



巨匠直播教學

APCS Python語法基礎班

物件導向入門

www.pcschoolonline.com.tw

本堂教學重點

1. 物件導向程式
2. 數值物件
3. 字串物件
4. 集合物件

課程內容

1. 物件導向初探

- 1-1. 物件導向原理
- 1-2. 建立物件
- 1-3. 修改屬性
- 1-4. 呼叫方法

2. 數值字串物件進階

- 2-1. 數值型態方法
- 2-2. 字串常用方法

3. 集合物件進階

- 3-1. 序列進階操作
- 3-2. 集合進階操作
- 3-3. 字典進階操作

課程內容

1. 物件導向初探

- 1-1. 物件導向原理
- 1-2. 建立物件
- 1-3. 修改屬性
- 1-4. 呼叫方法

2. 數值字串物件進階

- 2-1. 數值型態方法
- 2-2. 字串常用方法

3. 集合物件進階

- 3-1. 序列進階操作
- 3-2. 集合進階操作
- 3-3. 字典進階操作

物件導向程式設計理論

◆ 物件導向程式設計

- ◆ 傳統程序式程式設計是一系列對電腦下達的指令(procedure 流程)
- ◆ 物件導向是一種抽象且擬人化的程式設計的方法
 - 以物件作為程式的基本單元,將物件資料與操作封裝其中
 - 物件能夠接受資料、處理資料並將資料傳達給其它物件
 - 透過物件之間的交互作用來完成工作

物件導向程式設計理論

◆ 物件導向程式設計優點

- ◆ 物件(資料與流程)重複使用
- ◆ 分散式開發
- ◆ 提高程式的靈活性和可維護性
- ◆ 在大型專案設計中廣為應用

Python 物件導向

◆ Python 是物件導向語言

◇ 在 Python 中，所有資料都是物件

- 字串型態(String)：str
- 數值型態(Numeric)：int, float, bool, complex
- 容器型態(Container)：list, set, dict, tuple

◇ 可以重複使用已存在的類別

- 標準函式庫
- 第三方函式庫
- 自行撰寫的其他程式

標準程式庫常用模組：turtle 模組

◆ turtle 模組

- ◆ 初學者在學習Python語言的教學工具
- ◆ 操控一隻烏龜在畫面上移動
- ◆ 提供了視覺化操作的指令，完成繪圖或動畫
- ◆ 包含類別：
 - Turtle
 - Screen

建立Turtle物件

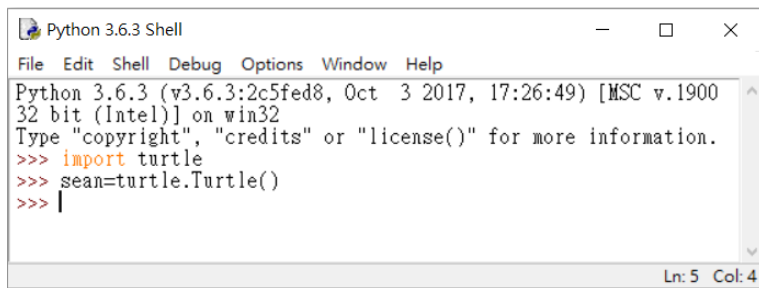
◆ 建立turtle.Turtle物件

◆ 匯入turtle模組

```
import turtle
```

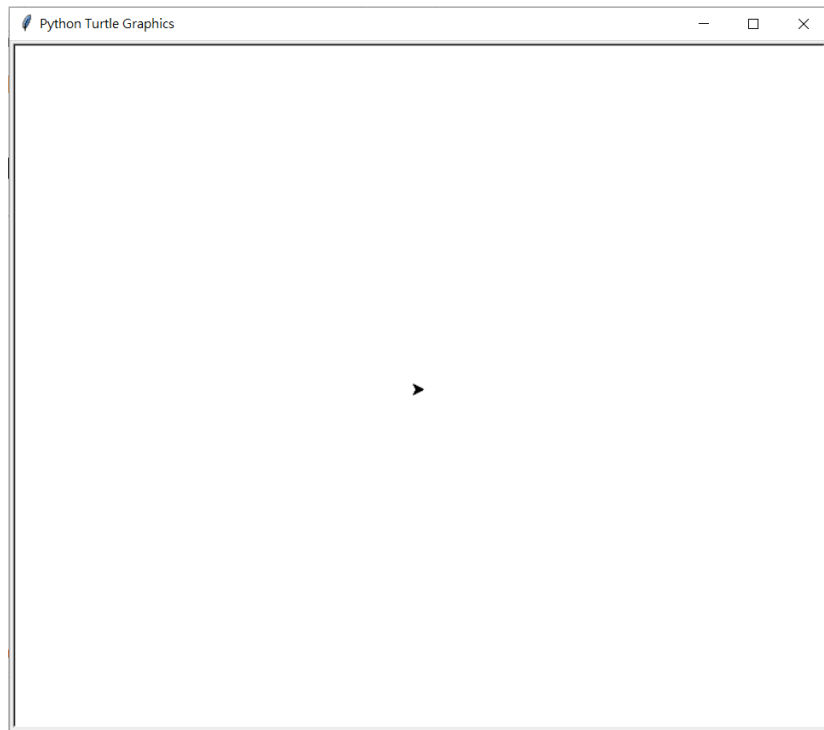
◆ 建立turtle物件

- 物件名稱 = 模組名稱.類別名稱()



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import turtle
>>> sean=turtle.Turtle()
>>> |
```

Ln: 5 Col: 4



物件屬性

◆ 建立物件屬性

- ◆ 建立物件後，透過．在物件上新增屬性

物件名稱.屬性 = 屬性值

◆ 讀取物件屬性

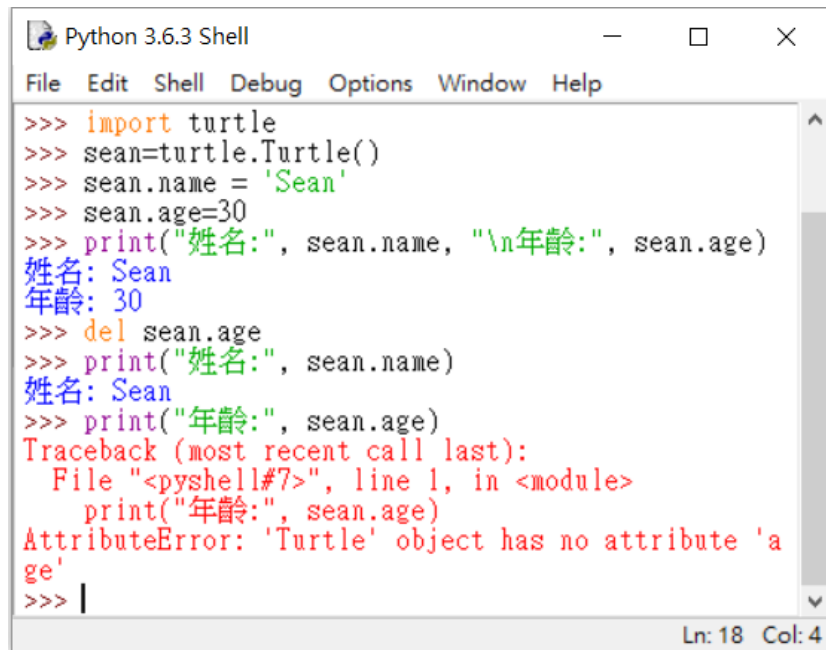
- ◆ 透過物件名稱．讀取物件屬性

物件名稱.屬性

◆ 刪除物件實體屬性

- ◆ 使用del 移除物件屬性

del 物件名稱.屬性



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
>>> import turtle
>>> sean=turtle.Turtle()
>>> sean.name = 'Sean'
>>> sean.age=30
>>> print("姓名:", sean.name, "\n年齡:", sean.age)
姓名: Sean
年齡: 30
>>> del sean.age
>>> print("姓名:", sean.name)
姓名: Sean
>>> print("年齡:", sean.age)
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    print("年齡:", sean.age)
AttributeError: 'Turtle' object has no attribute 'age'
>>> |
```

Ln: 18 Col: 4

Turtle 內建屬性

屬性	說明	設定方法	參數值
shape	游標形狀	shape(str)	'arrow', 'turtle', 'circle', 'square', 'triangle', 'classic'
shapesize	游標大小(寬,長,外框)	shapesize(w,l,o)	正整數
pencolor	游標輪廓顏色	pencolor(color)	'red', 'yellow', 'green', ..., (#RGB), (#RRGGBB)
fillcolor	游標填充顏色	fillcolor(color)	
		color(pen, fill)	
pensize	畫筆粗細	pensize(width)	正整數
speed	移動速度	speed(n)	0~10之間整數，1~10越來越快，0為最快

實體方法

◆ 實體方法

- ◆ 類別中宣告物件可提供的操作行為
- ◆ 實體方法使用方式與函式類似
 - 傳入參數對應、傳回值接收規則均與函式相同
- ◆ 需透過指定物件變數來呼叫
 - 不同物件的執行結果可能不相同

物件名稱.方法名稱(...)

