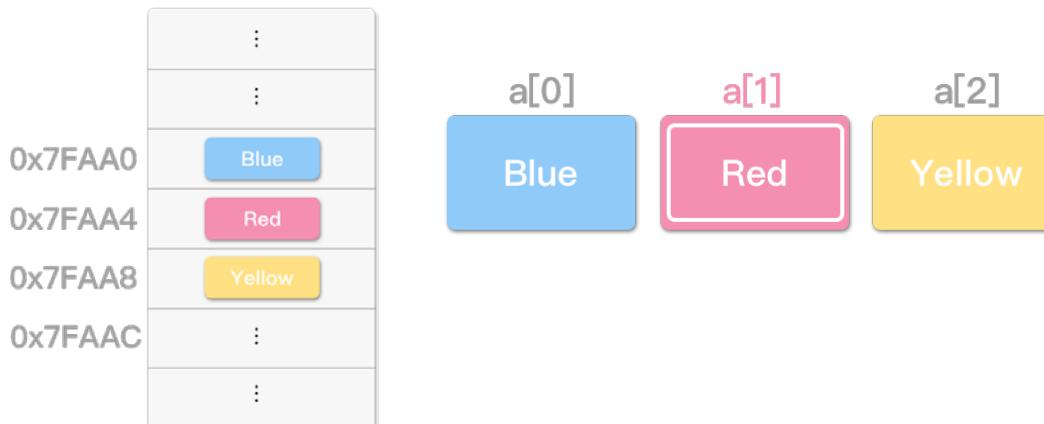
[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# Array(陣列)

## □ Array(陣列)為一種資料結構 (Data Structure)

- 儲存一群「相同資料型別元素 (element)」串列。佔用連續記憶體位置。
- 像容器，裡面每個元素都有其位置，稱為索引(index)
- 藉由索引(indices)得出任一元素位址(address)，達成隨機存取 (Random Access)
- 例如儲存「顏色」陣列，其命名為 a，儲存於「連續記憶體位置」，藉由索引存取元素，不需一個個尋訪。取得「Red」，只要透過索引值「1」即可



# 陣列維度 (Array Dimension)

- 陣列維度，可以是一維、二維...多維 (multi-dimension)
  - 一維陣列 (one-dimensional Array)，一列線性相同型態元素，用索引值存取元素
    - 陣列第一個元素索引值0，第二個元素1，依此類推，陣列擁有n個元素，要存取最後一個元素，需設定索引值n-1
  - 二維陣列 (Two-dimensional Array)，藉由列(row) 與行(column)，將資料置於連續記憶體。需用兩個索引值存取一個元素，例如：A[row][column]。
    - 二維陣列索引用法，如同矩陣 (matrix)，第 i 列、第 j 行的元素，表示為 a[i] [j]，例如：欲存取 乙班、1 號 同學的成績，可透過 a["乙班"] [1] 的方式

		座號	0	1	2	
		班級	甲班	乙班	丙班	
	創創	94分	守守	87分	不化蟲	66分
	DoReMi	38分	羽月	100分	小愛	60分
	皮卡丘	70分	皮老闆	45分	皮子英雄	50分

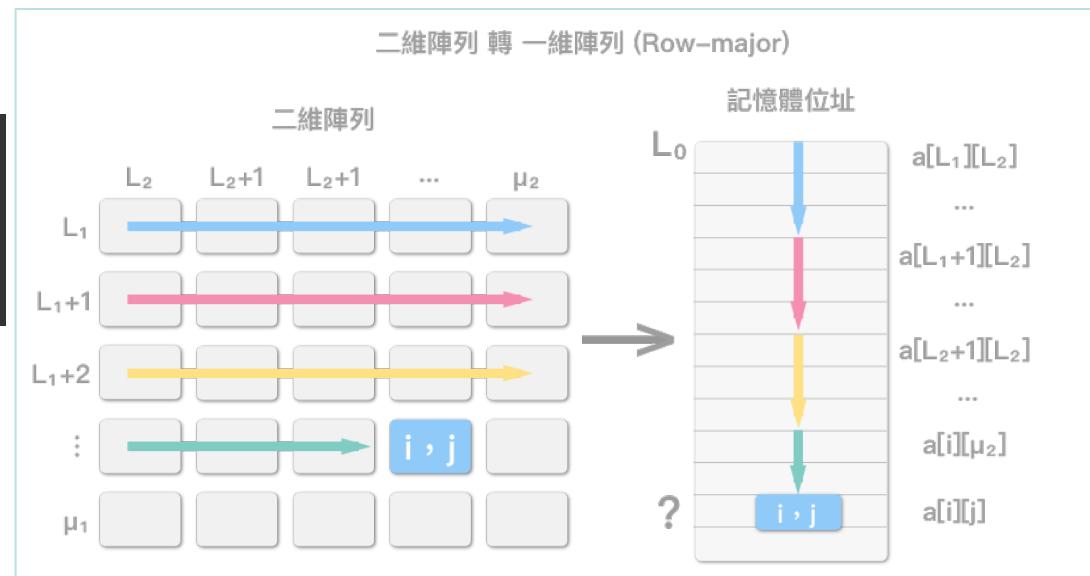
## 二維陣列位址計算

- 兩種方式將二維陣列 轉換為 一維陣列的方式
    - 以列為主 (Row-major)
    - 以行為主 (Column-major)

## □ Row-major

- 以 Row-major 為例，透過一列一列，將二維陣列循序存入記憶體中。

一陣列  $A(L_1 : \mu_1, L_2 : \mu_2)$ ，有  $m$  列、 $n$  行，  
 陣列的基底位址為  $L_0$ ，每個元素大小為  $d$ ，  
 則  $A[i][j]$  之位址 =  $L_0 + (i - L_1) * (n * d) + (j - L_2) * d$ ，  
 其中， $m = \mu_1 - L_1 + 1$ ， $n = \mu_2 - L_2 + 1$ 。



# Python Array

- Python 沒有內建支援陣列，可以Lists代替，或使用Numpy
- 陣列元素存取

```
a = ['s1', '4.0', 's2', '8.0']
print(type(a))
b=[0, 1, 2, 3, 4]
print(type(b))
```

```
<class 'list'>
<class 'list'>
```

```
a = [2, 4, 6, 8]
print(a[0])
print(a[3])
a[1]=3
a[2]=5
print(a)
print(2 in a)
print(10 in a)
```

```
2
8
[2, 3, 5, 8]
True
False
```

```
a = ['s1', '4.0', 's2', '8.0']
print(a[0])
print(a[3])
a[1]='3'
a[2]='5'
print(a)
print('s2' in a)
print('8.0' in a)
```

```
s1
8.0
['s1', '3', '5', '8.0']
False
True
```

