Python Tutorial

Wu-Jun Pei @ NTU CSIE Camp 2019

課程素材連結：https://www.csie.ntu.edu.tw/~b06902029/reveal.js/CSCamp2019/

Lecture 1 - Introduction

1.1 Brief Introduction

Python 是一種廣泛使用的**直譯式**、進階編程、通用型程式語言。Python的設計哲學強調程式碼的可讀性和簡潔的語法（尤其是使用 **空格縮排** 劃分程式碼塊，而非使用大括號或者關鍵詞）。相比於C++或Java，Python讓開發者能夠用更少的代碼表達想法。不管是小型還是大型程式，該語言都試圖讓程式的結構清晰明了。 與Scheme、Ruby、Perl、Tcl等動態型別程式語言一樣，Python擁有動態型別系統和垃圾回收功能，能夠自動管理記憶體使用，並且支援多種編程範式，包括 **物件導向**、命令式、函數式和程序式編程。其本身擁有一個巨大而廣泛的**標準庫**。 Python 直譯器本身幾乎可以在 **所有的作業系統中執行**。Python的其中一個直譯器CPython是用C語言編寫的、是一個由 **社群** 驅動的自由軟體，目前由Python軟體基金會管理。節錄自 wikipedia。

- 直譯式：相對於編譯式程式語言如 C/C++，Python 不需要編譯
- 空格縮排：嚴格要求空白的數量（通常為 4 個空格）
- 標準庫：Python 的標準函式庫涵蓋相當多面向，日常生活中所需的許多簡單程式已經可以輕易地用標準函式庫完成
- 社群：除了標準函式庫之外，Python 也因為社群的眾多第三方函式庫而變得更加好用、促進更多使用者加入。著名的第三方函式庫如 numpy、scipy、matplotlib、pandas、pygame 等，目前 Python Package Index (PyPI) 上面已有超過 18 萬個 project 了

1.2 Hello World

Hello World 是初學者在接觸一個新的程式語言時會學習的程式。這個程式相當基本、簡單，沒有輸入，只有一行輸出 “Hello World!”

```
print('Hello World!')
```

1.3 程式運行

程式執行程式碼的順序和人類閱讀時相同，從頭到尾一行一行執行。Python 因為不用執行，所以只有在該行被執行出錯誤時才知道程式碼出現問題，無法在寫完程式碼的瞬間就藉由「編譯」得知程式碼的錯誤

註解

在編寫程式碼時，我們常常會因為提供更多資訊（X、避免自己看不懂自己年少輕狂寫的程式碼（O，而加上一些註解。註解的那行程式不會執行而直接跳掉。在 Python 中，註解的符號是井字號 `#`，使用如下：

```
# 這行是註解
print('我不是註解')
```

Lecture 2 - I/O & Variables

2.1 Input/Output

Input

> 建議先服用 `str` 後可以讀懂 80% 內容，服用 `function` 後可以更清楚的理解其他內容

```
userInput = input() # Basic
name = input('Please input your name: ') # 加上輸入提示
```

- `input` ：輸入函式。 以行為單位，從使用者輸入每次讀取一行，以上例為例，`userInput` 會讀取第一行使用者輸入、`name` 會在印出 `"Please input your name: "` 後讀取第二行使用者輸入
- 參數：一個字串（使用者提示）
- 回傳值：一個字串（使用者輸入）

Output

```
print('Hello World')                # 印出一個變數
print(3, 0.8787, 'Yeah')            # 印出多個變數
print(1, 2, 3, sep = '|', end = '') # 改變分隔多個變數的字元，行尾不印換行
```

- `print`：輸出函式。 依序輸出一個或多個變數，以 `sep` 隔開，最後印出 `end` 作為收尾。
- 參數
  - 一個或多個變數
  - `sep`：分隔字串，預設是空格 `' '`
  - `end`：結尾字串，預設是換行 `'\n'`