Turtle 常用方法

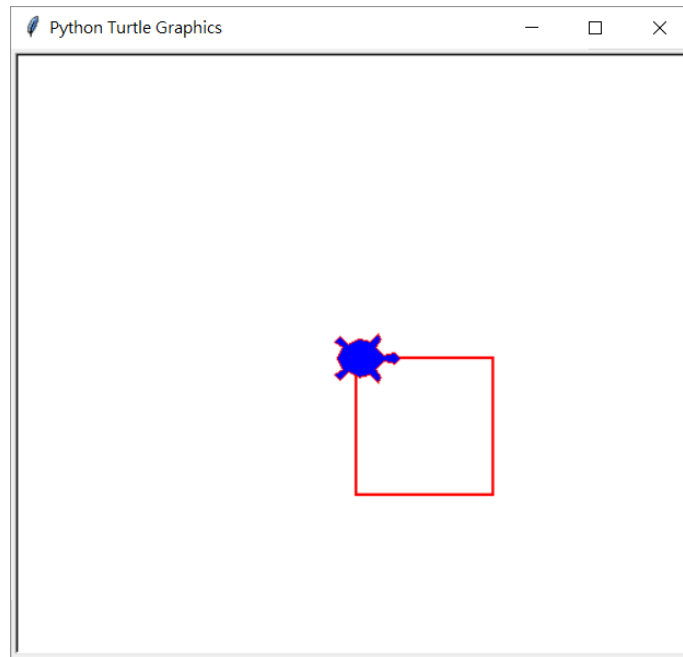
方法	傳回值	說明
forward(distance)	None	向前方移動 distance 個像素距離
backward(distance)	None	向後方移動 distance 個像素距離
right(angle)	None	順時針旋轉 angle 度
left(angle)	None	逆時針旋轉 angle 度
goto(x, y=None)	None	將游標移動到(x , y)的位置
home()	None	將游標移動到初始位置(0, 0)
position()	(x , y)	傳回游標目前的(x , y)位置
circle(radius, extent=None)	None	游標畫一個圓形, radius半徑為正時逆時針繪製, 半徑為負時順時針繪製, extent為繪製弧度, 180表示畫半圓弧
penup()	None	提筆之後路徑不會被畫出
pendown()	None	下筆之後路徑會被畫出
undo()	None	還原上一個動作

程式範例

```
object1.py - D:\PythonJunior...  - □ ×
File Edit Format Run Options Window Help
import turtle

sean = turtle.Turtle()
sean.shape('turtle')
sean.shapesize(2)
sean.pencolor('red')
sean.fillcolor('blue')
sean.pensize(2)

for i in range(1,5):
    sean.forward(100)
    sean.right(90)
|
Ln: 12 Col: 18
```



類別 vs. 物件

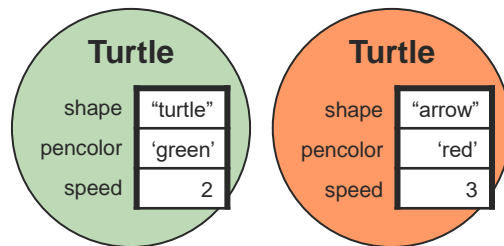
◆ 類別

- ◆ 程式設計師以類別來定義同類型物件的共同藍圖
- ◆ 類別也是一種資料型別定義

Turtle	
shape	
pencolor	
speed	
...	
__init__()	
forward(distance)	
right(angle)	
...	

◆ 物件

- ◆ 物件是類別的一個實體
- ◆ 兩隻烏龜是同一個類別的不同實體



實體成員

◆ 實體成員 instance member

◆ 實體屬性：每個物件各自擁有一份資料

- 物件初始化流程中建立物件屬性

◆ 實體方法：需透過特定物件來操作

- 物件方法中只能使用存在的屬性

程式範例

```
*object2.py - D:\PythonJunior\Examples\...
File Edit Format Run Options Window Help

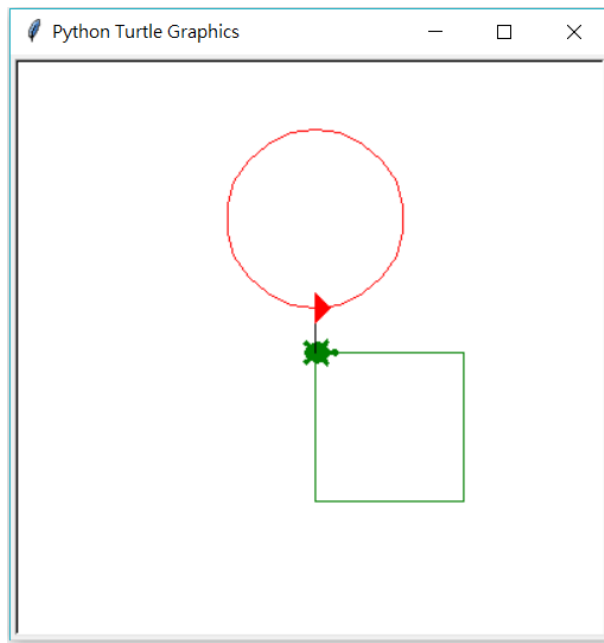
import turtle

def draw_square(a_turtle):
    for i in range(1,5):
        a_turtle.forward(100)
        a_turtle.right(90)

sean = turtle.Turtle()
sean.shape('turtle')
sean.color('green')
draw_square(sean)

amy = turtle.Turtle()
amy.goto(0, 30)
amy.shape('arrow')
amy.color('red')
amy.circle(60)

Ln: 18 Col: 0
```



類別 vs. 物件

◆ 應用程式設計階段

◆ 定義類別及類別之間的互動關係

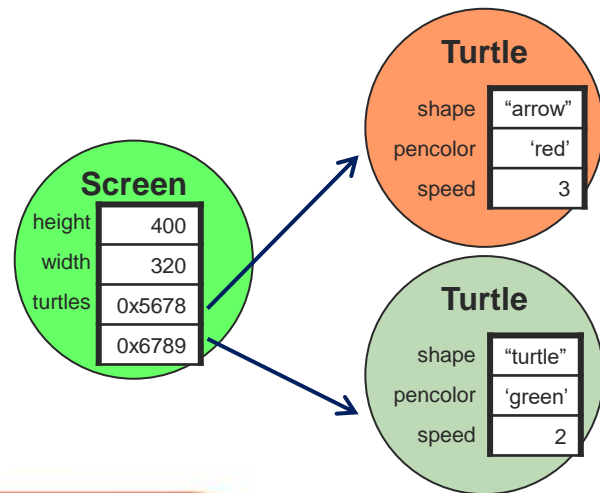
◆ 應用程式執行時期

◆ 根據類別定義建立物件

- 配置記憶體

◆ 物件之間互動產生之狀態變化

- 記憶體中儲存值改變



Screen 內建屬性及常用方法

屬性	說明	設定方法	參數值
bgcolor	背景顏色	bgcolor(color)	'red', 'yellow', 'green', ..., (#RGB), (#RRGGBB)
bgpic	背景圖片或文字	bgpic(picname=None)	None, 字串, 圖片檔名
width	視窗寬度	screensize(width=None, height=None)	設定視窗寬度及高度,單位為畫素
height	視窗高度		

方法	傳回值	說明
clear()	None	清空視窗上繪製圖案
reset()	None	清空視窗上繪製圖案, 並將游標回到原點
bye()	None	關閉視窗
exitonclick()	None	點選視窗時呼叫bye()關閉視窗

程式範例

```
*object3.py - D:\PythonJunior\Exam...
File Edit Format Run Options Window Help

import turtle

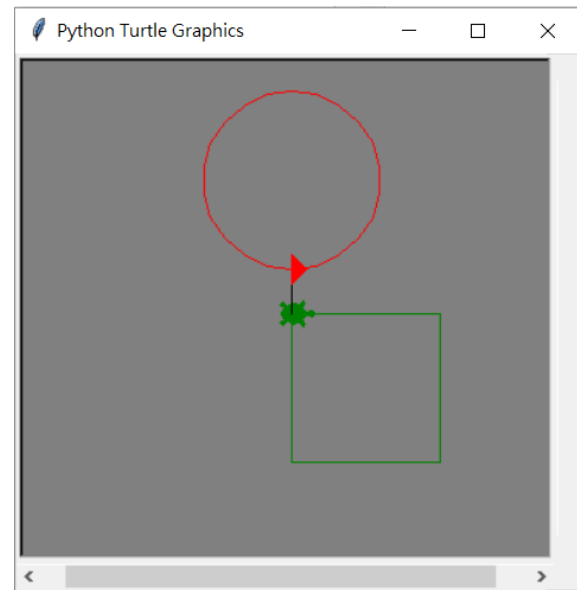
def draw_square(a_turtle):
    for i in range(1,5):
        a_turtle.forward(100)
        a_turtle.right(90)

window=turtle.Screen()
window.bgcolor("grey")

sean = turtle.Turtle()
sean.shape('turtle')
sean.color('green')
draw_square(sean)

amy = turtle.Turtle()
amy.goto(0, 30)
amy.shape('arrow')
amy.color('red')
amy.circle(60)

window.exitonclick()
Ln: 23 Col: 0
```



練習：烏龜繪圖

- ◆ 使用turtle模組繪製下列圖案
 - ◆ 正N邊形
 - ◆ N個同心圓

課程內容

1. 物件導向初探

- 1-1. 物件導向原理
- 1-2. 建立物件
- 1-3. 修改屬性
- 1-4. 呼叫方法

2. 數值字串物件進階

- 2-1. 數值型態方法
- 2-2. 字串常用方法

3. 集合物件進階

- 3-1. 序列進階操作
- 3-2. 集合進階操作
- 3-3. 字典進階操作

Python內建型態

- ◆ Python中，所有的資料都是物件
- ◆ Python 常用內建資料型態
 - ◆ 數字型態：整數(int)、布林值(bool)、浮點數(float) 及複數(complex)
 - ◆ 文字型態：字串(str)、字元(char)
 - ◆ 集合型態：tuple、list、set、dict

Python數字型態

◆ int

◆ 整數的長度不受限制，不會有溢位的問題

- Python 3 之後，不再區分整數與長整數(long)

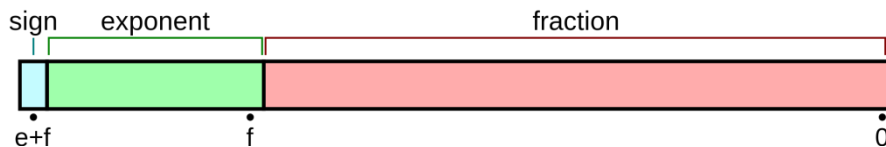
◆ 預設是十進位整數

- 二進位整數，數字以0b開頭，數值為0或1
- 八進位整數，數字以0o開頭，數值為0到7的數字
- 十六進位整數，數字以0x開頭，數值為0到9及A到F的數字

Python數字型態

◆ float

- ◆ 包含小數或科學符號(e/E)的數值
- ◆ 以IEEE-754 的double倍精度浮點數(64位元)實現
- ◆ 數值為近似值



◆ complex

- ◆ 複數，包含實部和虛部，兩部分均可為浮點數：2.3+2.1j

Python數字型態

◆ bool

◆ 數值為True、False

- 底層以整數實作(bool為int的子類別)
- `int(True)`為1，`int(False)`為0

◆ 數字、物件等不同資料也可轉換為bool

- 數字0、0.0，空的物件(空字串""、空的list/tuple、None)轉換為False
- 數字不為0、0.0，非空的物件轉換為True

數值進階方法

◆ int進階方法

- ◆ `bit_length()` : 數值二進位所占用的位元數
- ◆ `to_bytes(length, byteorder, signed)` : 數值二進位位元陣列
 - `length` : 顯示時使用的bytes數量
 - `byteorder` : 'big' 最高位元開始、'little' 最低位元開始
 - `signed` : 是否包含負數

