Python 快速上手 part2

William

資料與程式碼: 程式碼與練習題解答
影片播放列表: 影片播放列表
投影片 PDF: 投影片PDF下載連結

「版權聲明頁」

本投影片已經獲得作者授權台灣人工智慧學校得以使用於教學用途，如需取得重製權以及公開傳輸權需要透過台灣人工智慧學校取得著作人同意；如果需要修改本投影片著作，則需要取得改作權；另外，如果有需要以光碟或紙本等實體的方式傳播，則需要取得人工智慧學校散佈權。

– 台灣人工智慧學校

下載課程資料

- 為維護課程資料，courses 中的檔案皆為 read-only，如需修改請 cp 至自身的環境中
- 打開 terminal，輸入

```
cp -r courses-tpe/python_programming mypython
```

- 今後的課程，如果需要下載課程資料都會使用這樣的方式

Function

Repetition

```
# repetition

# print("Hello Adam, nice to meet you")
# print("Hello Bruce, nice to meet you")
# print("Hello Cate, nice to meet you")

greet("Adam")
greet("Bruce")
greet("Cate")
```

Syntax

```
# Function is a group of related statements that perform a specific task.

def function_name(parameters):
    statement(s)
```

def - marks the start of function header.
function name - to uniquely identify it.
parameters - through which we pass values to a function. (optional)
colon (:) - to mark the end of function header.
return statement - to return a value from the function. (optional)

Define, Call Function

```
# define function without parameters
def greet():
    print("Hello!")

 # call function
 greet()            # Hello!


# define function with parameter
def greet(name):
    print("Hello", name + ", nice to meet you.")

greet("Felix")      # Hello Felix, nice to meet you.
```

Repetition

```
# repetition

# print("Hello Adam, nice to meet you")
# print("Hello Bruce, nice to meet you")
# print("Hello Cate, nice to meet you")

greet("Adam")
greet("Bruce")
greet("Cate")
```

Return Statement

```python
# None
def greet():
    print("Hello")

# One
def add_two_nums(arg1, arg2):
   sum = arg1 + arg2
   return sum;

# call function
result = add_two_nums(10, 20)
print(result)      # 30
```

Multiple return values

```
# constructs a tuple and returns this to the caller
def square(x,y):
    return x*x, y*y

result = square(2,3)
print(result)   # (4,9)


# "unwrap" the tuple into the variables directly by specifying the same number of variables
def square(x,y):
    return x*x, y*y

res_x, res_y = square(2,3)
print(res_x)    # 4
print(res_y)    # 9
```

Anonymous Function - Lambda

```
# Lambda functions can have only one expression.
# The expression is evaluated and returned.

double = lambda x: x * 2

print(double(5))     # 10


# is nearly the same as
def double(x):
    return x * 2
```

Anonymous Function - Lambda

```
# Lambda functions can have any number of arguments
double = lambda x, y: x * 2 + y

print(double(5,2))       # 12


# is nearly the same as
def double(x, y):
    return x * 2 + y
```

Global, Local variables

```python
# global
x = "global"

def foo():
    y = x + "_variable"
    print(y)

foo()    # global_variable
```

```python
# local
def foo():
    z = "local"

# NameError: name 'z' is not defined
print(z)
```

練習 - part 4

Q1. 請寫出一個函式，將列表中的數字相乘。
 Sample List : [1, 2, 3, 4, 5]
 Expected Result : 120

Q2. 請寫□個函式，輸入一字串，返回反轉全部字元的字串。
 a_func("test")
 Expected Result : "tset"

Q3. 請寫□個函式把裡□的字串，每個單字本□做反轉，但是單字的順序不變。 (Optional)
 a_func("it is a test string")
 Expected Result : "ti si a tset gnirts"

Generators

Generator with for loop

```
# with for loop
def generator_example():
    a = 1
    yield print(a)    # 1
    a += 1
    yield print(a)    # 2
    return

for i in generator_example():
    continue                    # Output:
                                  1
                                  2
```

Generator with for loop

```
# with for loop
def generator_example():
    a = 1
    yield print(a)    # 1
    a += 1
    yield print(a)    # 2
    return

for i in generator_example():
    continue
```

```
# Output:
  1
  2
```