跳脫字元

有些符號在電腦上打不出來，如換行符號在直譯器或者編輯器按一下 enter 顯然是沒用的、tab 符號按下 tab 也是沒用的，跳脫符號 `\` 就幫我們解決這類的問題，以下列出常見的跳脫字元：

- `\n`：換行符號
- `\t`：tab 符號
- `\\`：跳脫符號本身，如果只按一個 `\` 他會以為那是某一個跳脫符號ㄛ！
- `\'` / `\"`：單/雙引號
- `\a`：警告（螢幕會亮一下、發出提示聲）

2.2 Variables

變數（Identifier）的命名

- 使用英文字母（`'a-zA-Z'`）、數字（`'0-9'`）、底線（`'_'`），數字不能擺在開頭
- 不能與 keywords 相同，完整的 Keyword List

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

查看一個變數的型態

```
type(variable)
```

賦值

在 python 以及許多程式語言中，一個等號 `=` 代表的都是「賦值」，有種數學中 let 的感覺。若要「判斷是否等於」時，通常使用兩個等號 `==` 判斷

```
a = # 這邊放值，表示賦值
```

常見型態

int / float

- Integer（整數）：沒有範圍限制
- Float（浮點數）：計算小數的方式，容易有誤差（見以下範例的 a / b）
- 七則運算

```
a = 7
b = 3

# 基本四則運算
print(a + b)    # Output: 10
print(a - b)    # Output: 4
print(a * b)    # Output: 21
print(a / b)    # Output: 2.3333333333333335

# 整數除法中的「商」
print(a // b)   # Output: 2
# 整數除法中的「餘數」，稱之為「模運算」，唸作「a 模 b」
print(a % b)    # Output: 1

# a 的 b 次方
print(a ** b)   # Output: 343
```

- 修改變數值 有兩個方法，意義不同但是有相同的結果
  - 重新賦值：計算完再存回變數之中
  - 使用「運算子等於」：將值直接做更動（常用）

```
x = 0

x = x + 3   # 此時 x = 3，將 x + 3 計算完存到 x 這個變數中
x += 4      # 此時 x = 7，將 x 加 4
x = x - 2   # 此時 x = 5，將 x - 2 計算完存到 x 這個變數中
x -= 1      # 此時 x = 4，將 x 減 1

x *= 3.5    # 此時 x = 14，將 x 乘 3.5
x /= 0.7    # 此時 x = 20，將 x 除 0.7

x %= 11     # 此時 x = 9，將 x 模 11
x //= 2     # 此時 x = 4，將 x 除二取商
x **= 3     # 此時 x = 64，將 x 立方
```

bool

- Bool（布林變數）：只有 `True` 以及 `False`

```
a = True
b = False
```

None

- 什麼都不是

```
a = None
```

str

- String（字串）：儲存一些字，不限語言，使用單引號 `'` 或雙引號 `"` 包起來

```
s1 = 'Python is so easy'
s2 = "維大力，義大利"
```

- str 是 **immutable（不可改變的）**，即你不能只更改一個字串裡的某一個字，需要重新賦值

```
s = 'abc'
#    012    編號從 0 開始

s[2] = 'd'  # 不能這樣改
s = 'abd'   # 重新賦值
```

NOTE：編號是從 **0** 開始，之後的所有有多個元素和在一起的 type 也是從 **0** 開始

- 加乘

```
pineapple = 'pine' + 'apple'    # 'pineapple'
oops = 'o' * 10 + 'ps'          # 'oooooooooops'
```

list

- List（序列）：一串值，可以是不同型態，中括號 `[]` 是關鍵

```
myList = ['string', 3, -0.87, ['List', 'in', 'the', 'List']]
#         -- 0 ---  1  - 2 -  ------------- 3 -------------
```

