台灣人工智慧學校

Python 快速上手

William

「版權聲明頁」

各時段預計完成內容

時段	Section
Section1	環境建置與使用 & 基礎語法
Section2	資料結構 & 流程控制
Section3	函式 & 生成器
Section4	正規表示式&類別

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

why python

Why Python?

- 深度學習的框架幾乎都支援 Python

Caffe

Caffe2

TensorFlow

theano

Microsoft CNTK

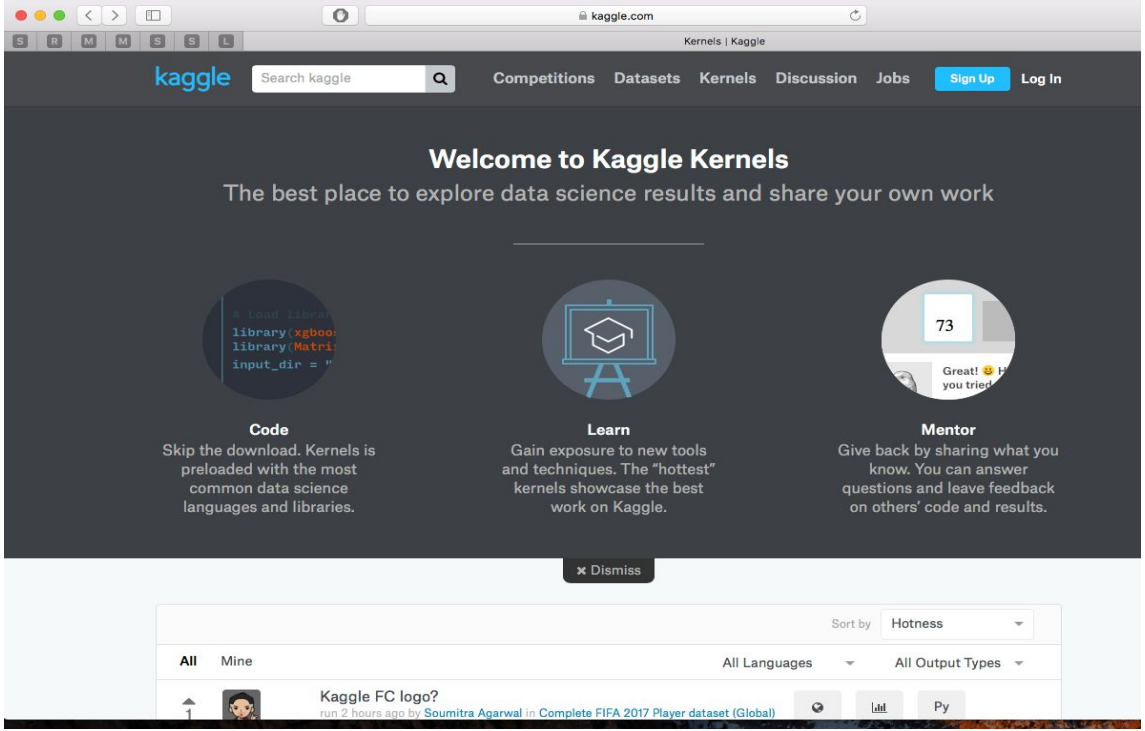
dmlc mxnet

K

PYTORCH

Why Python?

- 資料科學中的主流語言



Why Python?

- 深度學習的框架幾乎都支援 Python

Caffe

Caffe2

TensorFlow

theano

Microsoft CNTK

dmlc mxnet

K

PYTORCH

Why Python?

- 簡單好學!
- Hello world in Python
- print("Hello world")

本機端環境建置

Method one : Python Shell

- 在anaconda pro
- running by line
- exit : Ctrl+Z or e

python3

```
afun@afun
Python 3.5.2 (default
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World')
Hello World
>>>
```

Before installing ...

- Highly recommend learning Python 3.x
  - Different syntax
  - Different implementation
  - No more support for Python 2.7

Anaconda

- 除了 Python，許多資料分析常用的套件也都包含在內
- Windows / Linux / Mac OS
- Download
- Anaconda Prompt
- conda install

To use Python in Anaconda, there are three methods ...

- Python Shell
- Ipython
- jupyter notebook

Method one ：Python Shell

- 在anaconda prompt 輸入python
- running by line

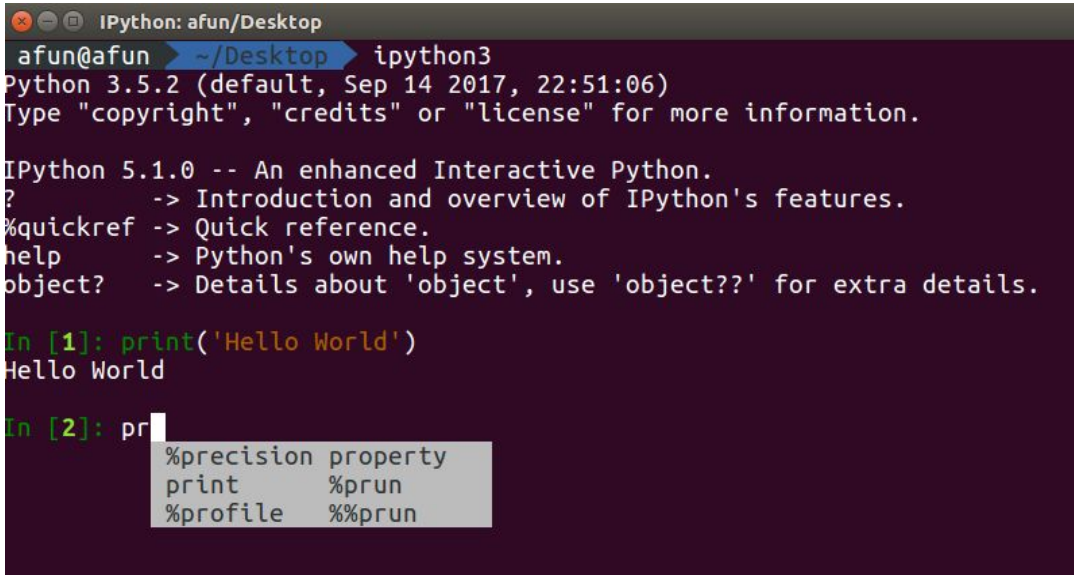
- exit：Ctrl+Z or exit()

```
python3
afun@afun  ~/Desktop  python3
Python 3.5.2 (default, Sep 14 2017, 22:51:06)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World')
Hello World
>>>
```

