第2章 常用物件之1 （Objects 1/2）

2.1 字串（String）

第一個常用的物件就是我們程式中用的「字串」。

字串的定義是用**單引號（‘）**或**雙引號（”）**一前一後來包裝一段字元訊息。

語法

```
' <字元列> '
" <字元列> "
```

上面學習過第一個哈囉程式中的（哈囉，您好！），就是一串<字元列>，也就是我們用單引號，一前一後包裝起來的字串物件的例子。再比如：

‘我叫志明’
“我叫春嬌”

以上皆是字串物件的例子。

In [2]:
```
name1 = '我叫志明'
name2 = "我叫春嬌"
```

介紹「指派」指令 (assignment statement)

語法

```
A = <敘述>
```

Python 設計指派指令讓我們可以將某<敘述>指派一個名稱或者說是將某<敘述>貼上一個標記 A。其特色如下：

1. 指派指令不用英文字 assign 而是用大家耳熟能詳的數學符號：**等號（=）**來表示。
2. 指派的目標寫在**等號（=）**的左邊
3. 等號右邊的<敘述>可以是任何資料運算操作。

舉上面的例子，我們就是把右邊一串招呼訊息（**'哈囉，您好！'**）包裝成字串物件，然後指派一個名稱叫做 message。這個名稱 Python 叫它是「變數」。

變數的命名，Python 有以下命名規則：

1. 變數名稱可以是任何大小寫的英文字母或下底線（_）或數字所組成。
2. 變數名稱開頭不可以是數字

詳細變數的介紹容後在第6章 (6.2) 詳细再加以說明。

現在回來小結一下

如果把前面我們學過的列印指令，加上剛學的指派指令，於是哈囉程式也可以變成二行，如下：

In [3]:
```
message = '哈囉，您好!'
print(message)
```

```
哈囉，您好!
```

太簡單了！

哈囉程式二版，我們加了一個然後列印指令的參數列用這個字串物件的變數完成了。

接下來再印兩個字串物件看看，例子如下：

In [4]:
```
name1 = '我叫志明'
print(name1)

name2 = "我叫春嬌"
print(name2)

```

```
我叫志明
我叫春嬌
```

問：字串物件可以裝多長的訊息呢？只能放一行訊息或是很多行呢？

答：凡事都有它的侷限，字串物件亦然。

字串長度決定的因素有二：

1. 不同操作系统 (Operating System)的架構
2. 不同國家地區使用的字集 (Character Set)

詳细情形，有興趣可 Google 一下電腦操作系統及字集國際標準的課程有更透徹的原理說明。

對剛入門的我們可以簡單這樣了解：若在 64-bit 操作系统下，只用英文字集的話，最長约900萬 Tera Bytes (1 TB = 9x1018 Bytes)。這個數字超大的。若用中文字集 (2個 Bytes 代表一個中文字) 就要減半，變成可容納 450萬 TB也夠大了。我們一般應用程式不會受到此限制所影響。

問：字串物件是否可存放多行訊息？

答：可以。

舉例，以下一則很多行的笑話如下：

> 入學第一天第一堂課，老師要全班同學自我介紹。
>
> 一位男同學走上講台大聲說：
> 『我叫尤勇，來自台北，我愛下棋！』
> 說完就走下台去了。
>
> 下一位是個女生，女同學走上講台嬌羞地，
> 小聲地自我介紹：
> 『我...叫夏琪， ...我喜歡游泳...』
>
> 全班哄堂大笑！

這一種情況，Python 設計讓我們用三個雙引號（"）或單引號（'），前後把整篇多行的笑話包裝進一個字串物件中。

多行字串物件的程式如下：

In [1]:

```
# 二行以上落落長的笑話用三個雙引號前後包裝成一個字串
# 然後指派到一個變數，名叫 joke
joke = """
    入學第一天第一堂課，老師要全班同學自我介紹。

    一位男同學走上講台大聲說：
    『我叫尤勇，來自台北，我愛下棋！』
    說完就走下台去了。

    下一位是個女生，女同學走上講台嬌羞地，
    小聲地自我介紹：
    『我…叫夏琪， …我喜歡游泳…』

    全班哄堂大笑！
"""

print(joke)

