Python3 教學

2018.0710

製作：廖家誼

Outline

- Python Function
- Python Classes
- Python Modules
- Python 資料處理、機器學習

Function(1/4)

A function is a block of code which only runs when it is called.

Example

```
def my_function():
  print("Hello from a function")

my_function()
```

Function(2/4)

You can pass data, known as parameters, into a function.

Example

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Function(3/4)

If we call the function without parameter, it uses the default value:

Example

```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Function(4/4)

To let a function return a value, use the `return` statement:

Example

```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Classes(1/5)

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a class named MyClass, with a property named x:

```
class MyClass:
  x = 5
```

Create an object named p1, and print the value of x:

```
p1 = MyClass()
print(p1.x)
```

Classes(2/5)

The __init__() Function

All classes have a function called __init__(), which is always executed when the class is being initiated.

Create a class named Person, use the __init__() function to assign values for name and age:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

Classes(3/5)

Objects can also contain methods. Methods in objects are functions that belongs to the object.

Insert a function that prints a greeting, and execute it on the p1 object:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Classes(4/5)

The `self` parameter is a reference to the class itself, is used to access variables that belongs to the class. It does not have to be named `self` , you can call it whatever you like, but it has to be the first parameter of any function in the class:

```
class Person:
  def __init__(mysillyobject, name, age):
    mysillyobject.name = name
    mysillyobject.age = age

  def myfunc(abc):
    print("Hello my name is " + abc.name)

p1 = Person("John", 36)
p1.myfunc()
```

Classes(5/5)

You can modify properties on objects like this:

Example

Set the age of p1 to 40:

```
p1.age = 40
```

SubClassed(1/4)

```
class Employee:

    staff_num = 0

    raise_amount = 1.8

    def __init__(self, first_name, last_name, payment):

        self.first_name = first_name

        self.last_name = last_name

        self.payment = payment

        self.email = first_name + '_' + last_name + '@example.com'

        Employee.staff_num += 1


def info(self):

    print('{} {}'.format(self.first_name, self.last_name))

    print('email: {}'.format(self.email))

def pay_raise(self):

    self.payment = int(self.payment * Employee.raise_amount) #or self.raise_amount
```

SubClassed(2/4)

```
class Developer(Employee):

    raise_amount = 2 # can't change!

    def __init__(self, first_name, last_name, payment, program_lang):
        super().__init__(first_name, last_name, payment)
        # same
        # Employee.__init__(self, first_name, last_name, payment)
        self.program_lang = program_lang
```

SubClassed(3/4)

```
class Manager(Employee):
    def __init__(self, first_name, last_name, payment, employees = None):
        super().__init__(first_name, last_name, payment)
        if employees == None: self.employees = []
        else: self.employees = employees

    def add_emp(self, employee):
        if employee not in self.employees:
            self.employees.append(employee)

    def remove_emp(self, employee):
        if employee in self.employees:
            self.employees.remove(employee)

    def print_all_employees(self):
        for employee in self.employees:
            print('-->',employee.email)
```

SubClassed(4/4)

```
# create Employee
dev_1 = Developer('brown','jack',22000,'Python')
dev_2 = Developer('hello','kitty',22000,'Java')
# deverlop
print(dev_1.program_lang, dev_2.program_lang)

# manager
manarger_1 = Manager('bob','johnson',32000,[dev_1])

manarger_1.print_all_employees()

manarger_1.add_emp(dev_2)

manarger_1.print_all_employees()

manarger_1.remove_emp(dev_1)

manarger_1.print_all_employees()

# instance

print(isinstance(manarger_1, Manager))
```

Modules(1/4)

What is a Module?

Consider a module to be the same as a code library.
A file containing a set of functions you want to include in your application.

