Python Tutorial

He-Zhe Lin 林鶴哲

2020.07.16 @ CSCamp

Overview

Chapter 4: if / else

Chapter 5: Loops

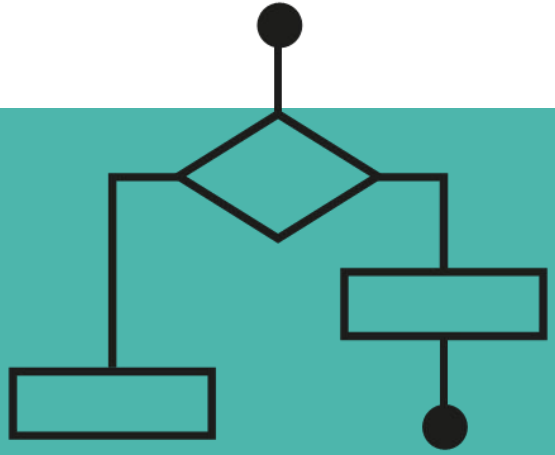
Chapter 6: Functions

Chapter 7: Class

Chapter 8: 資訊營之後

Learning 5 chapters in 2 hours, interesting :-)

if / else



if-else - 就只是條件判斷

```
>>> if condition_1:
...     # do A
>>> elif condtion_2:
...     # do B
>>> elif condition_3:
...     # do C
>>> else:
...     # do D
```

if + (任意) elif + (0 / 1個) else

冒號很重要

縮排也很重要(通常是 tab / 四個空白)

問題:condition 和 do A, B, C, D 要放什麼?

condition 就是是非題

是非題的答案是 **bool** 值

bool 值：只有兩種值的變數，也就是 True / False

- 如果 condition 為 True，就會跳到該 if / elif / else 區塊中執行
- 如果 condition 為 False，就會往下一個條件判斷看看是否為 True

問題：condition 可以放什麼？

比大小的 condition

語法:`A > B`, `A < B`, `A >= B`, `A <= B`, `A <= B <= C`

`A == B` (等於), `A != B` (不等於)

```
>>> 1 < 3
>>> 3.14 >= 4
>>> 1 + 1 == 2
>>> 1 + 1 = 2
>>> a, b, c = 1, 2, 3
>>> (a + b * c) != 9
>>> a < b < c
>>> b < c < a
```

邏輯的 condition 聽說 intro to CS 上過了 (Yeah~)

語法:

`A and B`	A 對 B 也對才是 True
`A or B`	A 對或 B 對就是 True
`not A`	A 對, not A 就是 False, A 錯, not A 就是對

```
>>> if (1 < 2) and (3 < 5):
....    print('True!')
>>> if (3 > 5) or (1 < 2):
....    print('True!')
>>> if (not (3 > 5)) and (1 > 2 or 2 > 3):
....    print('True!')
```

屬於關係的 condition

語法:[element] in [str/list/tuple]

```
>>> 'apple' in 'penpineappleapplepen'
>>> 'CS' in 'COMPUTER SCIENCE'
>>> 1 in [1, 2, 3]
>>> my_dict = {1 : 'one', 2 : 'two', 3 : 'three'}
>>> 0 in my_dict
>>> 'one' in my_dict
```

Example: 日期檢查器

給定年 (2020 ≤ 年 ≤ 2030)、月、日，檢查該日期是否存在。

```
year, month, day = int(input()), int(input()), int(input())
days_in_year = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

先判斷閏年的 2/29 (合法)

再判斷一般狀況

巢狀 if - 俄羅斯娃娃

```
if (year % 4, month, day) == (0, 2, 29):
    print('日期合法')
elif (1 <= month <= 12) and (1 <= day <= days_in_year[month-1]):
    print('日期合法')
else:
    if not (1 <= month <= 12):
        print('日期非法:月份錯誤')
    elif (month, day) == (2, 29):
        print('日期非法:該年是平年')
    else:
        print('日期非法:日子錯誤')
```

if 變形 - 動機

取數字的正負號

若 x ≥ 0 則 x = 1

否則 x = -1

```
if x >= 0:
    x = 1
else:
    x = -1
```

a, b 比大小

若 a > b 則 maximum = a

否則 maximum = b