Method two：ipython

- $ pip install ipython
- 在anaconda prompt 輸入ipython
- include magic code
- running by line
- TAB for hint

- exit：exit



jupyter notebook

Introduction to Jupyter notebook

- Code is divided into cells to control execution
- Ideal for exploratory analysis and model building

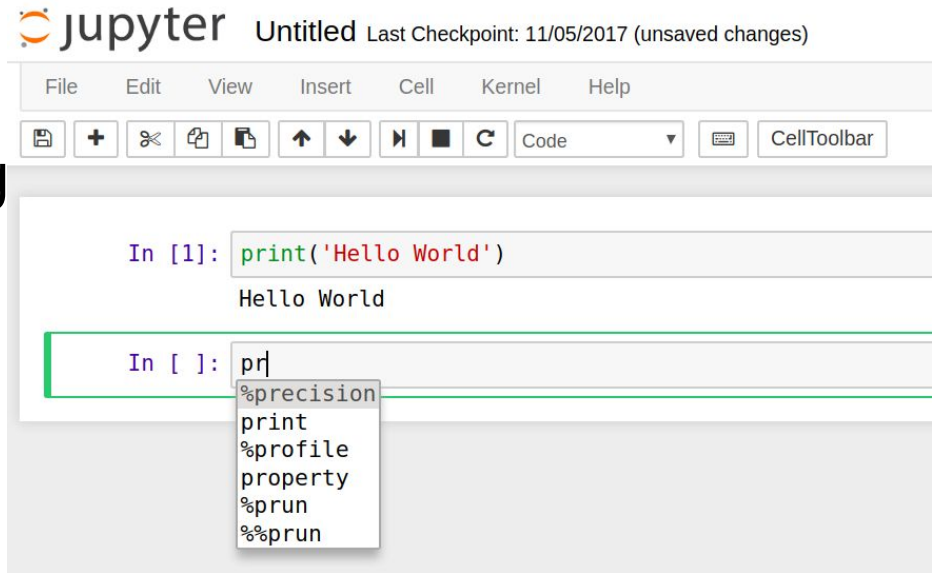
但對於許多人來說...

- 一看到命令字元介面...

沒有按鈕...沒有游標...滑鼠不能用...

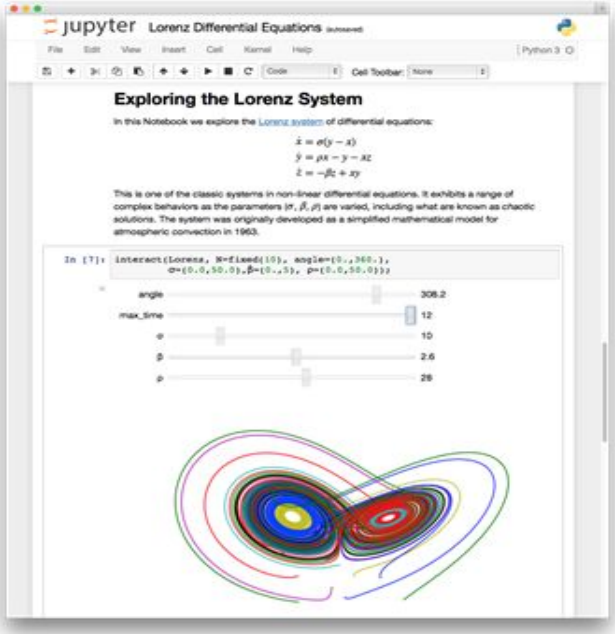
Method three：jupyter notebook

- $ pip install jupyter
- include magic code
- running by cell
- TAB for autocomplete
- SHIFT+TAB for docstring



Introduction to Jupyter notebook

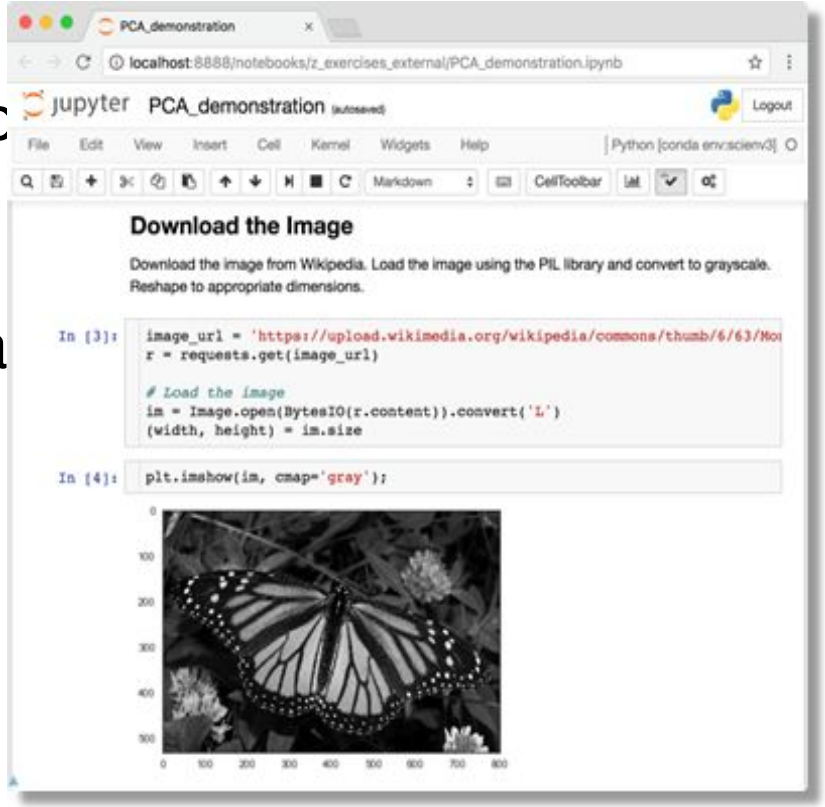
- Jupyter is an anagram of: Julia, Python, and R
- Supports multiple content types: code, narrative text, images, movies, etc



