

文字檔案處理 File I/O

Python 編程課程 第 5 課

- 讀取文字檔案
- 寫入文字檔案
- 異常／錯誤處理

處理文字檔案（開啟、讀取、關閉）

Python 自帶 **open()** 函數，可以**開啟**文字檔案
（回傳文字物件類型賦值給變量）

read() 函數可以**讀取**文檔內容
（賦值給字串變量）

完成檔案操作後，用 **close()** 函數**關閉**檔案

程式碼

```
file = open("datafile.txt", "r")  
contents = file.read()  
file.close()  
  
print(contents)
```

輸出

```
Hello world !  
I go to school by bus.  
Goodbye and thank you Mr. Chan.  
See you next time.
```

開啟文字檔案 - open()

open() 是處理文字檔案最重要的函數
當中需要傳入兩個**參數**：

- 文字檔案發副檔名
- 開啟模式

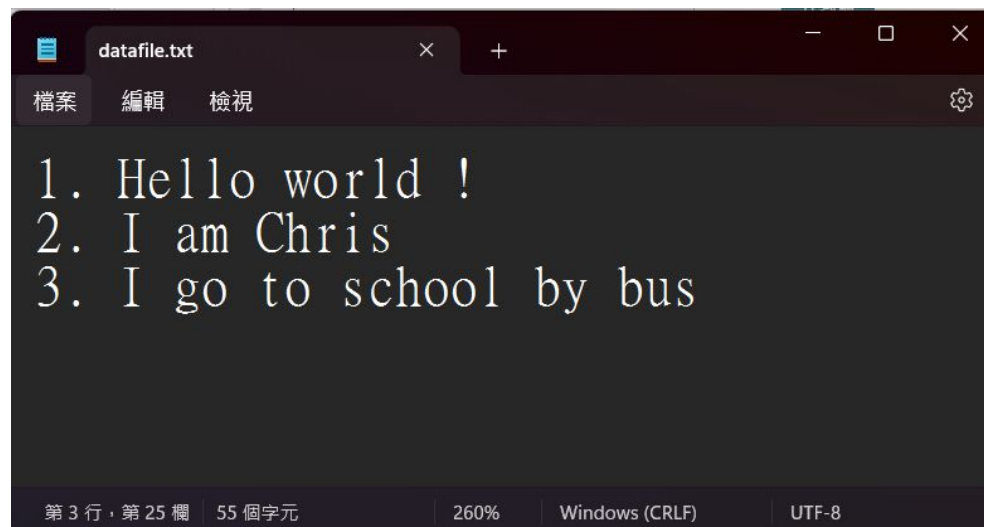
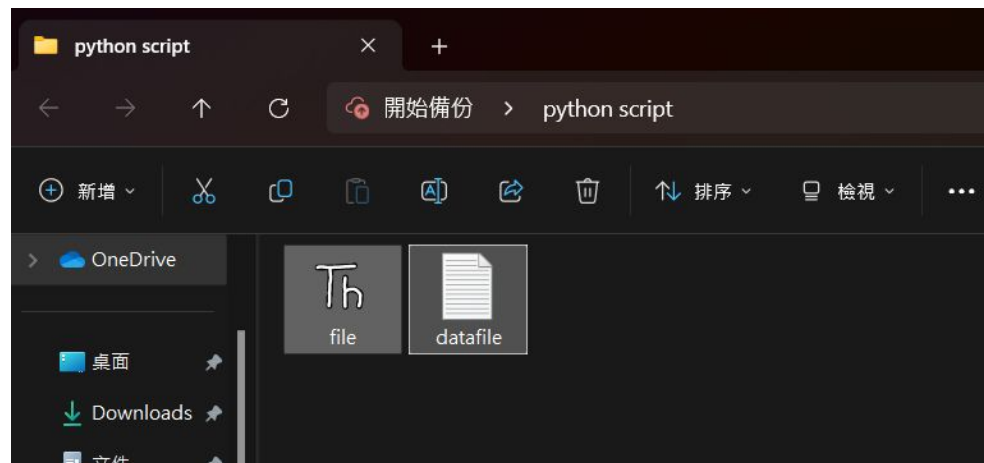
```
file = open("datafile.txt", "r")
```