```

```
入學第一天第一堂課，老師要全班同學自我介紹。

一位男同學走上講台大聲說：
『我叫尤勇，來自台北，我愛下棋！』
說完就走下台去了。

下一位是個女生，女同學走上講台嬌羞地，
小聲地自我介紹：
『我…叫夏琪， …我喜歡游泳…』

全班哄堂大笑！
```

「註解」指令 (Comments)

上面這一段程式除了執行指令（给 Python 執行的）之外，不知各位是否注意到加入二條註解指令（"#" 開頭， 專為人類看的敍述）Python 不會當指令執行 。 如上面第一行和第二行都是註解指令。

小結一下：

任何一個Python 程式來都可以有下列三種指令類型所组成。

1. Python 執行的指令 (statements)
2. 人們看的註解指令 (comments)
3. 人們留白的空白行指令 (blank lines)

Python 只懂得第一種執行指令而依序執行。 詳細的執行指令我們陸續會學習， 現在有一個簡單的概念就行。

其中後面兩種指令：註解和空白行是專為程式讀者做註解及留白所設的。一個好的程式設計師必須養成在程式中寫註解指令的好習慣。

另外上面的多行字串，Python 也讓我們用它來寫檔案或模塊手冊 (Documentation)，而不用每一行一行都要用 "#" 的註解指令，省了不少麻煩。

例如説有一個 Python 檔案，叫 "greetings.py"。這個程式設計師就利用多行字串，把它的使用手冊寫進程式檔案中，如下：

```
"""" 呼叫輸入函式 input(<提示字串>)，
 Python 能印出一段提示字串，
 然後等使用者輸入資訊，
 按 enter 鍵後，
 Python 再返回程式生成字串物件。
""""
```

於是乎任何人可以用 help 指令，就能很方便列印出使用手冊：

問：字串物件中的訊息輸出簡單，那麼如何輸入訊息給程式呢？

答：輸入訊息一樣簡單，有輸入函式可以呼叫。

下面就來介绍如何呼叫輸入函式：

```
string = input (<提示字串>)
```

Python 能印出一段提示字串，然後等使用者輸入資訊，按 enter 鍵後，Python 再返回程式生成字串物件。這個物件包裝有使用者輸入的資訊。

至於函式的詳細內容後面會介紹的。

現在我們看程式如下：

In [6]:
```
# 從鍵盤輸入，呼叫 input(<提示字串>)函式
name = input("請問你的名字叫: ")

# 螢幕上輸出
print(name)

```

```
請問你的名字叫: 尤勇
尤勇
```

字串物件的運算

字串物件的常用運算操作，有下列五種：

1. 字串物件的粘接 (Concatenation)
2. 字串物件的方法（Methods）
3. 字串物件的格式化 (Formats)
4. 字串物件可套用的函式（Functions）
5. 字串物件的截取（Slicing）

字串物件的粘接 (Concatenation)

語法

<字串物件> + <字串物件> + ...

多個字串物件可以粘接成一個字串物件。其運算子 (Operator) Python 設計用 "+" 來表示。

字串物件粘接的例子，如下：

In [7]:
```
greeting = '哈囉'
name = '尤勇'

# 兩個字串物件之間，用 '+' 運算子粘接
print('多個字串物件粘接成一個字串物件，列印出來如下：')
message = greeting + ', ' + name + '. 歡迎!'
print(message)

```

```
多個字串物件粘接成一個字串物件，列印出來如下：
哈囉, 尤勇. 歡迎!
```

字串物件的方法（Method）

物件的方法（Method）是附屬於物件而提供跟該物件相關的而且可以重覆呼叫的程式。詳細的物件方法我們後面會再更詳細介紹。

目前先介紹字串物件幾個常用的方法，例子如下：

In [2]:
```
message = 'Hello, 尤勇. 歡迎!'
# 字元位置  0123456 78 9

# lower case method (適用於英文轉換小寫的方法)
print(message.lower())

# upper case method (適用於英文轉換大寫的方法)
print(message.upper())
```

```
hello, 尤勇. 歡迎!
HELLO, 尤勇. 歡迎!
```