◆ float 進階方法

- ◆ `is_integer()` : 數值是否為整數
- ◆ `hex()` : 數值16進位顯示 $1.a * 2^b \Rightarrow 1.apb$

Python字串型態

◆ Python字串型態

◆ 字串建立

- 單引號或雙引號

◆ 字串索引編號

- `str[n]`

◆ 字串部分取值

- `str[n : m]`

字串進階操作方法

方法名稱	傳回值	說明
<code>replace(old, new, count=-1)</code>	str	將字串中old子字串以new子字串取代 count:最多取代數量, -1為全部取代
<code>strip([chars])</code>	str	傳回移除前導和末尾字元組合後的字串 chars省略時移除前後空白 存在時移除前後包含指定字元的所有組合
<code>rstrip([chars])</code>	str	傳回移除末尾字元組合後的字串 chars省略時移除末尾所有空白 存在時移除末尾包含指定字元的所有組合

字串內容新增

◆ 字串內容新增

- ◆ 不能直接做內容新增
- ◆ 使用 + 的方式串接字串與變數。
- ◆ id()函式：可顯示() 變數的記憶體位址
 - 可用來確認字串變更後是否仍為同一個物件。

程式範例

```
strAppend1.py - D:\PythonJunior\Ex...
File Edit Format Run Options Window Help

greet = "Hello"
for i in range(len(greet)):
    print(greet[i], end='')
print("\ni = ", i)

greet[i+1] = "!"
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

==== RESTART: D:\PythonJunior\Examples\Ch6\strAppend1.py ====
Hello
i = 4
Traceback (most recent call last):
  File "D:\PythonJunior\Examples\Ch6\strAppend1.py", line 6,
in <module>
    greet[i+1] = "!"
TypeError: 'str' object does not support item assignment
>>> |
```

Ln: 36 Col: 4

程式範例

```
strAppend1-2.py - D:\PythonJunior\Examples\Ch...
File Edit Format Run Options Window Help
greet = "Hello"
print(id(greet), "->", greet)

#greet[5]="!"
greet += "!"
print(id(greet), "->", greet)
```



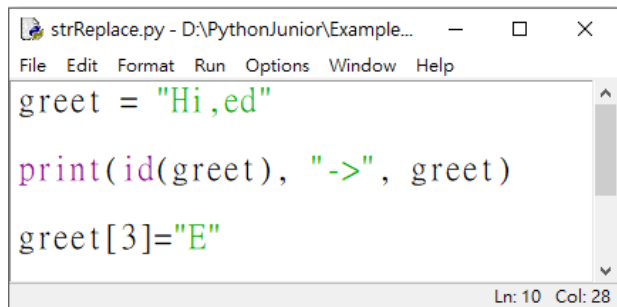
```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
== RESTART: D:\PythonJunior\Examples\Ch6\strAppend1-2.py ==
2014177169200 -> Hello
2014176978864 -> Hello!
>>>
```


字串內容變更

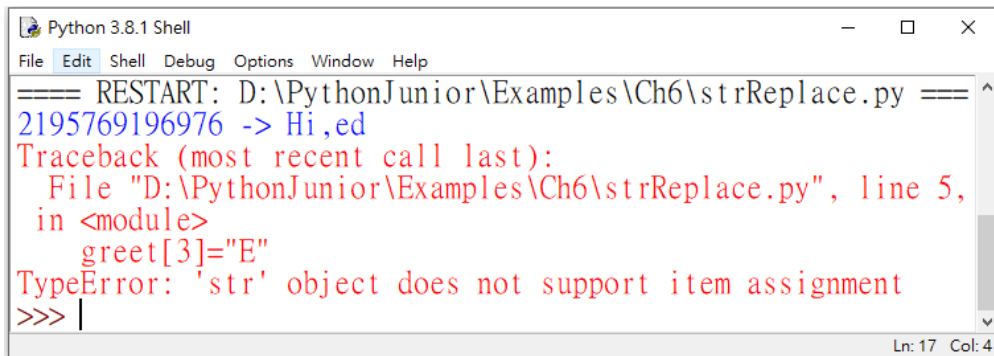
◆ 字串內容變更

- ◆ 字串內容不能直接變更
- ◆ 以部分取值的方式合併串接字串。
- ◆ 用str物件的replace()方法作內容變更

程式範例

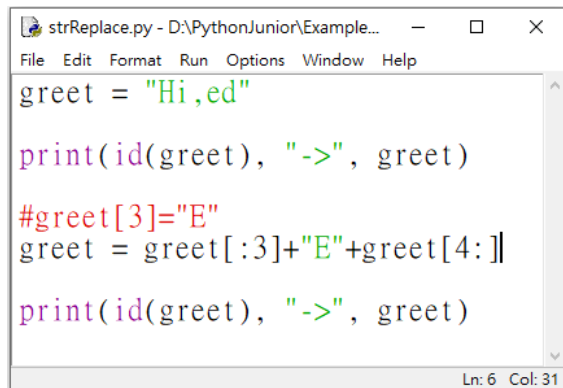


```
strReplace.py - D:\PythonJunior\Example...  
File Edit Format Run Options Window Help  
greet = "Hi,ed"  
print(id(greet), "->", greet)  
greet[3]="E"  
Ln: 10 Col: 28
```

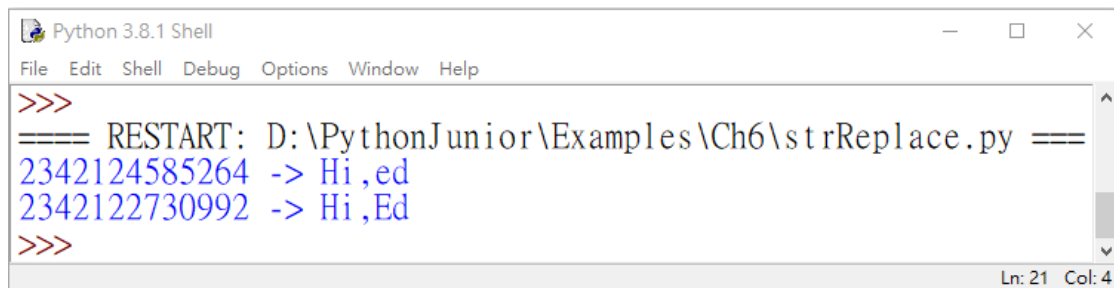


```
Python 3.8.1 Shell  
File Edit Shell Debug Options Window Help  
==== RESTART: D:\PythonJunior\Examples\Ch6\strReplace.py ====  
2195769196976 -> Hi,ed  
Traceback (most recent call last):  
  File "D:\PythonJunior\Examples\Ch6\strReplace.py", line 5,  
    in <module>  
      greet[3]="E"  
TypeError: 'str' object does not support item assignment  
>>> |  
Ln: 17 Col: 4
```

程式範例

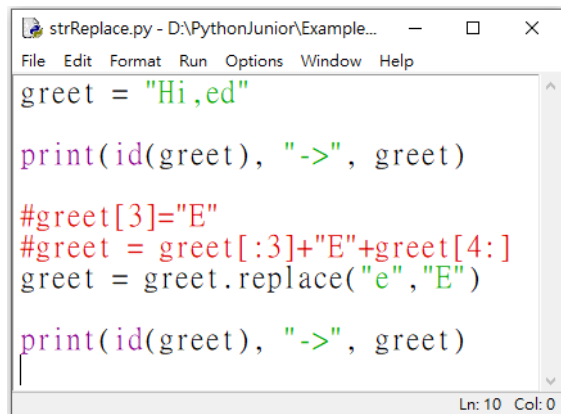


```
strReplace.py - D:\PythonJunior\Example...  
File Edit Format Run Options Window Help  
greet = "Hi,ed"  
print(id(greet), "->", greet)  
#greet[3]="E"  
greet = greet[:3]+"E"+greet[4:]  
print(id(greet), "->", greet)  
Ln: 6 Col: 31
```



```
Python 3.8.1 Shell  
File Edit Shell Debug Options Window Help  
>>>  
==== RESTART: D:\PythonJunior\Examples\Ch6\strReplace.py ===  
2342124585264 -> Hi,ed  
2342122730992 -> Hi,Ed  
>>>  
Ln: 21 Col: 4
```

程式範例

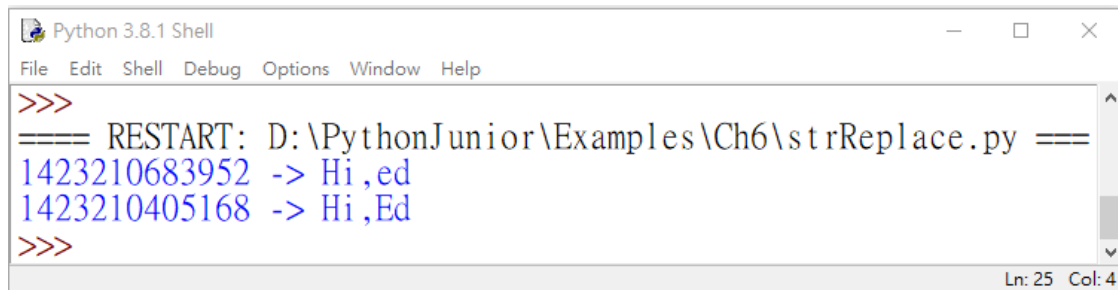


```
strReplace.py - D:\PythonJunior\Example...
File Edit Format Run Options Window Help
greet = "Hi,ed"

print(id(greet), "->", greet)

#greet[3]="E"
#greet = greet[:3]+"E"+greet[4:]
greet = greet.replace("e","E")

print(id(greet), "->", greet)
Ln: 10 Col: 0
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
==== RESTART: D:\PythonJunior\Examples\Ch6\strReplace.py ====
1423210683952 -> Hi,ed
1423210405168 -> Hi,Ed
>>>
Ln: 25 Col: 4
```

