Python: Hello World

學一個新的程式語言，第一步先試著輸出/入訊息，確認撰寫程式環境一切無誤。

- Python使用input函式輸入資料
- Python使用print函式輸出資料

  下面Python範例輸出一個訊息。

In [1]:
```
print('Hello World!')
```

```
Hello World!
```

下面Python範例輸入與輸出一個訊息,

注意 input 可以顯示提示訊息。

In [4]:
```
name = input("Enter your name: ")
print("Hello", name, "!")
```

```
Enter your name: Compeador
Hello Compeador !
```

變數

- 照字面解釋就是會變的數字
- 不只是數字,可以是任何東西
- x, y, z...

變數的宣告

必須是底線或字母開頭,之後由字母,數字或底線所組成

In [15]:
```
zero = 0
one  = "one"
two  = '二'
three= 3.0
print(zero, one, two, three)
```

```
0 one 二 3.0
```

一般虛擬碼(pseudo code)會用的表示法:

```
x ← 123
```

而在 Python 內...

```
x = 123
```

代入(O) 等於(X)

Q: 那 "等於" 該怎麼表示?

A: == , 條件判斷 (if) 時常用

變數的命名慣例

- 常數(constants): THIS_IS_CONSTANT
- 變數(variables): this_is_variable
- 物件名稱(class names): ClassName
- 私有變數(private variables): _private_variable

皆區分大小寫！(Case Sensitive)

Python Keywords

在任何語言內的關鍵字皆有特殊用途, 不可拿來當變數名稱。

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

Python 變數的特性

1. 動態型別, 同一個變數可任意儲存不同的資料
2. 變數具有不同的存取範圍
3. 變數具有不同的生命週期
4. Always pass by reference

註解

在程式中不做任何動作,給人看以方便瞭解程式內容。

在協作的專案中非常重要

In [17]:
```
# the answer to life, the universe, and everything
x = 42
```

Python 內只有單行註解, 但無多行註解；通常使用多行字串 (Multi-line string) 來當註解用

In [32]:

```
'''
Lorem ipsum 簡稱爲 Lipsum,
據說是自15世紀開始在印刷業所使用的隨機假文;
用來檢視字體顯示的效果, 這段文章也沿用至今.
'''
print('''Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Integer sit amet odio scelerisque quam
Vestibulum finibus viverra posuere.
''')
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Integer sit amet odio scelerisque quam
tempus suscipit et vel dolor.
Vestibulum finibus viverra posuere.
```

In [33]:

```
# Don't Do THIS
邪惡ㄉ中文變數 = "[illegible]
print(邪惡ㄉ中文變數)
```

```
[illegible]
```

資料形態

資料型態包含兩種定義:

1. 可能出現的資料範圍(值域)
2. 可以對資料進行什麼操作

In [20]:

```
a = 81000
b = 'TSJ'
c = 3.1415

# 使用 type() 函式顯示資料型態
print(type(a), type(b), type(c))
```

```
<class 'int'> <class 'str'> <class 'float'>
```

In [30]:

```
# 使用 dir() 函式顯示資料可執行的操作 / 函式 / 方法
print(dir(6666))
print(dir('bj4'))
print(dir(0.95))
```

```
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__
', '__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__', '__f
loat__', '__floor__', '__floordiv__', '__format__', '__ge__', '__get
attribute__', '__getnewargs__', '__gt__', '__hash__', '__index__', '
__init__', '__init_subclass__', '__int__', '__invert__', '__le__', '
__lshift__', '__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '_
_new__', '__or__', '__pos__', '__pow__', '__radd__', '__rand__', '__
rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv_
```

```
_', '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__round__', '
__rpow__', '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__',
'__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__su
bclasshook__', '__truediv__', '__trunc__', '__xor__', 'as_integer_ra
tio', 'bit_length', 'conjugate', 'denominator', 'from_bytes', 'imag
', 'numerator', 'real', 'to_bytes']
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '
__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__g
etitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__in
it_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__
', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '_
_repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__st
r__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count
', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map
', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit',
'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'i
stitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans',
'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'r
split', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swa
pcase', 'title', 'translate', 'upper', 'zfill']
['__abs__', '__add__', '__bool__', '__class__', '__delattr__', '__di
r__', '__divmod__', '__doc__', '__eq__', '__float__', '__floordiv__
', '__format__', '__ge__', '__getattribute__', '__getformat__', '__g
etnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__int__', '__le__', '__lt__', '__mod__', '__mul__', '__ne__', '__ne
g__', '__new__', '__pos__', '__pow__', '__radd__', '__rdivmod__', '_
_reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rmod__
', '__rmul__', '__round__', '__rpow__', '__rsub__', '__rtruediv__',
'__set_format__', '__setattr__', '__sizeof__', '__str__', '__sub__',
'__subclasshook__', '__truediv__', '__trunc__', 'as_integer_ratio',
'conjugate', 'fromhex', 'hex', 'imag', 'is_integer', 'real']
```

