Python 的 50+ 練習：資料科學學習手冊

> *使用資料類別運算顯示與判斷*

數據交點 | 郭耀仁 yaojenkuo@datainpoint.com

這個章節會登場的保留字與函數

- `def` 保留字。
- `return` 保留字。
- `False` 保留字。
- `True` 保留字。
- `in` 保留字。
- `not` 保留字。
- `and` 保留字。
- `or` 保留字。

這個章節會登場的保留字與函數（續）

- `abs()` 函數。
- `pow()` 函數。
- `type()` 函數。
- `help()` 函數。
- `isinstance()` 函數。

這個章節會登場的保留字與函數（續）

- `bool()` 函數。
- `int()` 函數。
- `float()` 函數。
- `str()` 函數。

解析哈囉世界

哈囉世界

- 哈囉世界中的 `print()` 是什麼？
- 哈囉世界中的 `"Hello, world!"` 是什麼？

In [1]:
```
print("Hello, world!")
```

```
Hello, world!
```

哈囉世界中的 `print()` 是什麼

- `print()` 是 Python 的內建函數。
- 內建函數是不需要先行「定義」就可以使用的函數。

什麼是函數

一段被賦予名稱的程式碼，能夠完成某一個文字處理或者數值計算任務，在使用函數之前，必須先確定這個函數在執行的環境中已經被定義妥善。

函數的組成有五個要件

1. 函數名稱。
2. 輸入
3. 參數。
4. 運算處理邏輯。
5. 輸出。

函數運作的過程就像買一杯珍珠奶茶

來源：Google 圖片

函數有四個來源

1. 來自內建函數。
2. 來自標準模組。
3. 來自第三方模組。
4. 來自使用者的定義。

來自內建函數可以直接使用

- 哪些內建函數可以直接使用：https://docs.python.org/3/library/functions.html
- 使用方式為輸入資料或引數（Parameters）到函數名稱後的小括號。

In [2]:
```
abs(-55) # -55 as data
```

Out[2]:
```
55
```

資料與引數的輸入方式

1. 依照位置輸入（Positional arguments）。
2. 依照參數名稱輸入（Keyword arguments）。

依照位置輸入（Positional arguments）

In [3]:
```
pow(5, 2) # 5 as base, 2 as exp
```

Out[3]: 25

In [4]:
```
pow(2, 5) # 2 as base, 5 as exp
```

Out[4]: 32

依照參數名稱輸入（Keyword arguments）

In [5]:
```
pow(exp=2, base=5) # 5 as base, 2 as exp
```

Out[5]:
```
25
```

目前還沒有載入標準/第三方模組函數的需求

之後在「使用模組包裝函數與類別」章節會說明。

練習題就是來自使用者定義的函數

```
def function_name(INPUTS: TYPE, ARGUMENTS: TYPE) -> TYPE:
    ### BEGIN SOLUTION
    OUTPUTS = INPUTS (+-*/...) ARGUMENTS
    return OUTPUTS
    ### END SOLUTION
```

現階段我們先瞭解一些關於自行定義函數的組成

- `def` 保留字用來定義函數的名稱。
- 縮排部分稱為程式區塊（Code block），是函數的主體，也是練習題要學員運用預期輸入與參數來完成的部分。
- 不要忘記把函數的預期輸出寫在 `return` 保留字後。
- 函數的類別提示（Typing）並不是必要的，但它能幫助我們更快理解練習題。

哈囉世界中的 `"Hello, world!"` 是什麼

- `"Hello, world!"` 是 `str` 類別的字面值（Literal value）。
- 除了 `str` 類別，字面值也可以是其他的資料或資料結構類別。
- 單純的字面值不太實用，更好的做法是以一個物件名稱去參照字面值！

In [6]:
```
hello_world = "Hello, world!"
hello_world
```

Out[6]:
```
'Hello, world!'
```

如何說明 `hello_world = "Hello, world!"`

- `hello_world` 物件是 `str` 類別的實例（Instance）。
- 我們可以使用 `=` 符號讓 `object_name` 成為 `literal_value` 類別的實例，供後續的程式使用。

```
object_name = literal_value
```