Introduction to Jupyter notebook

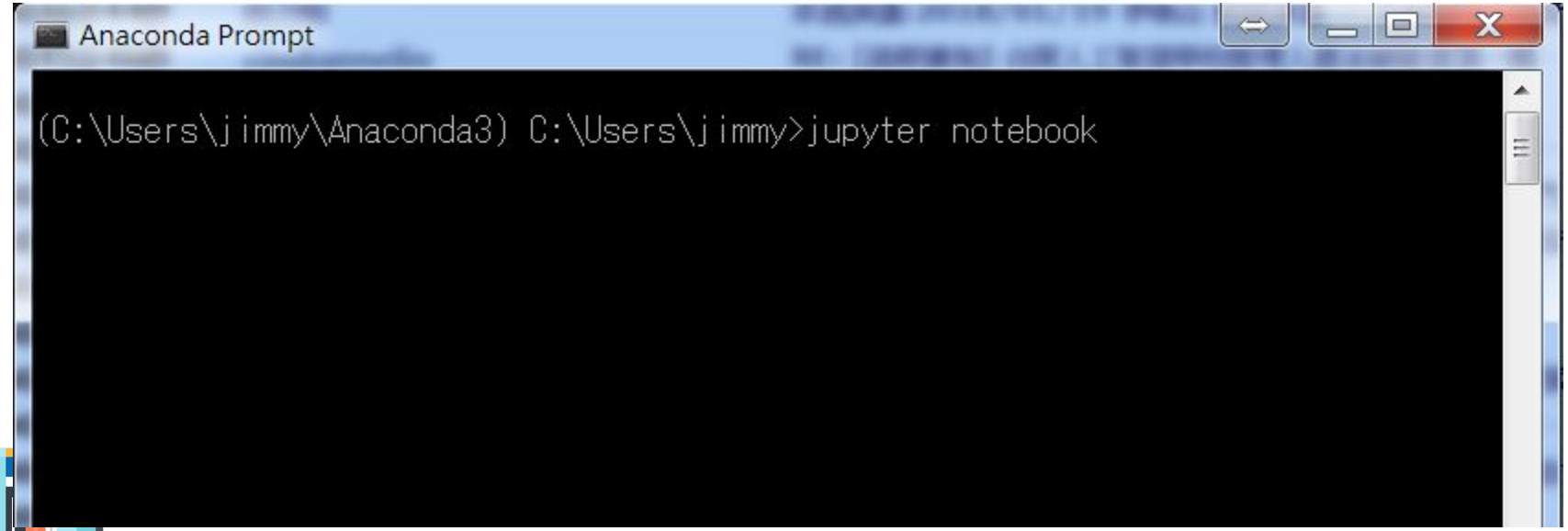
- Code is divided into cells to cc

- Ideal for exploratory analysis a building



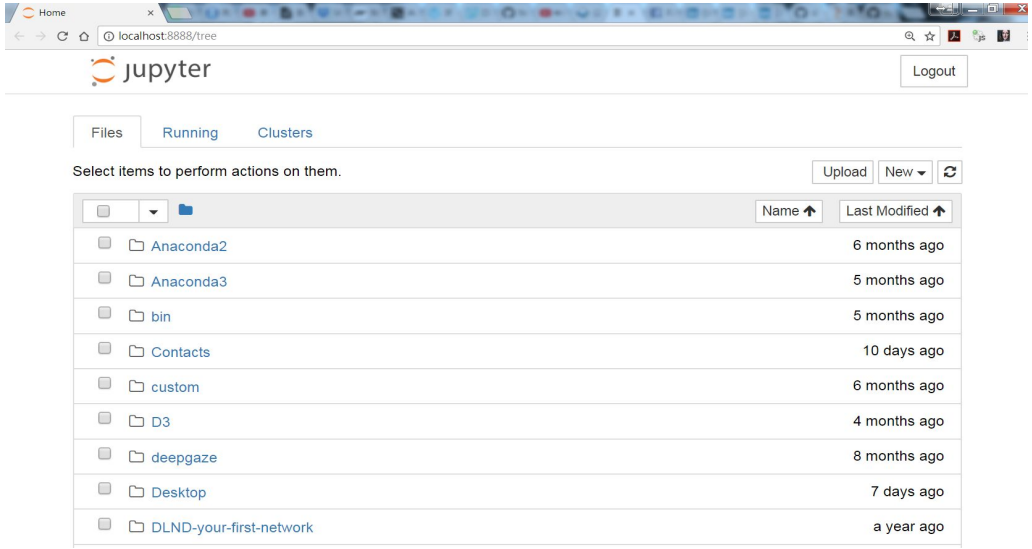
如何開啟 Jupyter notebook (本機)