```
if a >= b:
    maximum = a
else:
    maximum = b
```

不符合 Python 簡單、優雅（翻桌）

if-else - all in one line

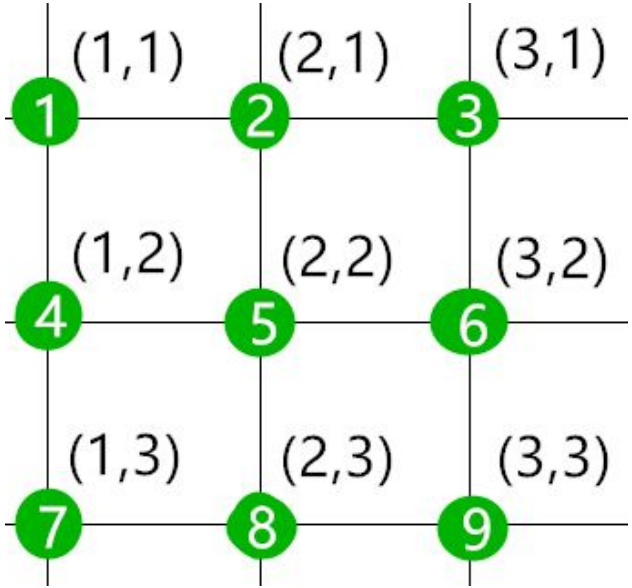
語法:[variable] = [true_value] if [condition] else [false_value]

如果 condition 為真, 把 variable 設成 true_value, 否則就設為 false_value

```
x = 1 if x >= 0 else -1
maximum = a if a > b else b
```

Exercise: 邊界判斷

在 3 X 3 網格中，輸入 x、y 座標，問處在該座標共有幾個移動方向（上、下、左、右）？(如 (2,3) 只有上、左、右，共 3 個方向)。



1. 輸入座標 x，y
2. 設一個變數 count 紀錄可以走的方向
3. 判斷關鍵：x 座標影響左右、y 座標影響上下。（你可能需要 4 個 if）
4. 把 count 印出來

懸賞：花費行數最少的小隊

Solution

```
x, y = int(input()), int(input())
count = 0
count = count + 1 if x != 1 else count
count = count + 1 if x != 3 else count
count = count + 1 if y != 1 else count
count = count + 1 if y != 3 else count
print(count)
```

【三分鐘懸賞：你唸我打】

輸入 A, B, C 三個人的成績到一個 list 當中，並計算他們的平均。

如：A 考 80 分，B 考 90 分，C 考 100 分，計算平均為 90 分。

【三分鐘懸賞：你唸我打 ver 2】

輸入 50 個人的成績，計算他們的平均。

Loops

while - 重復做到 condition 不成立

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

冒號很重要

縮排也很重要(tab / 四個空白)

如果 condition 為 True，do something，直到 condition 為 False 為止。

Example: 算平均

輸入 50 個人的成績, 計算它們的平均。

```
now_input, total = 0, 0
while now_input < 50:
    score = int(input())
    total += score
    now_input += 1
print(total / 50)
```

Exercise: 考拉茲猜想

對於每個正整數, 如果是奇數, 將其乘 3 加 1 , 如果是偶數, 將其除以 2, 如此循環, 終究會得到 1。(如:5 -> 16 -> 8 -> 4 -> 2 -> 1)

輸入一個整數, 請問要多少次操作後, 會得到 1 呢 ? (如 5: 5次)

1. 輸入一個整數
2. `while` 迴圈 `condition`: 該數字不等於 1
3. `while` 迴圈 `do something`: 用 `if` 分成奇數和偶數兩種狀況
4. 用一個 `count` 紀錄操作次數

懸賞:小於 10000 的數中, 需要最多操作次數才能得到 1 的是多少 ?

Solution: 考拉兹猜想

$$f(n) = \begin{cases} \frac{n}{2} & if\ n\ is\ even \\ 3n+1 & if\ n\ is\ odd \end{cases}$$

```
x = int(input())
times = 0
while x != 1:
    if x % 2 == 0:
        x = x / 2
    else:
        x = 3 * x + 1
    times += 1

print(times)
```

【國三英文句型】for ... in ...

Example:

For each **student** in the **class**, ...

For each **dumpling** in the **plate**, ...

For each **file** in the **folder**, ...

對於**大變數**當中的**每個小元素**, 我們都做某件事情...

for … in …