開啟模式	功能	如果檔案 不存在	如果檔案 存在
"r"	讀取內容	錯誤	/
"w"	覆寫內容	創建檔案	/
"a"	添加內容	創建檔案	/
"x"	創建檔案	/	錯誤

讀取文字檔案 - 請打開Thonny一起做

第一步：創建 .txt 檔案

1. 在桌面新增一個文件夾，在裡面新增：
 - **Python file** (Thonny圖標)
 - **文字文件** (命名為“**datafile**”)
2. 打開 **datafile.txt**，寫入右邊內容
 - Hello world !
 - I am Chris (或者你的名字)
 - I go to school by bus



讀取文字檔案 - read(), readline(), readlines()

有三個讀取文字檔案內容的函數：

程式碼

```
file.read() # 讀取所有, 回傳字串
```

程式碼


```
file.readline() # 讀取一行, 回傳字串
```

程式碼

```
file.readlines() # 讀取所有, 回傳列表
```


輸出

```
1. Hello world !  
2. I am Chris  
3. I go to school by bus  
<class 'str'>
```




輸出

```
1. Hello world !  
  
<class 'str'>
```



輸出

```
['1. Hello world !\n', '2. I am Chris\n', '3. I go to school  
by bus']  
<class 'list'>
```



第二步：撰寫 .py 程式

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

```
# 建立變量 file
# 用 open() 開啟文字檔 datafile.txt, 讀取模
式
file = open("_____", _____)
# 用 .read() 讀取文檔內容
content = file._____
# 用 .close() 關閉文檔
file._____
# 列印內容和內容類型
print(content)
print(_____content_____)
```

輸出

```
1. Hello world !
2. I am Chris
3. I go to school by bus
<class 'str'>
```

第二步：撰寫 .py 程式

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

```
# 建立變量 file
# 用 open() 開啟文字檔 datafile.txt, 讀取模
式
file = open("datafile.txt", "r")
# 用 .read() 讀取文檔內容
content = file.read()
# 用 .close() 關閉文檔
file.close()
# 列印內容和內容類型
print(content)
print(type(content))
```

輸出

```
1. Hello world !
2. I am Chris
3. I go to school by bus
<class 'str'>
```


第三步：更多讀取方式

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

```
file = open("datafile.txt", ____)
```

用 .readline() 讀取文檔內容

```
line1 = file._____
```

```
line2 = file._____
```

```
line3 = file._____
```

關閉文檔

```
file._____
```

```
file = open("datafile.txt", ____)
```

用 .readlines() 讀取文檔內容

```
content_2 = file._____
```

關閉文檔

```
file._____
```

輸出

1. Hello world !

2. I am Chris

3. I go to school by bus

<class 'str'>

['1. Hello world !\n', '2. I am Chris\n', '3. I
go to school by bus']

<class 'list'>

第三步：更多讀取方式

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

```
file = open("datafile.txt", "r")
# 用 .readline() 讀取文檔內容
line1 = file.readline()
line2 = file.readline()
line3 = file.readline()
# 關閉文檔
file.close()

file = open("datafile.txt", "r")
# 用 .readlines() 讀取文檔內容
content_3 = file.readlines()
# 關閉文檔
file.close()
```

輸出

```
1. Hello world !
2. I am Chris
3. I go to school by bus
<class 'str'>

['1. Hello world !\n', '2. I am Chris\n', '3. I
go to school by bus']
<class 'list'>
```

自動關閉檔案 - with

處理檔案時，我們開啟了一些資源，如果這些資源**忘記關閉**，或是**意外沒有關閉**，可能造成**效能問題**，甚至出現**錯誤**

此時，用 **with** 自動關閉檔案就是其中一個「**資源管理**」技巧

傳統 `open()` 和 `.close()`

```
file = open('MP1_syllabus.txt')
contents = file.read()
file.close()
```