- 安裝好 Anaconda 後，請打開 Anaconda Prompt，並輸入 jupyter notebook



順利開啟！

- 會自動在您預設的瀏覽器中打開 Jupyter notebook
- 顯示的資料則是 terminal 輸入的當前路徑 (預設使用者名稱)

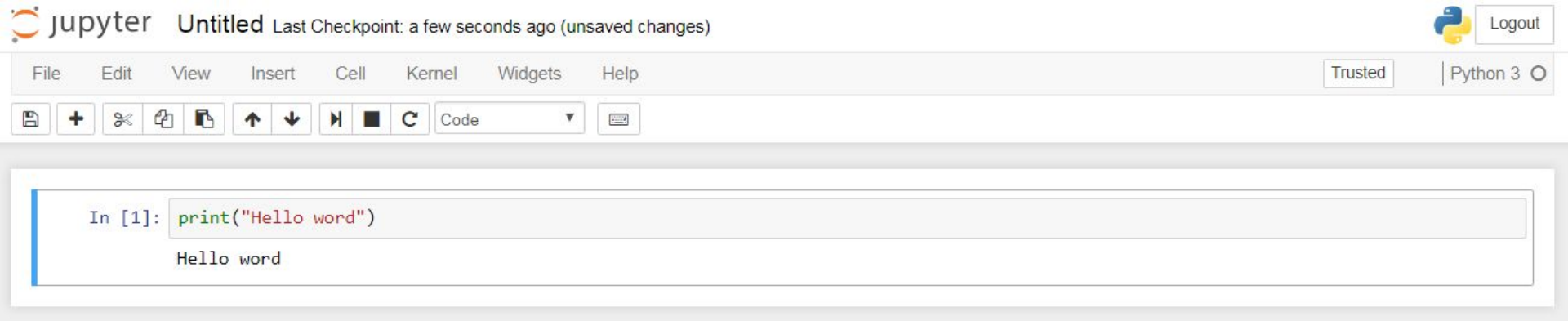


開啟 notebook

- 在 New 鍵下，選擇 Python 3，即可開啟新的 notebook

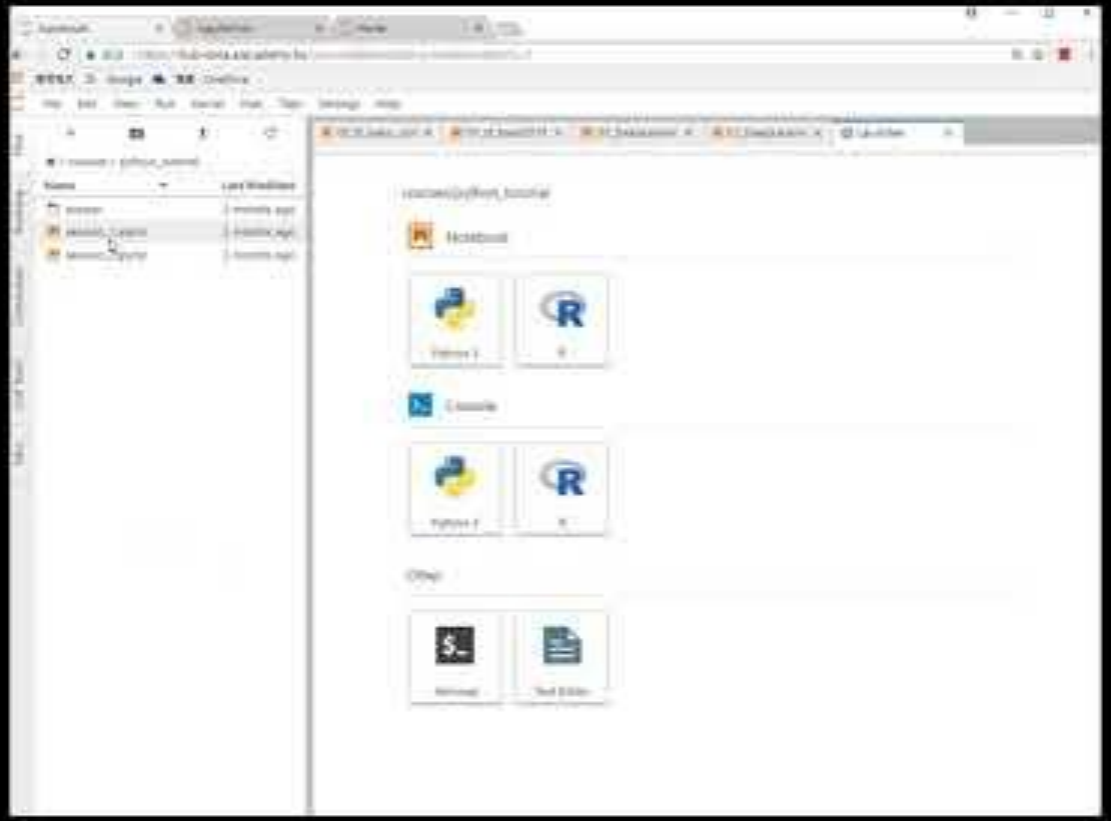