NOTE：再次提醒，編號是從 **0** 開始

- Append：從後方插入

```
myList.append(None)
```

- Get/Set Item：存取 / 更改元素

```
myList[1] = 33
print(myList[3])
```

- 加乘

```
l1 = [1, '2', 3] + ['4', 5, 6]  # [1, '2', 3 + '4', 5, 6]
l2 = [True, False] * 3          # [True, False, True, False, True, False]
```

tuple

- Tuple（多元組，唸 *他剖*）：跟 List 很像，只是 ***immutable（不可改變的）***，逗點 `,` 是精髓，小括號 `()` 是輔助

```
t1 = (1, 2)     # 用括號括起來
t2 = 3,'four',5 # 逗點才是精髓
t3 = None,      # 使用「逗點」讓 t3 變成一個只有一個元素的 tuple
```

- Useful Tips

```
a, b = 1, 3             # 使用 tuple 一次宣告兩個變數
c, d = 'a', 'b', 'c'    # 錯誤，此時需要小括號幫忙括
c, d = 'a', ('b', 'c')  # 正確，c 對上 'a'、d 對上 ('b','c')

a, b = b, a             # Swap
```

  - 一次宣告多個變數
  - 將兩個變數的值互換

dict

- Dictionary（字典）：給定 key（英文），得知他的 value（中文），之中 key 以及 value 的型態都不需要唯一，大括號 `{}` 是精髓

```
my = {'Face' : 'Ugly', 'Salary' : 1e10, 'Over18' : True, 'girlfriend': None}
```

- Insert/Modify：插入 / 修改一個 key/value

```
my[19] = [1, 9]
my['Face'] = 'Pretty Handsome'
```

- keys & items

```
keys = list(my.keys())      # list of keys
items = list(my.items())    # list of (key, value) tuples
```

Len & Slice

Len

得知一個 str 的長度、list、tuple、dict 的大小

```
print(len('Five'))          # Output: 4
print(len([1, 2, 3, None])) # Output: 4
print(len(my))              # Output: 5
```

Slice

取得一個 str、list、tuple的區間，以 str 為例，list 及 tuple 有類似的功用

```
s = 'abcdefghijklmnopqrstuvwxyz'
#    0123456789
#              0123456789
#                        012345

print(s[-2])        # Output: 'y'       倒數第二個
print(s[:5])        # Output: 'abcde'   前五個字元
print(s[-4:])       # Output: 'wxyz'    最後四個字元
print(s[::5])       # Output: 'afkpuz'  每五個取一次
```

- 若數字為負，則代表**倒數幾個**，此技巧亦可用在 *Get/Set Item* 上。 NOTE：最後提醒，編號是從 **0** 開始
- `object[start : stop]`：從 start 開始取，直到取到 stop（不包含 stop）。
  - 若 start 不填寫則代表從頭開始
  - 若 step 不填寫則代表最後到尾巴
- `object[start : stop : step]`：從 start 開始取，每 step 取一次，直到超過 stop（不超過 stop）。
  - start 跟 stop 的差距不一定要是 step 的倍數

轉型

使用 `型態(variable)` 即可得到 variable 在該型態的值

```
str_a = '33'
int_a = int(str_a)          # 轉換成 int
print(int_a, type(int_a))

float_a = float(int_a)      # 再轉換成 float
print(float_a, type(float_a))
```

Input 套路使用

使用轉型將輸入直接變成想要的型態

```
inputInteger = int(input('Please input an integer: '))
```

Lecture 3 - Flow Control (i)

3.1 判斷式

一個判斷式子是否成立，回傳 bool 型態（True/False）

基本判斷子

- `==`：判斷值是否等於
- `!=`：判斷值是否不不等於
- `<` / `<=` / `>=` / `>`：小於 / 小於等於 / 大於等於 / 大於

```
print(1 < 3)            # Output: True
print(3.14 >= 4)        # Output: False
print(1 + 1 == 2)       # Output: True
print(1 + 2 * 3 == 9)   # Output: False
```