# Traversing an Array using Loop

```
score = [89, 55, 73, 66]
sum = 0
for i in range(4):
    sum = sum + score[i]
    print("The score of the", i+1, "student is", score[i])
avg=sum/len(score)
print("The total score is", sum)
print("The average score is", avg)
```

The score of the 1 student is 89  
The score of the 2 student is 55  
The score of the 3 student is 73  
The score of the 4 student is 66  
The total score is 283  
The average score is 70.75

```
score = [89, 55, 73, 66]
sum = 0
i=0
while i<len(score):
    sum = sum + score[i]
    print("The score of the", i+1, "student is", score[i])
    i+=1
avg=sum/len(score)
print("The total score is", sum)
print("The average score is", avg)
```

# 陣列宣告與初始化

- 宣告一個陣列，個別初始化每一個元素
  - $a=[0]^n$  #宣告一維陣列名稱為a有n個元素
    - $a[0]=1; a[1]=2; a[2]=3, \dots, a[n-1]=n$
  - $a=[[0]]^y$  for  $i$  in range( $x$ )] #宣告二維陣列名稱為a有x列y行
    - $a[0][0]=1; a[0][1]=2; \dots a[0][y-1]=y-1; a[1][0]=m, \dots, a[x-1][y-1]=n$
- 宣告陣列並初始化給值
  - $a=[1, 2, 3, 4, 5]$
  - $a=[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]$

1	2	3	4
5	6	7	8
9	10	11	12

- 宣告陣列，使用LOOP初始化給值

```
a=[0]^n
for i in range(n):
    a[i]=0      # $a[i]=0$ 
```

```
a=[[0]]^column for i in range(row)] #宣告二維陣列名稱為a有row列column行
for i in range(row):
    for j in range(column):
        a[i][j]=0      #initialize  $a[row][column]=0$ 
```

# Exercise

## □ 產生陣列元素

### ○ Expected Output :

➤ Elements in array are: 0 1 2 3 4 5 6 7 8 9

```
#not using max() and min()
b=[25, 6, 3, 30, 9, 12]
maximum=b[0]
minimum=b[0]
for i in range(len(b)):
    if b[i]  :
        maximum=b[i]
    elif b[i]  :
        minimum=b[i]
print(maximum, minimum)
```

```
#using list
a=[]
for i in range(0,10):
    a.append(i)
print(a)
```

```
#using array
b=[0]  
for i in range(0,10):
    b[i]= 
print(b)
```

## □ 找出陣列最大與最小元素

### ○ Test data: a=[25, 6, 3, 30, 9, 12]

### ○ Expected output:

➤ Maximum = 30  
➤ Minimum = 3

```
#using max() and min()
a=[25, 6, 3, 30, 9, 12]
print(max(a), min(a))
```

# OX遊戲

## □ 檢查是否連線

```
def check(data, m):          #檢查連線
    slant1 = slant2 = 0
    for x in range(3):
        col = row = 0
        for y in range(3):
            if (data[x][y]==m): #column連線檢查
                col = col + 1
            if (data[y][x]==m): #row連線檢查
                row = row + 1
            if col == 3 or row == 3: return 1
            if data[x][x]==m:    #斜連線檢查
                slant1 = slant1 + 1
            if data[x][2-x]==m:  #斜連線檢查
                slant2 = slant2 + 1
            if slant1==3 or slant2 ==3: return 1
    return -1
```

```
def draw(data):              #畫棋盤
    for x in range(3):
        for y in range(3):
            if data[x][y]==-1:
                print('_', end=' ')
            else:
                print(data[x][y], end=' ')
    print()
    print()
```

# OX遊戲

## □ 檢查m 是否可連線

```
def getXY(data, m):      #檢查 m 是否可連線
    temp = copy.deepcopy(data) #深度複製
    for x in range(3):
        for y in range(3):
            if temp[x][y]==-1: #有空位
                temp[x][y]=m  #下這個位置檢查是否連線
                if check(temp, m)==1:
                    return x, y #找到下的位置
                temp[x][y]==-1 #沒有連線恢復空位
    return -1, -1          #沒有找到位置
```

## □ 檢查是否結束

```
def over(data):           #檢查空位數 0
    sum=0
    for i in range(3):     #空位個數檢查
        sum += data[i].count(-1)
    return 1 if sum==0 else 0
```

# OX遊戲

## □ 檢查使用者是否下在合法位置

```
def isValid(data, x, y):      #下在合法位置
    if (x not in range(0, 3)) or (y not in range(0, 3)):
        return 0
    elif data[x][y]!=-1:
        return 0
    return 1
```

## □ 使用者輸入

```
def userMove(data):
    while (True):
        print('x y:', end="")
        s = input().split()
        x, y = int(s[0]), int(s[1])  # 輸入XY座標
        if isValid(data, x, y)==0: #檢查輸入合法座標
            print('input again:')
        else:
            data[x][y]=1
            print()
            return x, y                #回傳合法座標
```

# OX遊戲

## □ 電腦第二次下(第一次下在中間)

```
def computerSecondMove(data, x, y):      #電腦第二次下
    if (x==0 and y==0) or (x==2 and y ==2): #下在對手的非斜對面
        data[0][2]=0
    else:
        data[0][0]=0
```

## □ 電腦下

```
def computerMove(data):
    x, y = getXY(data, 0)  #檢查電腦 0 是否可連線
    if x!=-1 and y!=-1:
        data[x][y]=0          #電腦下在可連線位置
        return
    x, y = getXY(data, 1)  #檢查對手是否可連線位置
    if x!=-1 and y!=-1:
        data[x][y]=0          #下在對手可連線位置，阻止對手連線
        return
```

# OX遊戲

## □ 遊戲流程

```
def game():
    data = [[-1]*3 for i in range(3)]
    data[1][1]=0          #電腦 0 先下在中間
    draw(data)
    x, y = userMove(data) #對手下
    computerSecondMove(data, x, y)
    draw(data)
    while (True):
        if (check(data,1)==1):
            print('1 win')
            break
        elif (check(data,0)==1):
            print('0 win')
            break
        elif (over(data)==1): #平手結束
            print('Tie')
            break
        userMove(data)      #對手下
        draw(data)          #畫目前棋盤
        computerMove(data) #電腦下
        draw(data)

game()
```

# OX遊戲

## □ 遊戲流程

```
def f(data, who , x, y):  
    temp = copy.deepcopy(data)  
    temp[x][y]=who  
    #print(temp)  
    if who==0 and check(temp, 0)==1: return True  
    elif who==1 and check(temp, 1)==1: return False  
    elif over(temp)==1: return True  
    else:  
        for i in range(9):  
            x, y = i//3, i%3  
            if (temp[x][y]==-1):  
                if f(temp, int(not who), x, y)==True: return True  
        return False
```

```
def computerMove2(data):  
    for i in range(9):  
        x, y = i//3, i%3  
        if data[x][y]==-1 and f(data, 0, x, y)==True:  
            data[x][y]=0  
    return
```

```
def testF():  
    data = [[1, -1, 1],  
           [1, 0, 1],  
           [0, -1, 0]]  
  
    r = f(data, 0, 0, 1)  
    print(r)
```