```
for thing in things:
    # do something
```

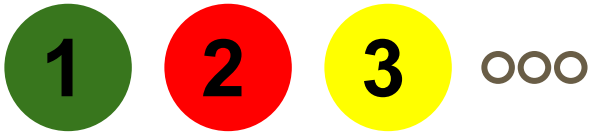
冒號很重要

縮排也很重要(tab / 四個空白)

對於所有 **things** 裡的每一個 **thing**, 我們都 do something

【注意】things 要是一個可迭代物件 (Iterable Object) (????)

迭代這檔事



可迭代物件：可以把內部元素一個一個取出來的物件

可以迭代：list, tuple, dict, str

不可迭代：int, float, bool, ...

for … in …

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

for letter in 'Python':
    print(letter)

plural_dict = {'datum': 'data', 'index': 'indices',\
               'matrix': 'matrices'}
for key in plural_dict:
    print('single', key, ',', 'many', plural_dict[key])
```

range - 產生連續數列

也是一個可迭代物件(可以從中一個一個拿東西出來)

```
>>> type(range(5))
<class 'range'>
>>> for i in range(5):
...     print(i, end=' ')

>>> for i in range(3, 10):
...     print(i, end=' ')
```

和從 list 當中取元素有夠像！

break / else - 跳過整個迴圈

想想看:什麼時候迴圈會結束

while 迴圈:condition 為 False 時

for 迴圈:迭代結束時

可不可以中途結束迴圈呢？

```
while condition:
    # do something
    if condition_2:
        break
else:
    # do something
```

Example

請檢查一個分數的 list 是否合法(落在 0-100 間)

```
score_list = [0, 100, 90, -1, 40]
for score in score_list:
    if score > 100 or score < 0:
        print('Invalid score!')
        break
else:
    print('Valid score.')
print('End of the program')
```

continue - 跳到下個迭代

continue 跳過該次迭代, 繼續一個迴圈

Example: 將 numbers 裡面的正整數相加。

```
numbers = [1, 3, -2, 1, -1, 5, 0]
total = 0
for number in numbers:
    if number < 0:
        continue
    total += number
print(total)
```

list comprehension

用一行產出一個 list

```
>>> my_list = [ x for x in range(5) ]

>>> my_list = [ 2 * x for x in range(5) if x % 2 == 0 ]

>>> my_list = [ x if x % 2 == 0 else -x for x in range(5) ]

>>> my_list = [ (x, y) for x in range(2) for y in range(2) ]
```

Exercise: 質數判斷

輸入一個正整數 x, 判斷 x 是不是質數。

```
x = int(input())
is_prime = True
for i in range(?, ?):
    ????
print(is_prime)
```

懸賞:用最短的時間判斷是不是質數的小隊

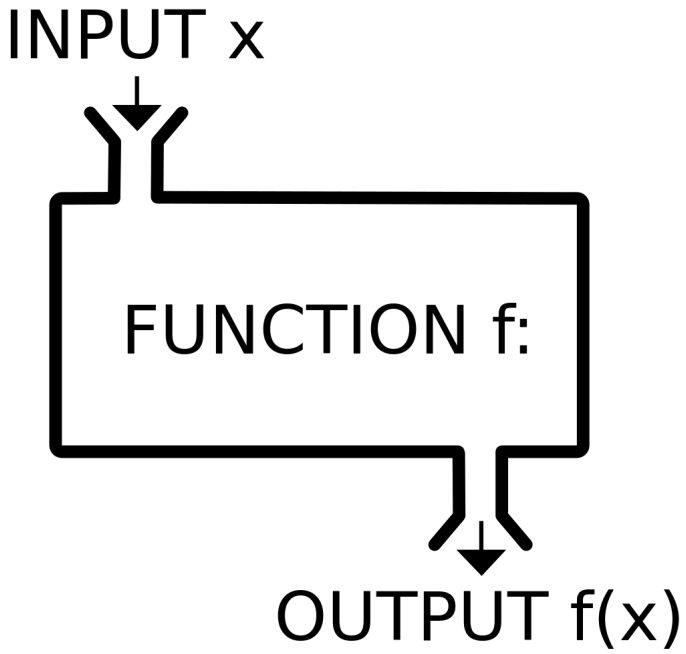
Solution: 質數判斷

```
x = int(input())
is_prime = True
for i in range(2, x):
    if x % i == 0:
        is_prime = False
print(is_prime)
```

```
x = int(input())
is_prime = True
for i in range(2, int(x**0.5)+1):
    if x % i == 0:
        is_prime = False
        break
print(is_prime)
```

Functions

數學中的函式



Python 中的函式

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

def : 宣告一個函式

functionName : 函式的名稱

parameters : 函式的 input 們（稱作**參數**）

: 記得加冒號 / 縮排四格

return_value : 回傳值 / 函式的 output

Example: 調分

設計一個函數, 將程式中 scores 中的分數乘 a 再加上 b。

`adjudtScore(scores, a, b): [float, ...]`

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

>>> new_scores = adjustScore(scores=[47, 72, 100, 60, 99], a=0.5, b=50)
>>> new_scores
[73.5, 86.0, 100.0, 80.0, 99.5]

