

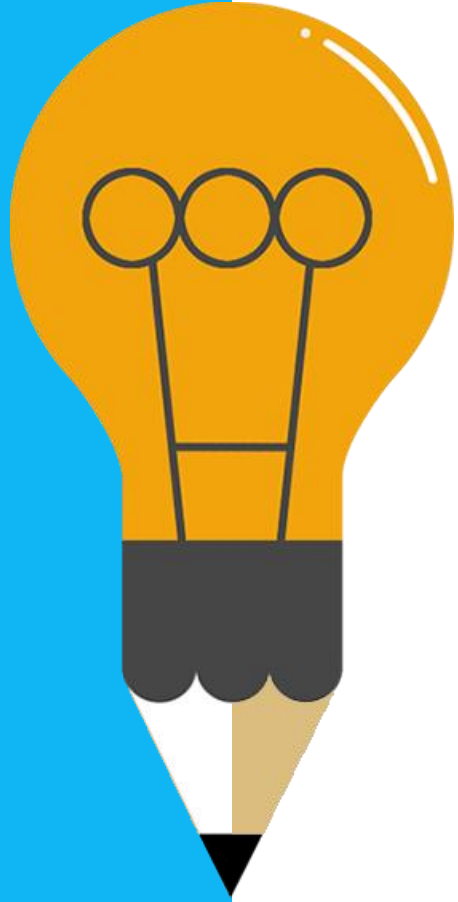
# 變數基本型態

- 數值資料型態
- 字串資料型態

# 摘要：變數基本資料型態與應用

- Who(對象)：變數基本資料型態
- Why(目的緣由)：為什麼需要變數的資料型態
  - 為了讓程式知道對變數可以進行那些運算
- When(應用時機)/Where(應用場域)：變數運算的應用時機
  - 根據程式流程，要對變數進行計算處理的時候
- What(內容定義)：變數可以進行的運算
  1. 數值資料型態
  2. 字串資料型態
- How(操作機制)：數值/字串變數的運算
  1. 數值運算：+、-、\*、/、//
  2. 字串運算：+、\*、slicing以及各種字串運算的方法等

# 數值資料型態



1

算術運算式

2

算術運算子執行的優先次序

3

範例

## 摘要：數值資料型態

本章介紹了程式語言中的算術運算，  
包含基本的四則運算、次方與取餘數等運算。  
另外，說明算術運算子與算術運算元的概念，  
運算子可分為一元運算子與二元運算子兩種類型。

# 算術運算子

- 算術運算式是由多個資料與多個要做的算術運算動作組合而成。
- 資料稱之為算術運算元，可以是變數或是資料值；
- 運算動作便稱之為算術運算子。
- 算術運算子又可分為二元運算子與一元運算子。
- 二元運算子表示需要二個資料才能夠執行運算動作。
- 一元運算子表示僅需要一個資料便能夠執行運算動作。

# Python 算術運算

數學運算	Python 程式碼	運算結果
加 1+1	1 + 1	2
減 20-3	20 - 3	17
乘 3×7	3 * 7	21
除法 100/8	100 / 8	12.5
取餘數	100 % 8	4 (100/8 = 12...4)
次方 4 <sup>2</sup>	4**2	16

## 算術運算子執行的優先次序

- 程式中算術運算子的優先順序和數學的運算式完全相同。
- 先乘除後加減並搭配用圓括弧強制先執行某一部分的運算式。
- 算術運算子中最高的優先次序是次方，  
次高的優先次序是乘法與除法與取餘數，  
最低的是加法與減法。

① 次方(\*\*)

② 乘(\*)、除(/、//)、餘(%)

③ 加(+)、減(-)

## 算術運算式範例

舉例1：

Python 程式碼	運算次序	運算結果
<code>1 + 10 * 3</code>	<code>1 + ( 10 * 3 )</code>	31
<code>20 * 2 - 4 * 3</code>	<code>( 20 * 2 ) - ( 4 * 3 )</code>	28
<code>3 * 4 ** 2 - 1</code>	<code>( 3 * ( 4 ** 2 ) ) - 1</code>	47

