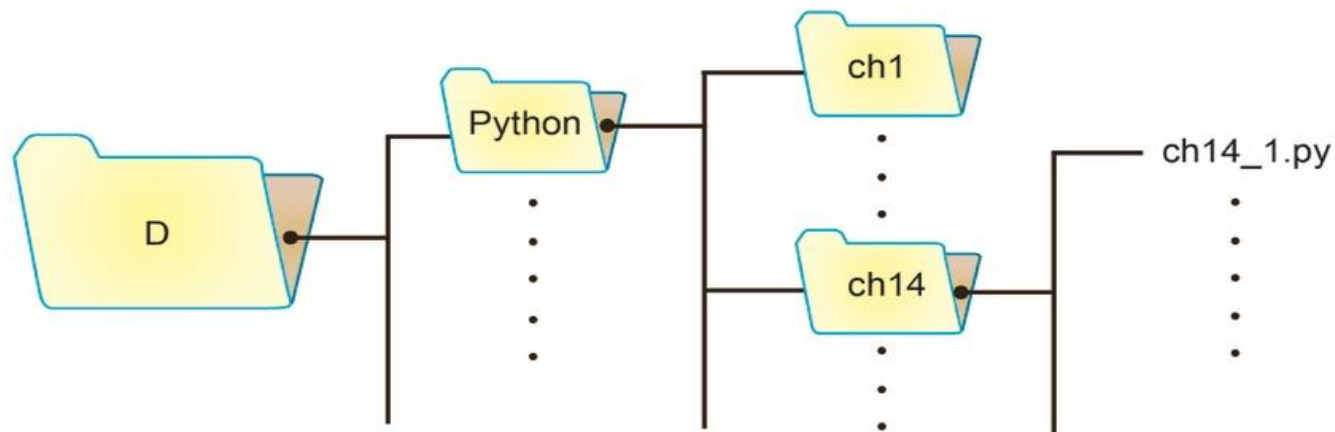


Chapter 14

檔案的讀取與寫入

14-1：資料夾與檔案路徑



- 對於ch14_1.py而言：
 - 檔案路徑名稱 D:\Python\ch14\ch14_1.py
 - 目前工作目錄(資料夾) D:\Python\ch14

14-1：資料夾與檔案路徑

- 絕對路徑：路徑從根目錄開始表達
 - D:\Python\ch14\ch14_1.py
- 相對路徑：指相對於目前工作目錄的路徑
 - 若目前工作目錄為D:\Python\ch14\，則上述檔案的相對路徑為 ch14_1.py
- ".": 目前資料夾
 - 使用上可省略，所以 .\ch14_1.py 與 ch14_1.py 意義相同。
- "..": 上一層資料夾
 - D:\Python\ch14\ch14_1.py 與 D:\Python\..\Python\ch14\..\ch14\ch14_1.py 相同。

14-1：資料夾與檔案路徑

- Python 內與檔案路徑相關的模組是“os”，所以使用前都要導入此模組。
 - `import os`

14-1：資料夾與檔案路徑

- 取得目前工作目錄os.getcwd()

```
1 import os
2
3 print(os.getcwd())E:\PythonTest\CH14
```

- 取得絕對路徑os.path.abspath()

```
1 import os
2
3 print(os.path.abspath("."))E:\PythonTest\CH14
4 print(os.path.abspath(".."))E:\PythonTest
5 print(os.path.abspath("test-CH14.py"))E:\PythonTest\CH14\test-CH14.py
```

- 傳回特定路段相對路徑os.path.relpath()

```
1 import os
2
3 print(os.path.relpath("E:\\PythonTest\\"))..
4 print(os.path.relpath("E:\\PythonTest\\CH13"))..\CH13
5 print(os.path.relpath("E:\\"))..\..
```