Python 內建資料型態

也不建議拿來當別的變數名稱

- 一般資料
  - int
  - float
  - str
  - bool
  - complex
- 容器 (Collections)
  - range
  - list
  - tuple
  - dict
  - set

布林值

通常在條件判斷中所使用

- True : 真
- False : 假

In [49]:
```
a = 1
b = 1.0
c = 2
print(a == b, a == c)
```

```
True False
```

In [60]:
```
print(
    bool(0), bool(0.0), bool(1), bool(''), bool('hi')
)
```

```
False False True False True
```

Python 數值運算

- + - * / : 加減乘除
- // ** % : 整除、次方、餘數(限整數)

改變變數內容的運算子

- = : 代入
- += -= *= /= : 運算後代入

Python 整數沒有限制整數大小

In [54]:
```
print(2**1000)
```

```
10715086071862673209484250490600018105614048117055336074437503883703
51051124936122493198378815695858127594672917553146825187145285692314
04359845775746985748039345677748242309854210746050623711418779541821
53046474983581941267398767559165543946077062914571196477686542167660
429831652624386837205668069376
```

其他數值表示方法

In [38]:
```
bin = 0b10010 # 二進位
hex = 0xC5AF8 # 十六進位
oct = 0o71247 # 八進位
```

In [39]:
```
print(3+2, 3//2, 3/2, 3**2, 3-2, 3 % 2)
```

```
5 1 1.5 9 1 1
```

In [40]:
```
print(3+2,',', 3//2, ',',3/2, ',',3**2, ',',3-2, ',',3 % 2)
```

```
5 , 1 , 1.5 , 9 , 1 , 1
```

其他 print 的方法

- 更改參數間的分隔方法
- 更改字尾
- 變數代入
- 變數排版

In [42]:
```
print(3+2, 3//2, 3/2, 3**2, 3-2, 3 % 2, sep=',')
```

```
5,1,1.5,9,1,1
```

In [121…
```
name = input("Enter your name: ")
print(f'Greeting, {name}!')
```

```
Enter your name: sire
Greeting, sire!
```

In [45]:
```
x = 1
x = x + 1
print(x)
x += 1
print(x)
x = x ** 2
print(x)
x **= 2
print(x)
```

```
2
3
9
81
```

Python 字串

- 單引號 '字串'
- 雙引號 "字串"

In [74]:
```
print('單引號字串', "雙引號字串")
print("可在雙引號當中穿插 '單引號'")
print('當然, "反之亦然"~')
```

```
單引號字串 雙引號字串
可在雙引號當中穿插 '單引號'
當然, "反之亦然"~
```

字元

- 字串是由字元所組成
- 每個字元長度爲 1
- 在 python 中, 字元可視為長度爲 1 的字串

字元的數值表示

In [84]:
```
print(ord('0'), ord('A'), ord('a'))
```

```
48 65 97
```

In [85]:
```
print( chr(48), chr(57), chr(65), chr(90), chr(97), chr(122) )
```

```
0 9 A Z a z
```

取得字串長度

In [88]:
```
msg = "How"
print( len(msg) )
```

```
3
```

字串加法與乘法

In [91]:
```
msg = "Hello"
print(msg + " World!")
print(msg * 3)
```

```
Hello World!
HelloHelloHello
```

字串足標

- 取得字串中的第 N 個字元
- 從足標 0 開始, 幾乎所有語言都是 (除了 lua)

In [92]:
```
msg = "How Do You Turn This On"
print(msg[0], msg[2], msg[-1])
```

```
H w n
```

取得範圍內的子字串

In [113…
```
msg = "How Do You Turn This On"
print(msg[1:])
print(msg[-2:])
print(msg[4:-2])
print(msg[6:-1])
```

```
ow Do You Turn This On
On
Do You Turn This
 You Turn This O
```

常用字串函式

In [119…

```
msg = "How Do You Turn This On"
print(msg.upper()) # 全大寫
print(msg.lower()) # 全小寫
print(msg.capitalize()) # 字首大寫,其他小寫
print(msg.title()) # 子字首大寫,其他小寫

msg = "  A string with extra spaces   "
print(msg, '!', sep='')
print(msg.strip(), '!', sep='') # 移除頭尾空白及換行符號等
```

```
HOW DO YOU TURN THIS ON
how do you turn this on
How do you turn this on
How Do You Turn This On
  A string with extra spaces   !
A string with extra spaces!
```