In [7]:
```
message = 'Hello, 尤勇. 歡迎!'
# 字元位置  0123456 78 9

# count method (字串中算字元出現次數的方法)
print('message 字串中有幾個 "h"?')
print(message.count('h'))

print('message 字串中有幾個 "H"?')
print(message.count('H'))

print('message 字串中有幾個 "l"?')
print(message.count('l'))

```

```
message 字串中有幾個 "h"?
0
message 字串中有幾個 "H"?
1
message 字串中有幾個 "l"?
2
```

In [6]:
```
message = 'Hello, 尤勇. 歡迎!'
# 字元位置  0123456 78 9

# find method (字串中搜尋字元或子字串的方法)
print("字串中有'world'嗎？")
print(message.find('world'))

print("字串中有'World'嗎？")
print(message.find('World'))

print("字串中有'勇'嗎？")
print(message.find('勇'))

print("字串中有'夏'嗎？")
print(message.find('夏'))

print("字串中有' '(空格) 嗎？")
print(message.find(' '))

```

```
字串中有'world'嗎？
-1
字串中有'World'嗎？
-1
字串中有'勇'嗎？
8
字串中有'夏'嗎？
-1
字串中有' '(空格) 嗎？
6
```

In [4]:
```
message = 'Hello, 尤勇. 歡迎!'
# 字元位置  0123456 78 9

# replace method (字串中取代字元或子字串的方法)
print("用'哈囉' 取代 'Hello'")
message = message.replace('Hello', '哈囉')
print(message)
```

```
用'哈囉' 取代 'Hello'
哈囉, 尤勇. 歡迎!
```

字串物件可套用的函式（Functions）

Python 系統也提供一些內建程式，讓我們寫程式可方便呼叫而且可套用到多個物件，叫做「函式」（Function），這與上面剛學習的物件方法 (Method) 雖然都是可以重覆使用的程式，但不同的是方法配屬在某特定的物件之下的程式，而函式是獨立於物件之外的程式。換言之，函式可以套用到不同的物件類型。

Python 方法與函式呼叫的語法也不同：

語法

- 物件方法的呼叫：

  <物件>.<方法名稱>(<參數列>)

用物件名稱加小數點來呼叫方法，如上。

- 函式的呼叫：

  <函式名稱>(<參數列>)

直接呼叫函式名稱加前後括號，把物件放入括號中當其中之一個參數（Argument）。

比如說 'len ()' 就是一個函式用來測量物件的長度，把量測物件放入括號裡，當參數來呼叫。

函式的呼叫例子如下：

In [9]:
```
message = '哈囉，您好!'

print("len()函式功能是量測物件的長度:")
print(len(message))

print("type()函式功能是顯示物件的類型:")
print(type(message))
```

```
len()函式功能是量測物件的長度:
6
type()函式功能是顯示物件的類型:
<class 'str'>
```

字串物件的格式化（Formats）

字串最常看到的運算就是放在 print指令的參數中，運行格式化。

其目的主要是因應人們不同報告格式需求而將字串物件重整，列印出訊息。

下面我們來學習兩種格式化字串物件的例子：

1. 字串物件 format 方法
2. Python 提供的 f-字串指令

In [10]:
```
greeting = '哈囉'
name = '尤勇'

# using format method (用字串物件的 'format' 方法)
print("用字串物件的 'format' 方法...")
message1 = '{}, {}. 歡迎!'.format(greeting, name)
print(message1)
print()

# using f-string in Python 3 (用 Python 版本3以後才有的 'f-字串指令')
print("用 Python 版本3以後才有的 'f-字串指令'...")
message2 = f'{greeting}, {name}. 歡迎!'
print(message2)

```

```
用字串物件的 'format' 方法...
哈囉, 尤勇. 歡迎!

用 Python 版本3以後才有的 'f-字串指令'...
哈囉, 尤勇. 歡迎!
```

