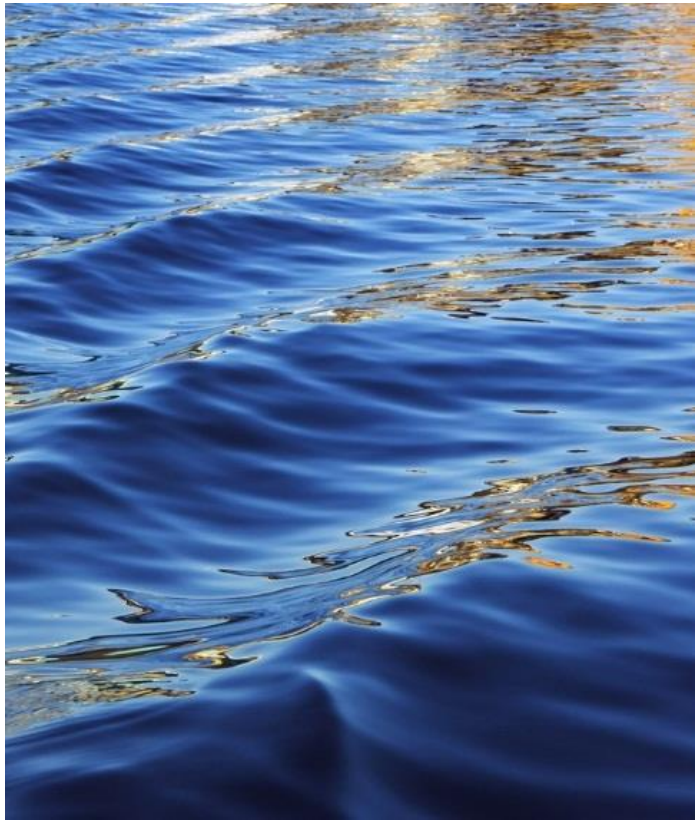
Python

Ricky

8.串列

1. 串列(list)



串列-二維串列

- 二維串列(two dimension list)，可以想成是二維空間
- 下面是一個成績表，很典型的二維表格

姓名	國文	英文	數學	總分
小明	85	90	80	0
小花	95	50	96	0
小羊	100	95	87	0
小華	60	45	70	0

串列-二維串列

- 首先，我們建立一個串列變數，假設命名為sc，
- 在python中，我們可以這樣寫

```
sc = [['小明', 85, 90, 80, 0],
      ['小花', 95, 50, 96, 0],
      ['小羊', 100, 95, 87, 0],
      ['小華', 60, 45, 70, 0],
     ]
```

姓名	國文	英文	數學	總分
[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][1]	[1][1]	[1][2]	[1][3]	[1][4]
[2][1]	[2][1]	[2][2]	[2][3]	[2][4]
[3][1]	[3][1]	[3][2]	[3][3]	[3][4]

串列-二維串列

- 我們以兩個學生來當例子，若要計算總分怎麼算呢?
- 首先，要用到之前一維串列的一次選擇多個項目。
- 第二，利用函數sum() ，回傳串列中總和

```
sc = [['小明', 85, 90, 80, 0],
      ['小花', 95, 50, 96, 0],
     ]
sc[0][4] = sum(sc[0][1:4])
sc[1][4] = sum(sc[1][1:4])
print(sc[0])
print(sc[1])
```

姓名	國文	英文	數學	總分
小明	85	90	80	0
小花	95	50	96	0

最後一個逗號可加也可不加

串列-賦值與切片拷貝

- 我們先列出我和朋友所喜歡的運動

```
mysports = ['basketball', 'baseball']
friendsports = mysports
print("我喜歡的運動      = ", mysports)
print("我朋友喜歡的運動 = ", friendsports)
```

- 加入美式足球football當作喜歡的運動，我的朋友想加入傳統足球soccer當作喜歡的運動，同時列出執行結果。
- 使用append()
- **你發現了甚麼?**

```
mysports = ['basketball', 'baseball']
friendsports = mysports
print("我喜歡的運動      = ", mysports)
print("我朋友喜歡的運動 = ", friendsports)
mysports.append('football')
friendsports.append('soccer')
print("我喜歡的最新運動      = ", mysports)
print("我朋友喜歡的最新運動 = ", friendsports)
```