跳脫字元

跳脫字元爲反斜線 \ , 用來表示一般鍵盤無法打出的字。

許多語言的跳脫字元基本上也都是反斜線

Sequence | Meaning | | Sequence | Meaning ---- | ---- | ---- | --- | ---- \\\ | 反斜線 | | \ooo | 八進位值爲 oo 的字元 \\' | 單引號 | |\xhh | 十六進位值爲 hh 的字元 \\" | 雙引號 | |\uhhhh | 十六進位值爲 hhhh 的 16-bit 字元 \n | 換行符號 | |\uhhhhhhhh | 十六進位值爲 hhhhhhhh 的 32-bit 字元 \t | tab

In [144…

```
print("\tEclipse first,")
print("the rest nowhere.", end='\n\n')
print("\\\u2605\u2606\u2605\u2606\u2605/")
```

```
        Eclipse first,
the rest nowhere.

\★☆★☆★/
```

字串與其他資料型態的轉換

In [157…

```
height = input("請輸入您的身高(公分): ")
weight = input("請輸入您的體重(公斤): ")
age    = input("請輸入您的年齡: ")
height = float(height) / 100
weight = float(weight)
age    = int(age)
bmi    = weight/height/height
bfp    = 1.2*bmi + 0.23*age
print("您的 BMI 是:", bmi)
print(f"體脂率: {bfp-16.2}%(男) / {bfp-5.4}%(女)")
```

```
請輸入您的身高(公分): 173
請輸入您的體重(公斤): 50
請輸入您的年齡: 23
您的 BMI 是: 16.706204684419795
體脂率: 9.137445621303755%(男) / 19.937445621303752%(女)
```

格式化字串

類似 C 的 `printf`

reference: https://pyformat.info/

Conversion	Meaning
d	Signed integer decimal
o	Unsigned octal
u	Unsigned decimal
x	Unsigned hexadecimal (lowercase)
X	Unsigned hexadecimal (uppercase)
e	Floating point exponential (lowercase)
E	Floating point exponential (uppercase)
f	Floating point decimal
c	Single character (can use integer or single character string
s	String (similar to using str())

Number	Format	Output	Description
3.1415926	{:.2f}	3.14	2 decimal places
3.1415926	{:+.2f}	+3.14	2 decimal places with sign
-1	{:+.2f}	-1.00	2 decimal places with sign
3.1415926	{:.0f}	3	No decimal places (will round)

5	{:02d}	05	Pad with zeros on the left
1000000	{:,}	1,000,000	Number format with comma separator
0.25	{:.2%}	25.00%	Format percentage
1000000000	{:.2e}	1.00e+09	Exponent notation
11	{:>10d}	11	Right aligned
11	{:<10d}	11	Left aligned
11	{:^10d}	11	Center aligned

In [158…

```
print("您的 BMI 是: {:.2f}".format(bmi))
print("體脂率: {:.2f}%(男) / {:.2f}%(女)".format(bfp-16.2, bfp-5.4))
```

```
您的 BMI 是: 16.71
體脂率: 9.14%(男) / 19.94%(女)
```

浮點數

Float-point (浮點數) 一般可理解爲小數, 和一般表示整數的方式不一樣。
有興趣可參考國際浮點數標準 IEEE-754 (https://zh.wikipedia.org/zh-tw/IEEE_754)

初學者只需知道浮點數計算上會有誤差

In [53]:

```
print(0.1 + 0.2 == 0.3)
print(0.1 + 0.2)
```

```
False
0.30000000000000004
```

解決方法

- 四捨五入、無條件捨去、進位
  - `round()`：內建
- Decimal (高精度浮點數)
  - `decimal` 模組

In [67]:

```
a = 0.1 + 0.2
b = round(a, 1)
print(a, b)
print(b == 0.3)
```

```
0.30000000000000004 0.3
True
```

In [68]:

```
from decimal import Decimal

a = Decimal('0.1') # 注意小數要用字串表示
b = Decimal('0.2')
print(a + b == Decimal('0.3'))
print(float(a+b) == 0.3)
```

```
True
True
```

In [165…

```
ELECTRON    = 1.602 * 10**-19
LIGHT_SPEED = 3 * 10**8
print(ELECTRON)
print("{:e}".format(LIGHT_SPEED))
```

```
1.602e-19
3.000000e+08
```

List 與 Tuple

- 皆是將許多資料儲存在一起的 Collection
- 儲存一連串的變數, 內容可是任何形態
- List 可以改變裡面儲存的變數, Tuple 不行 (mutable/immutable)
- 跟字串一樣用足標存取, 從 0 開始
    - `list[start:end:increment]`