筆記本的顯示規則

- 會將程式儲存格最後一列的字面值、物件或函數輸出。
- 如果想要有多個輸出，必須明確地使用 `print()` 函數。

常見的筆記本使用習慣

- 如果程式儲存格只需要一個輸出，直接參照字面值、物件或函數。
- 如果程式儲存格需要多個輸出，全部都使用 `print()` 函數。

In [7]:

```
# Single output in a code cell
hello_world = "Hello, world!"
hello_world
```

Out[7]:

```
'Hello, world!'
```

In [8]:

```
# Multiple outputs in a code cell
print(hello_world)
print(hello_world)
print(hello_world)
```

```
Hello, world!
Hello, world!
Hello, world!
```

物件與函數的命名規則

- 使用全小寫英文，採用蛇形命名法（Snake case），不同單字之間以底線 `_` 相隔。
- 不能使用保留字作命名。
- 使用單數名詞為資料類別的物件命名、使用複數名詞為資料結構類別的物件命名、使用動詞為函數或方法命名，盡量讓名稱簡潔且具有意義。
- 不要使用內建函數作物件的命名，避免覆蓋內建函數的功能。

來源：PEP 8 -- Style Guide for Python Code

什麼是保留字

- 保留字是具有特殊作用的指令。
- 目前有看過的 `import`、`def` 與 `return` 等都是 Python 的保留字。
- Python 的保留字一覽：
  https://docs.python.org/3/reference/lexical_analysis.html#keywords

有時候我們在程式碼之中會看到用來解釋的說明文字

- 註解（Comments）是口語化的文字敘述，以 `#` 標記，並不能夠被翻譯成電腦語言。
- 註解可以細分為單行註解、行末註解。

In [9]:
```
# A hello world example
hello_world = "Hello, world!" # hello_world is an instance of str class
hello_world                   # show hello_world object
```

Out[9]:
```
'Hello, world!'
```

各司其職的資料類別

五種基礎資料類別

1. `int` (integer)
2. `float`
3. `str` (string)
4. `bool` (boolean)
5. `NoneType`

「簡介」章節提過什麼是程式設計

以程式語言來指定電腦來解決特定問題的過程，基礎資料類別的存在就是為了讓電腦能夠：

- 以 `int` 與 `float` 進行數值計算。
- 以 `str` 表達與進行文字操作處理。
- 以 `bool` 進行流程控制。

使用內建函數 `type()` 確認物件所參照的資料類別

我們通常都會用一個物件去參照字面值，而物件名稱或 `print()` 函數並不能反映資料類別。

In [10]:
```
favorite = "5566" # favorite boy group?
print(favorite)
favorite = 5566   # favorite number?
print(favorite)
```

```
5566
5566
```

In [11]:

```
my_lucky_number = 5566
full_marathon_distance_in_km = 42.195
my_favorite_boy_group = "5566"
print(type(my_lucky_number))
print(type(full_marathon_distance_in_km))
print(type(my_favorite_boy_group))
```

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

數值與數值運算符

数値

- `int` (integer)
- `float`

In [12]:
```
my_lucky_number = 5566
full_marathon_distance_in_km = 42.195
print(type(my_lucky_number))
print(type(full_marathon_distance_in_km))
```

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

数値運算符

- 直觀的加減乘除 `+`, `-`, `*`, `/`
- 次方 `**`
- 計算餘數 `%`
- 計算商數 `//`
- 優先運算 `()`

次方運算的範例：與內建函數 `pow()` 相同作用的自行定義函數 `power()`

In [13]:

```
def power(x, n):
    return x**n

print(power(5, 3))
print(pow(5, 3))
```

```
125
125
```

數值運算符的順位

1. 最高順位是優先運算 `()`
2. 第二順位是次方 `**`
3. 第三順位是計算餘數 `%` 計算商數 `//` 乘 `*` 以及除 `/`
4. 第四順位是加 `+` 減 `-`