Generator with next, avoid StopIteration Error

```python
# with next
def generator_example():
    yield print(1)
    yield print(2)
    return
```

```python
# avoid StopIteration Error
try:
    gen.__next__()
except StopIteration:
    pass # do nothing
```

```python
gen = generator_example()
gen.__next__()      # 1
gen.__next__()      # 2
gen.__next__()      # raise StopIteration Error
```

```
---------------------------------------------------------------------------
--
StopIteration                             Traceback (most recent call las
t)
<ipython-input-56-e21f692c4865> in <module>()
      8 gen.__next__() # 1
      9 gen.__next__() # 2
---> 10 gen.__next__() # raise StopIteration Error

StopIteration:
```

Benefits - Memory Usage

```python
# 利用 list 迭代
range_num = 10
for i in [x*x for x in range(range_num)]:
    # do something
    pass

# 利用 generator 迭代
for i in (x*x for x in range(range_num)):
    # do something
    pass
```

Memory Usage - by using list

```
import psutil
before_used = psutil.virtual_memory().used  # expressed in bytes
after_used = 0

print("before:", before_used)                                   ## 10372907008

range_num = 1000000
for i in [x*x for x in range(range_num)]: # 第一種方法:對 list 進行迭代
    if i == (range_num - 1) * (range_num - 1):
        after_used = psutil.virtual_memory().used
        print("after:", after_used)                             ## 10405208064

print("used memory:", (after_used - before_used))               ## 32301056
```

Memory Usage - by using generator

```
import psutil
before_used = psutil.virtual_memory().used  # expressed in bytes
after_used = 0

print("before:", before_used)                                    ## 10458206208

range_num = 1000000
for i in (x*x for x in range(range_num)): # 第二種方法:對 generator 進行迭代
    if i == (range_num - 1) * (range_num - 1):
        after_used = psutil.virtual_memory().used
        print("after:", after_used)                              ## 10461298688

print("used memory:", (after_used - before_used))                ## 3092480
```

Module

Modules

```
# A module is a file containing Python definitions and
statements.

import re

import re as r

from re import findall

from re import *
```

Module - os

```python
import os

# 顯示絕對路徑
os.path.abspath("session_1-ans.ipynb")          # '/Users/felix/Python/session_1-ans.ipynb'

# 將多個字串組合為路徑
'/'.join(['path', 'result', 'a.csv'])           # 'path/result/a.csv'

# 將多個字串組合為路徑
os.path.join('path', 'result', 'a.csv')         # 'path/result/a.csv'

# 檢查某路徑/資料夾是否存在
os.path.exists("python\session_1-ans.ipynb")    # False
```

練習 - part 5

Q1: 若某 k 位數的正整數，其所有位數數字的 k 次方和等於該數相等，則稱為阿姆斯壯數 (Armstrong number)。 例如 1^3 + 5^3 + 3^3 = 153，則 153 是一個阿姆斯壯數。

請創建一個 Generator 函式，找出 100 ~ 999 的所有三位數的阿姆斯壯數；
利用 yield 回傳數值，並且用多次呼叫的方式，依序列印出所找到的阿姆斯壯數。

Q2: 透過 Generators 讀取一個純文字檔案中的所有文字。(Optional)

hint 1. 利用 open("your_file_path", "r") 來開啟檔案
hint 2. 需設定每次要讀取檔案的大小
hint 3. 利用迴圈存取，直到檔案讀取完畢為止

Regular Expression

Module - re

```
import re

string = "This is demo string, do nothong!"
pattern = "is"

# Return a list of all non-overlapping matches in the string.
print(re.findall(pattern, string))      # ['is', 'is']
```

Regular Expression - Simple example

```
"This is demo string, do nothing!"

# pattern_1
"is"

# pattern_2
"abc"

# find - does the string contains the pattern?
# YES or NO
```

Regular Expression - more example

```
"This is demo string, 01234567899876543210."

# pattern
"01234567899876543210"

# if you want to search more complex pattern?
# using regular expression!
syntax = "[0-9]{20}"
```

Special Characters

`.` match any character except a newline

`*` match 0 or more repetitions of the preceding character

`+` match 1 or more repetitions of the preceding character

`{m}` match exactly m copies of the previous character

`{m,n}` match from m to n repetitions of the preceding character

`\` escapes special characters

`[ ]` Used to indicate a set of characters

[amk] will match 'a', 'm', or 'k'