# Python

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# zip

- `zip([iterable, ...])` , 將參數壓成一個tuple，回傳成list

```
a = [1,2,3]
b = [4,5,6]
c = [7,8, 9, 10]
zipped = zip(a,b)
print([x for x in zipped])
print([x for x in zip(a,c)]) # truncate the extra element 10 in c
```

#二維矩陣行列互換

```
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print([ [row[col] for row in a] for col in range(len(a[0]))])
```

Output

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

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

# zip

## □ 以指定機率獲取元素

```
import random
def random_pick(seq, probabilities):
    x = random.uniform(0,1)
    print(x)
    print(list(zip(seq, probabilities)))
    cumulative_prob = 0.0
    for item, item_prob in zip(seq, probabilities):
        print(item, item_prob)
        cumulative_prob+=item_prob
        if x < cumulative_prob:
            return item

for i in range(5):
    print(random_pick('abc', [0.1, 0.3, 0.6]), end=' ')
```

### Output

```
0.6152055150729896
[('a', 0.1), ('b', 0.3), ('c', 0.6)]
a 0.1
b 0.3
c 0.6
c 0.525939867714389
[('a', 0.1), ('b', 0.3), ('c', 0.6)]
a 0.1
b 0.3
c 0.6
c 0.2658650623120966
[('a', 0.1), ('b', 0.3), ('c', 0.6)]
a 0.1
b 0.3
b 0.45398064248829084
[('a', 0.1), ('b', 0.3), ('c', 0.6)]
a 0.1
b 0.3
c 0.6
c 0.8133785661416347
[('a', 0.1), ('b', 0.3), ('c', 0.6)]
a 0.1
b 0.3
c 0.6
c
```

# Exercise

- 輸入兩個 lists，使用 `zip()` 串成 dictionary

```
def create_dictionary(alist, blist):
    return dict(zip(alist, blist))

names=['john', 'Eric', 'Mary', 'Ethon', 'Edward']
scores = [90, 50, 80, 40, 100]
a=create_dictionary(names, scores)
print(a)
```

# Exercise

- 傳入字串，使用dictionary計算字出現的頻率

```
# count the frequency of words in a string using dictionary
def word_count(s):
    words=s.split(' ')
    d = {}
    for word in words:
        d[word] = d.get( ) + 1
    return d
a=word_count('This a is a test')
print(a)
x=a.keys()
y=a.values()
print(x, y)
```

```
{'This': 1, 'a': 2, 'is': 1, 'test': 1}
dict_keys(['This', 'a', 'is', 'test']) dict_values([1, 2, 1, 1])
```

# Exercise

□ 輸入字串，計算word出現頻率。

- 產生兩個list，no\_duplicated\_list存無重複的word，word\_count\_list 存字頻
- 使用 zip() 組合兩個list產生字典

```
def word_count(s):
    words=s.split(' ')
    no_duplicated_list=[]
    word_count_list = []
    for i in range(len(words)):
        if words[i] not in no_duplicated_list:
            no_duplicated_list.append(words[i])
            word_count_list.append(1)
    for i in range(len(word_count_list)):
        for j in range(len(words)):
            if words[j] == no_duplicated_list[i]:
                word_count_list[i] += 1
    print(no_duplicated_list)
    print(word_count_list)
    return zip(no_duplicated_list, word_count_list)
a=dict(word_count('This is a test ... This is not a test'))
print(a)
```

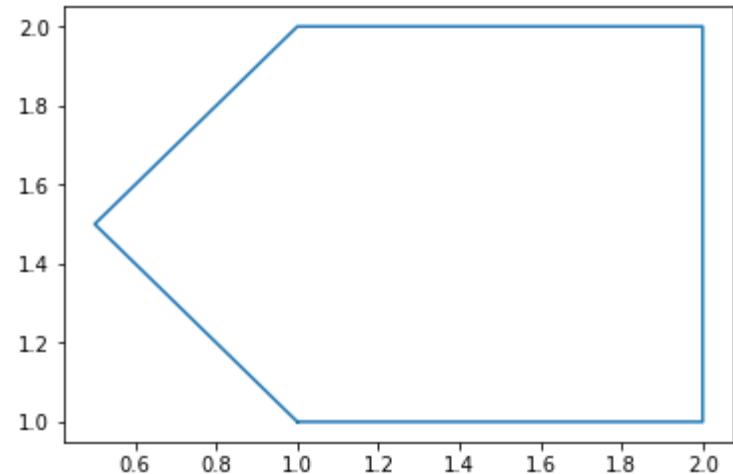
```
['This', 'is', 'a', 'test', '...', 'not']
[2, 2, 2, 2, 1, 1]
{'This': 2, 'is': 2, 'a': 2, 'test': 2,
...': 1, 'not': 1}
```

# Unzipping the Value Using zip()

```
import matplotlib.pyplot as plt  
coord = [[1,1], [2,1], [2,2], [1,2], [0.5,1.5]]  
coord.append(coord[0]) #繞成一圈  
print(coord)  
x, y = zip(*coord) #解壓縮造出x, y lists  
print(x)  
print(y)  
plt.figure()      #繪圖版  
plt.plot(x, y)    #畫折線圖  
plt.show()         #顯示繪圖
```

pip install matplotlib

```
[[1, 1], [2, 1], [2, 2], [1, 2], [0.5, 1.5], [1, 1]]  
(1, 2, 2, 1, 0.5, 1)  
(1, 1, 2, 2, 1.5, 1)
```



# Python

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# Panda

## □ pandas 套件

- 基於NumPy 的資料分析工具，提供大量函式庫和標準資料模型、三個資料結構：**Panel**、**DataFrame** 與 **Series**。
- Series處理時間序列如感測器資料，建立索引的一維陣列。
- DataFrame 處理結構化(Table)資料，有列索引與欄標籤的二維資料集，例如關聯式資料庫、CSV 等。
- Pandas Series是索引資料的一維陣列，可從串列或陣列創建，包含一系列值和一系列索引，可使用values和index屬性存取。
- Pandas Series可看成一般化NumPy陣列，也可當特殊字典。
- Pandas的DataFrame是NumPy陣列的一般化，是Python字典的特殊化。
- 當字典將鍵映射到值時，DataFrame將行名稱映射到行資料的 Series 。

# Panda

## □ pandas 套件

- 處理異質資料讀取、轉換、輸出整合，如：從列欄試算表中尋值、讀取資料庫進入 Dataframe，處理後存回資料庫。。
- 載入資料結構物件，快速進行資料前處理，如資料補值，空值去除或取代等。
- 處理浮點及非浮點資料類型的缺失值(NaN)。
- 可刪除或插入列，自動對齊資料，對資料集進行拆分組合操作。
- 轉換Python和NumPy不同索引的資料為DataFrame物件，進行直觀合併，連接資料集，重新定義資料集形狀和轉置。