14-1：資料夾與檔案路徑

- 檢查路徑方法 `exist/isabs/isdir/isfile`

下列是常用的 `os.path` 模組方法。

`exist(path)`：如果 `path` 的檔案或資料夾存在傳回 `True` 否則傳回 `False`。

`isabs(path)`：如果 `path` 的檔案或資料夾是絕對路徑傳回 `True` 否則傳回 `False`。

`isdir(path)`：如果 `path` 是資料夾傳回 `True` 否則傳回 `False`。

`isfile(path)`：如果 `path` 是檔案傳回 `True` 否則傳回 `False`。

14-1：資料夾與檔案路徑

- 檢查路徑方法exist/isabs/isdir/isfile

```

1 import os
2
3 print(f"檔案或資料夾存在 = {os.path.exists('CH14')} = ")
4 print(f"檔案或資料夾存在 = {os.path.exists('E:/PythonTest/CH14')} = ")
5 print(f"檔案或資料夾存在 = {os.path.exists('test-CH14.py')} = ")
6 print("-----")
7
8 print(f"test-CH14.py 是絕對路徑： {os.path.isabs('test-CH14.py')}")
9 print(f"E:/PythonTest/CH14 是絕對路徑： {os.path.isabs('E:/PythonTest/CH14')}")
10 print("-----")
11
12 print(f"test-CH14.py 是資料夾： {os.path.isdir('test-CH14.py')}")
13 print(f"E:/PythonTest/CH14 是資料夾： {os.path.isdir('E:/PythonTest/CH14')}")
14 print("-----")
15
16 print(f"test-CH14.py 是檔案： {os.path.isfile('test-CH14.py')}")
17 print(f"E:/PythonTest/CH14 是檔案： {os.path.isfile('E:/PythonTest/CH14')}")

```

```

檔案或資料夾存在 = os.path.exists('CH14') = False
檔案或資料夾存在 = os.path.exists('E:/PythonTest/CH14') = True
檔案或資料夾存在 = os.path.exists('test-CH14.py') = True
-----
test-CH14.py 是絕對路徑： False
E:/PythonTest/CH14 是絕對路徑： True
-----
test-CH14.py 是資料夾： False
E:/PythonTest/CH14 是資料夾： True
-----
test-CH14.py 是檔案： True
E:/PythonTest/CH14 是檔案： False

```

14-1：資料夾與檔案路徑

- 檔案與目錄的操作 `mkdir/rmdir/remove/chdir`

這幾個方法是在 `os` 模組內，建議執行下列操作前先用 `os.path.exists()` 檢查是否存在。

`mkdir(path)`：建立 `path` 目錄。

`rmdir(path)`：刪除 `path` 目錄，限制只能是空的目錄。如果要刪除底下有檔案的目錄需使用 `rmtree()`。

`remove(path)`：刪除 `path` 檔案。

`chdir(path)`：將目前工作資料夾改至 `path`。

14-1：資料夾與檔案路徑

- mkdir:

```
1 import os
2
3 newDir = "testDir"
4
5 if os.path.exists(newDir):
6     print(f"資料夾 {newDir} 已經存在")
7 else:
8     os.mkdir(newDir)
9     print(f"建立 {newDir} 成功")
```

建立 testDir 成功

名稱	修改日期	類型
test.py	2021/11/14 上午 10:06	Python File
test-CH14.py	2021/11/13 上午 12:10	Python File

名稱	修改日期	類型
testDir	2021/11/14 上午 10:07	檔案資料夾
test.py	2021/11/14 上午 10:06	Python File
test-CH14.py	2021/11/13 上午 12:10	Python File

資料夾 testDir 已經存在

```
Traceback (most recent call last):
  File "e:\PythonTest\CH14\test.py", line 6, in <module>
    os.mkdir(newDir)
FileExistsError: [WinError 183] 當檔案已存在時，無法建立該檔案。: 'testDir'
```

14-1：資料夾與檔案路徑

- rmdir:

```
1 import os
2
3 newDir = "testDir"
4
5 if os.path.exists(newDir):
6     os.rmdir(newDir)
7     print(f"刪除 {newDir} 成功")
8 else:
9     print(f"資料夾 {newDir} 不存在")
```