- `a < b < c`：三個以上的變數也是可以的

字典序

字串的等於 / 不等於很簡單，那字串怎麼比大小？

- 比較第一個不一樣的字母，ASCII 值比較小的在前。常用的 ASCII 順序：`0-9` < `A-Z` < `a-z`
- 若沒有該字母（imply 某一字串為另一字串前綴），則短者較小

```
print('a' < 'b')            # Output: True
print('as' < 'False')       # Output: False
print('sigh' < 'sight')     # Output: True
```

is

1. 用於判斷變數型態

   ```
   print(type(1) is int)   # Output: True
   ```

2. 判斷兩變數是否是同一個東西

   ```
   l1 = [1, 2, 3]
   l2 = [1, 2, 3]
   print(l1 == l2)     # Output: True  值相同
   print(l1 is l2)     # Output: False 但為不同 object
   ```

in

- 判斷一個子字串是否在一個 str 之中

  ```
  print('doll' in 'dollar')   # Output: True
  print('fb' in 'facebook')   # Output: False
  ```

- 判斷一個變數是否在一個 list/tuple/dict 之中

```
print(1 in [1, 2, 3])                          # Output: True
print(0 in {1 : 'one', 2 : 'two', 3 : 'three'}) # Output: False
print('one' in {1 : 'one', 2 : 'two', 3 : 'three'})
# Output: False 判斷依據為 key
```

邏輯運算 not/and/or

- `not a`：True/False 反過來
- `a and b`：只有 a, b 都是 True 才是 True，其他都是 False
- `a or b`：只有 a, b 都是 False 才是 False，其他都是 True
- 運算子優先度：not > and > or。請適時加上小括號避免意想不到的狀況發生

```
if myGender == 'Male' and myXingZuo == 'Leo':
    print('He must be a handsome boy')

it = 0.1
if it is not int:
    print('Then it must be a float')
```

3.2 if

```
if condition:
    # do something
```

- 冒號：提示接下來要縮排
- condition：判斷是
- 縮排：四格空格，接下來有一樣縮排的程式碼都是同一個區塊

Example

因為期中考太爛了，所以**如果期末考低於 80 分，我就會被當**

```
if finalScore < 80:
    print('I will get a F')
```

3.3 if/elif/else

```
if condition1:
    # do something
elif condition2:
    # do something
elif condition3:
    # do something

# ...

else:
    # do something
```

- 第一個只能是 if；中間可以有任意 elif；最後可以有 else，也可以不要
- 善用 else 系列讓人生變得美好

Example

拍馬屁系統

```
score = int(input('Please input your math score: '))

if score >= 90:
    print('Great job')
elif score >= 60:   # 因為 else 的性質，不用特別判斷 x 是否小於 90
    print('Not bad, keep going on')
else:
    print('See you next year')
```

條件表達式

```
variable = trueValue if condition else falseValue

# Example
drink = 'water' if BMI > 24 else 'Bubble Milk Tea'
```

- `variable` 在 condition 成立時會被賦值成 `trueValue`，不成立時則是 `falseValue`
- 簡單美學

3.4 Nested Structure

```
if condition_1:
    if condition_1_1:
        # do something
    elif condition1_2:
        # do something
elif condition_2:
    # do something
else:
    if condition_3_1:
        # do something
    else:
        if condition_3_2_1:
            # do something
```

- 是蜂巢的「巢」不是很潮的「潮」
- 要多深就可以多深

Lecture 4 - Flow Control (ii)

4.1 while

```
while condition:
    # do something
```

- 冒號：提示接下來要縮排
- condition：判斷式
- 縮排：四格空格，接下來有一樣縮排的程式碼都是同一個區塊

Example

夏威夷Pizza終結者

```
remainPineapples = 13

while remainPineapples > 0:
    print('Pick some pineapples')
    remainPineapples -= 1

print('Finally...')
```