字串格式化的應用

字串物件有很多常用格式化功能，尤其應用在列印上。

下面來介紹列印指令如何配合字串格式化帶給我們不同的報表效果。

1. 列印指令括號中，若無參數或只有一個空字串，就列印一條空白行
2. 列印指令括號中，若有二個字串以上作為參數，之間逗號“，”分開也代表二個字串粘接運算，但中間加印一個空格。

另外 Python 設計字串物件中，可以有反斜線“\”開頭，視為特別的格式化字元符號。

特別字串	代表格式
\t	移到下一個製表點
\n	移到下一行
\r	移到此行第一個位置
\b	移回上一個位置
\f	移到下一頁
\'	列印單引號
\"	列印雙引號
\\	列印反斜線

用上面的例子，我們加上一些反斜線特別字元試試，看看格式化後帶來什麼不同效果：

In [11]:
```
greeting = '哈囉'
name = '尤勇'

# using format method (用字串物件的 'format' 方法)
print("用字串物件的 'format' 方法...")
message1 = '{}, {}. 歡迎!\n'.format(greeting, name)
print('===>' + message1)
```

```
用字串物件的 'format' 方法...
===>哈囉, 尤勇. 歡迎!
```

In [9]:
```
greeting = '哈囉'
name = '尤勇'

# using f-string in Python 3 (用 Python 版本3以後才有的 'f-字串指令')
print('用 Python 版本3以後才有的 \'f-字串指令\'...')
message2 = f'{greeting}, {name}. 歡迎!'
print(f'===>{message2}\n')

# 1. 列印指令括號中若無參數，就印一條空白行
print('下一條是空白行...')
print()
print('上一條是空白行!\n')

# 2. 列印指令括號中若有二個字串以上作為參數，之間可以用逗號 “，” 代表二字串粘
print(greeting, name)
```

```
用 Python 版本3以後才有的 'f-字串指令'...
===>哈囉, 尤勇. 歡迎!

下一條是空白行...

上一條是空白行!

哈囉 尤勇
```

字串物件的截取（Slicing）

太棒了！字串物件格式化也算簡單吧！

除了訊息可格式化外，有時候我們要的訊息必須從字串中切開截取出來。

下面就來學習字串截取，例子如下：

In [19]:
```
name = '哈囉 我的名字叫尤勇'
#       0-1 3 4 5 6  8

# 字串可視為一條字元的排列式，位置從0開始算起
print("字串中，切割出打招呼'哈囉'兩個字元：從第1個字元到第2個字但不包含第2個字
message1 = name[0:2]
print(f'===>{message1}\n')

```

```
字串中，切割出打招呼'哈囉'兩個字元：從第1個字元到第2個字但不包含第2個字元:
===>哈囉
```

In [18]:
```
name = '哈囉 我的名字叫尤勇'
#       0-1 3 4 5 6  8

print("字串中，切割出'我的名字'四個字元：從第3個字元到第7個字，但不包含第7個字
message2 = name[3:7]
print(f'===>{message2}\n')

```

字串中，切割出'我的名字'四個字元：從第3個字元到第7個字，但不包含第7個字元:
===>我的名字

In [17]:
```
name = '哈囉 我的名字叫尤勇'
#       0-1 3 4 5 6  8

print("字串中，切割出名字：從第8個字元到最後一個字元:")
message3 = name[8:]
print(f'===>{message3}\n')

```

字串中，切割出名字：從第8個字元到最後一個字元:
===>尤勇

2.2 數字 (Number)

第二個常用的資料物件類型：數字。

數字物件由以下規則來定義：

1. 是一串由0-9十進位數字所組成，例如：12345
2. 數字可以有一位小數點，叫浮點數 (float)，例如：123.45
3. 數字沒有小數點的叫整數 (integer)，例如：123
4. 整數又可分為正整數（如：123）、零（即：0）和負整數（如：-123）

Python 又再细分數字物件為：

1. 整數物件 (int)
2. 浮點數物件 (float)

數字物件之間轉換的例子如下：

In [23]:
```
# 字串物件轉換成整數物件
num_1 = int('100')
print(f'num_1 = {num_1} 物件類型： {type(num_1)}\n')

print(f'{num_1} 是整數物件')
print(f'===>{num_1 + num_1} 也是整數物件!\n')

# 整數物件轉換成字串物件
str_1 = str(num_1)
print(f'str_1 = "{str_1}" 物件類型： {type(str_1)}\n')

print(f'{str_1} 是字串物件')
print(f'===>"{str_1 + str_1}" 也是字串物件!')