舉例2：

數學式  $5 + 6 \times 4^{3^2}$  如何用程式碼表示？

Ans: `5 + 6 * 4 ** 3 ** 2`

- `4 ** 3 ** 2` 必須先計算，而且 `3 ** 2` 要最先處理，  
程式的算術運算式完全依照數學的運算規則來執行。



# 除法

- Python中的除法運算子有一般除法與整數除法兩種用法。
- /符號為一般除法，運算結果的資料型態是浮點數float。
- //符號為整數除法，運算結果的資料型態是整數int。
- //整數除法運算結果只會輸出小數點前的整數。

## 算術運算：除法

Python 程式碼	運算結果	說明
<code>11 / 2</code>	5.5	一般除法，得到小數結果
<code>11 // 2</code>	5	整數除法，得到整數結果 (較小的最靠近整數)
<code>int(11 / 2)</code>	5	強制轉型，得到整數結果 (無條件捨去小數部位)

Python 程式碼	運算結果	說明
<code>-11 / 2</code>	-5.5	一般除法，得到小數結果
<code>-11 // 2</code>	-6	整數除法，得到整數結果 (較小的最靠近整數)
<code>int(-11 / 2)</code>	-5	強制轉型，得到整數結果 (無條件捨去小數部位)

## 算術運算：取餘數

- $a = (a // b) * b + (a \% b)$

- 取餘數：

運算式	Python 結果
$10 \% 3$	1
$-10 \% 3$	2
$10 \% -3$	-2
$-10 \% -3$	-1

運算式	Python 結果
$-10 \% 3$	結果：2 運算過程： $-10 = (-10 // 3) * 3 + (-10 \% 3)$ $-10 = (-4) * 3 + (2)$
$-17 \% 3$	結果：1 運算過程： $-17 = (-17 // 3) * 3 + (-17 \% 3)$ $-17 = (-6) * 3 + (1)$

## 算術運算：整數與浮點數的計算

- 只有當整數與整數進行整數除法運算時，結果才會是整數。

第一個物件型別	算術符號	第二個物件型別	結果型別	範例	結果
int	+, -, *, **, //, %	int	int	3 * 2	6
int	/	int	float	6 / 2	3.0
int	+, -, *, **, /, //, %	float	float	3 - 2.0	1.0
float	+, -, *, **, /, //, %	int	float	3.0 - 2	1.0
float	+, -, *, **, /, //, %	float	float	3.0 * 2.0	6.0

- //雖然為整數除法，但當任一物件型別為float時，結果仍為float。

```
print (5 // 5)
print (5.0 // 5)
print (5 // 5.0)
print (5.0 // 5.0)
```



```
1
1.0
1.0
1.0
```

