08：檔案處理

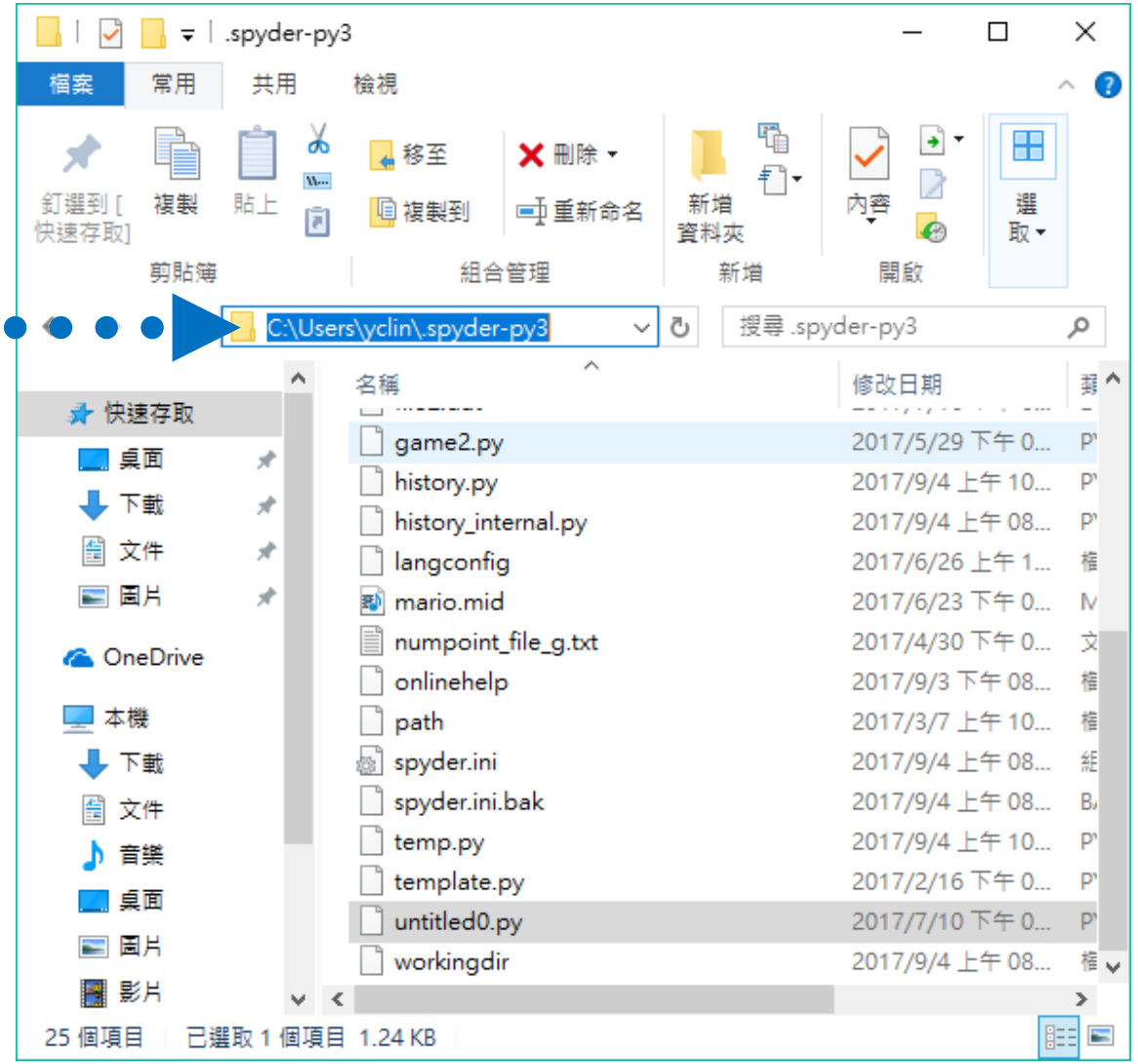
2018.1.30

主題

- 開啟檔案
- 檔案操作

絕對路徑 (absolute path)

相對路徑 (relative path)



三部曲

- 檔案操作的三部曲為：
  - 開啟 (open())
  - 讀取(read)與寫入(write)
  - 關閉 (close())