# Panda

```
# 讀入 csv 文字檔
import pandas as pd
csv_file = "https://storage.googleapis.com/learn_pd_like_tidyverse/gapminder.csv"
gap = pd.read_csv(csv_file)
print(type(gap))
print(gap.head())
```

country	continent	year	lifeExp	pop	gdpPercap
Afghanistan	Asia	1952	28.801	8425333	779.4453
Afghanistan	Asia	1957	30.332	9240934	820.853
Afghanistan	Asia	1962	31.997	10267083	853.1007
Afghanistan	Asia	1967	34.02	11537966	836.1971
Afghanistan	Asia	1972	36.088	13079460	739.9811
Afghanistan	Asia	1977	38.438	14880372	786.1134
Afghanistan	Asia	1982	39.854	12881816	978.0114

# Panda

## □ DataFrame °

```
# 讀入 excel 試算表  
xlsx_file = "https://storage.googleapis.com/learn_pd_like_tidyverse/gapminder.xlsx"  
gapminder = pd.read_excel(xlsx_file)  
print(type(gapminder))  
gapminder.head()
```

df.shape : 這個 DataFrame 有幾列有幾欄  
df.columns : 這個 DataFrame 的變數資訊  
df.index : 這個 DataFrame 的列索引資訊  
df.info() : 關於 DataFrame 的詳細資訊  
df.describe() : 關於 DataFrame 各數值變數的描述統計

# Panda

## □ DataFrame。

用 list 標註變數名稱可將變數從DataFrame選出，實踐 SQL select，例如選出 country 與 continent 欄位：

```
print(gap[['country', 'continent']])
```

只選一個變數且沒有以 list 標註，同樣能選出欄位資料，但型別變為 Series

```
country = gapminder['country']
print(type(country))
```

呼叫 DataFrame 聚合函數針對欄位計算sum，例如計算 2007 年全球人口總數：

```
gapminder[gapminder['year'] == 2007][['pop']].sum()
```

# Panda

## □ DataFrame °

計算 2007 年全球的平均壽命、平均財富：

```
gapminder[gapminder['year'] == 2007][['lifeExp', 'gdpPercap']].mean()
```

呼叫 DataFrame 的 groupby 實踐 group\_by 功能，例如計算 2007 年各洲人口總數：

```
gapminder[gapminder['year'] == 2007].groupby(by = 'continent')['pop'].sum()
```

# Pandas 聚合操作

聚合操作	說明
count()	項目總數
first() , last()	第一個和最後一個項目
mean() , median()	平均值和中位數
min() , max()	最小值和最大值
std() , var()	標準差和變異數
mad()	平均絕對偏差
prod()	所有項目的乘積
sum()	所有項目的總和

$$(MAD) = \frac{1}{n} \sum_{i=1}^n |x_i - m(X)|$$

$m(X)$ 是資料集中心趨勢(central tendency)，可取均值(mean)、中位數(median)或眾數(mode)，選取不同中心描述函數對MAD有影響。

# Pandas 排序

- Pandas Series是一個類似一維陣列的物件，包含資料陣列和相關資料標籤陣列。可透過傳入索引重新排序Series(未找到的索引是NaN)，也按索引對Series進行排序。
- 用 sort\_index()可按索引對DataFrame進行排序，也可指定 ascending=False進行降序排序。
- 用 sort\_values方法可以按行對DataFrame的值進行排序。

# Pandas 索引與資料選取

- Pandas Series物件可用於一維NumPy陣列與標準Python字典。像字典一樣，Series物件提供從一組鍵到一組值的映射，用類似字典語法修改。
- 將Series當作一維陣列，透過與NumPy陣列相同的基本機制提供陣列樣式的項目選擇，即slices、masking和fancy indexing。索引採用指令loc、iloc、和ix。
- DataFrame類似二維或結構化陣列與共享相同索引的Series結構字典。
- 可將DataFrame視為增強二維陣列。可使用values屬性檢查原始底層資料陣列。
- Pandas使用iloc索引器，可將底層陣列索引成像簡單NumPy陣列(使用隱式Python樣式索引)，但結果保留DataFrame索引和行標籤；使用loc索引器，可使用顯式索引和行名稱，以類似陣列的樣式索引基礎資料。

# Panda

## □ DataFrame 。

DataFrame 空值為NA。

選取標籤為A和C的列，且選完類型還是dataframe

```
df = df.loc[:, ['A', 'C']]  
df = df.iloc[:, [0, 2]]
```

選取標籤為且只取前兩行，選完類型還是dataframe

```
df = df.loc[0:2, ['A', 'C']]  
df = df.iloc[0:2, [0, 2]]
```

loc根據dataframe的標籤選取列，

iloc是根據標籤所在的位置，從0開始計數。

" :"表示選取整列，

"0:2"表示選取第0行到第1行，2不在範圍內。

# 合併

## □ DataFrame °

```
import pandas as pd
import numpy as np
ser1 = pd.Series(['A', 'B', 'C'], index=[1, 2, 3])
ser2 = pd.Series(['D', 'E', 'F'], index=[4, 5, 6])
print(pd.concat([ser1, ser2]))
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
b=np.array([[3,6,7],[1,4,7],[2,5,9]])
df1=pd.DataFrame(a,index=['r0','r1','r2'],columns=list('ABC'))
print(df1)
df2=pd.DataFrame(b,index=['r4','r5','r6'],columns=list('ABC'))
print(df2)
df = pd.concat([df1, df2])
print(df)
df3=pd.DataFrame(b,index=['r0','r1','r2'],columns=list('ABC'))
print(df3)
df = pd.concat([df1, df3], axis=1)
print(df)
```

```
1  A
2  B
3  C
4  D
5  E
6  F
dtype: object
```

```
   A  B  C
r0 1  2  3
r1 4  5  6
r2 7  8  9
   A  B  C
r4 3  6  7
r5 1  4  7
r6 2  5  9
```

```
   A  B  C
r0 1  2  3
r1 4  5  6
r2 7  8  9
r4 3  6  7
r5 1  4  7
r6 2  5  9
```

```
   A  B  C
r0 3  6  7
r1 1  4  7
r2 2  5  9
   A  B  C  A  B  C
r0 1  2  3  3  6  7
r1 4  5  6  1  4  7
r2 7  8  9  2  5  9
```