```

```
num_1 = 100 物件類型： <class 'int'>

100 是整數物件
===>200 也是整數物件!

str_1 = "100" 物件類型： <class 'str'>

100 是字串物件
===>"100100" 也是字串物件!
```

In [28]:
```
# 浮點數物件轉換成整數物件
float_1 = 3.45
float_2 = 3.55
print(f'float_1 = {float_1} 物件類型： {type(float_1)}\n')
print(f'float_1 浮點數物件轉換成整數物件 = {round(float_1)}\n')

```

```
float_1 = 3.45 物件類型： <class 'float'>

float_1 浮點數物件轉換成整數物件 = 3
```

數字物件的運算

常用的數字物件的算術運算有七種，如下：

1. 加（如：x + y）
2. 減（如：x - y）
3. 乘（如：x * y）
4. 除（如：x / y）
5. 餘（如：x % y）
6. 商（如：x // y）
7. 乘冪（如：x ** y）

數字物件的算術運算的例子如下：

In [29]:
```
# 1. 加 (addition):
print(f'3 + 2 = {3 + 2}')

# 2.減 (Subtraction):
print(f'3 - 2 = {3 - 2}')

# 3. 乘 (Multiplication):
print(f'3 * 2 = {3 * 2}')

# 4. 除 (Division):
print(f'3 / 2 = {3 / 2}')

```

```
3 + 2 = 5
3 - 2 = 1
3 * 2 = 6
3 / 2 = 1.5
```

In [30]:
```
# 5. 餘 (Modulus):
print(f'3 % 2 = {3 % 2}')

# 6. 商 (Floor Division):
print(f'3 // 2 = {3 // 2}')

# 7. 乘冪 (Exponent):
print(f'3 ** 2 = {3 ** 2}')

```

```
3 % 2 = 1
3 // 2 = 1
3 ** 2 = 9
```

In [31]:
```
# 還有其他的運算，如下：
# 加法簡寫
num = 1
print(f'num = {num}')

# num = num + 10 也可以寫成如下：
num += 10
print(f'num += 10 是 num = num + 10 加法簡寫\nnum = {num}\n')

# 絕對值函式 (abs):
num = -4
print(f'絕對值函式 (abs): {num} 絕對值 = {abs(num)}')

```

```
num = 1
num += 10 是 num = num + 10 加法簡寫
num = 11

絕對值函式 (abs): -4 絕對值 = 4
```

問：Python 浮點數如何轉成整數？

答：Python 浮點數取整的方法，有四種方式：

1. 向下取整:直接用內建的 int() 函数
2. 四捨五入到最接近的偶整數: 用 round() 函数
3. 向上取整: 需要用到 math 模組中的 ceil() 方法
4. 需要分別獲取整數部分和小數部分: 用 math 模組中的 modf() 方法

浮點數取整的例子如下：

In [35]:
```
"""Python 浮點數取整的方法1
"""
import math  # import 指令是載入 'math' 工具模組（第6章詳述）

# 1. 向下取整:直接用內建的 int() 函数即可
print(f'1. 向下取整:直接用內建的 int() 函数...')
float_1 = 3.75
int_1 = int(float_1)
print(f'{float_1} ===> {int_1}')
print(f'{float_1} 類型是 {type(float_1)}')
print(f'{int_1} 類型是 {type(int_1)}\n')
# 3

```

```
1. 向下取整:直接用內建的 int() 函数...
3.75 ===> 3
3.75 類型是 <class 'float'>
3 類型是 <class 'int'>
```