開啟檔案

- Python 內建的 open() 可以開啟檔案，語法如下：
  - open(filename [, mode] [, encode])

  - filename
    - 讀寫的檔案名稱，它是字串型態，可以是相對或絕對路徑
    - 如果沒有設定路徑，則會預設為目前執行程式的目錄

開啟檔案

- Python 內建的 open() 可以開啟檔案，語法如下：
  - open(filename [, mode] [, encode])
  - mode
    - 設定檔案開啟的模式(r, w, a)，也是字串型態
    - 省略 mode 參數，將預設為讀取模式

模式	說明
r	讀取模式，此為預設模式。
w	寫入模式，若檔案已存在，內容將會被覆蓋。
a	附加模式，若檔案已存在，內容將會被附加至尾端。

範例

```
content = '''Hello Python
《TQC+程式語言Python 3專業認證》
即將推出
'''

f = open("file1.txt", 'w')
f.write(content)
f.close()
```

範例

```
2 f = open("file1.txt", 'r')
3
4 for line in f:
5     print(line)
6     # print(line, end='')
7
8 f.close()
```

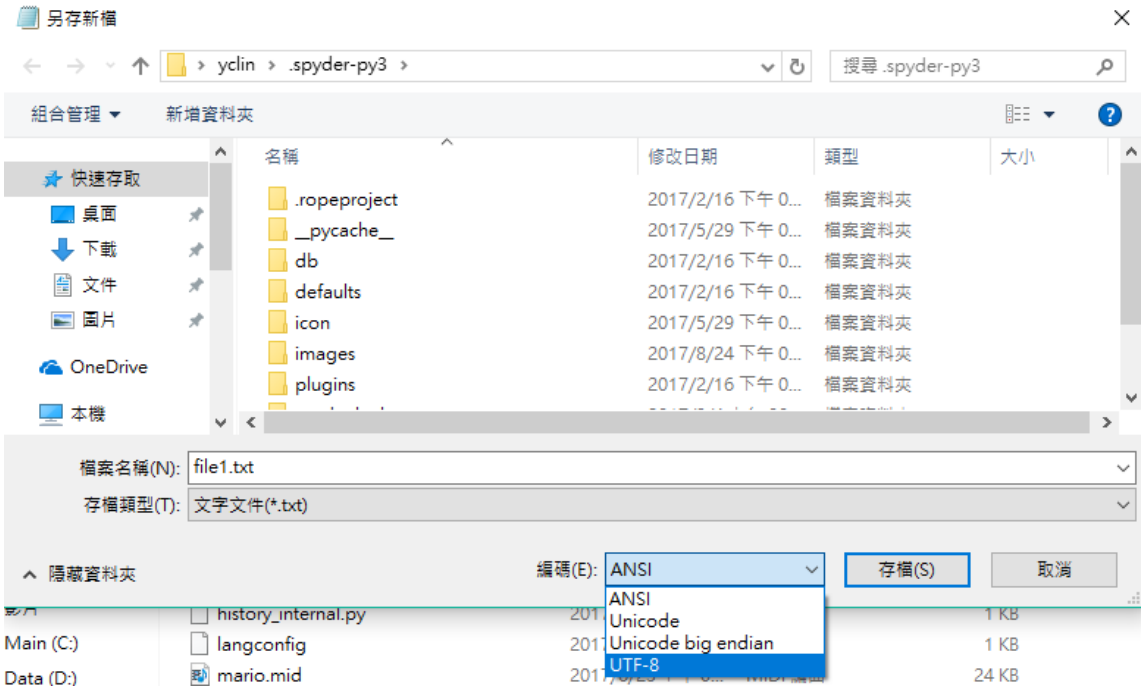
偷懶一下

```
3 with open("file1.txt", 'r') as f:
4     for line in f:
5         print(line)
```

開啟檔案

- Python 內建的 open() 可以開啟特定檔案，語法如下：
  - open(filename [, mode] [, encode])
  - encode
    - 指定檔案的編碼模式，一般可設定 cp950 或 UTF-8
    - 預設的編碼依作業系統而定，如果是正體中文 Windows 系統，預設的編碼是 cp950，也就是記事本儲存為 ANSI 的編碼。