開啟 notebook

- 完成！可以開始輸入 code 囉

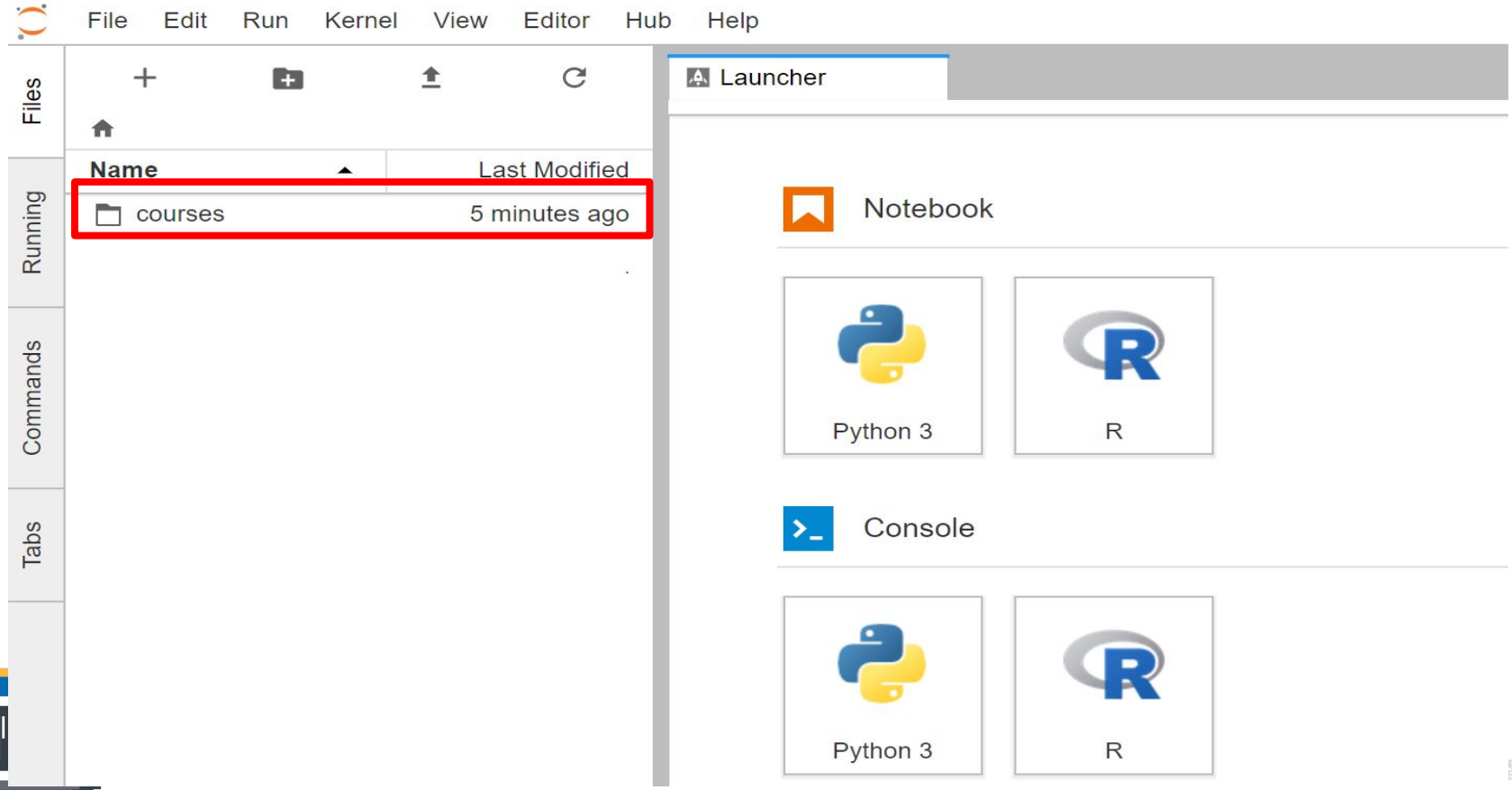


- jupyter notebook 會自動儲存
- code 的結果會即時顯示

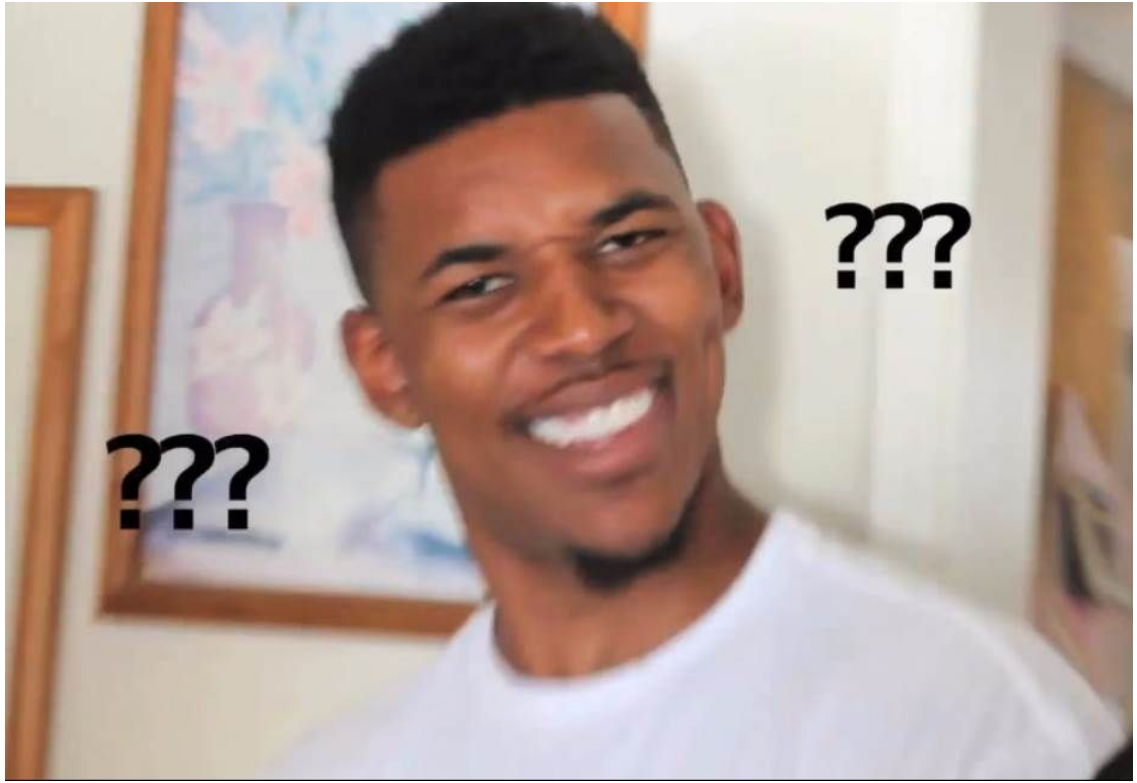
AIA Server 使用



AIA Server- jupyter notebook



明明不一樣，你跟我說這都是jupyter notebook?