優先運算的範例：將華氏溫度轉換為攝氏溫度的自行定義函數 `convert_fahrenheit_to_celsius()`

$$\text{Celsius}(^\circ\text{C}) = (\text{Fahrenheit}(^\circ\text{F}) - 32) \times \frac{5}{9} \tag{1}$$

In [14]:

```python
def convert_fahrenheit_to_celsius(fahrenheit):
    celsius = (fahrenheit - 32) * 5/9  # the formula converts fahrenheit to celsius.
    return celsius

print(convert_fahrenheit_to_celsius(32))
print(convert_fahrenheit_to_celsius(212))
```

```
0.0
100.0
```

科學計號 `e`

- 數值可以支援科學記號（Scientific notation）`e`
- 注意不要和自然對數函數的底數 $e = 2.71828...$ 搞混了。

In [15]:
```
print(3e0)
print(3e2)
print(3e-2)
```

```
3.0
300.0
0.03
```

文字與文字運算符

使用成對的單引號 `'`、雙引號 `"` 或三個雙引號 `"""` 形成 `str`

In [16]:

```
str_with_single_quotes = 'Hello, world!'
str_with_double_quotes = "Hello, world!"
str_with_triple_double_quotes = """Hello, world!"""
print(type(str_with_single_quotes))
print(type(str_with_double_quotes))
print(type(str_with_triple_double_quotes))
```

```
<class 'str'>
<class 'str'>
<class 'str'>
```

`str` 中可能會包含單引號或雙引號，有兩種解決方式

1. 以反斜線 `\`（又稱跳脫符號）作為標註。
2. 以不同樣式的引號來形成 `str` 藉此區隔。

In [17]:

```
mcd = 'I\'m lovin\' it!'   # escape with \
mcd = "I'm lovin' it!"     # use different quotation marks
mcd = """I'm lovin' it!""" # use different quotation marks
```

善用成對的三個雙引號 `"""` 形成 `str`

- 常用來形成有換行、段落、有單引號以及有雙引號的文章或其他語言的程式碼。
- 放置在自行定義函數的主體第一列可以作為說明（Docstring）。
- 利用反斜線 `\` 讓程式碼在顯示換行但作用連續。

In [18]:

```
# Use \ for implicit continuation
shawshank_redemption_storyline = """Chronicles the experiences of a formerly successful \
banker as a prisoner in the gloomy jailhouse of Shawshank after being found guilty \
of a crime he did not commit. The film portrays the man's unique way of dealing \
with his new, torturous life; along the way he befriends a number of fellow prisoners, \
most notably a wise long-term inmate named Red."""
type(shawshank_redemption_storyline)
```

Out[18]:

```
str
```

In [19]:

```
# Use \ for implicit continuation
sql_query =\
"""
SELECT *
  FROM world
 WHERE country = 'Taiwan';
"""
type(sql_query)
```

Out[19]:

```
str
```

使用內建函數 `help()` 查詢函數的使用方法

In [20]:
```
help(abs)
```

```
Help on built-in function abs in module builtins:

abs(x, /)
    Return the absolute value of the argument.
```

使用筆記本寫程式可以在函數的小括號中按 `Shift - Tab` 查詢使用方法

In [21]:
```
#abs()
#pow()
```

將 `str` 放置在自行定義函數主體第一列可以作為說明 (Docstring)

In [22]:

```
def power(x, n):
    return x**n

help(power)
```

```
Help on function power in module __main__:

power(x, n)
```

In [23]:

```
def power(x, n):
    """
    Equivalent to x**n.
    """
    return x**n

help(power)
```

```
Help on function power in module __main__:

power(x, n)
    Equivalent to x**n.
```

能對 `str` 使用的文字運算符

- 加號 `+` 能夠連接 `str`
- 乘號 `*` 能夠複製 `str` 出現次數。

In [24]:
```
m = """I'm"""
c = """ lovin'"""
d = """ it!"""
print(m + c + d)
```

```
I'm lovin' it!
```