字串內容搜尋及記數

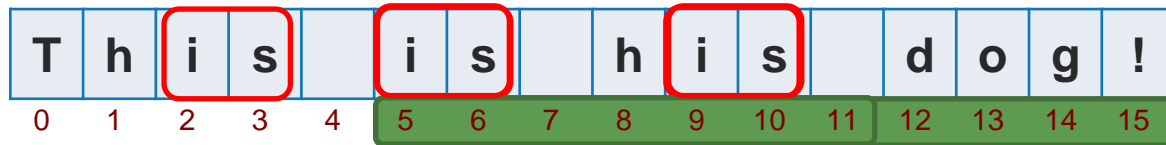
方法名稱	傳回值	說明
<code>find(sub [,start [,end]])</code>	int	傳回sub第一次出現的索引值, 找不到時傳回-1 start與end可指定尋找的切片範圍
<code>rfind(sub [,start [,end]])</code>	int	傳回sub最後出現的索引值, 找不到時傳回-1 start與end可指定尋找的切片範圍
<code>index(sub [,start [,end]])</code>	int	與find()相似但找不到時拋出ValueError
<code>rindex(sub [,start [,end]])</code>	int	與rfind()相似但找不到時拋出ValueError
<code>count(sub[,start[,end]])</code>	int	傳回子字串sub出現的次數 start與end可指定計算的切片範圍

程式範例

```
strFind.py - D:\PythonJunior\Examples\Ch6\str...
File Edit Format Run Options Window Help
str1 = "This is his dogs"
search1 = "is"
print(str1.find(search1))
print(str1.rfind(search1))
print(str1.find(search1, 5))
print(str1.rfind(search1, 10))

print(str1.index(search1, 5, 12))
print(str1.rindex(search1, 5, 12))
print(str1.index(search1, 10))
Ln: 11 Col: 0
```

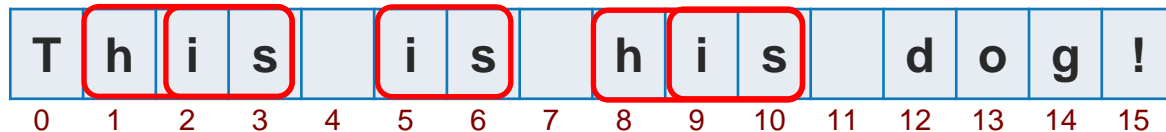
```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: D:\PythonJunior\Examples\Ch6\strFind.py =====
2
9
5
-1
5
9
Traceback (most recent call last):
  File "D:\PythonJunior\Examples\Ch6\strFind.py", line 10, i
n <module>
    print(str1.index(search1, 10))
ValueError: substring not found
>>>
Ln: 134 Col: 4
```



程式範例

```
strCount.py - D:\PythonJunior\Examples\Ch6\strCount.py (3.8.1)
File Edit Format Run Options Window Help
str1 = "This is his dogs"
print(str1)
search1 = "is"
print("%s出現%d次" %(search1, str1.count(search1)))
search2= "his"
print("%s出現%d次" %(search2, str1.count(search2)))
search3="this"
print("%s出現%d次" %(search3, str1.count(search3)))
Ln: 9 Col: 0
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Examples\Ch6\strCount.py ====
This is his dogs
is出現3次
his出現2次
this出現0次
>>>
Ln: 9 Col: 4
```



字串聯結與切割

方法名稱	傳回值	說明
join(iterable)	str	用字串str連結傳入的集合或字串 iterable集合中的字符拼接而成的字符串
split([sep [,maxsplit]])	list	將字串以指定的字元sep分割後組成列表傳回 不指定字元，預設以空格分割字串 maxsplit：分割次數
rsplit([sep [,maxsplit]])	list	與split()相同但由右邊開始切割

字串聯結

- ◆ `str.join()` 方法，用字串`str`聯結傳入的集合或字串
- ◆ `join(iterable)`
 - ◇ `iterable`：被聯結的集合或字串。

`"-".join(['Jan','Feb','Mar']) ➡ 'Jan-Feb-Mar'`

`" ".join('Hello') ➡ 'H e l l o'`

字串切割

- ◆ `str.split()` 函式，將字串以指定的字元進行分割

- ◆ `split([sep [,maxsplit]])`

- ◆ 不指定字元，則預設以空格分割字串 `'Hi Amy'.split()` ➡ `['Hi','Amy']`

- ◆ `sep`：指定分割的字元

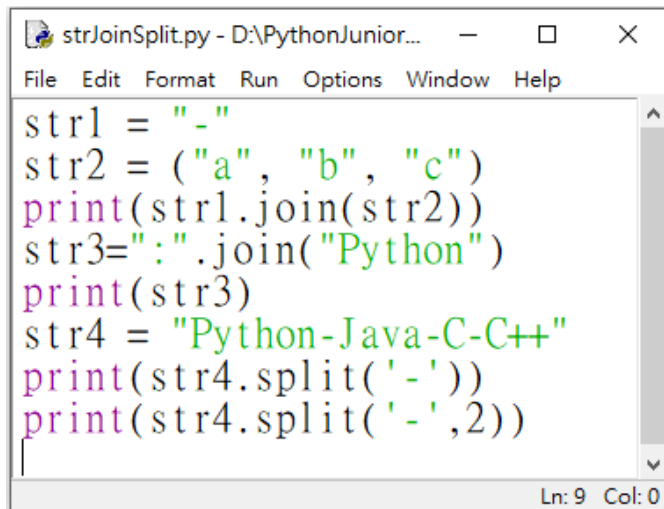
`'Jan-Feb-Mar'.split("-")` ➡ `['Jan','Feb','Mar']`

- ◆ `maxsplit`：分割次數

`'Jan-Feb-Mar'.split("-", 1)` ➡ `['Jan','Feb-Mar']`

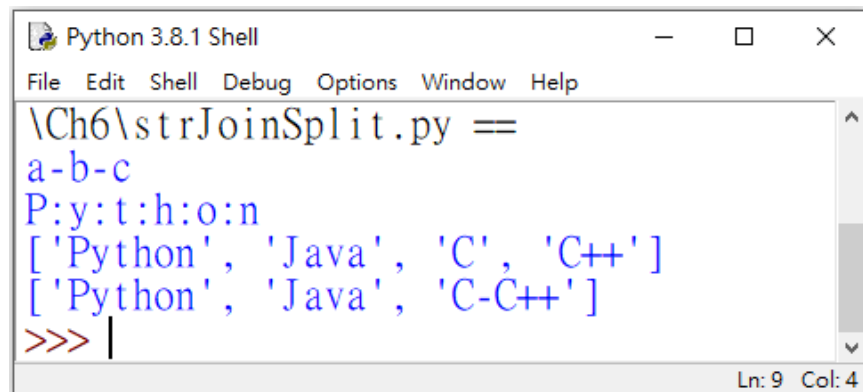
- 產生`maxsplit+1`個元素

程式範例



```
str1 = "-"
str2 = ("a", "b", "c")
print(str1.join(str2))
str3=":".join("Python")
print(str3)
str4 = "Python-Java-C-C++"
print(str4.split('-'))
print(str4.split('-',2))
```

Ln: 9 Col: 0



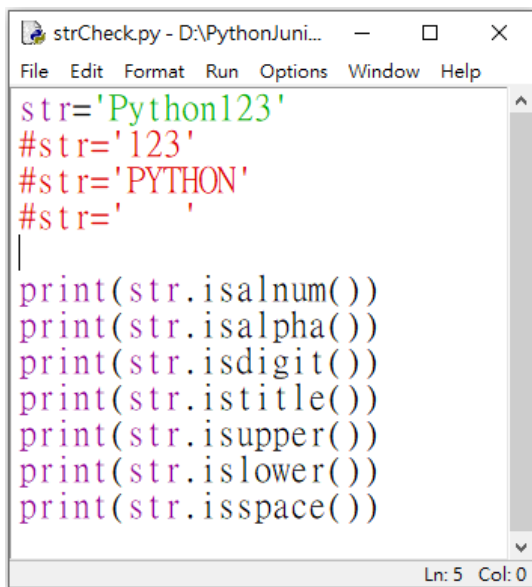
```
\Ch6\strJoinSplit.py ==
a-b-c
P:y:t:h:o:n
['Python', 'Java', 'C', 'C++']
['Python', 'Java', 'C-C++']
>>> |
```

Ln: 9 Col: 4

字串內容檢測方法

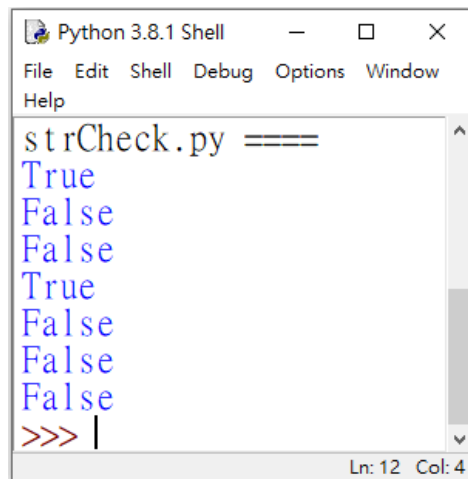
方法名稱	傳回值	說明
isalnum()	bool	檢測字串是否僅包含文字及數字(0-9A-Za-z)
isalpha()	bool	檢測字串是否僅包含文字(A-Za-z)
isdigit()	bool	檢測字串是否僅包含數字
istitle()	bool	檢測字串是否均為首字母大寫
isupper()	bool	檢測字串是否均為全大寫字母
islower()	bool	檢測字串是否均為全小寫字母
isspace()	bool	檢測字串是否均為空白
startswith(str)	bool	檢測字串是否以傳入字串開頭
endswith(str)	bool	檢測字串是否以傳入字串結尾

程式範例



```
str='Python123'  
#str='123'  
#str='PYTHON'  
#str=' '  
  
print(str.isalnum())  
print(str.isalpha())  
print(str.isdigit())  
print(str.istitle())  
print(str.isupper())  
print(str.islower())  
print(str.isspace())
```