刪除 testDir 成功

名稱	修改日期	類型
testDir	2021/11/14 上午 10:07	檔案資料夾
test.py	2021/11/14 上午 10:06	Python File
test-CH14.py	2021/11/13 上午 12:10	Python File




名稱	修改日期	類型
test.py	2021/11/14 上午 10:06	Python File
test-CH14.py	2021/11/13 上午 12:10	Python File



14-1：資料夾與檔案路徑

- remove:

```
1  import os
2
3  fName = "test-1.py"
4
5  if os.path.exists(fName):
6      os.remove(fName)
7      print(f"刪除 {fName} 成功")
8  else:
9      print(f"檔案 {fName} 不存在")
```

刪除 test-1.py 成功

名稱	修改日期	類型
 test.py	2021/11/14 上午 10:14	Python File
 test-1.py	2021/11/13 上午 12:10	Python File
 test-CH14.py	2021/11/14 上午 10:13	Python File

名稱	修改日期	類型
 test.py	2021/11/14 上午 10:14	Python File
 test-CH14.py	2021/11/14 上午 10:13	Python File

14-1：資料夾與檔案路徑

```

1  import os
2
3  newDir = "E:\\PythonTest\\CH100"
4  currentDir = os.getcwd()
5  print(f"目前工作資料夾： {currentDir}")
6
7  #如果 newDir 不存在就建立新的資料夾
8  if os.path.exists(newDir):
9      print(f"資料夾 {newDir} 已經存在")
10 else:
11     os.mkdir(newDir)
12     print(f"建立資料夾 {newDir} 成功")
13
14 #將目前工作目錄切換到 newDir
15 os.chdir(newDir)
16 print(f"新工作資料夾： {os.getcwd()}")
17
18 #將目前工作目錄切回 currentDir
19 os.chdir(currentDir)
20 print(f"返回原工作資料夾： {os.getcwd()}")

```

名稱	修改日期	類型
.vscode	2021/8/18 下午 04:08	檔案資料夾
CH13	2021/11/13 上午 12:05	檔案資料夾
CH14	2021/11/14 上午 10:16	檔案資料夾

目前工作資料夾： E:\PythonTest\CH14
 建立資料夾 E:\PythonTest\CH100 成功
 新工作資料夾： E:\PythonTest\CH100
 返回原工作資料夾： E:\PythonTest\CH14

名稱	修改日期	類型
.vscode	2021/8/18 下午 04:08	檔案資料夾
CH13	2021/11/13 上午 12:05	檔案資料夾
CH14	2021/11/14 上午 10:16	檔案資料夾
CH100	2021/11/14 上午 10:21	檔案資料夾

目前工作資料夾： E:\PythonTest\CH14
 資料夾 E:\PythonTest\CH100 已經存在
 新工作資料夾： E:\PythonTest\CH100
 返回原工作資料夾： E:\PythonTest\CH14

14-1：資料夾與檔案路徑

- 傳回檔案路徑 `os.path.join()`
 - 將 `os.path.join()` 參數內的字串結合為一個檔案路徑，參數可以有2個到多個。

```
1 import os
2
3 print(os.path.join("E:\\", "Pythontest", "Ch14\\", "test.py"))
4 print(os.path.join("E:\\Pythontest", "Ch14\\", "test.py"))
5 print(os.path.join("E:\\Pythontest/Ch14\\", "test.py"))
6
7
8 print(os.path.join("E:\\", "Pythontest\\Ch14\\", "test.py"))
9
10 listFiles = ["ch13.py", "ch14.py", "ch15.py"]
11 for file in listFiles:
12     print(os.path.join("E:\\", "Pythontest\\", file))
```

```
E:\Pythontest\Ch14\test.py
E:\Pythontest\Ch14\test.py
E:\Pythontest/Ch14\test.py
E:\Pythontest\Ch14\test.py
E:\Pythontest\ch13.py
E:\Pythontest\ch14.py
E:\Pythontest\ch15.py
```