串列-位址的觀念

- 我們延續上面的程式，小小修改一下，**增加列出串列變數的位址。**
- **使用函數，ID()**
- **發現了甚麼?**

```
mysports = ['basketball', 'baseball']
friendsports = mysports
print("列出mysports位址      = ", id(mysports))
print("列出friendsports位址 = ", id(friendsports))
print("我喜歡的運動      = ", mysports)
print("我朋友喜歡的運動 = ", friendsports)
mysports.append('football')
friendsports.append('soccer')
print(" -- 新增運動項目後 -- ")
print("列出mysports位址      = ", id(mysports))
print("列出friendsports位址 = ", id(friendsports))
print("我喜歡的最新運動      = ", mysports)
print("我朋友喜歡的最新運動 = ", friendsports)
```

串列-位址的觀念

- 在一開始講變數的時候，有說過Python某一個變數X對另外一個變數Y給值時，其實他是把當下變數X的位址複製給了另外一個變數Y。
- 所以他們指到的是同一個位置。因此串列，也是一樣的概念
- 新增加的項目都是指到同一個位置下增加項目

串列-串列的切片拷貝

- 那我們要怎麼真的增加一個新的串列位置呢???
- 切片拷貝(copy)觀念是，執行拷貝後產生新串列物件，當一個串列改變後，不會影響另一個串列的內容，如下：
  - friendsports = mysports[:]

```
mysports = ['basketball', 'baseball']
friendsports = mysports[:]
print("列出mysports位址      = ", id(mysports))
print("列出friendsports位址 = ", id(friendsports))
print("我喜歡的運動      = ", mysports)
print("我朋友喜歡的運動 = ", friendsports)
mysports.append('football')
friendsports.append('soccer')
print(" -- 新增運動項目後 -- ")
print("列出mysports位址      = ", id(mysports))
print("列出friendsports位址 = ", id(friendsports))
print("我喜歡的最新運動      = ", mysports)
print("我朋友喜歡的最新運動 = ", friendsports)
```

串列-淺拷貝(copy)與深拷貝(deepcopy)

- 賦值

假設b=a，a和b位址相同，指向一物件彼此會連動。

- 淺拷貝

假設b=a.copy( )，a和b是獨立的物件，但是它們的子物件元素是指向同一物件

- 深拷貝

假設b=deepcopy(a)，a和b以及其子物件皆是獨立的物件，所以未來不受干擾，使用前需要”import copy”模組，這是引用外部模組，未來會講更多相關的應用。

字串

- 字串，在python中也可以當作一個序列，由字元(character)所組成的序列，不過跟串列不同的是字串內的單一元素內容是不可以更改的。

- 使用正值與負值的索引列出字串元素內容。

```
string = "Python"
# 正值索引
print(" string[0] = ", string[0],
      "\n string[1] = ", string[1],
      "\n string[2] = ", string[2],
      "\n string[3] = ", string[3],
      "\n string[4] = ", string[4],
      "\n string[5] = ", string[5])
# 負值索引
print(" string[-1] = ", string[-1],
      "\n string[-2] = ", string[-2],
      "\n string[-3] = ", string[-3],
      "\n string[-4] = ", string[-4],
      "\n string[-5] = ", string[-5],
      "\n string[-6] = ", string[-6])
# 多重指定觀念
s1, s2, s3, s4, s5, s6 = string
print("多重指定觀念的輸出測試 = ",s1,s2,s3,s4,s5,s6)
```

字串-切片

- 串列切片的概念一樣可以在字串切片上應用

```
string = "Deep Learning"                          # 定義字串
print("列印string第0-2元素      = ", string[0:3])
print("列印string第1-3元素      = ", string[1:4])
print("列印string第1,3,5元素    = ", string[1:6:2])
print("列印string第1到最後元素 = ", string[1:])
print("列印string前3元素        = ", string[0:3])
print("列印string後3元素        = ", string[-3:])
```

字串-函數或方法

- 除了會更動內容的串列函數不可應用在字串以外，其餘都可以沿用

```
sc=[1,2,3,4,5]                                  # 定義串列
sc[2]=99
print("sc :",sc)
string = "Deep Learning"                        # 定義字串
string[2]="E"
print("string : ",string)
```

- 將函數len( )、max( )、min( )應用在字串。

```
string = "Deep Learning"                        # 定義字串
strlen = len(string)
print("字串長度", strlen)
maxstr = max(string)
print("字串最大的unicode碼值和字元", ord(maxstr), maxstr)
minstr = min(string)
print("字串最小的unicode碼值和字元", ord(minstr), minstr)
```