[a-z] will match any lowercase ASCII letter

[0-5][0-9] will match all the two-digits numbers from 00 to 59

Module - re

```
import re

string = "This is demo string, do nothing!"
pattern = "is"

# Return a list of all non-overlapping matches in the string.
print(re.findall(pattern, string))      # ['is', 'is']
```

find numbers, letters

```
import re
# find numbers
pattern = "[0-9]+"
string = '12 drummers drumming, 111 pipers piping, 1006 lords
a-leaping'
re.findall(pattern, string)      # ['12', '111', '1006']

# find letters
pattern = "[cmf]an"
string = 'find: can, man, fan, skip: dan, ran, pan'
re.findall(pattern, string)      # ['can', 'man', 'fan']
```

find e-mail

```
import re
email_text = """
Big Data Analytics/ Deep LearningSocial Computing / Computational Social Science / Crowdsourcing
Multimediaand Network SystemsQuality of ExperienceInformation SecurityPh.D. candidate at NTU
EEchihfan02-27883799#1602Camera CalibrationComputer VisionData
Analysiscmchang02-27883799#1671System OptimizationMachine LearningyusraBig data
analysiscclin02-27883799#1668Data Analysisrusi02-27883799#1668Government Procurement ActFinancial
Managementkatekuen02-27883799#1602AdministrationEvent Planningseanyu02-27883799#1668Data
AnalysisPsychology & NeuroscienceMarketingxinchinchenEmbedded Systemkyoyachuan062602-27883799
#1601FinTechActuarial ScienceData Analysiskai0604602-27883799#1601Data AnalysisMachine
Learningchloe02-27839427Accountingafun02-27883799 felix2018@iis.sinica.edu.tw
#1673Data AnalysisWeb developmentyunhsu198902-27883799#1668MarketingTIGP Ph.D. Fellow at Academia Sinica &
NCCUbaowalyMachine LearningData AnalysisSocial Computingchangyc1427883799#1678
Data Analysisjimmy1592302-2788379 jimmy15923@iis.sinica.com.tw#1688Data AnalysisjasontangAnalysisMachine
Learninguchen02-27883799#1668Deep Learningpjwu02-27883799#1604Computational PhotographyData Analysis
"""

re.findall("([A-Za-z0-9._]+@[A-Za-z.]+[com|edu]\.tw)", email_text)
        # Output: ['felix2018@iis.sinica.edu.tw', 'jimmy15923@iis.sinica.com.tw']
```

練習 - part 6

請匹配出下列問題的 Regular Expression

Q1. 同時匹配 abcdefg, abcde, abc
Q2. 同時匹配 abc123xyz, abcde22a, abc456aaa
Q3. 匹配 "catcat" (包含 ")
Q4. 同時匹配 wazzzzzup, wazzzup
Q5. 同時匹配 aaaabcc, aabbbbc, aacc
Q6. 匹配手機號碼, 格式為:0987-654-321
Q7. 匹配右方格式, xxx.xxx.xxx.xxx (其中 x 是 0~9 的數字)

想要更多練習, 請到 RegexOne 網站右上方的 Interactive Tutorial。

Class

init, self

```python
# no arguments
class MyClass:
    def __init__(self):
        print("do nothing")

my_object = MyClass()
# do nothing
```

```python
# with arguments
class MyClass:
    def __init__(self, var1, var2):
        self.var1 = var1
        self.var2 = var2

my_object = MyClass(123, 456)
print(my_object.var1)    # 123
print(my_object.var2)    # 456
```

Class

```
# Attribute references
class MyClass:
    var = 123
    def method(self):
        return "hello world"

# Instantiation
my_object = MyClass()

# 用 . 來訪問物件的屬性或方法
print(my_object.var)          # 123
print(my_object.method())     # hello world
```

init, self

```python
# no arguments
class MyClass:
    def __init__(self):
        print("do nothing")

my_object = MyClass()
# do nothing
```

```python
# with arguments
class MyClass:
    def __init__(self, var1, var2):
        self.var1 = var1
        self.var2 = var2

my_object = MyClass(123, 456)
print(my_object.var1)    # 123
print(my_object.var2)    # 456
```

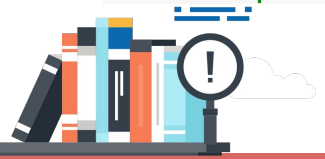
Object

```
class MyClass:
    def __init__(self, var1):
        self.var1 = var1

my_object_123 = MyClass(123)
my_object_987 = MyClass(987)

print(my_object_123.var1)        #123
print(my_object_987.var1)        #987

print(my_object_123)             #<__main__.MyClass object at 0x1070e6128>
print(my_object_987)             #<__main__.MyClass object at 0x1070e60f0>
```