14-1：資料夾與檔案路徑

- 獲得特定檔案的大小 `os.path.getsize()`

```
1 import os
2
3 print(os.path.getsize("test-CH14.py")) 1879
4 print(os.path.getsize("E:/PythonTest/CH14/test.py")) 119
```

```
1 import os
2
3 print(os.path.getsize("test-CH13.py"))
4 print(os.path.getsize("E:/PythonTest/CH14/test.py"))
```

```
Traceback (most recent call last):
  File "e:\PythonTest\CH14\test.py", line 3, in <module>
    print(os.path.getsize("test-CH13.py"))
  File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.2544.0_x64__qbz5n2kfra8p0\lib\genericpath.py", line 50, in getsize
    return os.stat(filename).st_size
FileNotFoundError: [WinError 2] 系統找不到指定的檔案。: 'test-CH13.py'
```

14-1：資料夾與檔案路徑

- 獲得特定工作目錄的內容os.listdir()

```
1  import os
2
3  print(os.listdir("E:/PythonTest/"))
4  print(os.listdir("."))
```

```
['.vscode', 'CH13', 'CH14', 'GuessNum.py', 'IdleTest.py', 'pythonCH1Test.py', 'pythonCH4Test.py', 'test-all.py',  
'test-CH10.py', 'test-CH11.py', 'test-CH12.py', 'test.py', 'text.txt']  
['test-CH14.py', 'test.py']
```

14-1：資料夾與檔案路徑

- 獲得特定工作目錄的內容 `os.listdir()`
 - 應用：計算特定目錄底下全部檔案的檔案大小。

```
1  import os
2
3  totalSize = 0
4  for file in os.listdir("E:/PythonTest/CH14"):
5      fileSize = os.path.getsize(file)
6      print(f"{file} size is {fileSize} byte")
7      totalSize += fileSize
8
9  print(f"全部檔案大小是 {totalSize} byte")
```

```
test-CH14.py size is 1879 byte
test.py size is 254 byte
全部檔案大小是 2133 byte
```


14-1：資料夾與檔案路徑

- 獲得特定工作目錄內容glob
 - glob模組，可用於列出特定工作目錄內容（不含子目錄）
 - 可使用萬用字元"*"、"?"、"[abc]"、...等等用法。

```
1 import glob
2
3 print("列出所有檔案")
4 for file in glob.glob("E:/PythonTest/exercise/*."):
5     print(f"{file}")
```

列出所有檔案

```
E:/PythonTest/exercise\aqi.json
E:/PythonTest/exercise\atq9305.jpg
E:/PythonTest/exercise\ch14_20.txt
E:/PythonTest/exercise\ch14_51.txt
E:/PythonTest/exercise\csvReport.csv
E:/PythonTest/exercise\ex12_2.py
E:/PythonTest/exercise\ex12_4.py
E:/PythonTest/exercise\ex2_2.py
E:/PythonTest/exercise\ex2_4.py
E:/PythonTest/exercise\ex2_6.py
E:/PythonTest/exercise\ex2_8.py
E:/PythonTest/exercise\ex3_2.py
E:/PythonTest/exercise\ex3_4.py
E:/PythonTest/exercise\ex3_6.py
E:/PythonTest/exercise\ex3_8.py
```