>>> new_scores = adjustScore([47, 72, 100, 60, 99], 0.5, 50)
>>> new_scores
[73.5, 86.0, 100.0, 80.0, 99.5]
```

參數預設值

在定義函數的時候，可以設定參數預設值

呼叫時如果沒有寫就直接以預設值帶入了(簡單)～～

【注意】沒有預設值的參數要寫在有預設值的參數前方。

```
>>> def adjustScore(scores, a=1, b=0):
>>>     return [a * score + b for score in scores]
>>> adjustScore([47, 72, 100, 60, 99])
>>> adjustScore([47, 72, 100, 60, 99], b=10)
>>> adjustScore([47, 72, 100, 60, 99], a=0.5, b=50)
>>> adjustScore(a=0.5, b=50, [47, 72, 100, 60, 99])
```

不要重造輪子

早上舉的例子：

```
class Complex:
    def __init__(self, real, img):
        ...
    def add(self, other):
        ...
```

事情的真相是... 其實 Python 有內建的複數運算...

作者OS: 我寫那麼久是寫辛酸的？？？

別人寫好的函數強到你只要幾行呼叫就可以做到

矩陣乘法、資料處理、機器學習、架網頁...

內建函數 - max / min

回傳若干個變數之中最大 / 小的那個。

【注意】每個元素倆倆之間必須要可以比較, 否則會 TypeError

```
>>> max(1, 3, 8, 4, 2)
8
>>> min([1, 3.14, -2.718, 30])
-2.718
>>> max(0, 3, [1,2,3], 'str')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'list' and 'int'
```

內建函數 - abs / sum

abs: absolute value, 回傳一個 int/float 數字的絕對值

sum: 回傳一個 Iterable object 的和。

```
>>> abs(-11.30307)
11.30307
>>> abs(3.14)
3.14
>>> sum(range(10))
45
>>> sum(i ** 5 for i in [14, 33, 18, 8, 34, 49])
369506226
```

內建函數 - sorted

sorted: 回傳一個排序好的 list(原本的 Iterable object 不會改變)

- key: 一個 function, 代表每個元素根據該函數回傳的值排序的。
- reverse: 預設 False(由小到大), True 會由大到小排

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

Exercise: 計算統計數據

設計一個 function, 傳進數字的 list, 回傳它的全距、標準差、變異數

$$Var(X) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n}$$

$$= \frac{\sum_{i=1}^{n} {X_i}^2}{n} - \overline{X}^2$$

【技法】用 tuple 回傳多個回傳值

```
def getStatistics(data):
    full_range = ???
    average = ???
    variance = ???
    return ???

data = [1, 3, 5, 7, 9]
print(getStatistics(data))
```

Solution: 計算統計數據

設計一個 function, 傳進數字的 list, 回傳它的全距、平均、變異數

```
def getStatistics(data):
    full_range = max(data) - min(data)
    average = sum(data) / len(data)
    variance = sum([(d - average)**2 for d in data]) / len(data)
    return (full_range, average, variance)

data = [1, 3, 5, 7, 9]
print(getStatistics(data))
```

Class

class: 類別 / object: 物件

類別	類別下的物件	屬性

「類別」就是資料型態，定義了該類別物件應有的「屬性」和「行為」

「物件」是類別下的一個實體，可以存放資料，對於其進行操作、運算

自定義 class

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

object1 = className()
```

`class` :宣告一個 class

`self` :表示被建構的物件本身

`className` :class 的名稱

`self.attribute` :物件屬性

`__init__` :建構物件時的函式

`className` :創立新物件

Example: 銀行帳戶

```
class Account:
    def __init__(self, user_name):
        self.name = user_name
        self.balance = 0

account = Account('Joe')
print(type(Account))
# Output: <class '__main__.Account'>

print(account.name, 'has', account.balance, 'dollars')
```