Tree and Lab

- Lab：較人性化的介面，但功能不完善

- Tree：功能相對較完善

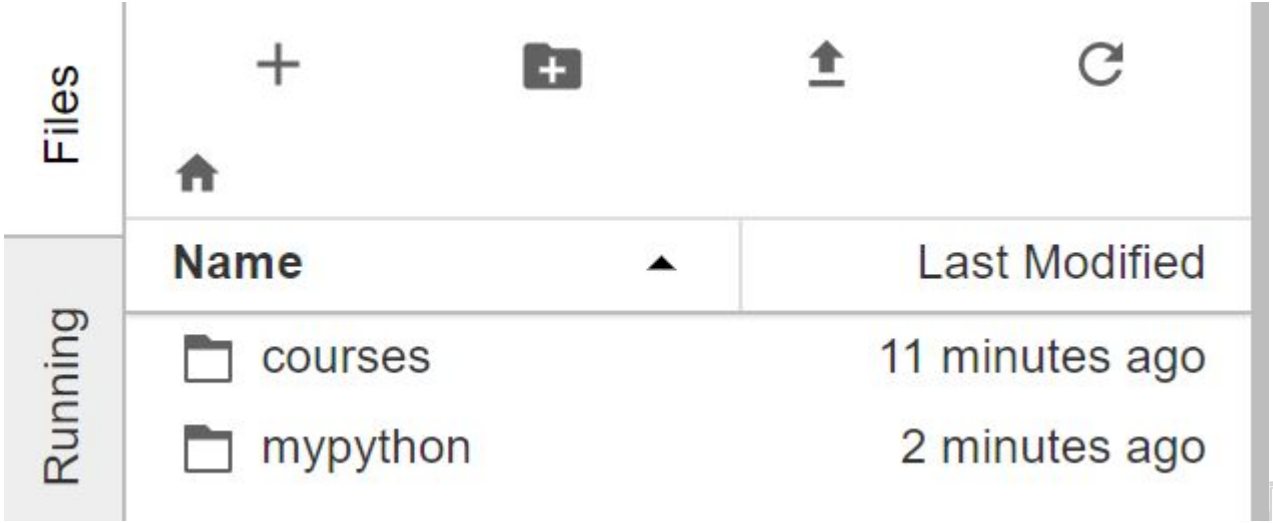
下載課程資料

- 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/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

```
jovyan@jupyter-evanstsai-40aiacademy-2etw:~$ cp -r courses/python_programming mypython
jovyan@jupyter-evanstsai-40aiacademy-2etw:~$ ls
*  courses  hsi-courses  lost+found  mypython  projectdata
jovyan@jupyter-evanstsai-40aiacademy-2etw:~$
```



Jupyter快捷鍵(非重點!請勿著墨太久!)

- 補充在另外獨立的slide中。
- slide連結：https://docs.google.com/presentation/d/1rBOmUrPdYcal24EOw7FV6dVQohDDwLDeKxE9RiXK6lY/edit?usp=sharing
- 稍微會寫python但沒用過jupyter notebook的建議可以先看一下這份補充；若沒有學過程式，也可以先開始後續python課程，稍微了解了程式語言後再回來參考喔。

台灣人工智慧學校

Python 程式設計

Felix

Basic Syntax

Data Type

```
print(type(100))          # <class 'int'>
print(type(counter))      # <class 'int'>

print(type(1000.0))       # <class 'float'>
print(type(miles))        # <class 'float'>

print(type("John"))       # <class 'str'>
print(type(name))         # <class 'str'>
```

Variables

```
# assignment
a = 1
b = c = 5

# assign multiple objects to multiple variables.
a, b, c = 1, 2, "John"
print(a)  # 1
print(b)  # 2
print(c)  # John
```

Data Type

```
counter = 100       # An integer assignment
miles   = 1000.0    # A floating point
name    = "John"    # A string

print(counter)  # 100
print(miles)    # 1000.0
print(name)     # John
```

Data Type

```
print(type(100))                # <class 'int'>
print(type(counter))            # <class 'int'>

print(type(1000.0))             # <class 'float'>
print(type(miles))              # <class 'float'>

print(type("John"))             # <class 'str'>
print(type(name))               # <class 'str'>
```

Keywords

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

Arithmetic Operators

Symbol	Task Performed
+	Addition
-	Subtraction
/	division
%	mod
*	multiplication
//	floor division
**	to the power of

Arithmetic Operators

```
add = 1 + 1     # 2
sub = 1 - 1     # 0
div = 4 / 2     # 2
mod = 4 % 3     # 1
mul = 2 * 3     # 6
f_div = 5 // 2 # 2
power = 2 ** 3 # 8
```

Comparison Operators

Symbol	Task Performed
==	True, if it is equal
!=	True, if not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Comparison Operators

```
a, b = 10, 20
```

```
a == b     # False
a != b     # True
a < b      # True
a > b      # False
a <= b     # True
a >= b     # False
```

Built-in Functions

e.g. print( ), type( ), int( ) and str( )

```
integer = 123
string = "456"

s_to_i = int(string)     # int now
i_to_s = str(integer)    # str now

print(type(s_to_i))      # <class 'int'>
print(type(i_to_s))      # <class 'str'>
```