3行代碼

with ... as

*file只在 with 的範圍內可以使用，離開這個範圍時 file 會自動關閉

```
with open('MP1_syllabus.txt') as file:
    contents = file.read()
```

with 範圍

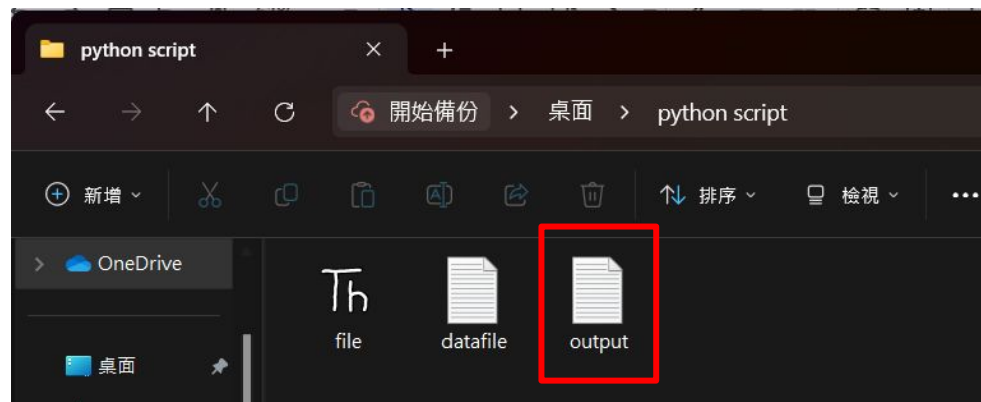
2行代碼 + 杜絕潛在問題



寫入文字檔案 - 請打開Thonny一起做

第一步：創建 .txt 檔案

1. 在桌面新增一個文件夾，在裡面新增：
 - **Python file** (Thonny圖標)
 - **文字文件** (命名為“**output**”)
2. 打開 **output.txt**，寫入右邊內容
 - Studying makes me happy.
(學習使我愉快 😊)



覆寫內容 Write vs 添加內容 Append

Write ('w')

output.txt (之前)

Python is the best !



程式碼

```
with open('output.txt', 'w') as file:  
    file.write('Mastering Python is tough but enjoyable')
```

output.txt (after)

Mastering Python is tough but enjoyable



'w' 會覆寫文檔所有內容

Append ('a')

output.txt (before)

Python is the best !



程式碼

```
with open('output.txt', 'a') as file:  
    file.write('Mastering Python is tough but enjoyable')
```

output.txt (after)

Python is the best !

Mastering Python is tough but enjoyable



寫入內容 - write(), writelines()

有兩個寫入文檔內容的函數：

程式碼

```
content_1 = "Python is the best !"  
  
with open('output.txt', 'w') as file:  
    file.write(content_1)           # write(): 寫入單行 內容
```

輸出

Python is the best !



程式碼

```
content_2 = ["apple \n",  
             "banana \n",  
             "cherry \n"]  
  
with open('output.txt', 'w') as file:  
    file.writelines(content_2)      # writelines(): 寫入多行 內容
```

輸出

apple
banana
cherry



第二步：覆寫文檔內容

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

```
# 建立想寫入的內容變量
content_1 = "Python is the best !"

# 用 open() 開啟 output.txt, 覆寫模式 "w"
file = _____"output.txt", _____

# 用 .write() 寫入文檔內容
file.write(_____)

# 用 .close() 關閉文檔
file._____
```

輸出

Python is the best !



第二步：覆寫文檔內容

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

```
# 建立想寫入的內容變量
content_1 = "Python is the best !"

# 用 open() 開啟 output.txt, 覆寫模式 "w"
file = open("output.txt", "w")

# 用 .write() 寫入文檔內容
file.write(content_1)

# 用 .close() 關閉文檔
file.close()
```

輸出

Python is the best !