# 合併

## □ DataFrame °

```
import pandas as pd
import numpy as np
ser1 = pd.Series(['A', 'B', 'C'], index=[1, 2, 3])
ser2 = pd.Series(['D', 'E', 'F'], index=[4, 5, 6])
print(pd.concat([ser1, ser2]))
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
b=np.array([[3,6,7],[1,4,7],[2,5,9]])
df1=pd.DataFrame(a,index=['r0','r1','r2'],columns=list('ABC'))
print(df1)
df2=pd.DataFrame(b,index=['r4','r5','r6'],columns=list('ABC'))
print(df2)
df = pd.concat([df1, df2])
print(df)
df3=pd.DataFrame(b,index=['r0','r1','r2'],columns=list('ABC'))
print(df3)
df = pd.concat([df1, df3], axis=1)
print(df)
```

```
1  A
2  B
3  C
4  D
5  E
6  F
dtype: object
```

```
   A  B  C
r0 1  2  3
r1 4  5  6
r2 7  8  9
   A  B  C
r4 3  6  7
r5 1  4  7
r6 2  5  9
```

```
   A  B  C
r0 1  2  3
r1 4  5  6
r2 7  8  9
r4 3  6  7
r5 1  4  7
r6 2  5  9
```

```
   A  B  C
r0 3  6  7
r1 1  4  7
r2 2  5  9
   A  B  C  A  B  C
r0 1  2  3  3  6  7
r1 4  5  6  1  4  7
r2 7  8  9  2  5  9
```

# 切割

```
import pandas as pd
import numpy as np
a=np.array([['John Kuo',2,3],['Tom Lee',5,6],['Mary Lin',8,9]])
df1=pd.DataFrame(a,index=['r0','r1','r2'],columns=['name', 'id',
'salary'])
print(df1)
df = df1['name'].str.split(' ',expand=True)
df = df.rename(columns = {0:'first', 1:'last'})
print(df)
del df1['name']
df1 = df1.reset_index(drop=True)
df = df.reset_index(drop=True)
newdf = pd.concat([df, df1], sort=True, axis=1)
print(newdf)
```

	name	id	salary
r0	John Kuo	2	3
r1	Tom Lee	5	6
r2	Mary Lin	8	9

	first	last
r0	John	Kuo
r1	Tom	Lee
r2	Mary	Lin

	first	last	id	salary
0	John	Kuo	2	3
1	Tom	Lee	5	6
2	Mary	Lin	8	9

# Python NumPy

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# NumPy

## □ NumPy

- Python的擴充程式庫，代表 "Numeric Python"。
- 陣列或矩陣運算演算法，效率與編譯過C語言效率一樣。
  - 線性代數、傅立葉轉換、亂數產生等功能
- 廣播(broadcasting): 不同大小的陣列之間的運算
- ndarray"(n-dimensional array): N維陣列物件，同質且固定大小，支援多維矩陣運算

# NumPy 建立陣列與存取元素

## □ NumPy

- 傳送串列(list)或元組(tuple)給np.array，可建立一維陣列。
- 透過np.arange建立一維NumPy陣列。
- 透過np.linspace建立間隔相同的一維陣列。

```
import numpy as np  
a = np.array([1, 2, 3, 4])  
b = np.arange(0, 30, 3)  
c = np.linspace(0, 10, 5)  
print(a)  
print(b)  
print(c)
```

```
[1 2 3 4]  
[ 0  3  6  9 12 15 18 21 24 27]  
[ 0.  2.5  5.  7.5 10.]
```

# NumPy 建立陣列與存取元素

## □ NumPy

- 陣列以索引(Index)存取元素，與Python索引用法相同。

```
import numpy as np  
c = np.linspace(0, 10, 5)  
print(c)  
print(c[0]) # 取得在index位置0的元素  
print(c[1]) # 取得在index位置1的元素  
print(c[2:6]) # 取得在index位置2-5(含2不含6)  
print(c[1:-1:2]) #取陣列index位置1,3,5,...隔2取子陣列
```

```
[ 0.  2.5  5.  7.5 10.]  
0.0  
2.5  
[ 5.  7.5 10.]  
[2.5 7.5]
```

# NumPy 建立陣列與存取元素

## □ Numpy 資料型態

- type() 回傳參數的資料型態
- .dtype 回傳陣列中元素的資料型態
- .astype() 進行資料型態轉換

```
import numpy as np  
x = np.array([1, 2, 3, 4, 5, 6])  
y = x.astype('float64')  
print(x)  
print(type(x))  
print(y.dtype)  
print(y)
```

```
[1 2 3 4 5 6]  
<class 'numpy.ndarray'>  
float64  
[1. 2. 3. 4. 5. 6.]
```

# NumPy 建立陣列與存取元素

## □ NumPy

- 經由多維串列(list)給np.array，建立多維陣列。
- 透過np.shape改變所需各種外形(shape)。

```
import numpy as np
two_dim = np.array([[ 1,  2,  3,  4],
                    [ 5,  6,  7,  8],
                    [ 9, 10, 11, 12]])
three_dim = np.array([[[ 1,  2,  3,  4],
                        [ 5,  6,  7,  8],
                        [ 9, 10, 11, 12]],
                     [[13, 14, 15, 16],
                      [17, 18, 19, 20],
                      [21, 22, 23, 24]]])
print(two_dim.shape)
print(three_dim.shape)
two_dim.reshape(4,3)
print(two_dim)
print(two_dim.shape)
```

```
(3, 4)
(2, 3, 4)
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
(3, 4)
```

# Exercise

## □ 輸出？

```
import numpy as np
a = np.arange(1, 26).reshape(5, 5)
print(a)
print(a[0, 1:4]) # 獲取row 0(第0個row)第1~3個元素(不含4)一維子陣列
print(a[1:4, 0]) # 獲取由column 0(第0個column)第1~3個元素(不含4)一維子陣列
print(a[::-2,::2]) # 獲取二維陣列，由row與column中隔個元素產生(0、2、4列與0、2、4行，也就是(0,0), (0,2), (0,4), (2,0), (2,2), (2, 4), (4,0), (4,2), (4,4)等9個元素)
print(a[:, 1]) # 獲取整個column 1中元素產生的一維子陣列
```

# Exercise

- 使用 np.arange 建立 5 的倍數一維  $1 \times 25$  陣列，利用 reshape 轉為  $5 \times 5$  陣列
- 使用 np.linspace 建立  $0 \sim 10$ ，一維  $1 \times 12$  陣列，利用 reshape 轉為  $3 \times 4$  陣

# Exercise

## □ 輸出？

```
import numpy as np
a = np.arange(0, 60).reshape((4, 3, 5))
print("陣列a\n",a)

print("a[0, 1, 2]=",a[0, 1, 2]) # 存取a[0][1][2]
print("a[:,2]=\n",a[:,2]) # 二維陣列中row 2(第三個橫)組成的串列
print("a[:, :, 2]=\n",a[:, :, 2]) # 二維陣列中column 2(第三直)組成的串列
print("a[::2, ::2, ::2]=\n",a[::2, ::2, ::2]) # 產生一個三維陣列，由橫與直中
隔個元素產生
```