## 建議

在運算式中變數與運算子和數字之間都建議用空格隔開。

當運算式非常複雜時，若沒有適當加入空格，閱讀起來將會非常困難！大家不妨自己嘗試看看！

## 習題演練：C2\_P1 Apple sharing

N 名學生拿起 K 個蘋果，並在彼此之間平均分配，  
剩餘（不可分割）部分保留在購物籃中。  
請問每個學生將獲得多少個蘋果？  
籃子裡還剩下多少個蘋果？

Sample input：

6

50

Sample output：

8

2

## 變數基本型態

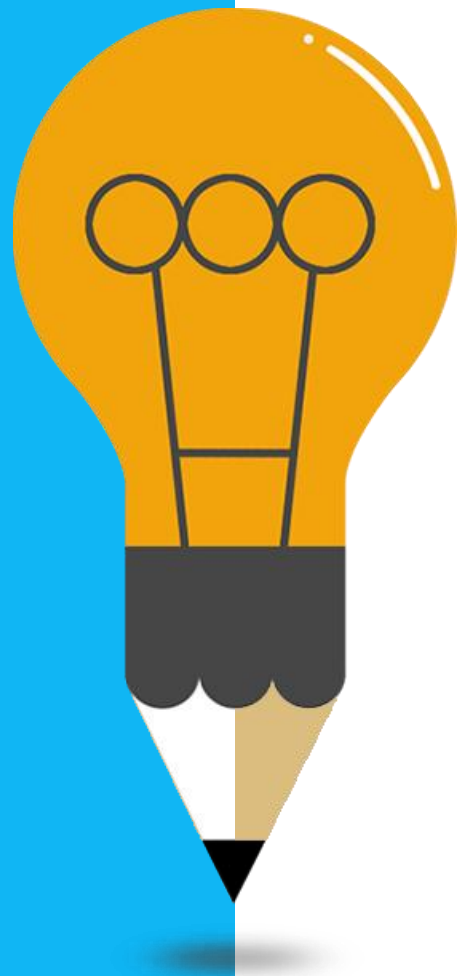
- 數值資料型態
- 字串資料型態

# 摘要：變數基本資料型態與應用

- Who(對象)：變數基本資料型態
- Why(目的緣由)：為什麼需要變數的資料型態
  - 為了讓程式知道對變數可以進行那些運算
- When(應用時機)/Where(應用場域)：變數運算的應用時機
  - 根據程式流程，要對變數進行計算處理的時候
- What(內容定義)：變數可以進行的運算
  1. 數值資料型態
  2. 字串資料型態
- How(操作機制)：數值/字串變數的運算
  1. 數值運算：+、-、\*、/、//
  2. 字串運算：+、\*、slicing以及各種字串運算的方法等



# 字串資料型態



1

字串的概念

2

字串常用方法

# 摘要：字串資料型態

本章說明字串的概念及常用方法，包括索引、切片、判斷、轉換與搜尋等方法。

索引：指定字元的所在位置。

切片：擷取特定的字串內容。

判斷：檢查字串是否包含特殊字元。

轉換：根據函式規則生成新的物件。

搜尋：尋找字串內的子字串。

# 字 串

- 由一連串有順序的字元(character)所組成，包含英文、數字或符號
- 使用單引號或是雙引號來表示字串
- 將一個變數初始化(建立)為一個字串物件
- 例如：
  - `string_1 = "one"`
  - `string_2 = '2'`
  - `string_3 = "This is a string"`
  - `string_4 = "This is a string too! "`

# 索引(index)

- 字串中的字元是有順序性的，也就是說字串中的每個字元，都有一個特定的位置，這個位置就叫索引 (index)。
- 索引值以 0 為開始值，-1 為從末尾的開始位置。

string	M	y		n	a	m	e		i	s		A	n	n	a	.
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

# 切片 (slicing)

- 抓取部份文字的動作 (substring)
- 用字串切片 (slicing)的方式來抓取子字串
- 格式：[起始索引值:終止索引值:間隔值]
  - 包含起始值的字元，但不包含終止值的字元

# 範例：切片(slicing)

string	M	y		n	a	m	e		i	s		A	n	n	a	.
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
name = "My name is Anna."  
print("name==>",name)  
print("str_len==>",len(name))  
print("name[3:9:1]==>",name[3:9:1])  
print("name[3:9:3]==>",name[3:9:3])  
print("name[3:9:2]==>",name[3:9:2])  
print("name[-3:-13:-1]==>",name[-3:-13:-1])  
print("name[4:-2:1]==>",name[4:-2:1])  
print("name[-12:-2:1]==>",name[-12:-2:1])  
print("name[-12:-2:-1]==>",name[-12:-2:-1])
```



```
name==> My name is Anna.  
str_len==> 16  
name[3:9:1]==> name i  
name[3:9:3]==> ne  
name[3:9:2]==> nm  
name[-3:-13:-1]==> nnA si ema  
name[4:-2:1]==> ame is Ann  
name[-12:-2:1]==> ame is Ann  
name[-12:-2:-1]==>
```

# 習題演練：C2\_P2取子字串

請從給定的字串中取出指定位置的子字串，並輸出結果。

Input：三行，第一行，字串；第二行，起始的index位置；  
第三行，結束的index位置

Sample input：

Python

0

3

Sample output：

Pyth

# 字串轉換

Function	Description
len(s)	字串s有多少個字元
replace(old, new)	將字串參數old取代為字串參數new的字串 #注意：替換後會生成新物件