To create a module just save the code you want in a file with the file extension `.py`:

Save this code in a file named `mymodule.py`

```
def greeting(name):
  print("Hello, " + name)
```

Import the module named mymodule, and call the greeting function:

```
import mymodule

mymodule.greeting("Jonathan")
```

Modules(2/4)

Variables in Module

The module can contain functions, as already described, but also variables of all types (arrays, dictionaries, objects etc):

Save this code in the file `mymodule.py`

```
person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

Import the module named mymodule, and access the person1 dictionary:

```
import mymodule

a = mymodule.person1["age"]
print(a)
```

Modules(3/4)

You can name the module file whatever you like by using the `as` keyword, but it must have the file extension `.py`

Create an alias for `mymodule` called `mx`:

```
import mymodule as mx

a = mx.person1["age"]
print(a)
```

Modules(4/4)

You can choose to import only parts from a module, by using the `from` keyword.

The module named `mymodule` has one function and one dictionary:

```
def greeting(name):
  print("Hello, " + name)

person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

Import only the person1 dictionary from the module:

```
from mymodule import person1

print (person1["age"])
```

來源

- https://www.w3schools.com/python/python_modules.asp

- https://www.w3schools.com/python/python_classes.asp

- https://www.w3schools.com/python/python_functions.asp

BMI

輸入身高、體重與名稱，最後顯示

```
input your height and weight and format > 155 40 Apple: 155 40 apple
hello, apple  your bmi:  16.649323621227886
```

BMI公式：weight / (height/100)**2

1. 建立一個模組：

```
class BMITest:
    def __init__(參數):
        def BMI(self):
            return BMI 數值
```

2. 另外一個程式碼import它：

```
import bmi
input_height_weight = input()
```

Process Data w/ Python(1/5)

- **Pandas**
- **Numpy**
- **Matplotlib**
- **Scipy**

Process Data w/ Python(2/5)

Pandas 是什麼？

Pandas是一個基於numpy的package（不是什麼很多隻熊貓啦！），在處理數據方面非常的好用，透過標籤和索引，Pandas讓我們可以非常輕易的處理數據。

Process Data w/ Python(3/5)

Numpy 是什麼？

NumPy‘s array type augments the Python language with an efficient data structure useful for numerical work, e.g., manipulating matrices. NumPy also provides basic numerical routines, such as tools for finding eigenvectors.

Numpy是一個提供矩陣運算非常非常非常好用的工具，雖然python處理大量資料時有list可以讓我們做到似矩陣的功能但List 效能表現並不理想，而Numpy具備平行處理的能力，可將操作動作一次套用在大型陣列上，幫助我們做更多方法建立多維數據以及矩陣運算，像是Pandas就是建立在Numpy的基礎延伸的套件。

Process Data w/ Python(4/5)

Matplotlib 是什麼？

The matplotlib module produces high quality plots. With it you can turn your data or your models into figures for presentations or articles. No need to do the numerical work in one program, save the data, and plot it with another program.

假設你做了好了資料分析要怎麼讓大家可以一目瞭然？這時候就需要用到它了！（登冷！）
Matplotlib是Python繪圖的它包含了大量的工具，你可以使用這些工具創建各種圖形，包括簡單的散點圖、直方圖，甚至是三維圖形，將你的資料轉成圖表，在數據分析中會經常使用Matplotlib完成數據可視化的工作！

Process Data w/ Python(5/5)

SciPy是什麼？

SciPy contains additional routines needed in scientific work: for example, routines for computing integrals numerically, solving differential equations, optimization, and sparse matrices.

我們說pandas用來整理數據表格，Numpy是以矩陣基礎做數據的數學運算，SciPy就是以Numpy為基礎做科學、工程的運算處理的package，包含統計、優化、整合、線性代數、傅立葉轉換圖像等較高階的科學運算。