# NumPy 建立陣列與存取元素

## □ Numpy 陣列取值

```
#np.ones([2, 3]) 設定 2x3 值均為 1  
# x.reshape([3, 2]) 將 2x3 改為 3x2  
#z[:,0] 取矩陣 z 的所有行的第 0 列元素，  
#z[:,1] 取所有行的第 1 列的元素。  
#z[:,m:n:s]即取矩陣 z 的所有行中，  
# 第 m 到 n-1 列資料，含左不含右。  
# s 是 step, 空的就是全取，-1 是倒轉  
#z[0,:]取矩陣 z 的第 0 行所有元素，  
#z[1,:]取矩陣 z 的第 1 行所有元素。
```

# NumPy 建立陣列與存取元素

## □ Numpy 陣列取值

```
#([column, row])
x = np.ones([2, 3])
print(x)
y = x.reshape([3, 2])
print(y)
z=np.array([[1,2,3,4],
            [5,6,7,8],
            [9,10,11,12],
            [13,14,15,16],
            [17,18,19,20]])
print(z[:,::-1])
print(z[:,0])
print(z[:,1])
print(z[:,2:3])
print(z[0][2], z[0,2])
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

```
[[ 4  3  2  1]
 [ 8  7  6  5]
 [12 11 10  9]
 [16 15 14 13]
 [20 19 18 17]]
```

```
[ 1  5  9 13 17]
```

```
[ 2  6 10 14 18]
```

```
[[ 3]
 [ 7]
 [11]
 [15]
 [19]]
```

# NumPy 操作

```
import numpy as np
x = np.ones([3, 4])
print(x, '\n')
y = x.reshape([4, 3])
print(y, '*\n')
z = y + 1
print(z, '\n')
h = np.random.randint(50, size=(4, 3))
print(h, '$\n')
g = z[0:2, 0] + 2 * h[1:3, 1]
print(z[0:2, 0])
print(2 * h[1:3, 1])
print(g, '\n')

b = z[0:2, 0:2] + 2 * h[1:3, 1:3]
print(z[0:2, 0:2])
print(2 * h[1:3, 1:3])
print(b, '~\n')
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]

[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]] *

[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]

[[ 2 10  1]
 [47  5 36]
 [23 10  6]
 [11 10  3]] $

[2. 2.]
[10 20]
[12. 22.]

[[2. 2.]
 [2. 2.]]
[[10 72]
 [20 12]]
[[12. 74.]
 [22. 14.]] ~
```

# NumPy 加總

## □ 加總某一個值

```
import numpy as np
def test():
    data = [[1, 0, 1],
            [1, 0, 1],
            [0, 1, 0]]
    a = np.array(data)
    for i in range(3):
        print((a[:,i]==0).sum())

test()
```

1  
2  
1

```
def over(data):          #檢查空位數
    a = np.array(data)
    return 1 if (a==0).sum()==0 else 0
```

```
def testOver():
    data = [[1, 1, 1],
            [1, 0, 1],
            [0, 1, 0]]
    print(over(data))
    data = [[-1, 1, 1],
            [1, 0, 1],
            [0, 1, 0]]
    print(over(data))
```

testOver()

1  
0

# NumPy 加總

## □ 加總某一個值

```
import copy
import numpy as np
def check(data, m):          #檢查連線
    slant1 = slant2 = 0
    a = np.array(data)
    for i in range(3):
        if (a[:,i] == m).sum() == 3 or (a[i,:] == m).sum() == 3: return 1
        if data[i][i] == m:      #斜連線檢查
            slant1 = slant1 + 1
        if data[i][2-i] == m:   #斜連線檢查
            slant2 = slant2 + 1
    if slant1 == 3 or slant2 == 3: return 1
    return -1
```

```
def testCheck():
    data = [[1, 0, 1],
            [1, 0, 1],
            [0, -1, 1]]
    r = check(data, 1)
    print(r)
    r = check(data, 0)
    print(r)
    data = [[0, 0, 1],
            [1, 0, 1],
            [0, -1, 0]]
    r = check(data, 1)
    print(r)
    r = check(data, 0)
    print(r)
```

1  
-1  
-1  
1

# NumPy 隨機

```
np.random.random((1000, 20))      #從0-20中隨機產生一千個浮點數。  
numpy.random.rand(d0, d1, ..., dn) : #產生一個[0,1)之間的隨機浮點數或N維浮點陣列。  
numpy.random.randn(d0, d1, ..., dn) :  
    產生一個浮點數或N維浮點陣列，取數範圍：正態分佈的隨機樣本數。  
numpy.random.standard_normal(size=None) :  
    產生一個浮點數或N維浮點陣列，取數範圍：標準正態分佈隨機樣本  
numpy.random.randint(low, high=None, size=None, dtype='l') :  
    產生一個整數或N維整數陣列，取數範圍：若high不為None時，取[low,high)之間隨  
機整數，否則取值[0,low)之間隨機整數。  
np.random.randint(5, size=(2, 4))  
numpy.random.random_integers(low, high=None, size=None) :  
    產生一個整數或一個N維整數陣列，取值範圍：若high不為None，則取[low,high]之  
間隨機整數，否則取[1,low]之間隨機整數。  
numpy.random.random_sample(size=None) :  
    產生一個[0,1)之間隨機浮點數或N維浮點陣列。  
numpy.random.choice(a, size=None, replace=True, p=None) :  
    從序列中獲取元素，若a為整數，元素取值為np.range(a)中亂數；若a為陣列，取值  
為a陣列元素中隨機元素  
rdm = numpy.random.RandomState(seed=None) # 產生偽隨機狀態種子
```

# NumPy 聚合操作

- NumPy陣列聚合操作。
  - np.sum語法與Python內建總和相同，但較快。
  - np.min與np.max，最小值與最大值，較Python內建快。
  - 二維陣列，預設是對整個陣列聚合操作，但也可依行或列操作。
- 聚合函數採用參數axis指定聚合的軸。
  - axis = 0算出一個row，
  - axis = 1算出一個column。

# Exercise

## □ 測試以下程式輸出