4.2 for

```
for variable in iterableObject:
    # do something
```

- variable：迭代時使用的變數
- iterableObject：可迭代的 object
- 冒號：提示接下來要縮排
- 縮排：四格空格，接下來有一樣縮排的程式碼都是同一個區塊

Iterable Objects

- iterate：to repeat a process, especially as part of a computer program，重複執行一段程式碼
- iterable object：可以從該 object 依次拿出東西 iterate

for - Example

```
# Iterate for 10 times
for i in range(10):
    print('Sorry, Sweet Heart. i =', i)

print(list(range(10)))
# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- `i`：迭代時使用的變數
- `range(10)`：類似一個從 0 跑到 9 的 list

range

- `range(n)`：從 $0, 1, \ldots, n-1$
- `range(m, n)`：從 $m, m+1, \ldots, n-1$
- `range(m, n, k)`：從 $m, m+k, m+2k, \ldots$ 直到超過範圍（k 可以是負數，$n-m$ 不一定要是 k 的倍數）
- 有種 *slice* 的感覺

```
print(list(range(6)))
# Output: [0, 1, 2, 3, 4, 5]

print(list(range(-3, 3)))
# Output: [-3, -2, -1, 0, 1, 2]

print(list(range(1, 13, 2)))
# Output: [1, 3, 5, 7, 9, 11]

print(list(range(19, 2, -3)))
# Output: [19, 16, 13, 10, 7, 4]
```

str/list/tuple/dict

- str：從 str 中拿出每個字元

  ```
  for c in 'Python':
      print(c, end = ' - ')

  # Output: 'P - y - t - h - o - n -'
  ```

- list/tuple：從 list/tuple 中拿出每個元素

  ```
  for p in [2, 3, 5, 7, 11, 13, 17, 19]:
      print(p, 'is a prime')
  ```

- dict：從 dict 拿出每個 key

  ```
  d = {'I' : 'My', 'You' : 'Your', 'We' : 'Our'}
  for key in d:
      print(key, '->', d[key])

  # Recall `items` will generate a list of (keym, value) pair
  for key, value in d.items():
      print(key, '->', value)
  ```

  以上兩個方法有一樣的輸出

enumerate

使用 `enumerate` 將原本的 iterable object 包起來形成一個新的 iterable object，每個會是一個 2-tuple `(i, a)`，`i` 代表他的順位，`a` 代表原本的變數。

Example

```
for i, name in enumerate(['Alice', 'Bob', 'Catherine', 'Dylan', 'Eve']):
    print('Number:', i, '; Name:', name)
```

All in One Line

- 常常用於將多個變換放在一行，避免程式碼過於冗長
- 使用起來非常舒爽，但常常走火入魔而導致一行過長

Example

輸入十六進位後轉成整數之後取負號

- 多行版

```
inputSequence = input('Input some hex numbers: ').split(' ')
# Input: 1c 353 90 aaba

intSequence = [int(x, 16) for x in inputSequence]   # 轉成整數
negSequence = [-x for x in intSequence]             # 轉成負數

print(negSequence)
# Output: [-28, -851, -144, -43706]
```

- 一行版

```
outputSequence = [-int(x, 16) for x in input('Input some hex numbers: ').split('
')]
print(negSequence)
```

4.3 continue & break

continue

直接進到下個 iteration，`continue` 之後的程式碼不會被執行

```
for i in range(1, 11):
    if i % 3 == 0:
        continue        # Jump to line 1, line 4 is skipped
    print((i // 3) / (i % 3))
```

break

跳出迴圈，`break` 之後的程式碼不會被執行

```
ans = 7.315
for i in range(1024):
    if i < ans and ans < i + 1:
        print('ans is between', i, 'and', i + 1)
        break
    print('Round', i, 'done')
```

Lecture 5 - Functions, Classes and Methods

5.1 Functions

def