練習 - part 1

Q1. 輸入兩個整數數字，計算兩數字之加、減、乘、除的結果，
並且列印出來。

```
Example Output:
第一個數字？ 20
第二個數字？ 10
20 + 10 = 30
20 - 10 = 10
20 * 10 = 200
20 / 10 = 2
```

hint1: 利用內建 input() 取得輸入數字，並且利用 int() 將輸入字串轉成整數。
hint2: num1 + num2 = sum 可利用 print(num1, "+", num2, "=", num1 + num2) 印出。

Data Structures

List - slicing

```
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8]
print(my_list[2:5])   # elements 3rd to 5th
                      ## [2, 3, 4]
print(my_list[:-5])   # elements beginning to 4th
                      ## [0, 1, 2, 3]
print(my_list[5:])    # elements 6th to end
                      ## [5, 6, 7, 8]
print(my_list[:])     # elements beginning to end
                      ## [0, 1, 2, 3, 4, 5, 6, 7, 8]
print(my_list[::3])   # slice a parent list with a step length
                      ## [0, 3, 6]
```

Numbers

```
# Output: <class 'int'>
print(type(5))

# Output: <class 'float'>
print(type(5.0))

# Output: <class 'complex'>
c = 5 + 3j
print(type(c))
```

Lists

```
# empty list
my_list = []

# list of integers
my_list = [1, 2, 3]

# list with mixed datatypes
my_list = [1, "Hello", 2.3]

# nested list
my_list = ["mouse", [8, 4, 6]]
```

List - index

```
my_list = ['h','e','l','l','o']

print(my_list[0])     # Output: h

print(my_list[1])     # Output: e

# my_list[5.0]     # Error! Only integer can be used for indexing

n_list = ["Happy", [2,0,1,8]]    # Nested List
print(n_list[1][3])              # Output: 8
```

List - negative indexing

```
my_list = ['p','r','o','b','e']

print(my_list[-1])    # Output: e

print(my_list[-5])    # Output: p
```

List - slicing

```
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8]
print(my_list[2:5])    # elements 3rd to 5th
                       ## [2, 3, 4]
print(my_list[:-5])    # elements beginning to 4th
                       ## [0, 1, 2, 3]
print(my_list[5:])     # elements 6th to end
                       ## [5, 6, 7, 8]
print(my_list[:])      # elements beginning to end
                       ## [0, 1, 2, 3, 4, 5, 6, 7, 8]
print(my_list[::3])    # slice a parent list with a step length
                       ## [0, 3, 6]
```

Built-in List Methods

```
num_list = [0, 0, 1, 2, 3, 4, 5, 6, 7, 8]

# append() is used to add an element at the end of the list.
num_list.append(9)

# remove() takes a single element as an argument and removes it
from the list.
num_list.remove(9)
```

Built-in List Methods

```
# index() is used to find the index value of a particular element.
num_list.index(5)

# pop() takes a single argument (index) and removes the element
present at that index from the list.
result = num_list.pop(7)

print(result)      # 6
print(num_list)    # [0, 0, 1, 2, 3, 4, 5, 7, 8]
```

Sets

```
# mathematical set operations
set_1 = set(['s', 'p', 'a','m'])
set_2 = set(['h','a','m'])

# union, intersection
print(set_1 | set_2)      # {'h', 'p', 'm', 's', 'a'}
print(set_1 & set_2)      # {'a', 'm'}

# symmetric difference
print(set_1 - set_2)      # {'p', 's'}
```

Tuples

```
# empty tuple
my_tuple = ()
print(my_tuple)        # Output: ()

# tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)        # Output: (1, 2, 3)
```

Strings

```
# all of the following are equivalent
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)
```

Strings

```
my_str = 'Hello World!'
print('my_str = ', my_str)   # my_str =  Hello World!

# first character, last character
print(my_str[0])      # H
print(my_str[-1])     # !

# slicing 3nd to 5th character
print(my_str[2:5])    # llo
```

Strings

```
str1 = 'Hello'
str2 = 'World!'

# using +
print(str1 + str2)     # HelloWorld!

# using *
print(str1 * 3)        # HelloHelloHello
```

Built-in Strings Methods

```
my_string = "hello world"

print(my_string.find("he"))        # Output: 0
print(my_string.capitalize())      # Output: Hello world
print(my_string.upper())           # Output: HELLO WORLD
print(my_string.endswith("d"))     # Output: True

print(my_string.split(" "))        # Output: ['hello', 'world']

print(my_string.replace("hello", "Nihao"))  # Output: Nihao world
```

Sets

```
# set of integers
my_set = {1, 2, 3}
print(my_set)      # {1, 2, 3}

# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)      # {'Hello', 1.0, (1, 2, 3)}
```

Sets

```
# mathematical set operations
set_1 = set(['s', 'p', 'a','m'])
set_2 = set(['h','a','m'])

# union, intersection
print(set_1 | set_2)      # {'h', 'p', 'm', 's', 'a'}
print(set_1 & set_2)      # {'a', 'm'}

# symmetric difference
print(set_1 - set_2)      # {'p', 's'}
```

Dictionary

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'a', 2: 'b'}

# dictionary with mixed keys
my_dict = {'name': 'Tom', 1: 23}
```

Dictionary

```
# Another define
my_dict = dict()

# add elements
my_dict['One'] = '1'
my_dict['OneTwo'] = 12
print (my_dict)        # {'One': '1', 'OneTwo': 12}

# update value
my_dict['One'] = 111
print (my_dict)        # {'One': 111, 'OneTwo': 12}
```

Dictionary

```
# Merge two lists to a dictionary.
names = ['One', 'Two', 'Three', 'Four', 'Five']
numbers = [1, 2, 3, 4, 5]
merged_dict = dict(zip(names, numbers))

print(merged_dict)     # {'One': 1, 'Two': 2, 'Three': 3, 'Four': 4, 'Five': 5}
```

Dictionary Methods