Ln: 5 Col: 0



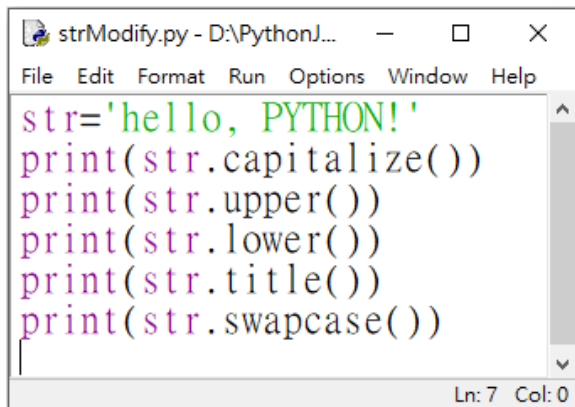
```
strCheck.py =====  
True  
False  
False  
True  
False  
False  
False  
>>> |
```

Ln: 12 Col: 4

字串內容變更

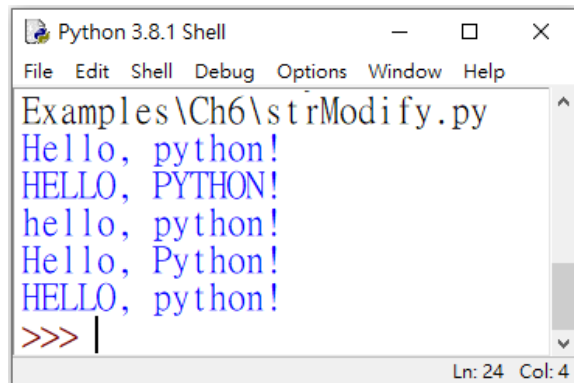
方法名稱	傳回值	說明
capitalize()	str	傳回一新字串, 將原字串首字母改為大寫
upper()	str	傳回一新字串, 將原字串字母均改為大寫
lower()	str	傳回一新字串, 將原字串字母均改為小寫
title()	str	傳回一新字串, 將原字串每個字首字母改為大寫
swapcase()	str	傳回一新字串, 將原字串每個字母大小寫對換
format(*args, **kwargs)	str	依格式化規則, 建立一新字串傳回

程式範例



```
str='hello, PYTHON!'
print(str.capitalize())
print(str.upper())
print(str.lower())
print(str.title())
print(str.swapcase())
```

Ln: 7 Col: 0



```
Examples\Ch6\strModify.py
Hello, python!
HELLO, PYTHON!
hello, python!
Hello, Python!
HELLO, python!
>>> |
```

Ln: 24 Col: 4

程式範例

```
strFormat.py - D:/PythonJunior/Examples/Ch6/strFormat.py (3.8.1)
File Edit Format Run Options Window Help

name = "小明"
age = 20
height = 178.5
weight = 80

print("%s今年%d歲\n%s身高%d公分\n" %(name, age, name, height))
print("{0}今年{1}歲\n{0}身高{2}公分\n{0}體重{3}公斤\n".format(name, age, height, weight))
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

Examples/Ch6/strFormat.py
小明今年20歲
小明身高178公分

小明今年20歲
小明身高178.5公分
小明體重80公斤

>>>
```


練習：英文字母接龍

◆ 撰寫英文字母接龍遊戲

- ◆ 第二個英文單字開始字母必須是前一個英文單字的結尾字母
- ◆ 有五次錯誤的機會

```
*Python 3.8.1 Shell*
File Edit Shell Debug Options Window Help

\string-game.py ===
字母接龍
請輸入字串:Hello
目前字串:Hello
請輸入-O-開始的字串:Orange
目前字串:Hello-Orange
請輸入-E-開始的字串:English
目前字串:Hello-Orange-English
請輸入-H-開始的字串:abc
輸入錯誤1次
目前字串:Hello-Orange-English
請輸入-H-開始的字串:|
```

Ln: 15 Col: 12

課程內容

1. 物件導向初探

- 1-1. 物件導向原理
- 1-2. 建立物件
- 1-3. 修改屬性
- 1-4. 呼叫方法

2. 數值字串物件進階

- 2-1. 數值型態方法
- 2-2. 字串常用方法

3. 集合物件進階

- 3-1. 序列進階操作
- 3-2. 集合進階操作
- 3-3. 字典進階操作

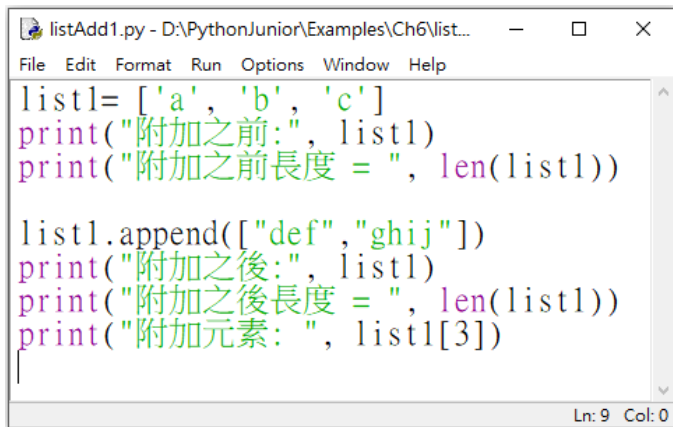
list 常用方法

方法	傳回值	說明
append(element)	None	附加一個元素到List尾端
extend(list)	None	擴展多個元素至List尾端
insert(idx, element)	None	將元素插入指定位置
pop([idx])	element	取出指定位置的元素，無索引值時取出最後一個元素
remove(element)	None	移除第一個出現的指定元素
sort()	None	排序
reverse()	None	反向排列
index(element, srt, end)	int	傳回指定元素第一次出現的索引
count(element, srt, end)	int	傳回指定元素出現的次數

list 新增項目

- ◆ list 可透過以下三種方式新增項目：
 - ◆ `append(element)`：附加一個元素到List尾端
 - ◆ `extend(list)`：擴展多個元素至List尾端
 - ◆ `insert(idx, element)`：將元素插入指定位置

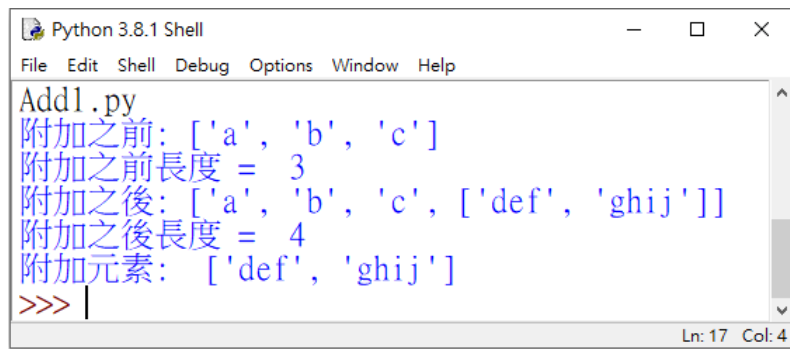
程式範例



```
list1= ['a', 'b', 'c']
print("附加之前:", list1)
print("附加之前長度 = ", len(list1))

list1.append(["def", "ghij"])
print("附加之後:", list1)
print("附加之後長度 = ", len(list1))
print("附加元素: ", list1[3])
```

Ln: 9 Col: 0

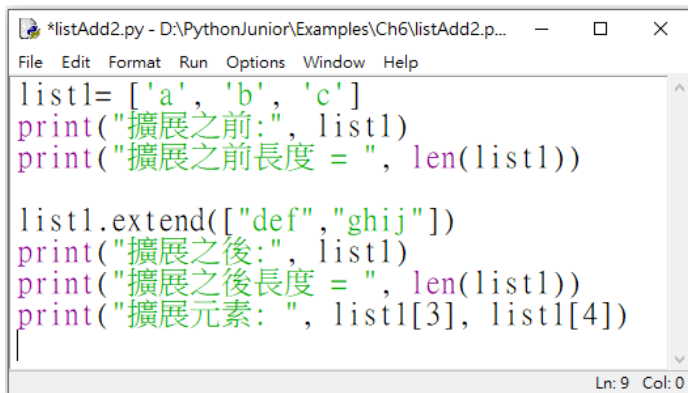


```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

Add1.py
附加之前: ['a', 'b', 'c']
附加之前長度 = 3
附加之後: ['a', 'b', 'c', ['def', 'ghij']]
附加之後長度 = 4
附加元素: ['def', 'ghij']
>>> |
```

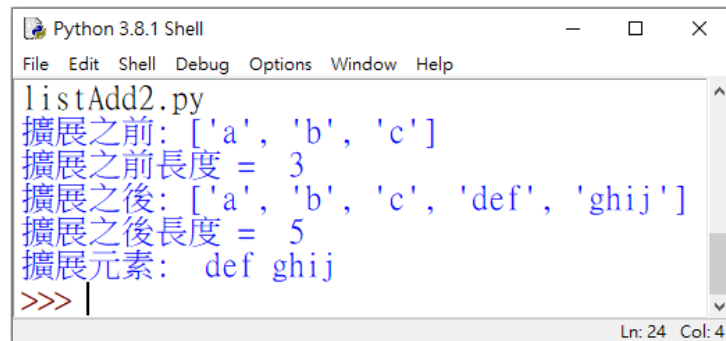
Ln: 17 Col: 4

程式範例



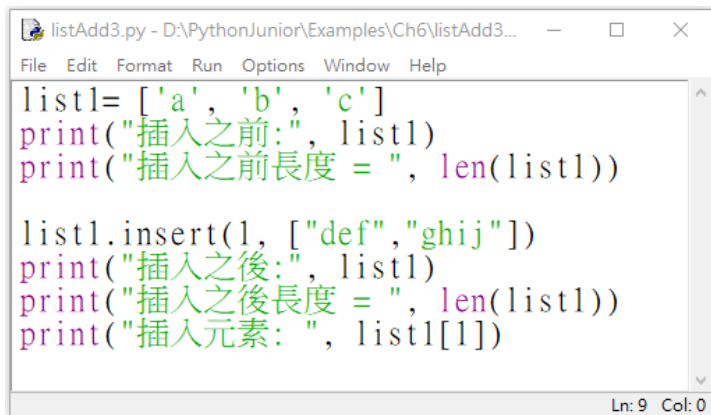
```
*listAdd2.py - D:\PythonJunior\Examples\Ch6\listAdd2.p...
File Edit Format Run Options Window Help
list1= ['a', 'b', 'c']
print("擴展之前:", list1)
print("擴展之前長度 = ", len(list1))

list1.extend(["def", "ghij"])
print("擴展之後:", list1)
print("擴展之後長度 = ", len(list1))
print("擴展元素: ", list1[3], list1[4])
|
Ln: 9 Col: 0
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
listAdd2.py
擴展之前: ['a', 'b', 'c']
擴展之前長度 = 3
擴展之後: ['a', 'b', 'c', 'def', 'ghij']
擴展之後長度 = 5
擴展元素: def ghij
>>> |
Ln: 24 Col: 4
```