檔案編碼 – UTF-8



- 國際間通行的編碼以及許多 Linux 系統，預設都是使用 UTF-8 編碼，因此建議將檔案另存為 UTF-8 (不要使用 ANSI)。

檔案編碼 – UTF-8

- 如果檔案編碼已更改為 UTF-8，則讀取時就必須明確指定編碼為 UTF-8，否則會出現錯誤。
  - f = open(“file1.txt”, ‘r’, encoding=“UTF-8”)
  - f = open(“file1.txt”, ‘r’, encoding=“utf-8”)

- UTF-8 = 8-bit Unicode Transformation Format

主題

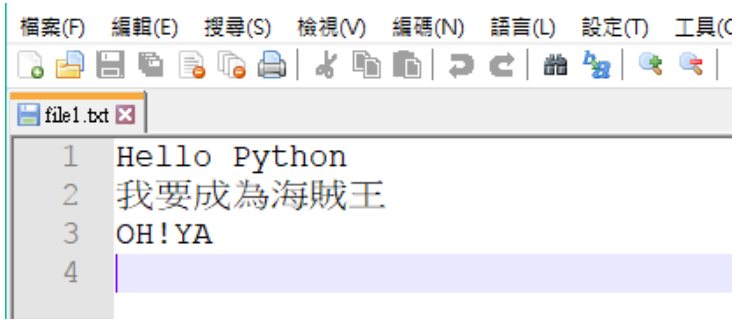
- 開啟檔案
- 檔案操作

處理檔案常用方法

方法	說明
close()	關閉檔案，檔案關閉後就不能再進行讀寫的操作。
flush()	檔案在關閉時會將資料寫入檔案中，也可以使用 flush() 強迫將緩衝區的資料立即寫入檔案中，並清除緩衝區。
read([size])	讀取指定長度的的字元，如果未指定長度則會讀取所有字元。
readable()	測試是否可讀取
★ readline([size])	讀取目前文字指標所在行中 size 長度的文字內容，若省略參數，則會讀取一整列，包括 "\n" 字符。
★ readlines()	讀取所有行，它會傳回一個串列。
next()	移動到下一行
seek(0)	將指標移到文件最前端
tell()	傳回文件目前位置
★ write(str)	將指定的字串寫入文件中， 它沒有返回值。
writable()	測試是否可寫入

read()

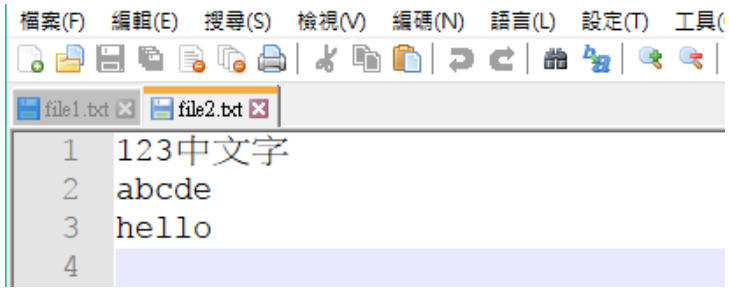
- read() 會從目前檔案指標的位置，讀取指定長度的字元
  - 如果未指定長度則會讀取所有的字元




```
3 f = open("file1.txt", 'r')
4
5 print(f.read(5))
6
7 f.close()
```

readline()

- 讀取目前檔案指標所在行中 size 長度的文字內容
  - 若省略參數，則會讀取一整列，包括 “\n” 字元



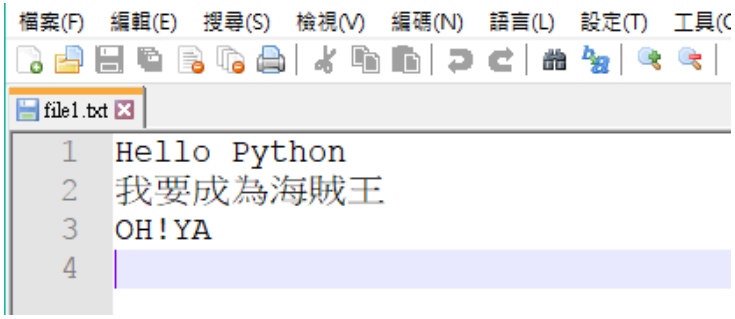

```
3 f = open("file2.txt", 'r', encoding="UTF-8-sig")
4 print(f.readline())
5 print(f.readline(3))
6 f.close()
```

```
123中文字

abc
```

readlines()