Example

```
class Person:
    bmi = 0.0
    height = 0.0
    weight = 0

    def __init__(self):
        pass

    def ask_person_info(self):
        self.height = float(input("What is your height? (meter) : "))
        self.weight = int(input("What is your weight? (kg) : "))

    def cal_BMI(self):
        self.bmi = round((self.weight / (self.height ** 2)), 2)
        print("Your BMI is " + str(self.bmi))
```

```
# main
person = Person()
person.ask_person_info()
person.cal_BMI()

# What is your height? (meter) : 1.8
# What is your weight? (kg) : 70
# Your BMI is 21.6
```



練習 - part 7

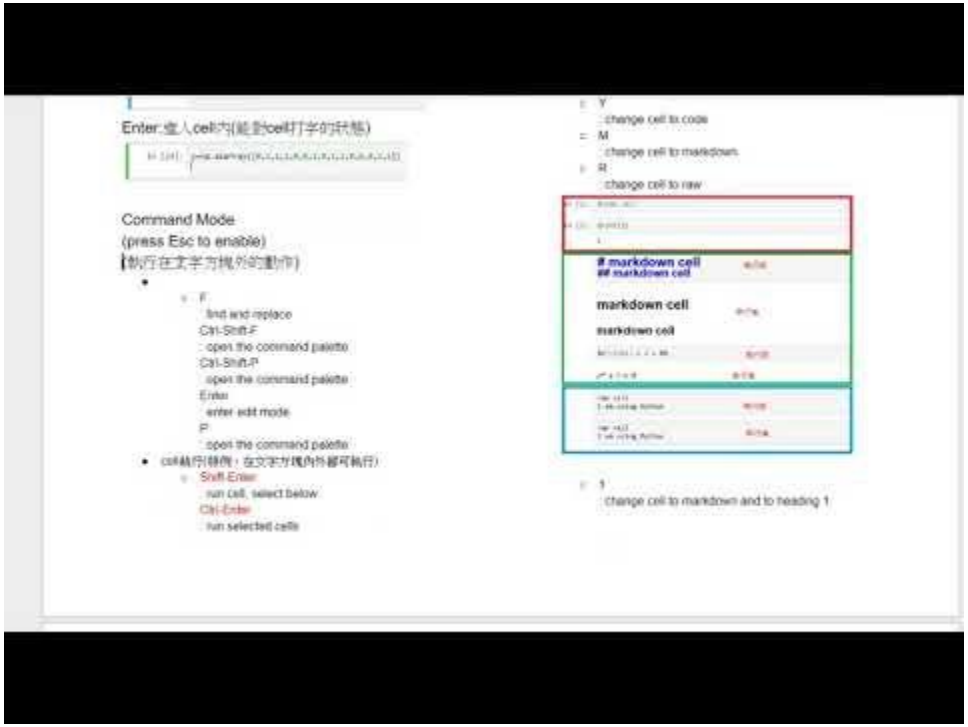
Q1. 寫一個 Class,
包含一個變數(str1)以及兩個函式(set_string 和 print_string).
set_string 接受一個字串參數, 賦值給 str1。
print_string 印出 str1 的大寫字串

hint: 先宣告一個成員變數, 再透過上述兩個函式對該變數做操作。

Command Mode

```
In [24]: y=np.asarray([0,1,1,1,0,0,1,0,1,1,0,0,0,1,1])
```

Command Mode 快捷鍵



Esc and Enter

Esc：跳出cell外(無法對cell打字的狀態)(進入Command Mode)

```
In [24]: y=np.asarray([0,1,1,1,0,0,1,0,1,1,0,0,0,1,1])
```

Enter:進入cell內(能對cell打字的狀態)(進入Edit Mode)

```
In [24]: y=np.asarray([0,1,1,1,0,0,1,0,1,1,0,0,0,1,1])
```