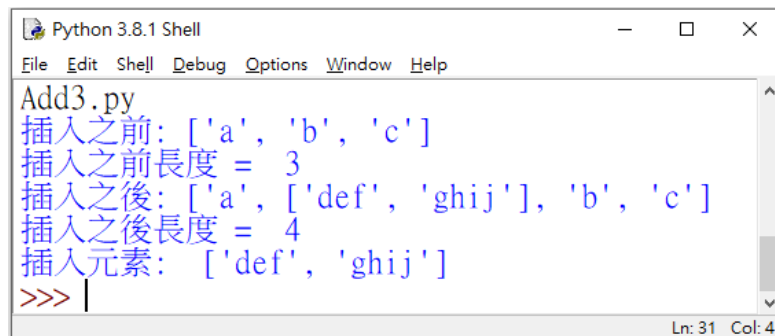
程式範例



```
list1= ['a', 'b', 'c']
print("插入之前:", list1)
print("插入之前長度 = ", len(list1))

list1.insert(1, ["def", "ghij"])
print("插入之後:", list1)
print("插入之後長度 = ", len(list1))
print("插入元素: ", list1[1])
```

Ln: 9 Col: 0



```
Python 3.8.1 Shell
Add3.py
插入之前: ['a', 'b', 'c']
插入之前長度 = 3
插入之後: ['a', ['def', 'ghij'], 'b', 'c']
插入之後長度 = 4
插入元素: ['def', 'ghij']
>>> |
```

Ln: 31 Col: 4

list 移除項目

◆ list 可透過以下三種方式移除項目：

◆ `pop([idx])`：取出指定位置的元素傳回

- 無索引值時取出最後一個元素

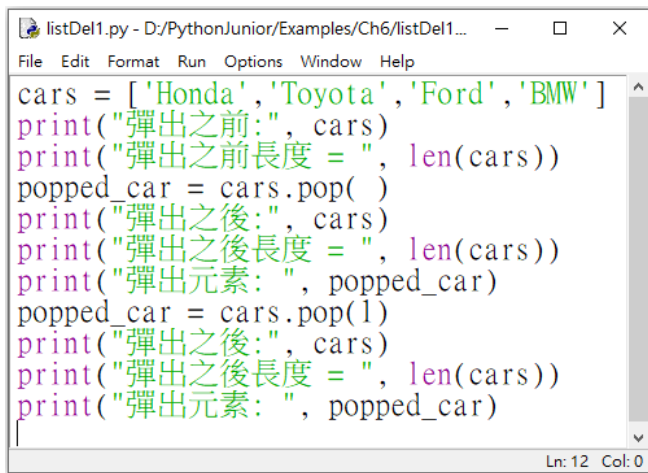
◆ `remove(element)`：移除序列中指定元素

- List中包含多個指定元素時,移除第一個

◆ `del list[n:m]`

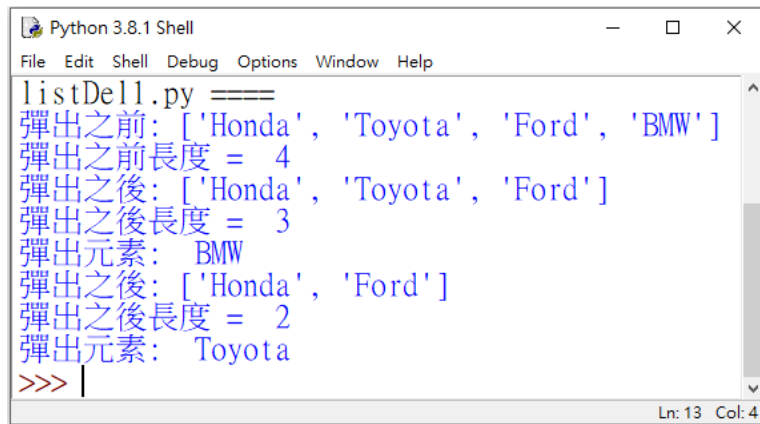
- 使用 `del` 關鍵字，移除指定(索引n 到 m-1位置)序列元素

程式範例



```
listDel1.py - D:/PythonJunior/Examples/Ch6/listDel1...
File Edit Format Run Options Window Help
cars = ['Honda', 'Toyota', 'Ford', 'BMW']
print("彈出之前:", cars)
print("彈出之前長度 = ", len(cars))
popped_car = cars.pop()
print("彈出之後:", cars)
print("彈出之後長度 = ", len(cars))
print("彈出元素: ", popped_car)
popped_car = cars.pop(1)
print("彈出之後:", cars)
print("彈出之後長度 = ", len(cars))
print("彈出元素: ", popped_car)
```

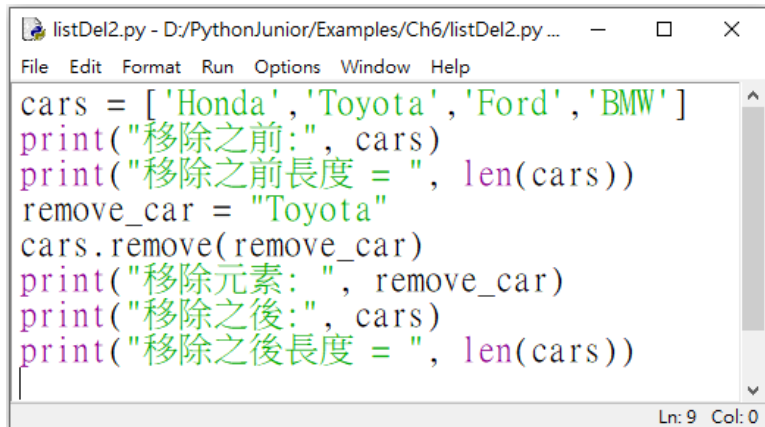
Ln: 12 Col: 0



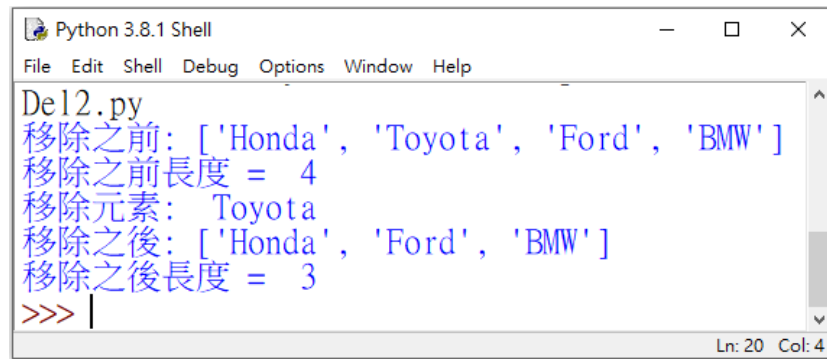
```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
listDell.py =====
彈出之前: ['Honda', 'Toyota', 'Ford', 'BMW']
彈出之前長度 = 4
彈出之後: ['Honda', 'Toyota', 'Ford']
彈出之後長度 = 3
彈出元素: BMW
彈出之後: ['Honda', 'Ford']
彈出之後長度 = 2
彈出元素: Toyota
>>> |
```

Ln: 13 Col: 4

程式範例



```
listDel2.py - D:/PythonJunior/Examples/Ch6/listDel2.py ...
File Edit Format Run Options Window Help
cars = ['Honda', 'Toyota', 'Ford', 'BMW']
print("移除之前:", cars)
print("移除之前長度 = ", len(cars))
remove_car = "Toyota"
cars.remove(remove_car)
print("移除元素: ", remove_car)
print("移除之後:", cars)
print("移除之後長度 = ", len(cars))
Ln: 9 Col: 0
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Del2.py
移除之前: ['Honda', 'Toyota', 'Ford', 'BMW']
移除之前長度 = 4
移除元素: Toyota
移除之後: ['Honda', 'Ford', 'BMW']
移除之後長度 = 3
>>> |
Ln: 20 Col: 4
```

程式範例

```
listDel3.py - D:/PythonJunior/Examples/Ch6/listDel3.py (3.8.1)
File Edit Format Run Options Window Help
cars = ['Honda', 'Toyota', 'Ford', 'BMW', 'Mazda', 'Benz']
print("刪除之前:", cars)
print("刪除之前長度 = ", len(cars))

del cars[4]
print("刪除之後:", cars)
print("刪除之後長度 = ", len(cars))

del cars[1:3]
print("刪除之後:", cars)
print("刪除之後長度 = ", len(cars))
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
= RESTART: D:/PythonJunior/Examples/Ch6/listDel3.py
刪除之前: ['Honda', 'Toyota', 'Ford', 'BMW', 'Mazda', 'Benz']
刪除之前長度 = 6
刪除之後: ['Honda', 'Toyota', 'Ford', 'BMW', 'Benz']
刪除之後長度 = 5
刪除之後: ['Honda', 'BMW', 'Benz']
刪除之後長度 = 3
>>>
```

Ln: 28 Col: 4

list 排序

◆ list 排序

- ◆ list 類別提供 `reverse()` 方法將序列反向排列
- ◆ 序列中的元素需為相同型態，並具備特定排序規則
- ◆ list 類別提供 `sort(...)` 方法進行排序
 - `list.sort()`
- ◆ Python 內建 `sorted(...)` 函式進行排序
 - `list1 = sorted(list)`

排序原則與參數

◆ 排序原則：

- ◆ 數字則由小到大排序
- ◆ 字串則依照 **ASCII** 編碼順序進行排序
- ◆ 自訂物件可用**key**參數指定排序的屬性/方法名稱

list 排序

◆ sort(...) 方法與sorted(...) 函式比較：

◆ sort(...) 方法

- 無傳回值
- list 變數參照的序列物件內容被依排序規則變更

◆ sorted(...) 函式

- 傳回一個新的(已排序)序列物件
- list 變數參照的序列物件內容不被變更
- sorted() 也可用來排序非序列物件

排序原則與參數

◆ 排序方法參數：

◆ reverse=True

- 預設此參數為False，資料由小到大排序
- 設定此參數為True，資料由大到小排序(反向)