14-1：資料夾與檔案路徑

```
1  import glob
2
3  print("列出特定檔案")
4  for file in glob.glob("E:/PythonTest/exercise/ex*_2.*"):
5      print(f"{file}")
6
7  print("列出特定檔案")
8  for file in glob.glob("E:/PythonTest/exercise/ex*2_2.*"):
9      print(f"{file}")
10
11 print("列出特定檔案")
12 for file in glob.glob("E:/PythonTest/exercise/ex?2_2.*"):
13     print(f"{file}")
14
15 print("列出特定檔案")
16 for file in glob.glob("E:/PythonTest/exercise/ex[123]_2.*"):
17     print(f"{file}")
```

```
列出特定檔案
E:/PythonTest/exercise/ex12_2.py
E:/PythonTest/exercise/ex2_2.py
E:/PythonTest/exercise/ex3_2.py
列出特定檔案
E:/PythonTest/exercise/ex12_2.py
E:/PythonTest/exercise/ex2_2.py
列出特定檔案
E:/PythonTest/exercise/ex12_2.py
列出特定檔案
E:/PythonTest/exercise/ex2_2.py
E:/PythonTest/exercise/ex3_2.py
```

14-1：資料夾與檔案路徑

- 遍歷目錄樹`os.walk()`

在 `os` 模組內有提供一個 `os.walk()` 方法可以讓我們遍歷目錄樹，這個方法每次執行迴圈時將傳回 3 個值：

- 1：目前工作目錄名稱(`dirName`)。
- 2：目前工作目錄底下的子目錄串列(`sub_dirNames`)。
- 3：目前工作目錄底下的檔案串列(`fileNames`)。

下列是語法格式：

```
for dirName, sub_dirNames, fileNames in os.walk(目錄路徑):
```

程式區塊

上述 `dirName`, `sub_dirNames`, `fileNames` 名稱可以自行命名，順序則不可以更改，至於目錄路徑可以使用絕對位址或相對位址，可以使用 `os.walk('.')` 代表目前工作目錄。

14-1：資料夾與檔案路徑

- 遍歷目錄樹os.walk()

```
1 import os
2
3 for dirName, subDirName, fileName in os.walk("E:/PythonTest/exercise/"):
4     print(f"目錄名稱{dirName}")
5     print(f"子目錄名稱{subDirName}")
6     print(f"檔案名稱{fileName}\n")
7
```

```
目錄名稱E:/PythonTest/exercise/
子目錄名稱['abc', '__pycache__']
檔案名稱['aqi.json', 'atq9305.jpg', 'ch14_20.txt', 'ch14_51.txt', 'csvReport.csv', 'ex12_2.py', 'ex12_4.py', 'ex2_2.py', 'ex2_4.py', 'ex2_6.py', 'ex2_8.py', 'ex3_2.py', 'ex3_4.py', 'ex3_6.py', 'ex3_8.py']

目錄名稱E:/PythonTest/exercise/abc
子目錄名稱[]
檔案名稱['text.py']

目錄名稱E:/PythonTest/exercise/__pycache__
子目錄名稱[]
檔案名稱['makefood.cpython-38.pyc', 'MyMath.cpython-38.pyc']
```

14-1：資料夾與檔案路徑

- Unix/Linux/Mac系統 - 變更檔案使用權限與擁有權
 - chmod(file, mode)
 - import stat

mode	說明
stat.S_IXOTH	其他使用者有執行權限 0o001
stat.S_IWOTH	其他使用者有寫入權限 0o002
stat.S_IROTH	其他使用者有讀取權限 0o004
stat.S_IRWXO	其他使用者有全部權限 0o007
stat.S_IXGRP	群組使用者有執行權限 0o010
stat.S_IWGRP	群組使用者有寫入權限 0o020
stat.S_IRGRP	群組使用者有讀取權限 0o040
stat.S_IRWXG	群組使用者有全部權限 0o070

mode	說明
stat.S_IXUSR	擁有者有執行權限 0o100
stat.S_IWUSR	擁有者有寫入權限 0o200
stat.S_IRUSR	擁有者有讀取權限 0o400
stat.S_IRWXU	擁有者有全部權限 0o700
stat.S_ISVTX	目錄內的文件只有使用者才可以更改或刪除0o1000
stat.S_ISGID	群組擁有者可以執行 0o2000
stat.S_ISUID	文件擁有者可以執行 0o4000
stat.S_IREAD	Windows 只讀取
stat.S_WRITE	Windows 可寫入