Command Mode

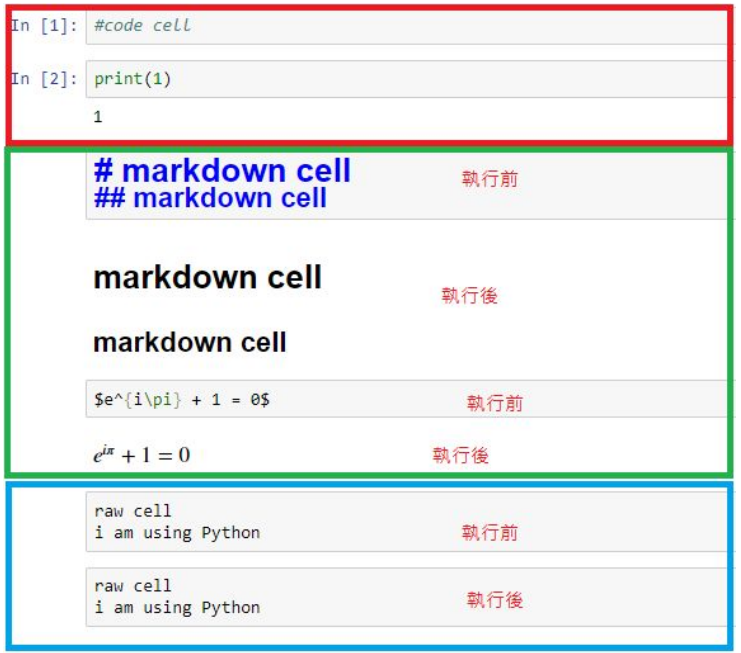
- cell執行(特例，在文字方塊內外都可執行)
  - Shift-Enter
    : run cell, select below
    Ctrl-Enter
    : run selected cells
    Alt-Enter
    : run cell and insert below

Command Mode

- cell的新增、移除、合併
  - A
    : insert cell above
    B
    : insert cell below
    X
    : cut selected cells
    C
    : copy selected cells
  - Shift-V
    : paste cells above
    V
    : paste cells below
    Z
    : undo cell deletion
    D,D
    : delete selected cells
    Shift-M
    : merge selected cells, or current cell with cell below if only one cell is selected

Command Mode

- 改變cell功能(code、文字、或標註)
  - Y
    : change cell to code
  - M
    : change cell to markdown
  - R
    : change cell to raw



Command Mode

- 其他
  - Ctrl-S
    : Save and Checkpoint
    S
    : Save and Checkpoint
    L
    : toggle line numbers
    O
    : toggle output of selected cells
    Shift-O
    : toggle output scrolling of selected cells
    H
    : show keyboard shortcuts
  - I,I
    : interrupt the kernel
  - 0,0
    : restart the kernel (with dialog)
  - Esc
    : close the pager
  - Q
    : close the pager
  - Shift-L
    : toggles line numbers in all cells, and persist the setting
  - Shift-Space
    : scroll notebook up
  - Space
    : scroll notebook down

Command Mode

- 其他
  - K
    : select cell above
    Up
    : select cell above
    Down
    : select cell below
    J
    : select cell below
  - Shift-K
    : extend selected cells above
    Shift-Up
    : extend selected cells above
    Shift-Down
    : extend selected cells below
  - Shift-J
    : extend selected cells below
  - F
    : find and replace
    Ctrl-Shift-F
    : open the command palette
    Ctrl-Shift-P
    : open the command palette
    Enter
    : enter edit mode
    P
    : open the command palette

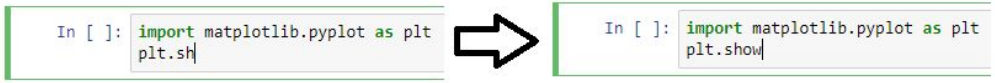
Edit Mode

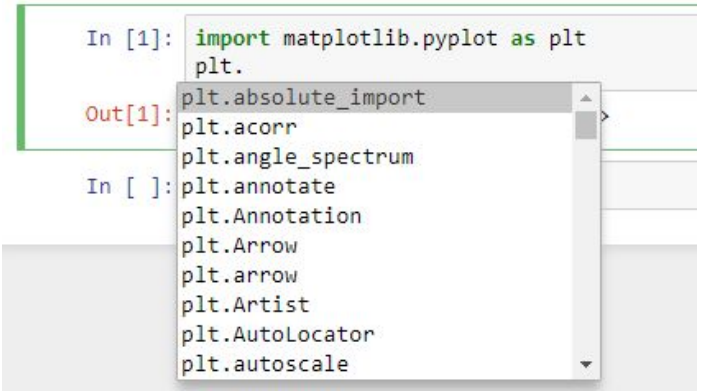
```
In [24]: y=np.asarray([0,1,1,1,0,0,1,0,1,1,0,0,0,1,1])
```