◆ Key 設定排序對象

- key=len：依照 list 內元素字串長度進行排序
- key=str.upper：將 list 內元素轉換為大寫，再依照 ASCII 編碼順序進行排序
- key=str.lower：將 list 內元素轉換為小寫，再依照 ASCII 編碼順序進行排序

程式範例

```
listSort1.py - D:\Python\Junior\Examples\Ch6\listSort1.py (3...
File Edit Format Run Options Window Help
nums = [5, 2, 1, 9, 6]
print("sorted()函式排序之前:", nums)
print("sorted()函式傳回:", sorted(nums))
print("sorted()函式排序之後:", nums)

print("sort()方法排序之前:", nums)
print("sort()方法傳回:", nums.sort())
print("sort()方法排序之後:", nums)
Ln: 9 Col: 0
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
\listSort1.py
sorted()函式排序之前: [5, 2, 1, 9, 6]
sorted()函式傳回: [1, 2, 5, 6, 9]
sorted()函式排序之後: [5, 2, 1, 9, 6]
sort()方法排序之前: [5, 2, 1, 9, 6]
sort()方法傳回: None
sort()方法排序之後: [1, 2, 5, 6, 9]
>>> |
Ln: 9 Col: 4
```


程式範例

```
*listSort2.py - D:/PythonJunior/Examples/Ch6/listSort...
File Edit Format Run Options Window Help

cars = ['honda', 'BMW', 'Toyota', 'audi']

print("排序之前:", cars)
cars.sort()
print("排序之後:", cars)
cars.sort(reverse=True)
print("反向排序:", cars)
cars.sort(key=len)
print("長度排序:", cars)
cars.sort(key=str.upper)
print("小寫排序:", cars)

Ln: 12 Col: 0
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

>>>
= RESTART: D:/PythonJunior/Examples/Ch6/listSort2.py
排序之前: ['honda', 'BMW', 'Toyota', 'audi']
排序之後: ['BMW', 'Toyota', 'audi', 'honda']
反向排序: ['honda', 'audi', 'Toyota', 'BMW']
長度排序: ['BMW', 'audi', 'honda', 'Toyota']
小寫排序: ['audi', 'BMW', 'honda', 'Toyota']
>>> |

Ln: 55 Col: 4
```

set

◆ set

◆ 使用{}大括號將元素包起來

◆ 元素之間以，隔開

- 元素可以不同類型的值或集合

◆ 元素不可重複

◆ 元素不具順序，不可用索引值存取

```
admins = {'Justin', 'David'}  
users = {'Mary', 'Nicole', 'Justin'}
```

set 常用方法

方法	傳回值	說明
add(element)	None	新增一個元素到Set
remove(element)	None	移除指定的元素
discard(element)	None	丟棄指定的元素
pop()	element	彈出一個元素
clear()	None	清空Set
intersection(set)	set	傳回交集
union(set)	set	傳回聯集
difference(set)	set	傳回差集
symmetric_difference(set)	set	傳回對稱差集

Set 操作

◆ Set 操作

- ◆ `len()` 取得 set 元素個數
- ◆ `in` 測試某元素是否在 set 中
- ◆ `for x in set` 逐一取出 set 所有元素

```
admins = {'Justin', 'David'}  
users = {'Mary', 'Nicole', 'Justin'}  
len(admins) ➡ 2  
len(users) ➡ 3  
'Justin' in admins ➡ True  
'David' in users ➡ False  
for name in admins :  
    print(name) ➡ Justin  
                   ➡ David
```

Set 操作

◆ Set 操作

`admins = {'Justin', 'David'}`

◆ `add(element)` : 新增元素到Set中

- 元素已存在無影響 `admins.add('David') ➡ {'Justin', 'David'}`
- 元素不存在新增 `admins.add('Sean') ➡ {'Justin', 'Sean', 'David'}`
- 元素在集合中的位置無法由程式控制

◆ `clear()` : 清空Set中所有元素 `admins.clear() ➡ set {}`

Set 操作

◆ Set 操作

`admins = {'Justin', 'David ', 'Sean'}`

◆ `remove(element)` : Set中移除指定元素

- Set中無此元素丟出KeyError
- `admins.remove('Justin')` ➡ `{'David', 'Sean'}`
`admins.remove('Justin')` ➡ `KeyError`

◆ `discard(element)` : Set中丟棄指定元素

- Set中無此元素則無動作
- `admins.discard('Sean')` ➡ `{'David'}`
`admins.discard('Sean')` ➡ `{'David'}`

◆ `pop()` : Set中任意取出一個元素傳回

- Set中無元素丟出KeyError
- `'David' = admins.pop()` ➡ `set {}`
`name = admins.pop()` ➡ `KeyError`

Set 操作

◆ Set 邏輯運算

◆ & 取得交集 intersection

`admins & users -> {'Justin'}`

`admins = {'Justin', 'David'}`

`users = {'Mary', 'Nicole', 'Justin'}`

同時是管理員也是使用者的帳號

◆ | 取得聯集 union

`admins | users -> {'Mary', 'Nicole', 'Justin', 'David'}`

是管理員或是使用者的帳號

◆ - 取得差集 difference

`admins - users -> {'David'}`

是管理員但不是使用者的帳號

`users - admins -> {'Mary', 'Nicole'}`

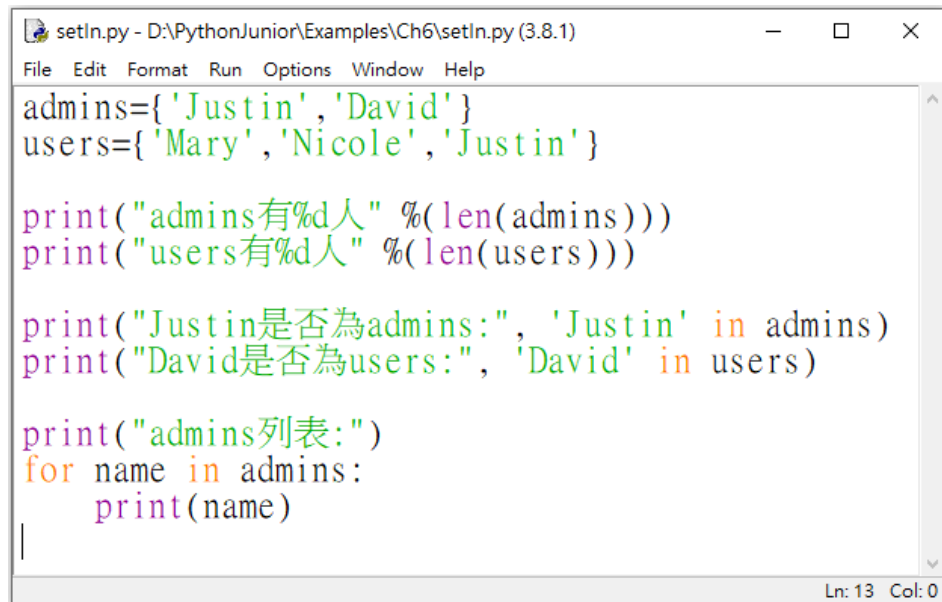
是使用者但不是管理員的帳號

◆ ^ 取得對稱差集 symmetric difference

`admins ^ users -> {'Mary', 'Nicole', 'David'}`

不同時具備管理員及使用者
兩個身分的帳號

程式範例



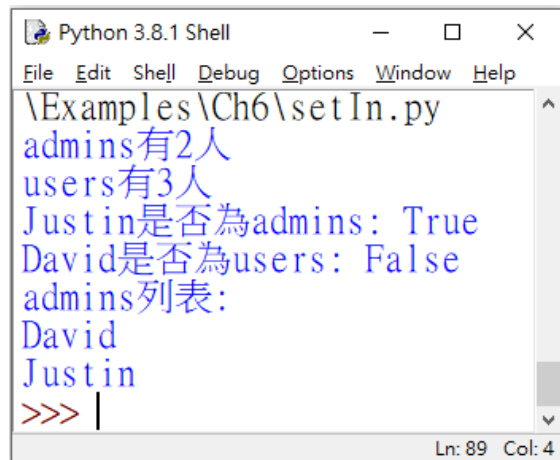
```
setIn.py - D:\PythonJunior\Examples\Ch6\setIn.py (3.8.1)
File Edit Format Run Options Window Help
admins={'Justin','David'}
users={'Mary','Nicole','Justin'}

print("admins有%d人" %(len(admins)))
print("users有%d人" %(len(users)))

print("Justin是否為admins:", 'Justin' in admins)
print("David是否為users:", 'David' in users)

print("admins列表:")
for name in admins:
    print(name)
```

Ln: 13 Col: 0



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
\\Examples\\Ch6\\setIn.py
admins有2人
users有3人
Justin是否為admins: True
David是否為users: False
admins列表:
David
Justin
>>> |
```

Ln: 89 Col: 4


```
setOp.py - D:\PythonJunior\Examples\C... - □ ×
File Edit Format Run Options Window Help
users={'Mary','Nicole','Justin'}
print("User 集合:", users)

users.add('Sean')
print("加入Sean")
print("User 集合:", users)
users.add('Mary')
print("加入Mary")
print("User 集合:", users)

users.remove('Sean')
print("移除Sean")
print("User 集合:", users)
#users.remove('Sean')

users.discard('Mary')
print("丟棄Mary")
print("User 集合:", users)
#users.discard('Mary')

popped = users.pop()
print("彈出:", popped)
print("User 集合:", users)
print("清空:", users.clear())
print("User 集合:", users)
#users.pop()
#print("User 集合:", users)
|
```

Ln: 28 Col: 0

程式範例

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: D:\PythonJunior\Examples\Ch6\setOp.py =====
User 集合: {'Nicole', 'Justin', 'Mary'}
加入Sean
User 集合: {'Nicole', 'Justin', 'Sean', 'Mary'}
加入Mary
User 集合: {'Nicole', 'Justin', 'Sean', 'Mary'}
移除Sean
User 集合: {'Nicole', 'Justin', 'Mary'}
丟棄Mary
User 集合: {'Nicole', 'Justin'}
彈出: Nicole
User 集合: {'Justin'}
清空: None
User 集合: set()
>>> |
```