In [34]:

```
"""Python 浮點數取整的方法2
"""

# 2. 四捨五入到最接近的數: 用 round() 函数
# 注意：round(數字, 目標位數) 函数返回最接近的目標數。
# 若有兩個目標數，會選擇為偶或較近偶整數的目標數。
print(f'2. 四捨五入到最接近的數: 用 round() 函数...')
float_1 = 3.2
int_1 = round(float_1)
print(f'{float_1} ===> {int_1}')
print(f'{float_1} 類型是 {type(float_1)}')
print(f'{int_1} 類型是 {type(int_1)}\n')
# 3
print(f'round(5.5) ===> {round(5.5)}\n')
# 5

float_1 = 3.85
float_2 = round(float_1, 1)
print(f'round(3.85) ===> {float_2} 較接近偶整數4')
# 3.9 較接近偶整數4
print(f'{float_1} 類型是 {type(float_1)}')
print(f'{float_2} 類型是 {type(float_2)}\n')

```

```
2. 四捨五入到最接近的數: 用 round() 函数...
3.2 ===> 3
3.2 類型是 <class 'float'>
3 類型是 <class 'int'>

round(5.5) ===> 6

round(3.85) ===> 3.9 較接近偶整數4
3.85 類型是 <class 'float'>
3.9 類型是 <class 'float'>
```

In [33]:
```
"""Python 浮點數取整的方法3
"""
import math  # import 指令是載入 'math' 工具模組 (第6章詳述)

# 3. 向上取整: 需要用到 math 模組中的 ceil() 方法
print(f'3. 向上取整: 需要用到 math 模組中的 ceil() 方法...')
float_1 = 3.85
int_1 = math.ceil(float_1)
print(f'math.ceil(3.85) ===> {int_1}')
# 4
print(f'{float_1} 類型是 {type(float_1)}')
print(f'{int_1} 類型是 {type(int_1)}\n')

print(f'math.ceil(3.14) ===> {math.ceil(3.14)}\n')
# 4
print(f'math.ceil(5.85) ===> {math.ceil(5.85)}\n')
# 6

```

```
3. 向上取整: 需要用到 math 模組中的 ceil() 方法...
math.ceil(3.85) ===> 4
3.85 類型是 <class 'float'>
4 類型是 <class 'int'>

math.ceil(3.14) ===> 4

math.ceil(5.85) ===> 6
```

In [32]:
```
"""Python 浮點數取整的方法4
"""
import math  # import 指令是載入 'math' 工具模組（第6章詳述）

# 4. 需要分別獲取整數部分和小數部分: 用 math 模組中的 modf() 方法
# 該方法返回一個包含小數部分和整數部分的元組>>> import math
print(f'4. 需要分別獲取整數部分和小數部分:...')
float_1 = 3.14
print(f'math.modf(3.14) ===> {math.modf(3.14)}')
#(0.14, 3.0)
print(f'math.modf(3.14) 類型是 {type(math.modf(float_1))}\n')

print(math.modf(3.85))
#(0.85, 3.0)

```

```
4. 需要分別獲取整數部分和小數部分:...
math.modf(3.14) ===> (0.14000000000000012, 3.0)
math.modf(3.14) 類型是 <class 'tuple'>

(0.8500000000000001, 3.0)
```

2.3 布林值（Boolean）

目前為止我們已經學習了字串、數字這兩種資料物件的類型。第三個資料物件類型：布林值，我們繼續來學習。

布林值物件有以下特色：

1. 不像字串、數字，布林值是比較簡單但特殊的物件，它只有兩個固定值："真"和"偽"。
2. 布林真值 Python 保留字是：True
3. 布林偽值 Python 保留字是：False

注意：True 和 False 第一個字母是大寫。

以下假設指令例子，如下：

In [1]:
```
x = 0
y = 5
print(f'x = {x}\ny = {y}')

bool_exp = x > y
print(f'bool_exp 物件類型: {type(bool_exp)}\n')

```

```
x = 0
y = 5
bool_exp 物件類型: <class 'bool'>
```

布林值的用法

有了布林值，我們可以設定條件讓 Python 作邏輯判斷其真偽而執行不同的任務。與布林值相關的控制指令中，最常用的就是假設指令（if statement)。

假設指令的功能就是根據一個條件敘述 (conditional expression，或 condition) 的真偽判斷，比如說兩個整數： x 和 y 比較大小，條件敘述就是 "x > y"。經過這個條件敘述運算最後，Python 得出一個或真或偽的布林值。再配合下面將介紹的假設指令，我們的程式就有邏輯判斷的功能了。