```
my_dict = {'name':'Jack', 'age': 16, 'gender':'man'}

# remove a particular item
print(my_dict.pop('gender'))      # man
print(my_dict)                    # {'name': 'Jack', 'age': 16}

# Returns view of dictionary's (key, value) pair
print(my_dict.items())            # [('name', 'Jack'), ('age', 16)]

# Return a new view of the dictionary's keys.
print(my_dict.keys())             # ['name', 'age']

# remove all items
my_dict.clear()
print(my_dict)                    # {}
```

練習 - part 2

Q1. 給定一個 a_list = [3, 7, 6, 2, 9, 4, 1], 請列印出下列結果:
(1) 第2個元素
(2) 最後一個元素
(3) 第3到第5個元素的列表

Q2.

編號	姓名
s1	John
s2	Tom
s3	Lisa

(1) 將上述表格資料, 存成 Dictionary 的資料結構。(key = 編號, value = 姓名)
(2) 列印出該 key 為 s2 的 value 的值
(3) 添加人員 編號 s4, 姓名 Mana
(4) 刪除人員 John

Control Flow

For Loop

```
# Program to find the sum of all numbers stored in a list
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# iterate over the list
sum = 0
for val in numbers:
    sum = sum + val

print("The sum is", sum)     # The sum is 55
```

if

```
num = 3
if num > 0:
    print(num, "is a positive number.")

num = -1
if num > 0:
    print(num, "is a positive number.")

## Output: 3 is a positive number.
```

if … else

```
num = -1
if num >= 0:
    print(num, "Positive or Zero")
else:
    print(num, "is a Negative number")

## Output: -1 is a Negative number.
```

if … elif … else

```
num = 0
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")

## Output: Zero
```

Logical - or, and

```
num = 5

if num == 0 or num == 1:
    print("Zero or One")
elif num >= 2 and num <= 10:
    print("From 2 to 10")
else:
    print('More')

## Output: From 2 to 10
```

is, not

```
num = 4

# num == 4
if num is 4:
    print("num is 4")

# !(num == 5)
if not num == 5:
    print("num is not 5")
```

```
# num != 6
if num is not 6:
    print("num is not 6")

# !(num == 7)
if not num is 7:
    print("num is not 7")
```

For Loop

```python
# Program to find the sum of all numbers stored in a list
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# iterate over the list
sum = 0
for val in numbers:
    sum = sum + val

print("The sum is", sum)     # The sum is 55
```

For loop with range()

```python
# range(stop)
# range(start, stop[, step])

numbers = [1, 2, 3, 4, 5, 6]

# iterate over the list using index
for i in range(len(numbers)):
    print("number", numbers[i])


# iterate over the list using 2 steps
for i in range(0, len(numbers), 2):
    print("2 steps", numbers[i])
```

```
# Output
  number 1
  number 2
  number 3
  number 4
  number 5
  number 6

# Output
  2 steps 1
  2 steps 3
  2 steps 5
```

For loop with enumerate( )

```
pets = ('Dogs', 'Cats', 'Turtles', 'Rabbits')

for index, pet in enumerate(pets):
    print(index, pet)



# Output:
  0 Dogs
  1 Cats
  2 Turtles
  3 Rabbits
```

While Loop

```
n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)     # The sum is 55
```

Nested Loop

```
for i in range(0, 2):
    for j in range(0, 2):
        print("i=", i, "j=", j, ", i*j=", i*j)

# Output:
  i= 0 j= 0 , i*j= 0
  i= 0 j= 1 , i*j= 0
  i= 1 j= 0 , i*j= 0
  i= 1 j= 1 , i*j= 1
```

break, continue and pass

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# break
for val in numbers:
    if val >= 4:
        break
    print(val)

# Output
1
2
3

# pass
for val in numbers:
    pass

# continue
for val in numbers:
    if val >= 3 and val <=8:
        continue
    print(val)

# Output
1
2
9
10
```

List comprehension

```
# make new lists by using iterable
squares = []
for x in range(10):
    squares.append(x**2)
print(squares)                # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]


# equivalently
squares = [x**2 for x in range(10)]
print(squares)                # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

List comprehension

```
# with if
squares = [x**2 for x in range(10) if x % 2 == 0]
print(squares)        ## [0, 4, 16, 36, 64]


# equivalently
squares = []
for x in range(10):
    if x % 2 == 0:
        squares.append(x**2)

print(squares)        ## [0, 4, 16, 36, 64]
```

練習 - part 3-1

Q1. 建立一個驗證密碼的小程式，程式 內建一組字串密碼，請使用者輸入一組字串密碼，比對密碼是否輸入正確。

```
Expected Result:
      請輸入密碼: Passw0rd
      密碼正確
      or
      請輸入密碼: adfgg
      密碼錯誤
```

Q2. 給予一個列表，計算出列表中元素為 2的倍數的和。

Sample List : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Expected Result : 30

練習 - part 3-2

Q3. 輸入人物的身高、體重，計算出該人物的 BMI
公式：BMI = 體重(公斤) / 身高*身高 (公尺)

P.S. 於2002年4月公布臺灣成人肥胖標準：
BMI＜18.5 為過輕，
18.5≦BMI＜24 為正常體重，
24≦BMI＜27 為過重，
BMI≧27 即為肥胖

Q4. 印出 1 到 50，但如果是 3 的倍數就印 Fizz，如果是 5 的倍數就印 Buzz，如果同時是 3 和 5 的倍數就印 FizzBuzz。

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

```
# 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

```python
# 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

```python
# 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

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

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

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<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

```python
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 nothong!"

# 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

```python
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']
```

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

```python
# 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

```python
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))
```

```python
# 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。