字串-轉換串列、賦值的應用

- List()函數可以將參數內的物件轉成串列
- 轉換成串列後，就可以修改內容

```
string = "Deep Learning"
x = list(string) #轉換串列
print("轉換完的輸出",x)

x[1] = "E"
x[2] = "E"              #修改內容
print("修改後的輸出",x)
```

字串-使用split( )分割字串

str1.split( )　　: 以空格當做分隔符號將字串拆開成串列

str2.split(ch)　　: 以ch字元當做分隔符號將字串拆開成串列

- 將2種不同類型的字串轉成串列，其中str1使用空格當做分隔符號，str2使用"\"當做分隔符號(因為這是逸出字元，所以使用\\)，同時這個程式會列出這2個串列的元素數量。

```
str1 = "Happy birthday to you"
str2 = "C:\Windows\office"

sList1 = str1.split()                          # 字串轉成串列
sList2 = str2.split("\\")                      # 字串轉成串列

print(str1, " 串列內容是 ", sList1)          # 列印串列
print(str1, " 串列字數是 ", len(sList1))     # 列印字數
print(str2, " 串列內容是 ", sList2)          # 列印串列
print(str2, " 串列字數是 ", len(sList2))     # 列印字數
```

字串-元素的組合 join( )

- 有了切割，一定有組合，在各種應用中很常使用join將串列組合起來
- 連接字串.join(串列) :串列元素會用連接字串組成一個字串
- 將串列內容連接。

```
path = ['C:','windows','Temp']
connect = '\\'                    # 路徑分隔字元
print(connect.join(path))
connect = '*'                     # 普通字元
print(connect.join(path))
```

字串-其它方法

- startswith( )：可以列出字串啟始文字是否是特定子字串。
- endswith( )：可以列出字串結束文字是否是特定子字串。
- replace(ch1,ch2)：將ch1字串由另一字串取代。
- 列出字串"CIA"是不是啟始或結束字串，以及出現次數。最後這個程式會將Linda字串用Lxx字串取代，這是一種保護情報員名字不外洩的方法。

```
msg = '''CIA Mark told CIA Linda that the secret USB had given to CIA Peter'''
print("字串開頭是CIA: ", msg.startswith("CIA"))
print("字串結尾是CIA: ", msg.endswith("CIA"))
print("CIA出現的次數: ",msg.count("CIA"))
msg = msg.replace('Linda','Lxx')
print("新的msg內容 : ", msg)
```

in和not in運算式

- **In 、not in主要是用於判斷一個物件是否屬於另一個物件內**
- **物件可以是變數、字串、串列、元祖(Tuple)、字典 (Dict)**
- boolean_value = obj1 in obj2
  :物件obj1在物件obj2內會傳回True
- boolean_value = obj1 not in obj2
  :物件obj1不在物件obj2內會傳回True
- 程式實例:請輸入字元，這個程式會判斷字元是否在字串內。

```
#猜密碼
password = 'applepie'
ch = input("請輸入字元 = ")
print("猜密碼1:in運算式")
#方式一
if ch in password:
    print("輸入字元在密碼中")
else:
    print("輸入字元不在密碼中")

#方式二
print("猜密碼2:not in運算式")
if ch not in password:
    print("輸入字元不在密碼中")
else:
    print("輸入字元在密碼中")
```

in和not in運算式

- 程式實例:這個程式基本上會要求輸入一個水果，如果串列內目前沒有這個水果，就將輸入的水果加入串列內。

```
fruits = ['apple', 'banana', 'watermelon']
fruit = input("請輸入水果 = ")
if fruit in fruits:
    print("這個水果已經有了")
else:
    fruits.append(fruit)
    print("謝謝提醒已經加入水果清單: ", fruits)
```

is或is not運算式

- Is , is not可以用比較兩個物件是否相同，所謂相同並不一定是內容相同而是指物件變數指向相同的記憶體。
- 物件可以是變數、字串、串列、元祖(Tuple)、字典 (Dict)
- boolean_value = obj1 is obj2