14-1：資料夾與檔案路徑

- Unix/Linux/Mac系統 -變更檔案擁有權
 - `chown(file, uid, gid)`: 將 `file` 的擁有者從 `uid` 改為 `gid`

14-2：讀取檔案

- 讀取整個檔案 `read()`
 - 讀取檔案需要先將檔案開啟，可使用 `open()` 函數（在4-3有教）。
 - 檔案開啟後，可用 `read()` 函數，將檔案內容全部讀出來，存入字串變數內。

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3  fileObj = open(fileName)
4  strContent = fileObj.read()
5  fileObj.close()
6  print(strContent)
```

```
明志工專
台北工專
我愛明志工專
```

```
1 明志工專
2 台北工專
3 我愛明志工專
4
```

14-2：讀取檔案

- with 關鍵字:

with open(欲開啟的檔案) as 檔案物件:

相關系列指令

- 可以不必在程式中關閉檔案

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3  #fileObj = open(fileName)
4  #strContent = fileObj.read()
5  #fileObj.close()
6  #print(strContent)
7  with open(fileName) as fileObj:
8      strContent = fileObj.read()
9      print(strContent)
```

明志工專
台北工專
我愛明志工專

- 最後一行的空白，可以用字串的方法 rstrip() 將空白刪除

```
7  with open(fileName) as fileObj:
8      strContent = fileObj.read()
9      print(strContent.rstrip())
```

明志工專
台北工專
我愛明志工專

14-2：讀取檔案

- 逐行讀取檔案內容：

for `line` in `file_Obj`: # `line`和`file_Obj`可以自行取名，`file_Obj`是檔案物件

迴圈相關系列指令

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3  with open(fileName) as fileObj:
4      for line in fileObj:
5          print(line)
```

明志工專
台北工專
我愛明志工專

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3  with open(fileName) as fileObj:
4      for line in fileObj:
5          print(line.rstrip())
```

明志工專
台北工專
我愛明志工專

14-2：讀取檔案

- 逐行讀取檔案內容：
 - 使用 `readlines()` 方法。
 - 將完整檔案內容逐行讀出，並且存成串列。

```
1 fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3 with open(fileName) as fileObj:
4     listContent = fileObj.readlines()
5
6 print(listContent)
```

```
['明志工專\n', '台北工專\n', '我愛明志工專\n']
```

```
1 fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3 with open(fileName) as fileObj:
4     listContent = fileObj.readlines()
5
6 for line in listContent:
7     print(line.rstrip())
```

```
明志工專
台北工專
我愛明志工專
```

14-2：讀取檔案

- 數據組合
 - 利用 python 提供的功能，將讀出來的字串重新整理，組合成希望的形式。

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3  with open(fileName) as fileObj:
4      listContent = fileObj.readlines()
5
6  strAll = ""
7  for line in listContent:
8      strAll += line.rstrip()
9
10 print(strAll)
```

明志工專台北工專我愛明志工專

14-2：讀取檔案

- 分批讀取檔案資料
 - 如果檔案很大，我們可以將檔案分批讀取進來。

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_51.txt"
2
3  nChunkSize = 100
4  strAll = ""
5  with open(fileName) as fileObj:
6      while True:
7          txt = fileObj.read(nChunkSize)
8          if not txt:
9              break
10         strAll += txt
11
12 print(strAll)
```

```
Are you sleeping, are you sleeping, Brother John, Brother John?
Morning bells are ringing, morning bells are ringing.
Ding ding dong, Ding ding dong.
```