```
def functionName(parameters):
    # do something
    return returnValue
```

- `functionName`：function 的名字
- `parameters`：傳進 function 的參數，可以有多個
- `return`：回傳 `returnValue`

Example - 分數衝高高

調整分數的 function，將 `scores` 中的分數乘 a 再加上 b

```
def adjustScore(scores, a, b):
    return [a * x + b for x in scores]

newScores = adjustScore([47, 72, 100, 60, 99], 0.5, 50)
print(newScores)

# Output: [73.5, 86.0, 100.0, 80.0, 99.5]
```

參數

- 可以有多個，不一定要同一個型態，依序傳入

在上面的 *Example - 分數衝高高* 之中，有三個參數 scores、a、b，分別代表原始分數、乘的倍數、加的數字。此時看到下面呼叫的時候，這三個參數也是依序傳入的。

參數預設值

```
def adjustScore(scores, a = 1, b = 0):
    return [a * x + b for x in scores]
```

- 我們也可以預設一些參數的值，這些沒有預設參數的變數必須要寫在有預設參數的前面（scores 寫在 a、b 的前面）
- 如果兩個參數都不打，則因為 a = 1, b = 0 而回傳一個沒有改變的 list。

  ```
  print(adjustScore([47, 72, 100, 60, 99]))
  # Output: [47, 72, 100, 60, 99]
  ```

- 打上參數的名稱

```
print(adjustScore([47, 72, 100, 60, 99], a = 2))
# Output: [94, 144, 200, 120, 198]

print(adjustScore([47, 72, 100, 60, 99], b = 10))
# Output: [57, 82, 110, 70, 109]

print(adjustScore([47, 72, 100, 60, 99], a = 0.5, b = 50))
# Output: [73.5, 86.0, 100.0, 80.0, 99.5]
```

- 不打上變數的名稱，則依序傳入參數

```
print(adjustScore([47, 72, 100, 60, 99], 2))    # 2 被傳給 a
# Output: [94, 144, 200, 120, 198]
```

回傳值

- 只能有恰好一個回傳值
- 通常可以用 type 查看回傳值型態

```
type(adjustScore([47, 72, 100, 60, 99], 0.5, 50))
```

在上面的 *Example - 分數衝高高* 之中，回傳值為一個 list

沒有回傳值

```
def sayHello(s):
    print('Hello,', s)

type(sayHello('Python'))    # 查看回傳值型態
```

- 函數都有回傳值，在不回傳東西時會回傳 `None`

多個回傳值

```
def quadratic(a, b, c):
    return (-b - (b ** 2 - 4 * a * c) ** 0.5) / (2 * a), \
        (-b + (b ** 2 - 4 * a * c) ** 0.5) / (2 * a)

x1, x2 = quadratic(1, 2, -3)
print(x1, x2)

type(quadratic(1, 2, -3))   #  查看回傳值型態
```

- 使用 tuple 將多個回傳值包成一個，問題解決！

常用 Built-in Functions

- Built-in Functions List
- `abs`：回傳一個 int/float 的絕對值

```
abs(3.14159)    # Output: 3.14159
abs(-11)        # Output: 11
```

- `max/min`：回傳若干個變數之中（或者 iterable object 之中）最大 / 小的那個。每個元素倆倆之間必須要可以比較，否則會有 `TypeError`

```
# 使用多個變數作為參數
max('0', 'a', 'A')
min(-3, 3)

# 使用 list 作為參數
max([34,  9,  2,  1, 26, 26, 39, 27, 18, 43, 18, 24])   # Output: 43
min(word for word in '七月十八日不會有颱風')              # Output: '七'
```

> 最後一個例子是比較各個中文字的字典序，剛好「七」最小

- `sorted`：回傳一個排序好的 list（原本的 iterable object 則不改變）。
  - `key`：一個 function，代表 a 是依據那個函式的值排序的。
  - `reverse`：布林變數，代表是否要前後反過來。