  :物件obj1等於物件obj2內會傳回True
- boolean_value = obj1 is not obj2

  :物件obj1不等於物件obj2內會傳回True

is或is not運算式

- 程式實例:
- 整數變數在記憶體位址的觀察
- 這個程式比較特別的是程式執行後，變數x和y值是10，所以可以看到經過id( )函數後，彼此有相同的記憶體位置。
- 變數z，由於值與x和y不相同，所以有不同的記憶體位址
- 經過運算後r的值變為10，最後得到x、y和r不僅值相同同時也指向相同的記憶體位址。

```
x = 10
y = 10
z = 15
r = z - 5
print(" is 用法")
boolean_value = x is y
print("x位址 = %d, y位址 = %d" % (id(x), id(y)))
print("x = %d, y = %d,  %s" % (x, y, boolean_value))#寫法一

boolean_value = x is z
print("x位址 = %d, z位址 = %d" % (id(x), id(z)))
print("x = %d, z = %d,  %s" % (x, z, boolean_value))#寫法一

boolean_value = x is r
print("x位址 = %d, r位址 = %d" % (id(x), id(r)))
print("x = %d, r = %d, " % (x, r), boolean_value)#寫法二
print(" is not 用法")
boolean_value = x is not y
print("x位址 = %d, y位址 = %d" % (id(x), id(y)))
print("x = %d, y = %d, " % (x, y), boolean_value)#寫法二

boolean_value = x is not z
print("x位址 = %d, z位址 = %d" % (id(x), id(z)))
print("x = %d, z = %d, " % (x, z), boolean_value)#寫法二

boolean_value = x is not r
print("x位址 = %d, r位址 = %d" % (id(x), id(r)))
print("x = %d, r = %d, " % (x, r), boolean_value)#寫法二
```

is或is not運算式

- 程式實例:
- 這個範例所使用的3個串列內容均是相同
- mysports和sports1所指位址相同所以會被視為相同物件
- sports2則指向不同位址所以會被視為不同物件
- 使用is指令測試時，不同位址的串列會被視為不同的串列。

```
mysports = ['basketball', 'baseball']
sports1 = mysports              # 賦值
sports2 = mysports[:]           # 切片拷貝新串列
print("我喜歡的運動 = ", mysports, "位址是 = ", id(mysports))
print("運動 1       = ", sports1,  "位址是 = ", id(sports1))
print("運動 2       = ", sports2,  "位址是 = ", id(sports2))
boolean_value = mysports is sports1
print("我喜歡的運動 is 運動 1     = ", boolean_value)

boolean_value = mysports is sports2
print("我喜歡的運動 is 運動 2     = ", boolean_value)

boolean_value = mysports is not sports1
print("我喜歡的運動 is not 運動 1 = ", boolean_value)

boolean_value = mysports is not sports2
print("我喜歡的運動 is not 運動 2 = ", boolean_value)
```

Is-應用在None

- 還記得None 是 未定義的值嗎?在布林中代表Flase。
- 但是他並不是空值

```
x=[] #定義一個空串列
if x is None:
    print("it's None")
else:
    print("it's not None")
```

enumerate物件

- obj = enumerate(iterable[, start = 0]) # 若省略start = 設定，預設索引值是0
- 程式實例:將串列資料轉成enumerate物件，同時列出此物件再將enumerate物件轉成串列的實例，start索引起始值分別為0和10。

```
drinks = ["coffee", "tea", "wine"]
enumerate_drinks = enumerate(drinks)                    # 數值初始是0
print(enumerate_drinks)                                 # 傳回enumerate物件所在記憶體
print("下列是輸出enumerate物件類型")
print(type(enumerate_drinks))                           # 列出物件類型
#串列輸出
print("轉成串列輸出, 初始索引值是 0 = ", list(enumerate_drinks))
#修改start
enumerate_drinks = enumerate(drinks, start = 10)    # 數值初始是10
print("轉成串列輸出, 初始索引值是10 = ", list(enumerate_drinks))
```

HOMEWORK

1. 設計一個帳號管理系統，這個程式分成2個部分，第一個部分是建立帳號，讀者的輸入將會存在accounts串列。第2個部分是要求輸入帳號，如果輸入正確會輸出”歡迎進入系統”，如果輸入錯誤會輸出”帳號錯誤”。
2. 建立ABC … Z大寫字母的字串，然後使用切片取得前6個英文字母，與後20個英文字母。最後組合輸出可以得到新的小寫字母排序。
3. 輸入任意五個分數，請輸出以下結果
    1. 列出分數由高到低的串列
    2. 列出最高分
    3. 列出最低分
4. 輸入一個字串，這個程式會判斷這個字串是否是網址
    1. 網址的格式會有”http://“ 或是 ” https://”