「假設」指令：if statement

語法

```
if <條件敘述1成真>：
        <程式區塊 A>

[elif <條件敘述2成真>：
        <程式區塊 B>]

[else ：
        <程式區塊 C>]
```

其中：

1. 假設指令由三種保留字組成：if, elif, 及else
2. if 敘述一定要有
3. 其他兩種 elif 和 else 敘述看需要條件而定
4. elif 可以多個而 else 最多只能出現一次
5. 條件敘述（conditional expression） 不管是簡單或複合的條件，只要 Python 能判斷出布林值即可

6. 每一個程式區塊都要同一排縮格（identation）

Python 用縮格來分別執行不同的程式區塊

Python 執行假設指令時：

1. 首先判斷<條件敘述1>的真偽
2. 若<條件敘述1>為真，則執行<程式區塊 A>
3. 若<條件敘述1>為偽，則執行 elif 判斷<條件敘述2>的真偽
4. 若<條件敘述2>為真，則執行<程式區塊 B>
5. 若<條件敘述2>為偽，則執行 else 的<程式區塊 C>

注意：

1. 三個程式區塊 A, B, C中，假設指令只有擇一區塊執行。
2. 有保留字: if , elif, 和 else 的那一行最後的字元必須是「：」。

以下假設指令例子，如下：

In [3]:
```
x = 0
y = 5

if x > y:
    # <程式區塊 A>
    print('x 大於 y')
elif x < y:
    # <程式區塊 B>
    print('x 小於 y')
else:
    # <程式區塊 C>
    print('x 等於 y')

```

```
x 小於 y
```

簡單的條件敘述例子

In [9]:
```
x = 0
y = 5

# 簡單的條件敘述
if x != 0:
    print('x 不是零')
else:
    print('x 是零')

# 整數0視為偽
if y:
    print('y 不是零')
else:
    print('y 是零')

```

```
x 是零
y 不是零
```

複合條件敘述例子

In [6]:
```
x = 0
y = 5

# 用 or 或 and 组成的複合條件敘述
if x==0 or y==0:
    print('x 或 y 至少有一個不是零')

if x!=0 and y!=0:
    print('x 和 y 兩個皆不是零')