- 讀取全部檔案內容，並以串列方式傳回
  - 檔案內的每一列會成為串列中的一個元素

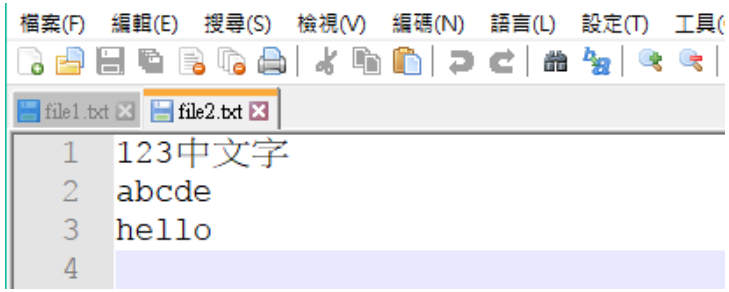



```
3 with open("file1.txt", 'r') as f:
4     data = f.readlines()
5
6     print(type(data))
7     print(data)
```

```
<class 'list'>
['Hello Python\n', '我要成為海賊王\n', 'OH!YA\n']
```

注意

- 讀取 UTF-8 編碼的 file2.txt 檔案內容




```
2 with open("file2.txt", 'r', encoding="UTF-8") as f:
3     data = f.readlines()
4     print(data)
5
6 f = open("file2.txt", 'r', encoding="UTF-8")
7 txt = f.read(5)
8 print(txt)
9 f.close()
```

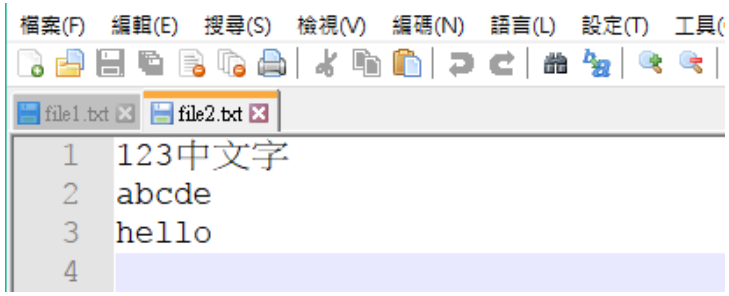
```
['\ufeff123中文字\n', 'abcde\n', 'hello\n']
123中
```

BOM (Byte Order Mark)

- 串列內容的第一筆資料前面多了一個「\ufeff」字元，這個字元是文件前端代碼，俗稱 BOM。
  - 它是在中文 Windows 系統中，用「記事本」將檔案儲存為 UTF-8 時自動產生。
  - BOM 會佔 1 個字元，因此第 7 列執行的結果只看到「123 中」這 4 個字元，因為第一個字元 BOM 未顯示出來。

UTF-8-SIG

- 另一種處理方式就是讀取有 BOM 的文件檔時，明確地加上「encoding = 'UTF-8-SIG'」將 BOM 去除。




```
with open("file2.txt", 'r', encoding="UTF-8-sig") as f:
    data = f.readlines()
    print(data)

f = open("file2.txt", 'r', encoding="UTF-8-sig")
txt = f.read(5)
print(txt)
f.close()
```

```
['123中文字\n', 'abcde\n', 'hello\n']
123中文
```

密技

```
2 file = open("number.csv", "r", encoding="UTF-8")
3
4 num_lst = []
5
6 for line in file:
7     # 去掉字串頭/尾的空白字元，再利用','進行切割
8     row = line.strip().split(',')
9     num_lst.extend([int(x) for x in row])
10
11 print("Sum is", sum(num_lst))
12
13 file.close()
```

number.csv

```
1  14, 52,76 , 43
2  32, 45, 99, 103
```

PY3-0002

九、資料計算

2. 設計說明：

(1) 請撰寫一程式，讀取 **read.txt**（每一列的格式為名字和身高、體重，以空白分隔）並顯示所有人的平均身高、平均體重以及最高者、最重者。

＊ 提示：輸出浮點數到小數點後第二位。

TO BE CONTINUED...