```
import numpy as np
import time
a= np.array([[ 1,  2,  3,  4],
             [ 5,  6,  7,  8],
             [ 9, 10, 11, 12]])
print(np.sum(a,axis = 0)) #預設算出 row
print(np.sum(a,axis = 1))
print(np.sum(a))          #全部加總
print(a.sum())            #使用本身sum
big_array = np.random.rand(90000000)
print("Python內建版本：")
start = time.process_time()
print(sum(big_array))
print(time.process_time()-start) #計算執行時間
print("Numpy版本：")
start = time.process_time()
print(np.sum(big_array))
print(time.process_time()-start)
```

[15 18 21 24]

[10 26 42]

78

78

Python內建版本：

45000051.44385809

6.875

Numpy版本：

45000051.4438571

0.09375

# NumPy 聚合列表

功能名稱	NaN安全版	說明
np.sum	np.nansum	計算元素之和
np.prod	np.nanprod	計算元素的乘積
np.mean	np.nanmean	計算元素的中位數
np.std	np.nanstd	計算標準差
np.var	np.nanvar	計算方差
np.min	np.nanmin	找到最小值
np.max	np.nanmax	找到最大值
np.argmin	np.nanargmin	查找最小值索引
np.argmax	np.nanargmax	查找最大值索引
np.median	np.nanmedian	計算元素的中位數
np.percentile	np.nanpercentile	計算元素基於排名的統計數據
np.any	N / A	評估是否有任何元素為真
np.all	N / A	評估所有元素是否都為真

# Exercise

- 輸入N
- 輸入N個整數
- 輸出最大值、最小值、標準差、中位數、百分比

# NumPy 排序

## □ NumPy 快速排序

### ○ np.sort: 回傳陣列的各別排序副本

- axis = 0 對 column 排序，
- axis = 1 對 row 排序。

```
import numpy as np  
x = np.array([[11, 5, 7, 13],  
              [23, 12, 4, 9],  
              [9, 16, 8, 3]])  
  
print(np.sort(x, axis=0))  
  
print(-np.sort(-x, axis=0))  
  
print(np.sort(x, axis=1))
```

```
[[ 9  5  4  3]  
 [11 12  7  9]  
 [23 16  8 13]]  
  
[[23 16  8 13]  
 [11 12  7  9]  
 [ 9  5  4  3]]  
  
[[ 5  7 11 13]  
 [ 4  9 12 23]  
 [ 3  8  9 16]]
```

# NumPy 排序

## □ NumPy快速排序

- np.partition: 取一個陣列和一個數字K；回傳新陣列，分區左邊是最小K個值，剩下值任意在右邊。
- argsort :回傳已排序元素的索引(indices)。
  - axis = 0針對column排序，
  - axis = 1針對row排序。

```
import numpy as np
x = np.array([[11, 5, 7, 13],
              [23, 12, 4, 9],
              [9, 16, 8, 3]])
print(np.argsort(x, axis=0))
print(np.argsort(x, axis=1))
print(np.partition(x, 1, axis=1))
#依第一column對整個矩陣排序
x=x[x[:,0].argsort()]
print(x)
```

[[2 0 1 2]  
[0 1 0 1]  
[1 2 2 0]]

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

[[ 5 7 11 13]  
[ 4 9 23 12]  
[ 3 8 16 9]]

[[ 9 16 8 3]  
[11 5 7 13]  
[23 12 4 9]]

# NumPy 排序

- 設定欄位名稱
- 依欄位名稱排序

```
import numpy as np
data = [(2,'c',89.5, 90),(3,'java',90, 85.5),(4,'php',88, 90)]
title = [('no',int),('name','S10'),('English',float), ('Math', float)] #設定欄位名稱
score = np.array(data,dtype = title)
s1 = np.sort(arr2,order = ['Math','English']) #多列欄位組合排序
s2 = np.sort(arr2,order='English')
print(s1)
print(s2)
```

```
[(3, b'java', 90., 85.5) (4, b'php', 88., 90.) (2, b'c', 89.5, 90. )]
[(4, b'php', 88., 90.) (2, b'c', 89.5, 90.) (3, b'java', 90., 85.5)]
```

# NumPy 排序

- 設定欄位名稱
- 依欄位名稱排序

```
import numpy as np
data = [(2,'c',89.5, 90),(3,'java',90, 85.5),(4,'php',88, 90)]
title = [('no',int),('name','S10'),('English',float), ('Math', float)] #設定欄位名稱
score = np.array(data,dtype = title)
s1 = np.sort(arr2,order = ['Math','English']) #多列欄位組合排序
s2 = np.sort(arr2,order='English')
print(s1)
print(s2)
print(s1.dtype.fields)
print(s1.dtype.names)
for i in s1.dtype.names:
    print('%8s' %i, end=' ')
print()
[(3, b'java', 90., 85.5) (4, b'php', 88., 90.) (2, b'c', 89.5, 90.)]
[(4, b'php', 88., 90.) (2, b'c', 89.5, 90.) (3, b'java', 90., 85.5)]
{'no': (dtype('int32'), 0), 'name': (dtype('S10'), 4), 'English': (dtype('float64'), 14), 'Math': (dtype('float64'), 22)}
('no', 'name', 'English', 'Math')
```

# NumPy 排序

- 印出資料型別
- 印出欄位名稱，

```
print(s1.dtype.fields)
print(s1.dtype.names)
```

```
{'no': (dtype('int32'), 0), 'name': (dtype('S10'), 4), 'English': (dtype('float64'), 14), 'Math': (dtype('float64'), 22)}
('no', 'name', 'English', 'Math')
```

# NumPy 排序

## □ 判斷資料型別，依資料型別印出所有資料

```
for i in s1.dtype.names:  
    print('%8s' %i, end=' ')  
print()  
for s in s1:  
    for i in s:  
        if i.dtype==int:          #i.dtype in (int, float) , 判斷多個  
            print('%8d' %i, end=' ')  
        elif i.dtype==float:      #判斷浮點數  
            print('%8.2f' %i, end=' ')  
        else:  
            print('%8s' %i.decode(), end=' ') # (byte解碼成unicode字串)  
    print()
```

no	name	English	Math
3	java	90.00	85.50
4	php	88.00	90.00
2	c	89.50	90.00

# 讀入 CSV 處理

- 使用 numpy 的 genfromtxt()，讀入 csv 檔案
- 計算敘述統計，劃出直方圖

```
import numpy as np
import matplotlib.pyplot as plt
data = np.genfromtxt('president_heights.csv', delimiter=',', skip_header=1 )
heights = np.array(data[:,2])
print(heights)
print("Mean height:    ", heights.mean())
print("Standard deviation:", heights.std())
print("Minimum height:   ", heights.min())
print("Maximum height:   ", heights.max())
print("25th percentile:  ", np.percentile(heights, 25))
print("Median:          ", np.median(heights))
print("75th percentile:  ", np.percentile(heights, 75))
plt.hist(heights)
plt.title('Height Distribution of US Presidents')
plt.xlabel('height (cm)')
plt.ylabel('number')
```

# 讀入 CSV 處理

- 使用 numpy 的 `genfromtxt()`，讀入 csv 檔案
- 計算敘述統計，劃出直方圖