```
a = [34,  9,  2,  1, 26, 26, 39, 27, 18, 43, 18, 24]

print(sorted(a))
# Output: [1, 2, 9, 18, 18, 24, 26, 26, 27, 34, 39, 43]

def f(x):
    return (x - 27.3) ** 2

print(sorted(a, key = f, reverse = True))
# Output: [1, 2, 9, 43, 39, 18, 18, 34, 24, 26, 26, 27]
```

- `sum`：回傳一個 iterable object 的合。

```
sum(range(10))                                # Output: 45
sum(i ** 5 for i in [14, 33, 18,  8, 34, 49])   # Output: 369506226
```

5.2 Classes & Methods

Class 包含兩個部分，attributes 以及 methods。

- **Attributes**：屬性，可以想成是 class 的 variable。舉「寵物」為例，寵物的 attributes 可能有 *品種*、*名字*、*性別*、*顏色*、*重量*、*身長*.....
- **Methods**：可以想成是 class 的 function。舉「寵物」為例，寵物的 methods 可能有 *走路*、*叫*、*撒嬌*.....

Classes

```
class className:
    def __init__(self):
        self.attribute = # some value
        # do other things
```

- `className`：class 的名字
- `__init__`：建構子，其實也是一個 method，在 class 被宣告時會呼叫他。
- `self`：通常作為 method 的第一個參數，並作為和 class 本身的 attributes 以及 methods 溝通的橋樑。雖然可以命名為其他名稱，但 self 已成一個公認的命名傳統
- `self.attribute`：class 的 attribute

Example

```
class Pet:
    def __init__(self, species, name):
        self.species = species
        self.name = name
        self.pronounce = 'meow'
        self.weight = 0

cat = Pet('Cat', '寶寶')
print(cat)          # Output: <__main__.Pet object at [somewhere]>
print(type(cat))    # Output: <class '__main__.Pet'>
print(cat.name)     # Output: 寶寶
```

Methods

```
class className:
    # __init__

    def methodName(self):
        # do something

    def anotherMethod(self, parameters):
        # do something
```

- `methodName`：method 的名稱。
- method 可以想成是一個 function，只是第一個變數一定要是 `self`，若有其他變數則加在 self 的後面，如 `anotherMethod` 的 parameters。

Example

```
class Pet:
    # __init__

    def speak(self):
        print(self.pronounce)

    def eat(self, calories):
        self.weight += calories / 1000
        print('Weight becomes', self.weight)

cat = Pet('Cat', '寶寶')
cat.speak()                 # Output: meow
cat.eat(2473)               # Output: Weight becomes 2.473
```

常用型態之常用 method

str

- `str.split(sep)`：回傳一個使用 sep 將 str 切開的 list。

```
print('a|b|c|d|ee',split('|'))
# Output: ['a', 'b', 'c', 'd', 'ee']

print([int(x) for x in input('Input some numbers: ').split(' ')])
# Input: 1 23 456 7890
# Output: [1, 23, 456, 7890]
```

- `str.lower()/str.upper()`：回傳將字串的英文字母全部變成小 / 大寫的新字串

```
print('AaBbCc'.lower())         # Output: aabbcc
print('upper me'.upper())       # Output: UPPER ME
```

- `str.join(iterableObject)`：回傳一個包含 iterableObject 中所有字串，並使用 str 作為分隔的字串，iterableObject 的每個元素 **必須** 要是 str

```
print(' - '.join('MAY THE FORCE BE WITH YOU'.split(' ')))
# Output: MAY - THE - FORCE - BE - WITH - YOU
```

list

- `list.sort`：直接將該 list 排序好。
  - 參數一樣有 `key`、`reverse`，參見 *sorted*

```
a = [34,  9,  2,  1, 26, 26, 39, 27, 18, 43, 18, 24]
a.sort()
print(a)
# Output: [1, 2, 9, 18, 18, 24, 26, 26, 27, 34, 39, 43]
```