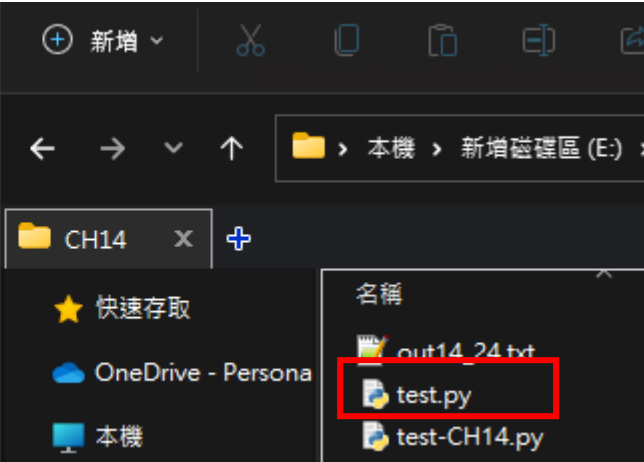
14-3：寫入檔案

- 將執行結果寫入空的文件內
 - 開啟檔案預設模式為 `mode = "r"` 讀取檔案模式。
 - 若是要開啟供寫入的檔案，開啟模式 `mode = "w"`。
 - 若是開啟的檔案要可以讀取或寫入，則可使用 `mode = "r+"`。
 - 若開啟的檔案不存在，則 `open()` 會建立新檔案，若所開啟的檔案已經存在，則檔案內會被清空。

14-3：寫入檔案

- 將執行結果寫入空的文件內
 - 輸出資料到檔案可使用 `write()` 方法。
 - `len = 檔案物件.write(欲輸出資料)` # 可將資料輸出到檔案物件
 - 其中 `len` 是 `write` 方法回傳輸出資料的長度。

```
test.py > ...
1  fileName = "out14_24.txt"
2
3  str = "I love Python"
4  with open(fileName, "w") as fileObj:
5      fileObj.write(str)
6
7
8  print()
9
```



```
1  fileName = "out14_24.txt"
2
3  str = "I love Python"
4  with open(fileName, "w") as fileObj:
5      print(fileObj.write(str))
```

14-3：寫入檔案

- 寫入數值資料
 - 不能直接輸出數值資料。

```
1 fileName = "out14_26.txt"
2
3 str = "I love Python"
4 x = 100
5 with open(fileName, "w") as fileObj:
6     fileObj.write(str)
7     fileObj.write(x)
```

```
Traceback (most recent call last):
  File "e:\PythonTest\CH14\test.py", line 7, in <module>
    fileObj.write(x)
TypeError: write() argument must be str, not int
```

out14_26.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢

I love Python

- 將資料轉成字串再輸出。

```
1 fileName = "out14_26.txt"
2
3 str1 = "I love Python"
4 x = 100
5 with open(fileName, "w") as fileObj:
6     fileObj.write(str1)
7     fileObj.write(str(x))
```

out14_26.txt - 記事本

檔案(F) 編輯(E) 格式(O) :

I love Python100

14-3：寫入檔案

- 輸出多行資料的實例

```
1  fileName = "out14_26.txt"
2
3  str1 = "I love Python"
4  x = 100
5  with open(fileName, "w") as fileObj:
6      fileObj.write(str1+"\n")
7      fileObj.write(str(x)+"\n")
```



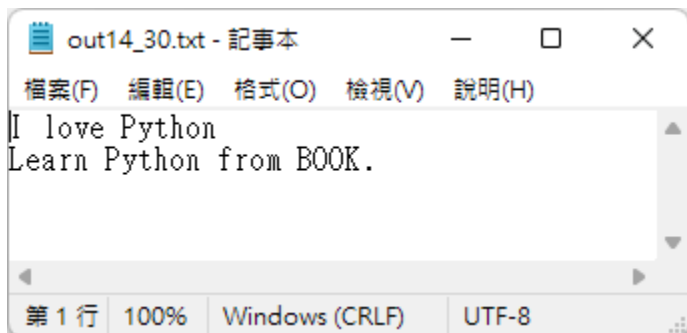
out14_26.txt - 記事
檔案(F) 編輯(E) 格式
I love Python
100