```
[189. 170. 189. 163. 183. 171. 185. 168. 173. 183. 173. 173. 175. 178.  
183. 193. 178. 173. 174. 183. 183. 168. 170. 178. 182. 180. 183. 178.  
182. 188. 175. 179. 183. 193. 182. 183. 177. 185. 188. 188. 182. 185.  
190.]
```

Mean height: 179.97674418604652

Standard deviation: 7.023178807524852

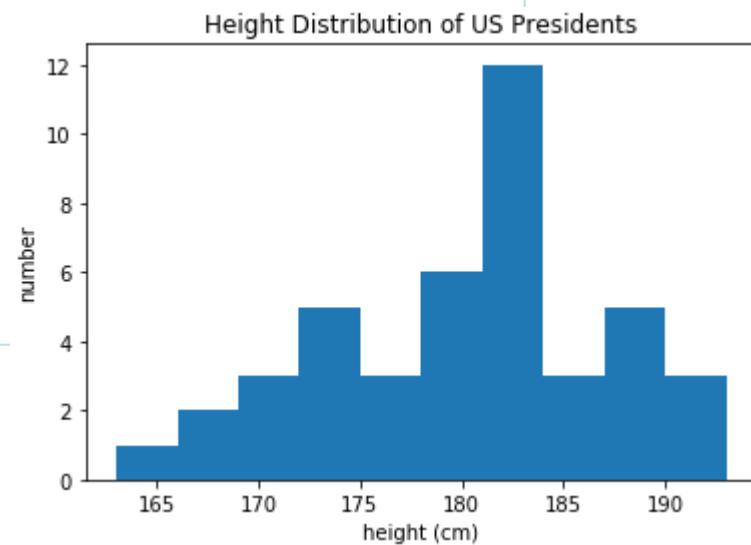
Minimum height: 163.0

Maximum height: 193.0

25th percentile: 174.5

Median: 182.0

75th percentile: 184.0



# Exercise

- 2020年全年毛豬交易行情資料，欄位：
  - total\_amt(成交總頭數)、average\_weight(平均重量)、average\_price(平均價格)

```
import numpy as np
nf1 = np.genfromtxt('pig.csv',delimiter=',',skip_header=1)
print("市場全年成交最高平均重量"+str(nf1[:,1].max(axis=0)))
print("市場全年成交最低平均價"+str(nf1[:,2].min(axis=0)))
print("市場全年總成交頭數"+str(nf1[:,0].sum(axis=0)))
total_sales=(nf1[:,0]*nf1[:,1]*nf1[:,2])           #整列運算
print("市場全年總成交金額"+str(total_sales.sum(axis=0)))
print("市場全年成交平均每頭金額
"+str(total_sales.sum(axis=0)/nf1[:,0].sum(axis=0)))
```

市場全年成交最高平均重量148.18

市場全年成交最低平均價66.35

市場全年總成交頭數3383747.0

市場全年總成交金額32115493123.458595

市場全年成交平均每頭金額9491.103538018237

# NumPy 操作

## □ 擴充矩陣

- numpy.append(arr, values, axis = None)
- arr，要新增元素的陣列
- values，被新增陣列
- axis，指定軸方向進行操作

```
k = np.append(y, h, axis=0)
print(k, '\n')
m = np.append(y, h, axis=1)
print(m, '@\n')
f = np.ones([3, 3])
n = np.append(f, h, axis=0)
print(n, '\n')
r = n.T
print(r, '!@\n')
```

```
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 2. 10.  1.]
 [47.  5. 36.]
 [23. 10.  6.]
 [11. 10.  3.]]
```

```
[[ 1.  1.  1.  2. 10.  1.]
 [ 1.  1.  1. 47.  5. 36.]
 [ 1.  1.  1. 23. 10.  6.]
 [ 1.  1.  1. 11. 10.  3.]] @
```

```
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 2. 10.  1.]
 [47.  5. 36.]
 [23. 10.  6.]
 [11. 10.  3.]]
```

```
[[ 1.  1.  1.  2. 47. 23. 11.]
 [ 1.  1.  1. 10.  5. 10. 10.]
 [ 1.  1.  1. 1. 36.  6.  3.]] !
```

# 擴充欄畫兩子圖

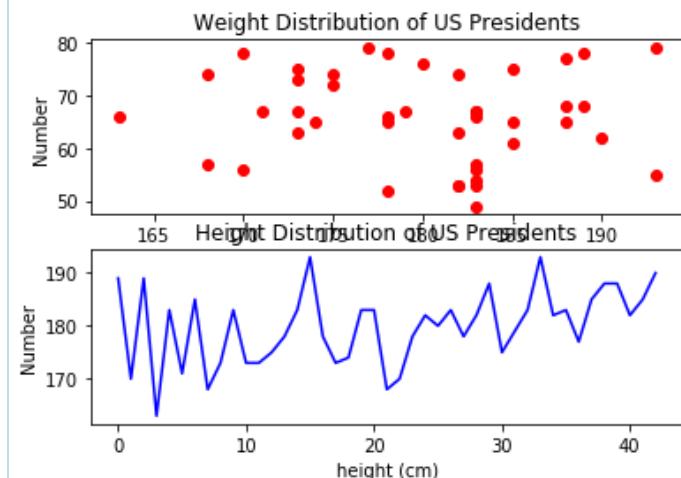
- 讀取 president\_heights.csv，加入體重 隨機 48~80 KG

```
import numpy as np
import matplotlib.pyplot as plt
data = np.genfromtxt('president_heights.csv', delimiter=',',skip_header=1 )
heights = np.array(data[:,2])
weights = np.random.randint(48, 80, size=heights.size)[:,np.newaxis]
data = np.append(data, weights, axis=1)
```

# 擴充欄畫兩子圖

- 劃出2個子圖，身高體重散射圖(紅色)，身高折線圖(藍色)，並設定刻度、XY軸標題

```
fig = plt.figure()
ax = fig.add_subplot(2,1,1)
ax.scatter(data[:,2:3], data[:,3:4], color='r')
plt.title('Weight Distribution of US Presidents')
plt.xlabel('weight (kg)')
plt.ylabel('Number')
ax = fig.add_subplot(2,1,2)
ax.plot(data[:,2], color='b')
plt.title('Height Distribution of US Presidents')
plt.xlabel('height (cm)')
ax.set_ylabel('Number')
```



# Exercise

- 製作20位學生:學號、姓名、Chinese、Math成績
- 讀入學生成績，計算加權平均成績
  - 國文2學分，數學3學分
- 依序(平均成績大~小)印出學號、姓名、平均成績
- 輸出兩科成績直方圖

# 畫子圖與圓

## □ 畫圓

```
import matplotlib.pyplot as plt
def drawBoard():
    fig=plt.figure()          #繪圖版
    ax = fig.add_subplot(2,1,1)
    x, y = (0, 9, 9, 0, 0), (0, 0, 9, 9, 0)
    ax.plot(x, y)            #畫折線圖
    x, y = (0, 9, 9, 0), (3, 3, 6, 6)
    ax.plot(x, y)            #畫折線圖
    x, y = (3, 3, 6, 6), (0, 9, 9, 0)
    ax.plot(x, y)            #畫折線圖
    c = plt.Circle((1.5, 7.5),1)
    ax.add_patch(c)
    ax = fig.add_subplot(2,1,2)
    plt.show()
```

```
drawBoard()
```