```
name = "My name is Anna. Anna is a good person."  
print("name value before replace : ",name)  
print("length of name : ",len(name))  
print("replace result : ",name.replace("Anna", "Jeff"),"\nname value after replace : ",name)  
str_len = len(name)  
str_replace = name.replace("Anna", "Jeff")  
print("length of str_replace : ",str_len)  
print("value of str_replace : ",str_replace)
```



```
name value before replace : My name is Anna. Anna is a good person. name id : 137816557819664  
length of name : 39  
replace result : My name is Jeff. Jeff is a good person.  
name value after replace : My name is Anna. Anna is a good person.  
length of str_replace : 41  
value of str_replace : My name is Sandy. Sandy is a good person. str_replace id : 137816557820336
```



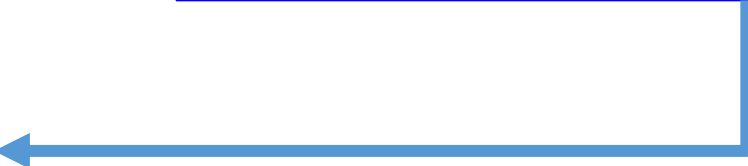
# 搜尋子字串

- `find()`：檢測字串是否包含欲搜尋之子字串
  - 有包含：將輸出子字串第一次出現的索引位置
  - 無包含：輸出-1
- 語法：`str.find(string, beg=0, end=len(string))`

Parameter	Definition
string	指定搜尋的字串
beg	開始索引值，預設為0
end	結束索引值，預設為字串長度

```
str1 = "This is an apple!"  
str2 = "app"  
str1len = len(str1)  
print("str1len : ",str1len)  
print("app index :",str1.find(str2))  
print("app index begin 14~ :",str1.find(str2,14,len(str1)))  
print("app index 0~14 :",str1.find(str2,0,14))
```

```
str1len : 17  
app index : 11  
app index begin 14 : -1  
app index 0~14 : 11
```



# 搜尋子字串

- `count()`：統計字串中某字串出現次數
- 語法：`str.count(string, start= 0, end=len(string))`

Parameter	Definition
string	指定欲統計的字串
start	開始的索引值，預設為0
end	結束的索引值，預設為字串長度

& index : 9  
&的個數 : 2  
&的個數 : 0  
4的個數 : 0  
on的個數 : 0  
ever的個數 : 1

```
StrA = "Python 4 & ever & EVER"  
StrB = "ever"
```


```
print("& index : ",StrA.find("&"))  
print("&的個數 : ", StrA.count("&",5))  
print("&的個數 : ", StrA.count("&",17))  
print("4的個數 : ",StrA.count("4",17))  
print("on的個數 : ",StrA.count("on",5))  
print("ever的個數 : ",StrA.count(StrB))
```

# 搜尋子字串

- in：檢測字串是否包含欲搜尋之子字串，回傳True或False
- 語法：str in string

```
StrA = "Python 4 & ever & EVER"  
StrB = "ever"
```

```
print("on的結果:",("on" in StrA))  
print("空字串的結果:",("" in StrA))  
print("ever的結果:",(StrB in StrA))
```



```
on的結果: True  
空字串的結果: True  
ever的結果: True
```

## 習題演練：C2\_P3字串搜尋

請判斷給定的字串中是否包含指定的子字串，若有則輸出True，若無則輸出False。

Sample input：

Python Programming  
Pro

Sample output：

True

# 總結：數值資料型態、字串資料型態



本章關鍵字  
本章重點概念簡述



請將你的章節總結寫在Memo上，  
Memo的title請設定為 **總結**