In [25]:
```
mcd = m + c + d
print(mcd*3)
```

```
I'm lovin' it!I'm lovin' it!I'm lovin' it!
```

另一個 `str` 的重要技巧是特定顯示格式

- 具體來說是在 `str` 中嵌入物件，如此顯示的內容將隨著物件中所儲存的值而變動。
- 可以透過加號來操作，但是這個做法可讀性較低且無法指定顯示格式。

In [26]:

```
def say_hello_to_anyone(anyone):
    return "Hello, " + anyone + "!"

print(say_hello_to_anyone("world"))
print(say_hello_to_anyone("Python"))
```

```
Hello, world!
Hello, Python!
```

透過 `str.format()` 方法以及 f-string 語法搭配大括號 `{}` 指定顯示格式

```
"{}".format(object_name) # str.format() 方法
f"{object_name}"         # f-string 語法
```

特定顯示格式的範例：`say_hello_to_anyone()` 函數

In [27]:

```
def say_hello_to_anyone(anyone):
    return "Hello, {}!".format(anyone) # str's format method

print(say_hello_to_anyone("world"))
print(say_hello_to_anyone("Python"))
```

```
Hello, world!
Hello, Python!
```

In [28]:

```
def say_hello_to_anyone(anyone):
    return f"Hello, {anyone}!" # f-string syntax

print(say_hello_to_anyone("world"))
print(say_hello_to_anyone("Python"))
```

```
Hello, world!
Hello, Python!
```

我常用的特定顯示格式

- 指定小數點 `n` 位的浮點數格式 `:.nf`。
- 具有千分位逗號的金額格式 `:,`。

In [29]:

```
def fahrenheit_to_celsius(fahrenheit):
    celsius = (fahrenheit - 32) * 5/9
    msg = "{:.1f} degrees fahrenheit is equal to {:.1f} degrees celsius.".format(fahrenheit, celsius)
    return msg

print(fahrenheit_to_celsius(32))
print(fahrenheit_to_celsius(212))
```

```
32.0 degrees fahrenheit is equal to 0.0 degrees celsius.
212.0 degrees fahrenheit is equal to 100.0 degrees celsius.
```

In [30]:

```python
def big_mac_index(country, currency, price):
    msg = f"A Big Mac costs {price:,.2f} {currency} in {country}."
    return msg

print(big_mac_index("Taiwan", "TWD", 72))
print(big_mac_index("US", "USD", 5.65))
print(big_mac_index("South Korea", "Won", 6520))
```

```
A Big Mac costs 72.00 TWD in Taiwan.
A Big Mac costs 5.65 USD in US.
A Big Mac costs 6,520.00 Won in South Korea.
```

其他更多特定顯示格式

https://www.w3schools.com/python/ref_string_format.asp

布林、關係運算符與邏輯運算符

什麼是 `bool`

布林（Boolean）是程式設計中用來表示邏輯的資料類別，以發明布林代數的數學家 George Boole 為名，它只有兩種值：假（`False`）和真（`True`），在程式設計與資料科學可以進行流程控制以及資料篩選。

有三種方式可以形成 `bool`

1. 直接使用保留字 `False` 或 `True`
2. 使用關係運算符。
3. 使用邏輯運算符。

直接使用保留字 `False` 或 `True`

In [31]:

```
bool_false = False
bool_true = True

print(bool_false)
print(bool_true)
print(type(bool_false))
print(type(bool_true))
```

```
False
True
<class 'bool'>
<class 'bool'>
```

使用關係運算符形成 `bool`

- 等於 `==`
- 不等於 `!=`
- 大於 `>` 小於 `<`
- 大於等於 `>=` 小於等於 `<=`
- 包含於 `in` 不包含於 `not in`

在關係運算符兩側放置物件或字面值形成 `bool` 稱為條件運算式（Conditional expression）

In [32]:

```
my_lucky_number = 5566
my_favorite_boy_group = "5566"

print(my_lucky_number == 5566)
print(my_lucky_number != 5566)
print(my_lucky_number > 5566)
print(my_lucky_number >= 5566)
print('56' in my_favorite_boy_group)
print('78' not in my_favorite_boy_group)
```

```
True
False
False
True
True
True
```