method - class 的專屬函式

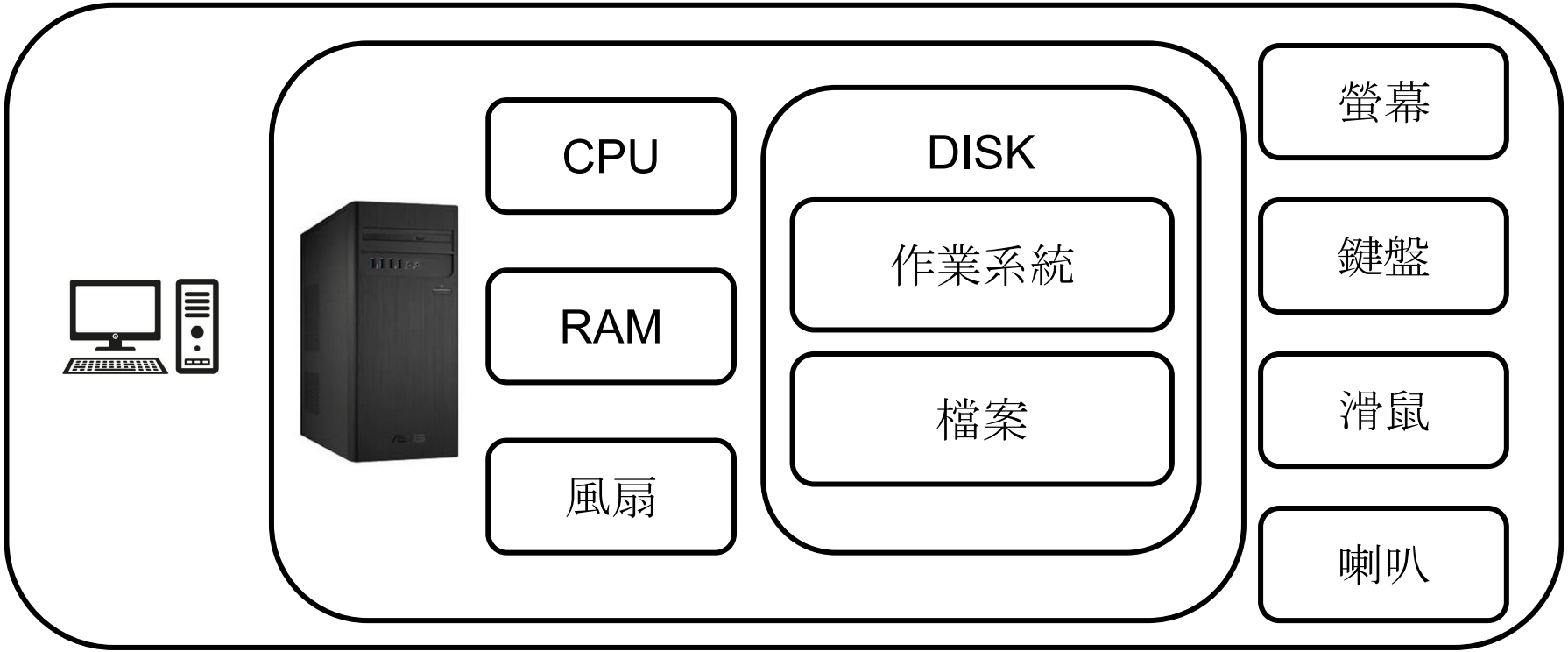
```
class className:
    # __init__
    def methodName(self):
        # do something
    def anotherMethod(self, parameters):
        # do something

object1 = className()
object1.methodName()
```

Example: 銀行帳戶 method

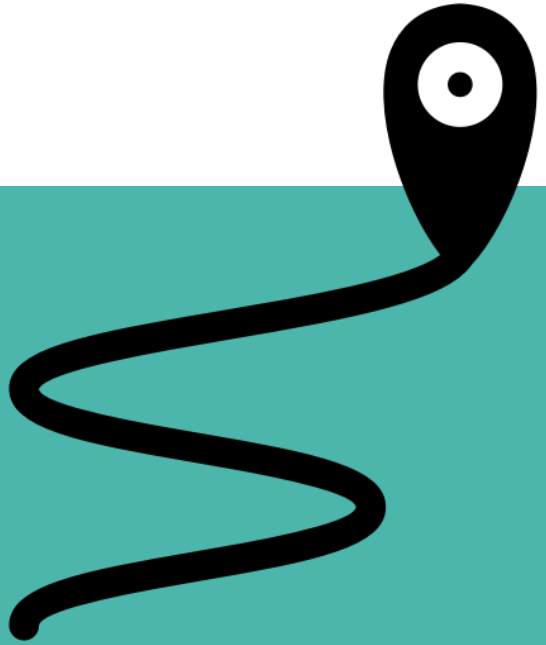
```
class Account:
    # __init__
    def getBalance(self):
        return self.balance

    def deposit(self, money):
        self.balance += money

    def draw(self, money):
        if self.balance > money:
            self.balance -= money
        else:
            print('Insufficient balance!')
```

```
account = Account('Joe')
print(account.getBalance())
account.deposit(500)
print(account.getBalance())
account.draw(1000)
```

class 的俄羅斯娃娃



資訊營之後



講了這麼多...
GET YOUR HANDS DIRTY!

擱來就是恁的代誌矣