Ln: 29 Col: 4

程式範例

```
setLogic.py - D:\PythonJunior\Examples\Ch6\setLogic.py (3.8.1)
File Edit Format Run Options Window Help

admins={'Justin','David'}
users={'Mary','Nicole','Justin'}

#intersect = admins & users
intersect = admins.intersection(users)
print("交集:", intersect)

#union = admins | users
union = admins.union(users)
print("聯集:", union)

#difference1 = admins - users
difference1 = admins.difference(users)
print("差集:", difference1)
#difference2 = users - admins
difference2 = users.difference(admins)
print("差集:", difference2)

#sym_difference = admins ^ users
sym_difference = admins.symmetric_difference(users)
print("對稱差集:", sym_difference)
```

Ln: 22 Col: 0

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

= RESTART: D:\PythonJunior\Examples\Ch6\setLogic.py
交集: {'Justin'}
聯集: {'David', 'Mary', 'Nicole', 'Justin'}
差集: {'David'}
差集: {'Mary', 'Nicole'}
對稱差集: {'David', 'Mary', 'Nicole'}
>>> |
```

Ln: 154 Col: 4

dict 常用方法

方法	傳回值	說明
keys()	dict_keys	取得所有的key傳回
values()	dict_values	取得所有的value傳回
items()	dict_items	鍵/值對以Tuple儲存至dict_items物件傳回
update({key:value})	None	新增或更新指定的鍵值對資料
get(key[, default])	value	檢視key對應的value，key不存在傳回default值
pop(key[, default])	value	取出key對應的value，key不存在傳回default值
clear()	None	清除dict中所有資料

判斷資料是否存在

◆ 判斷資料是否存在

◆ Key 存在表示資料存在

◆ `key in dict1.keys()`

- 存在傳回 `true`
- 不存在傳回 `false`

update() 新增與修改

◆ update() 新增/修改鍵值對資料

◆ 相當於用 `dict[Key]=Value` 指派方式新增或修改資料

◆ 將 `key` 與 `value` 儲存至一個鍵值對物件中。

- `{key:value}`

◆ `dict.update(鍵值對資料)`

- 鍵不存在時為新增資料

- 鍵已存在為修改資料

get() / pop() 取值

◆ get(key) / pop(key) 取得對應資料

◆ get(key[, default]) :

- 取得key對應的value，key不存在傳回default值
- default值預設為None
- 取得後鍵值對仍存在Dict中

◆ pop(key[, default]) :

- 取出key對應的value，key不存在傳回default值
- 無default值丟出KeyError
- 取出後鍵值對不再存在Dict中

dict 刪除

◆ 刪除可分刪除指定資料、清除所有內容與刪除字典變數三種：

◆ `del dict[key]` 刪除指定 `key` 及其對應的`value`資料

◆ `dict.clear()` 清除`dict`中所有資料內容

◆ `del dict` 刪除字典變數

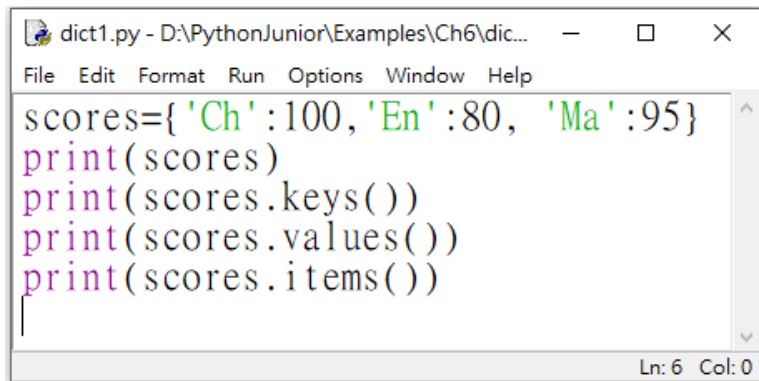
字典的排序

- ◆ 字典沒有`sort()`方法
- ◆ 可使用`sorted()`函式對字典排序
 - ◆ `sorted(dict)`
 - 依據 `key` 由小而大進行排序
 - `Key` 需相同型態，而且有排序規則
 - ◆ `sorted(dict, reverse = True)`
 - 依據 `key` 由大而小進行 (反向)排序

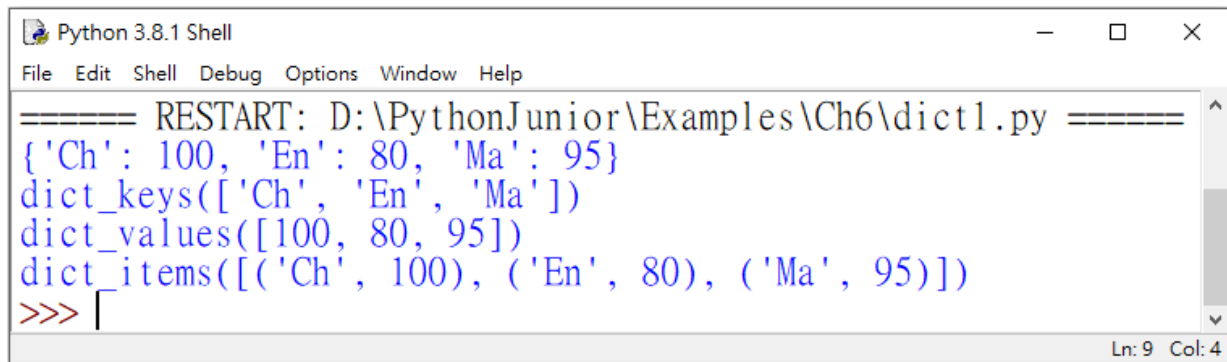
字典的排序

- ◆ 取得字典內的value資料排序。
 - ◆ value 需相同型態，而且有排序規則
 - ◆ `sorted(dict.values())`
 - 依據 value 由小而大進行排序
 - ◆ `sorted(dict.values(), reverse = True)`
 - 依據 value 由大而小進行排序

程式範例



```
dict1.py - D:\PythonJunior\Examples\Ch6\dic...  
File Edit Format Run Options Window Help  
scores={'Ch':100,'En':80,'Ma':95}  
print(scores)  
print(scores.keys())  
print(scores.values())  
print(scores.items())  
Ln: 6 Col: 0
```



```
Python 3.8.1 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: D:\PythonJunior\Examples\Ch6\dict1.py =====  
{'Ch': 100, 'En': 80, 'Ma': 95}  
dict_keys(['Ch', 'En', 'Ma'])  
dict_values([100, 80, 95])  
dict_items([('Ch', 100), ('En', 80), ('Ma', 95)])  
>>> |  
Ln: 9 Col: 4
```

程式範例

```
dictUpd.py - D:\PythonJunior\Examples\Ch6\dictU...
File Edit Format Run Options Window Help
scores={'Ch':100,'En':80, 'Ma':95}
print(scores)

add_dic={'So':90}
scores.update(add_dic)
print(scores)

add_dic={'Ma':'N/A'}
scores.update(add_dic)
print(scores)
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: D:\PythonJunior\Examples\Ch6\dictUpd.py =====
{'Ch': 100, 'En': 80, 'Ma': 95}
{'Ch': 100, 'En': 80, 'Ma': 95, 'So': 90}
{'Ch': 100, 'En': 80, 'Ma': 'N/A', 'So': 90}
>>> |
Ln: 19 Col: 4
```

程式範例

```
dictGet.py - D:/PythonJunior/Examples/Ch6/dictGet.py (3.8.1)
File Edit Format Run Options Window Help
scores={'Ch':100,'En':80,'Ma':95,'Na': 90}
print("查詢前成績:", scores)

ch_score = scores.get('Ch')
print('國語考%s分' %ch_score)
en_score = scores.get('EN', 'N/A')
print('英文考%s分' %en_score)
ma_score = scores.pop('Ma', 'N/A')
print('數學考%s分' %ma_score)
so_score = scores.pop('so', 'N/A')
print('社會考%s分' %so_score)

print("查詢後成績:", scores)
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: D:/PythonJunior/Examples/Ch6/dictGet.py =====
查詢前成績: {'Ch': 100, 'En': 80, 'Ma': 95, 'Na': 90}
國語考100分
英文考N/A分
數學考95分
社會考N/A分
查詢後成績: {'Ch': 100, 'En': 80, 'Na': 90}
>>>
```

Ln: 9 Col: 4

程式範例

```
dictDel.py - D:/PythonJunior/Examples/Ch6/dictDel.py (3.8.1)
File Edit Format Run Options Window Help

scores={'Ch':100,'En':80,'Ma':95,'Na': 90}
print("查詢前成績:", scores)

del scores['Ch']
print("刪除Ch成績:", scores)

scores.clear()
print("清空成績:", scores)

del scores
print("刪除變數:", scores)
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

===== RESTART: D:/PythonJunior/Examples/Ch6/dictDel.py =====
查詢前成績: {'Ch': 100, 'En': 80, 'Ma': 95, 'Na': 90}
刪除Ch成績: {'En': 80, 'Ma': 95, 'Na': 90}
清空成績: {}
Traceback (most recent call last):
  File "D:/PythonJunior/Examples/Ch6/dictDel.py", line 11, in
<module>
    print("刪除變數:", scores)
NameError: name 'scores' is not defined
>>> |
```

Ln: 18 Col: 4

程式範例

```
dictSort.py - D:/PythonJunior/Examples/Ch6/dictSort.py (3.8.1)
File Edit Format Run Options Window Help
scores={'Ch':100,'En':80,'Ma':95}
print("科目排序:", sorted(scores))
print("成績排序:", sorted(scores.values()))

print("科目反向排序:", sorted(scores,reverse = True))
print("成績反向排序:", sorted(scores.values(),reverse = True))
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: D:/PythonJunior/Examples/Ch6/dictSort.py =====
科目排序: ['Ch', 'En', 'Ma']
成績排序: [80, 95, 100]
科目反向排序: ['Ma', 'En', 'Ch']
成績反向排序: [100, 95, 80]
>>> |
```

Ln: 7 Col: 4

練習：英文單字出現次數

- ◆ 選擇一首英文歌曲，計算單字及出現次數
 - ◆ 忽略大小寫差異，全部轉換為小寫
 - ◆ 去除標點符號
 - ◆ 轉換為list
 - ◆ 以dict 儲存
 - 英文單字為key，出現次數為value



巨匠直播教學

www.pcschoolonline.com.tw