要將 `==` 以及 `=` 明確地區分出來

- `==` 是關係運算符，用來判斷符號兩側是否相等。
- `=` 是賦值運算符，用來讓左側的物件名稱成為右側類別的實例。

In [33]:
```
my_favorite_boy_group = "5566"
my_favorite_boy_group == 5566
```

Out[33]:
```
False
```

使用邏輯運算符形成 `bool`

- 交集 `and`
- 聯集 `or`
- 非 `not`

非 `not` 運算符將 `bool` 反轉

In [34]:
```
bool_false = False
bool_true = True
print(not bool_false)
print(not bool_true)
```

```
True
False
```

邏輯運算符針對已經是 `bool` 的資料進行集合運算

- 在交集 `and` 運算符兩側必須都是 `True` 判斷結果才會是 `True`
- 在聯集 `or` 運算符兩側必須都是 `False` 判斷結果才會是 `False`

在交集 `and` 運算符兩側必須都是 `True` 判斷結果才會是 `True`

In [35]:

```
bool_false = False
bool_true = True
print(bool_true and bool_true)
print(bool_true and bool_false)
print(bool_false and bool_true)
print(bool_false and bool_false)
```

```
True
False
False
False
```

在聯集 `or` 運算符兩側必須都是 `False` 判斷結果才會是 `False`

In [36]:

```
bool_false = False
bool_true = True
print(bool_false or bool_false)
print(bool_true or bool_true)
print(bool_true or bool_false)
print(bool_false or bool_true)
```

```
False
True
True
True
```

交集 `and` 運算符的範例：`is_a_good_marathon_weather()` 函數

如果一個適合跑馬拉松的天氣定義是「乾」**並且**「冷」，假設比賽日乾燥與低溫的機率均為 50%，適合跑馬拉松的機率為 25%。

In [37]:
```
def is_a_good_marathon_weather(is_dry, is_cold):
    return is_dry and is_cold

print(is_a_good_marathon_weather(True, True))
print(is_a_good_marathon_weather(True, False))
print(is_a_good_marathon_weather(False, True))
print(is_a_good_marathon_weather(False, False))
```

```
True
False
False
False
```

聯集 `or` 運算符的範例：`is_a_good_marathon_weather()` 函數

如果一個適合跑馬拉松的天氣定義是「乾」**或者**「冷」，假設比賽日乾燥與低溫的機率均為 50%，適合跑馬拉松的機率為 75%。

In [38]:
```
def is_a_good_marathon_weather(is_dry, is_cold):
    return is_dry or is_cold

print(is_a_good_marathon_weather(False, False))
print(is_a_good_marathon_weather(True, True))
print(is_a_good_marathon_weather(True, False))
print(is_a_good_marathon_weather(False, True))
```

```
False
True
True
True
```

未定義値 `NoneType`

什麼是 `NoneType`

- Python 表示未定義值、空值或者虛無值的特殊資料類別，只有一個值：`None`。
- `NoneType` 既不是 `False`。
- `NoneType` 也不是空的 `str` 類別 `''`。
- `NoneType` 也不是 `0`。

In [39]:

```
a_none_type = None
print(type(a_none_type))
print(a_none_type == None)
print(a_none_type == False)
print(a_none_type == '')
print(a_none_type == 0)
```

```
<class 'NoneType'>
True
False
False
False
```

如果在自行定義函數時沒有使用 `return`，函數將輸出 `NoneType`

這也是「練習題指引」中寫到若只是用 `print()` 函數將預期輸出印出，並無法通過測試的根本原因，輸出 `NoneType` 無法通過和預期輸出比對的測試資料。

In [40]:

```python
def convert_fahrenheit_to_celsius(fahrenheit):
    celsius = (fahrenheit - 32) * 5/9
    print(celsius) # Instead of return celsius, we just print celsius.

function_output = convert_fahrenheit_to_celsius(212)
print(function_output)
print(type(function_output))
```

```
100.0
None
<class 'NoneType'>
```