Edit Mode

Edit Mode

- 自動輸出補上(或提供輸入選項)
  - Tab
    : code completion or indent





Edit Mode

- function提示與說明文件
  - Shift-Tab
    : tooltip
  - Shift-Tab*2
    : tooltip+parameters

```
In [1]: import matplotlib.pyplot as plt
        plt.show

Out[1]: Signature: plt.show(*args, **kw)
        Docstring:
        Display a figure.
        When running in ipython with its pylab mode, display all

In [ ]:
```

Edit Mode

- 剪下、複製、貼上
  - Ctrl-X
  - Ctrl-C
  - Ctrl-V

Edit Mode

- 還原與取消還原
  - Ctrl-Z
  - : undo
  - Ctrl-U
  - : undo selection(和Ctrl-Z類似，不同的地方在於將 "選取"也算成一次動作)
  - Ctrl-Y/ Ctrl-Shift-Z
  - : redo
  - Alt-U
  - : redo selection(和Ctrl-Y/ Ctrl-Shift-Z類似，不同的地方在於將 "選取"也算成一次動作)

```
w3_BN = tf.Variable(w3_initial)
b3_BN = tf.Variable(tf.zeros([10]))
y_BN  = tf.nn.softmax(tf.matmul(l2_BN,w3_BN)+b3_BN)
# Loss, optimizer and predictions
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
cross_entropy_BN = -tf.reduce_sum(y_*tf.log(y_BN))

correct_prediction = tf.equal(tf.arg_max(y,1),tf.arg_max(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))
correct_prediction_BN = tf.equal(tf.arg_max(y_BN,1),tf.arg_max(y_,1))
accuracy_BN = tf.reduce_mean(tf.cast(correct_prediction_BN,tf.float32))
# Training the network
zs, BNs, acc, acc_BN = [], [], [], []

sess = tf.InteractiveSession()
sess.run(tf.global_variables_initializer())
for i in tqdm.tqdm(range(40000)):
```

Edit Mode

- 游標動作
  - Ctrl-Home
    : go to cell start
  - Ctrl-Up(not work when i tried)
    : go to cell start
  - Ctrl-End
    : go to cell end
  - Ctrl-Down(not work when i tried)
    : go to cell end
  - Ctrl-Left
    : go one word left(not a Character)
  - Ctrl-Right
    : go one word right(not a Character)
  - Down
    : move cursor down
  - Up
    : move cursor up
- 其他
  - Ctrl-]
    : indent
  - Ctrl-[
    : dedent
  - Ctrl-A
    : select all(cell內全選)
  - Ctrl-/
    : comment
  - Ctrl-D
    : delete whole line
  - Insert
    : toggle overwrite flag
  - Ctrl-Backspace
    : delete word before
  - Ctrl-Delete
    : delete word after
  - Ctrl-M
    : enter command mode
    Ctrl-Shift-F
    : open the command palette
  - Ctrl-Shift-P
    : open the command palette
  - Esc
    : enter command mode
  - Ctrl-Shift-Minus
    : split cell at cursor
  - Ctrl-S
    : Save and Checkpoint

Jupyter notebook 魔術指令

Jupyter notebook 魔術指令

- Jupyter 中有許多特殊指令，都是以 % 開頭
  %cd: 改變路徑
  %save: 將 cell 儲存為 .py
  %run xxx.py: 執行 xxx.py 檔
  %timeit: 計算該 cell 執行之時間
  %matplotlib inline: 將繪製的圖直接顯示在 notebook 上
  %matplotlib notebook

Jupyter notebook 魔術指令

- 以！開頭，可以直接輸入 terminal 的指令
  !nvidia-smi
  !ls
  !rm
  !pip install ...

想知道 function 的說明文件?

- method1:Shift-Tab*2
- method2:run ?+function

```
In [4]:  1 ??np.sqrt
```

```
Call signature:  np.sqrt(*args, **kwargs)
Type:            ufunc
String form:     <ufunc 'sqrt'>
File:            c:\users\jimmy\anaconda3\lib\site-packages\numpy\__init__.py
Class docstring:
Functions that operate element by element on whole arrays.

To see the documentation for a specific ufunc, use np.info().  For
example, np.info(np.sin).  Because ufuncs are written in C
(for speed) and linked into Python with NumPy's ufunc facility,
Python's help() function finds this page whenever help() is called
on a ufunc.

A detailed explanation of ufuncs can be found in the "ufuncs.rst"
file in the NumPy reference guide.
```