```

```
x 或 y 至少有一個不是零
```

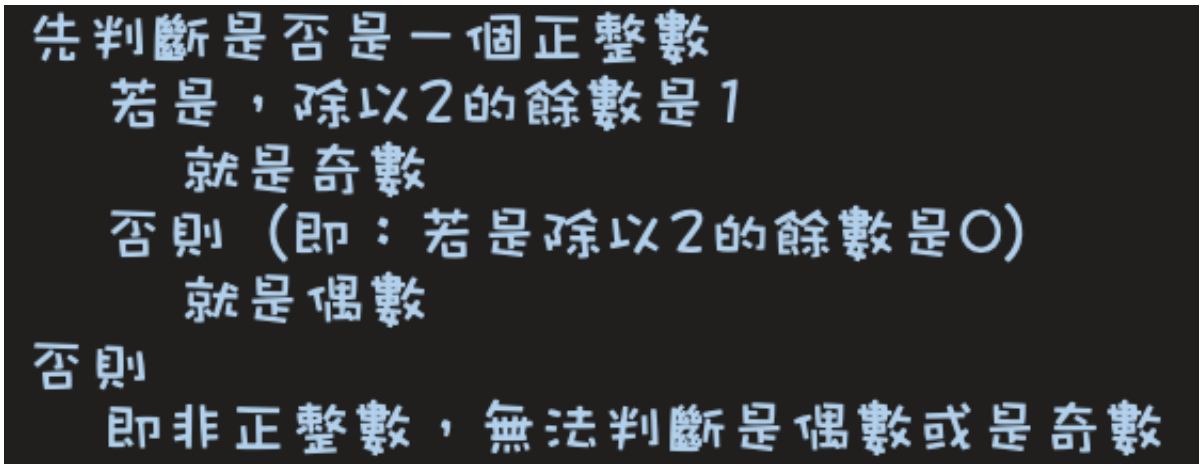
問：上面學的餘數運算"%"在程式上如何應用呢？

答：餘數運算應用很廣。

一個最常見的應用，如下：

當我們要判斷一個整數是偶數或是奇數。寫程式時，配合假設指令就可以完成。

所以我們的程式邏輯，流程如下：



接著我們試著以程式來說明餘數的應用：

In [17]:
```
num_str = input('請輸入一個正整數: ')

# 字串轉換成整數
num = int(num_str)

if num > 0:
    if (num % 2):
        print(f'{num} 是奇數!')
    else:
        print(f'{num} 是偶數!')
else:
    print(f'{num} 不是正整數，無法判斷是偶數或是奇數!')
```

```
請輸入一個正整數: 142857
142857 是奇數!
```

問：條件敍述中，數字物件的比較真偽，應該很常用吧？

答：是的。

有六個常用的數字比較運算判斷數字之間大小關係：

1. 等於（如：x == y）
2. 不等於（如：x != y）
3. 大於（如：x > y）
4. 小於（如：x < y）
5. 大於或等於（如：x >= y）
6. 小於或等於（如：x <= y）

數字的比較大小，例子如下：

In [10]:

```python
num_1 = 3
num_2 = 2
print(f'num_1 = {num_1}, num_2 = {num_2}\n')

# 1. 等於 (如：x == y)
print(f'問：{num_1} == {num_2} 嗎?')
print (f'答：{num_1 == num_2}')

# 2. 不等於 (如：x != y)
print(f'問：{num_1} != {num_2} 嗎?')
print (f'答：{num_1 != num_2}')

# 3. 大於 (如：x > y)
print(f'問：{num_1} > {num_2} 嗎?')
print (f'答：{num_1 > num_2}')

# 4. 小於 (如：x < y)
print(f'問：{num_1} < {num_2} 嗎?')
print (f'答：{num_1 < num_2}')

# 5. 大於或等於 (如：x >= y)
print(f'問：{num_1} >= {num_2} 嗎?')
print (f'答：{num_1 >= num_2}')

# 6. 小於或等於 (如：x <= y)
print(f'問：{num_1} <= {num_2} 嗎?')
print (f'答：{num_1 <= num_2}')
```

```
num_1 = 3, num_2 = 2

問：3 == 2 嗎?
答：False
問：3 != 2 嗎?
答：True
問：3 > 2 嗎?
答：True
問：3 < 2 嗎?
答：False
問：3 >= 2 嗎?
答：True
問：3 <= 2 嗎?
答：False
```

問：條件敘述中，字串物件也能比較真偽嗎？

答：是的。

字串物件的比較通常可以是兩種情況：

1. 比較字串的內容是否相同。
2. 呼叫字串物件方法 (method) 後，比較返回值 (returned value) 與期望值是否相同。

字串物件的條件敘述比較運算，例子如下：

In [12]:
```
# 1. 比較字串的內容條件是否相同
language = 'Python'

if language == 'Python':
    print('程式語言是 Python\n')
elif language == 'Java':
    print('程式語言是 Java\n')
elif language == 'JavaScript':
    print('程式語言是 JavaScript\n')
else:
    print('程式語言未知\n')

```

```
程式語言是 Python
```

In [17]:
```
# 2. 呼叫字串物件方法 (method) 後，比較返回值與期望值是否相同
message = '哈囉, 尤勇. 歡迎!'
target = '勇'
print(f'問：字串="{message}" 中含有字串="{target}" 嗎？')

# 字串中，搜尋字元或子字串的方法: find()
index = message.find(target)
if  index == -1:
    print(f'答："{message}"字串中，沒有:"{target}"\n')
else:
    print(f'答："{message}"字串中，在第 {index} 個索引是:"{target}"\n'

```

```
問：字串="哈囉, 尤勇. 歡迎!" 中含有字串="勇" 嗎？
答："哈囉, 尤勇. 歡迎!"字串中，在第 5 個索引是:"勇"
```