In [171…

```
list1=[1,0.2,3,'world',5,'hello']
list2=[1,'hello',1.3,list1]
print(list1)
print(list1[0])
print(list1[1:3])
print(list1[0::2])
print(list1[-1::-1]) # 反轉
print(list2) # nested list
```

```
[1, 0.2, 3, 'world', 5, 'hello']
1
[0.2, 3]
[1, 3, 5]
['hello', 5, 'world', 3, 0.2, 1]
[1, 'hello', 1.3, [1, 0.2, 3, 'world', 5, 'hello']]
```

List 常用函式

- len
- max/min/sum
- count
- index
- clear
- append 注意與extend的差異
- extend 注意與append的差異
- insert

In [168…

```
numbers=[1,2,3,4,5,1,0,-1,-2,-3]
print(len(list1)) # list元素個數
print(len(list2))
print(len(numbers))
print(max(numbers)) #最大的
print(min(numbers)) #最小的
print(sum(numbers)) #和
print(numbers.count(1)) # 1有幾個
print(numbers.index(1)) # 第一個1出現在第幾號

# 排序
numbers.sort() # numbers.sort(reverse=True)
print(numbers)
```

```
6
4
10
5
-3
10
2
0
[-3, -2, -1, 0, 1, 1, 2, 3, 4, 5]
```

In [170…

```
numbers.clear()
print(numbers)    # 清除所有元素

numbers.append(1) # 從尾端加入一個元素
print(numbers)

numbers.append([7,9,10])# 從尾端加入一個元素[7,9,10]
print(numbers)

numbers.extend([7,9,10]) # 從尾端加入一list個元素
print(numbers)

numbers.insert(1,12) #1號位置插入12
print(numbers)

del numbers[2] #刪除第2號元素
print(numbers)

print([1,2,3]+[4,5,6,7]) # list串接

ones = [1]*5 # list重複
print(ones)
```

```
[]
[1]
[1, [7, 9, 10]]
[1, [7, 9, 10], 7, 9, 10]
[1, 12, [7, 9, 10], 7, 9, 10]
[1, 12, 7, 9, 10]
[1, 2, 3, 4, 5, 6, 7]
[1, 1, 1, 1, 1]
```

字串與list

In [173…

```
str1 = '123,456,789'
list1 = str1.split(',') # 將字串 str 1以 ',' 字元拆解成 list
print(list1)

str2 =','.join(list1) # 將 list1 以 ',' 串接
print(str2)
```

```
['123', '456', '789']
123,456,789
```

Immutable and Mutable Objects

深入討論: https://softwareengineering.stackexchange.com/questions/151733/if-immutable-objects-are-good-why-do-people-keep-creating-mutable-objects

Immutable Object (不可變的物件)

- 當物件被初始化放在記憶體後, 其內容就不可再改變

- 可使用 `id()` 函式查詢物件的 `object_id`
- 通常如 `int`，`float`，`str` 簡單類別同數值擁有相同 id
- 通常會使底層操作變快
- 通常操作有限,較不方便
- Tuple 也是 Immutable Object

Mutable Object (可變物件)

- 物件被初始化放在記憶體後, 其內容可隨時改變
- 每個物件都有獨立的 id
- 通常可操作的方法較多教方便
- 容易因人員或設計上的失誤而發生錯誤或拖累效率
- Python 中除了上述四個形態之外的物件皆是 Mutable Object

下列程式碼會產生錯誤

In [177…

```
msg = "Hello World!"
msg[-1] = 's'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-177-23b3ed177da8> in <module>
      1 msg = "Hello World!"
----> 2 msg[-1] = 's'

TypeError: 'str' object does not support item assignment
```

下面例子，list2 不是 list1 的複製，他們指向同一物件，所以更改 list2 如同改 list1

In [175…

```
list1 = [1,2,3,4]
list2 = list1
list2[1] = 5
print(list1)
print(list2)
```

```
[1, 5, 3, 4]
[1, 5, 3, 4]
```

下面例子，list2是list1的複製，他們指向不同一物件，所以更改list2不會改到list1

In [176…

```
list1 = [1,2,3,4]
list2 = list1.copy() # 或是 list(list1)
list2[1] = 5
print(list1)
print(list2)
```

```
[1, 2, 3, 4]
[1, 5, 3, 4]
```

小結

掌握上述 Python 特性，大概可以用 Python 描述出可用 C 語言所能表達的程序性敘述。
注意 Python 畢竟是解譯式程式語言，撰寫 Python 程式時，更需要講究執行效能的寫法。