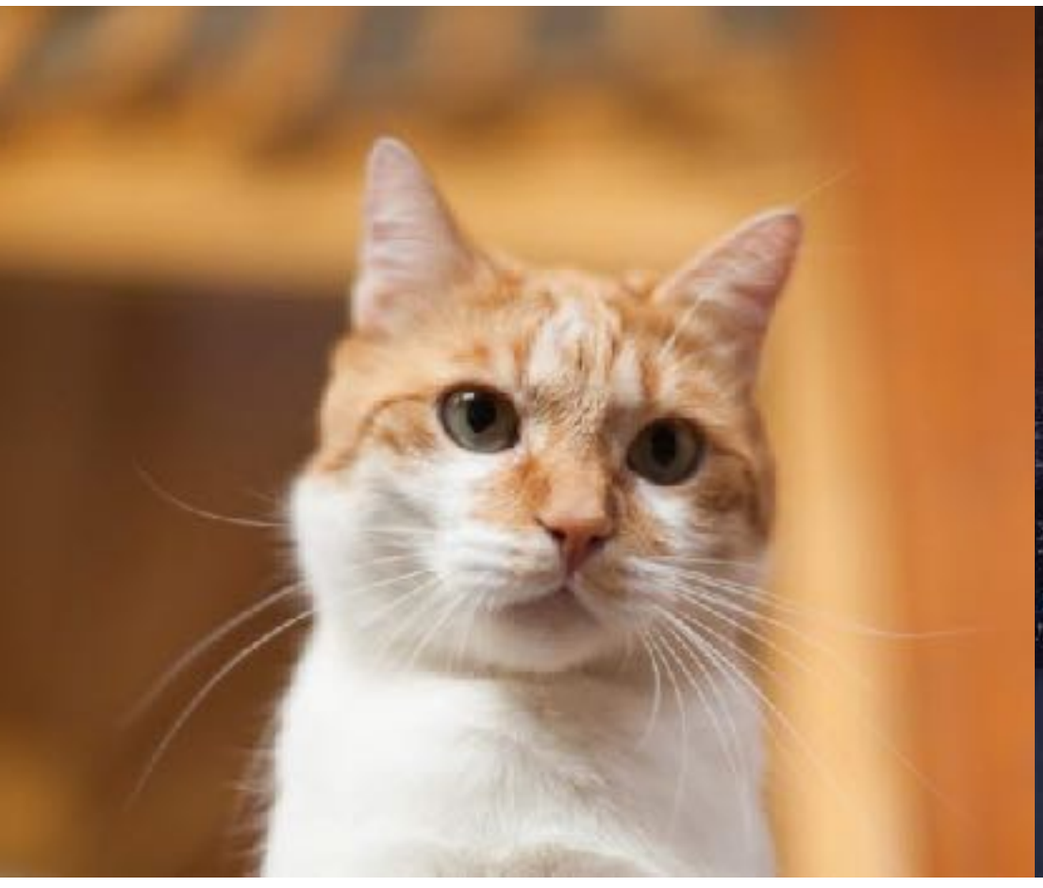
Machine Learning(1/5)