資料類別的判斷與轉換

資料類別判斷的兩種方式

1. 將 `type()` 函數的輸出與資料類別名稱比較。
2. 透過內建函數 `isinstance()` 判斷。

資料類別可以透過內建函數 `type()` 判斷

將 `type()` 函數的輸出與資料類別名稱比較。

In [41]:

```
a_bool_false = False
a_str_false = "False"
print(type(a_bool_false) == bool)
print(type(a_str_false) == bool)
```

```
True
False
```

In [42]:

```
an_integer_5566 = 5566
a_str_5566 = "5566"
print(type(an_integer_5566) == int)
print(type(a_str_5566) == int)
```

```
True
False
```

In [43]:

```
a_none_type = None
a_none_str = "None"
print(type(a_none_type) == type(None))
print(type(a_none_str) == type(None))
```

```
True
False
```

資料類別可以透過內建函數 `isinstance()` 判斷

`isinstance(x, classinfo)` 函數判斷輸入物件 `x` 是否為某個資料類別的實例，其中 `classinfo` 參數輸入欲判斷的資料類別名稱。

In [44]:

```
# Use bool as classinfo
a_bool_false = False
a_str_false = "False"
print(isinstance(a_bool_false, bool))
print(isinstance(a_str_false, bool))
```

```
True
False
```

In [45]:

```
# Use int as classinfo
an_integer_5566 = 5566
a_str_5566 = "5566"
print(isinstance(an_integer_5566, int))
print(isinstance(a_str_5566, int))
```

```
True
False
```

In [46]:

```
# Use type(None) as classinfo
a_none_type = None
a_none_str = "None"
print(isinstance(a_none_type, type(None)))
print(isinstance(a_none_str, type(None)))
```

```
True
False
```

資料類別可以透過與其同名的內建函數轉換

- `bool()` 可以將輸入資料轉換為 `bool`
- `int()` 可以將輸入資料轉換為 `int`
- `float()` 可以將輸入資料轉換為 `float`
- `str()` 可以將輸入資料轉換為 `str`

資料類別轉換的包容性

由資料類別的子集合往資料類別的母集合轉換可以確保不會出錯。

$$\text{bool} \in \text{int} \in \text{float} \in \text{str} \tag{2}$$

In [47]:

```
# Upcasting is always allowed.
print(int(False))
print(float(0))
print(str(0.0))
```

```
0
0.0
0.0
```

資料類別轉換的包容性（續）

由資料類別的母集合往資料類別的子集合轉換需要注意是否符合邏輯、是否正確。

$$\text{bool} \in \text{int} \in \text{float} \in \text{str} \tag{3}$$

In [48]:
```
print(bool('0'))
print(bool('False'))
```

```
True
True
```

`str` 無法被轉換為 `float`

`try...except...` 是例外處理的語法，之後在「使用流程控制管理程式區塊的執行」章節會說明。

In [49]:
```
try:
    print(float('Hello world!'))
except ValueError as error_message:
    print(error_message)
```

```
could not convert string to float: 'Hello world!'
```

重點統整

- Python 的內建函數：https://docs.python.org/3/library/functions.html
- Python 的保留字：https://docs.python.org/3/reference/lexical_analysis.html#keywords
- Python 風格指南：https://www.python.org/dev/peps/pep-0008

重點統整（續）

- 各司其職的資料類別：
    - 以 `int` 與 `float` 進行數值計算。
    - 以 `str` 表達與進行文字操作處理。
    - 以 `bool` 進行流程控制。

重點統整（續）

- `int` 與 `float` 支援數值運算符以及科學記號 `e`。
- `str` 支援文字運算符以及一個重要的應用為特定顯示格式。
- `bool` 可以透過保留字、關係運算符或邏輯運算符生成。

重點統整（續）

- 以 `type()` 與 `isinstance()` 函數判斷物件屬於哪一個資料類別。
- 以透過與資料類別同名的內建函數轉換。