第三步：添加文檔內容

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

建立想寫入的內容變量(列表)

```
content_2 = ["apple \n",  
             "banana \n",  
             "cherry \n"]
```

用 with...as 開啟 output.txt, 添加模式 "a"

```
_____ open(_____) _____:
```

用 .writelines() 寫入每行內容

```
file._____content_2_
```

輸出

Python is the best !apple

banana

cherry



第三步：添加文檔內容

粉紅色斜體：題目

黑色正常字體：提供的程式碼

紅色底線：請填入程式碼

程式碼

建立想寫入的內容變量(列表)

```
content_2 = ["apple \n",  
             "banana \n",  
             "cherry \n"]
```

用 with...as 開啟 output.txt, 添加模式 "a"

```
with open('output.txt', 'a') as file:
```

用 .writelines() 寫入每行內容

```
file.writelines(content_2)
```

輸出

Python is the best !apple

banana

cherry



異常／錯誤處理

檔案不存在

若所開啟的**檔案不存在**，或檔案**名稱輸入錯誤**，
程式運行會出現錯誤並停止運作，後面的程式碼也不會執行

```
with open('non_exist.txt') as file:  
    contents = file.read()
```

程式報錯

FileNotFoundError: [Errno 2] No such file or directory: 'non_exist.txt'

例外處理 - Try Except

使用 try 和 except 進行保護，避免**整個程式**因錯誤而停止。

嘗試執行這些程式碼

如果 try 程式塊有錯誤，
就執行下面的程式碼

```
try:
    with open('non_exist.txt') as file:
        contents = file.read()
except:
    print('the file does not exist.')
```

try-except

< 繼續運行其他程式碼 >

好習慣：

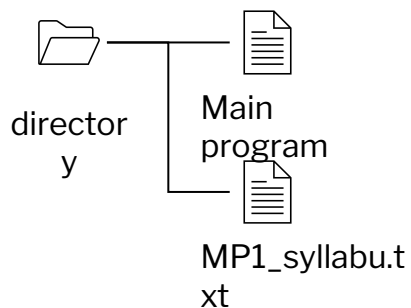
預想一下如果 try 程式塊會出現什麼錯誤，再寫在 except 程式塊，方便進行偵錯

常用例外(異常／錯誤) Exceptions

例外	錯誤原因
FileNotFoundError	檔案不存在
SyntaxError	Python 語法錯誤
TypeError	參數種類錯誤／錯配
IndentationError	縮進錯誤
NameError	變量未定義
IndexError	索引值超過了列表的大小

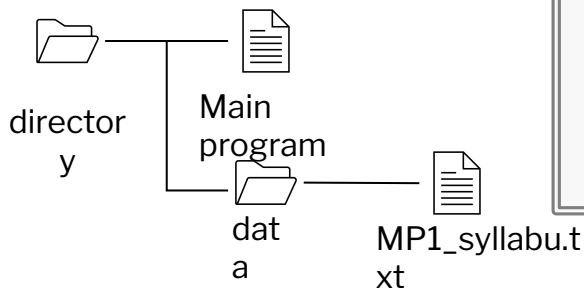
檔案路徑 File path

When a pass a **simple filename** to `open()`,
Python looks in the current directory



```
with open('MP1_syllabus.txt') as file:  
    contents = file.read()
```

If the file is not in the same directory,
you can use a **relative filepath**



```
with open('data/MP1_syllabus.txt') as file:  
    contents = file.read()
```


總結

```
# 讀取文字檔案 open() close()
```

```
file = open('MP1_syllabus.txt')
```

```
contents = file.read()
```

```
file.close()
```

```
# 資源管理: 自動關閉檔案 with...as
```

```
with open('MP1_syllabus.txt') as file:
```

```
    contents = file.read()
```

```
# 讀取所有文字內容 read()
```

```
with open('MP1_syllabus.txt') as file:
```

```
    contents = file.read()
```

```
# 讀取單行文字內容 readline()
```

```
with open('MP1_syllabus.txt') as file:
```

```
    contents = file.readline()
```

```
# 讀取多行文字內容 readlines()
```

```
with open('MP1_syllabus.txt') as file:
```

```
    contents = file.readlines()
```

```
# 寫入單行內容 write()
```

```
with open('output.txt', 'w') as file:
```

```
    file.write('string to write')
```

```
# 寫入多行內容／列表 writelines()
```

```
with open('output.txt', 'w') as file:
```

```
    file.writelines(['line 1\n', 'line 2'])
```