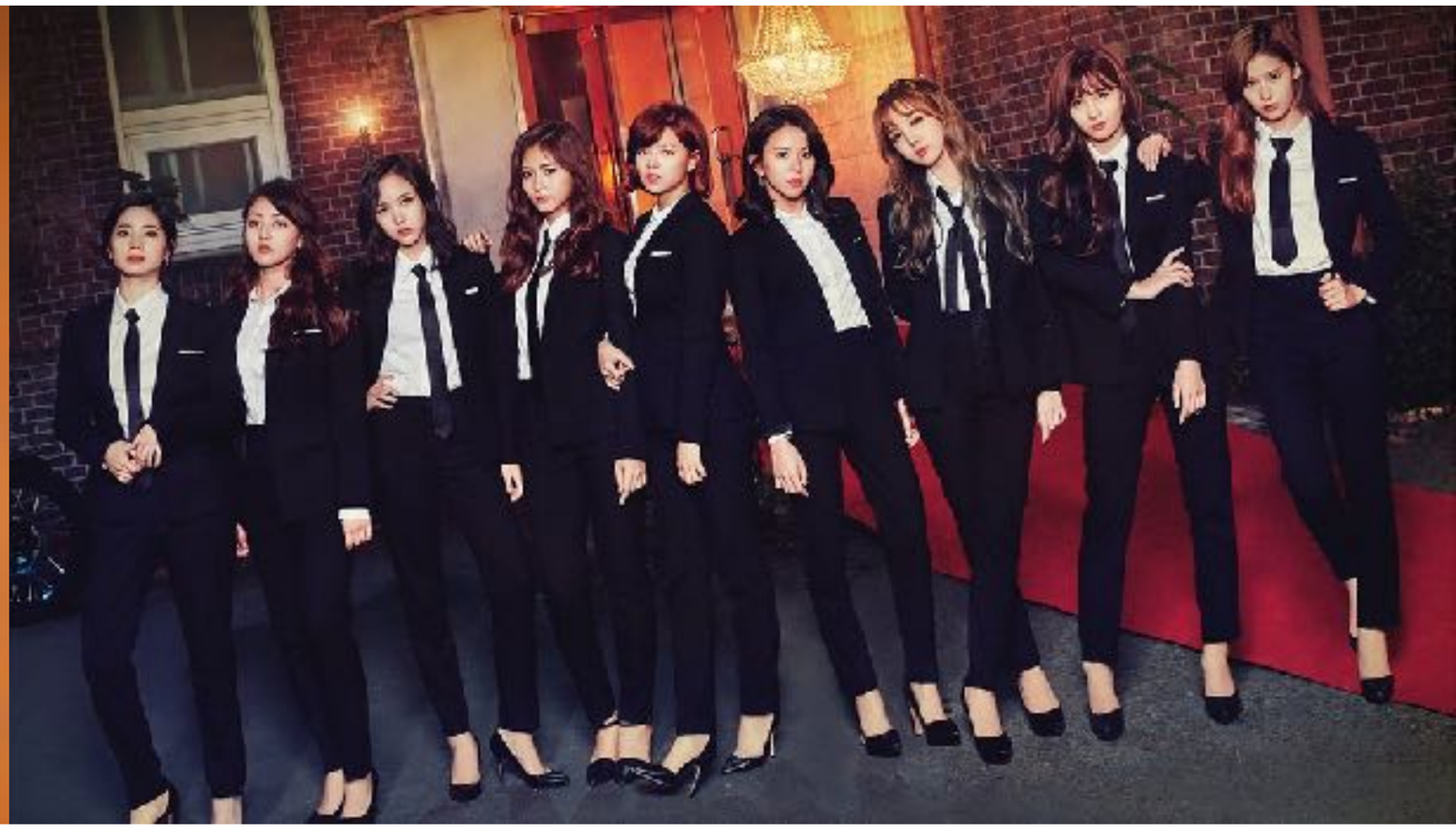
我們常常說機器學習機器學習，到底是要學什麼？

就像網路上有大量的資訊，我們人透過這些大量的資訊去學習分辨什麼是好的什麼是壞的，而為了讓電腦也跟人的腦袋一樣去學習這些資訊、分辨資訊，這樣的演算法就是機器學習，也就是實現人工智慧的一個途徑。

若我們能夠很簡單的定義一個東西是什麼，那用判斷式就好了啊！

Machine Learning(2/5)





Label: Cat

Label: What????

模仿人類學習的方式透過觀察 (data) 學習 (取出特徵、經過計算處理)
得到有意義的技巧 (提升某項可以量化評估的表現)。

Machine Learning(3/5)

重量	表面	標籤
150g	皺	橘子
170g	皺	橘子
130g	平滑	蘋果
140g	平滑	蘋果
...	...	...

利用重量跟表面來當訓練的資料，而重量跟表面就是我們所謂的特徵，而標籤就是定義這些特徵的結果是哪一種水果，也就是透過機器學習想讓電腦告訴我們的答案。

重量	表面	標籤
150g	1	0
170g	1	0
130g	0	1
140g	0	1

我們將皺用1表示、平滑用0表示。橘子用0、蘋果用1，用這種資料格式餵給電腦可以讓電腦容易理解。

Machine Learning(4/5)

```
2
3   from sklearn import tree
4   features = [[150,1],[170,1],[130,0],[140,0]]
5   labels = [0,0,1,1]
6   clf = tree.DecisionTreeClassifier()
7   clf = clf.fit(features,labels)
8   wantPredict = clf.predict([[120,0]])
9   if wantPredict == [1]:
10      print('This is an apple')
11  elif wantPredict == [0]:
12      print('This is an orange')
```

工商服務時間(？)

plusone (plusone)

IT邦新手 5 級・點數 91

8681
累計瀏覽數

18人
在追蹤

設定

個人背景	發問	文章	回答	邀請回答	最佳解答
	0	34	0	0	0

鐵人檔案

- 2018 iT 邦幫忙鐵人賽

Data Technology

使用Python進行資料分析 系列

鐵人練成 | 共 30 篇文章 | 32 人訂閱 | 團隊 i.m.a.c_wishing_well

plusone的鐵人檔案

☆ plusone的收藏

♡ plusone的追蹤

plusone的Like

https://ithelp.ithome.com.tw/users/20107514