14-3：寫入檔案

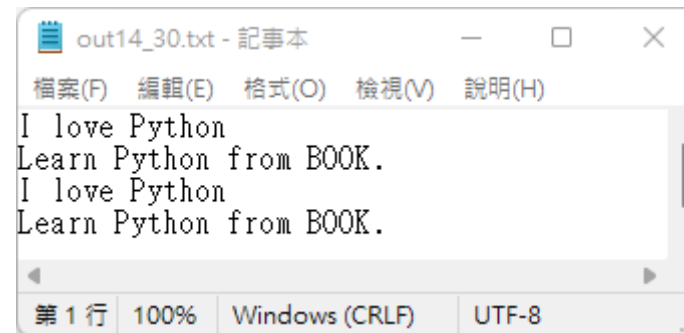
- 附加文件內容：
 - 以open()開啟時，使用參數mode="a"或是用"a"，其實a是append的縮寫。

```
1  fileName = "out14_30.txt"
2
3  str1 = "I love Python"
4  str2 = "Learn Python from BOOK."
5  with open(fileName, "a") as fileObj:
6      fileObj.write(str1+"\n")
7      fileObj.write(str2+"\n")
```

- 第一次執行結果

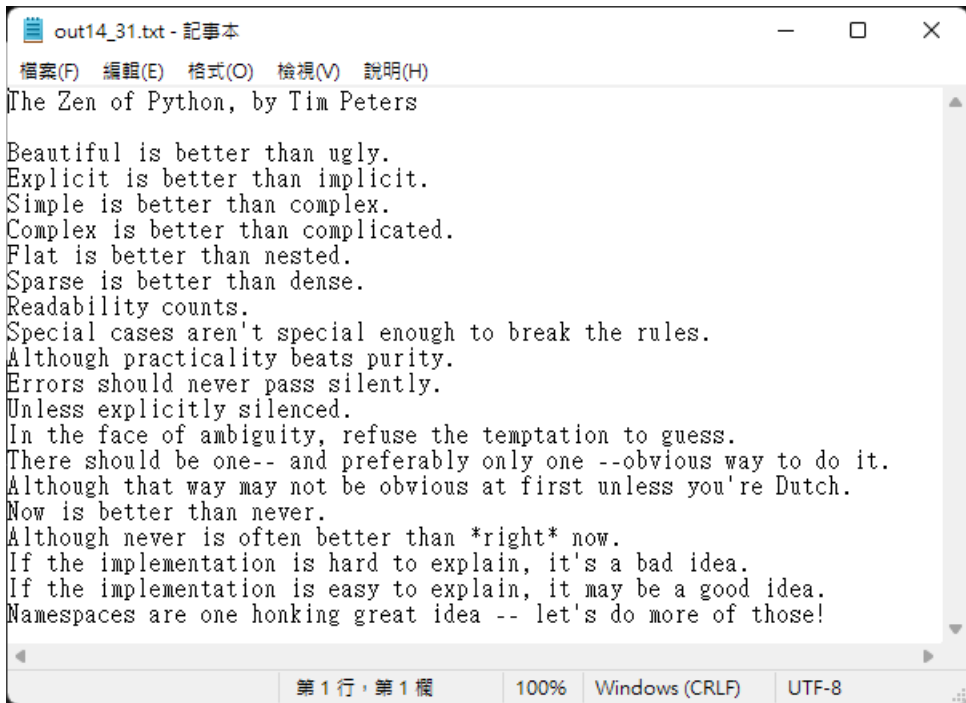


- 第二次執行結果



14-3：寫入檔案

- 檔案很大時的分段寫入



```
out14_31.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
The Zen of Python, by Tim Peters
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

2022/11/16

```
1  fileName = "out14_31.txt"
2
3  zenofPython = '''The Zen of Python, by Tim Peters
4
5  Beautiful is better than ugly.
6  Explicit is better than implicit.
7  Simple is better than complex.
8  Complex is better than complicated.
9  Flat is better than nested.
10 Sparse is better than dense.
11 Readability counts.
12 Special cases aren't special enough to break the rules.
13 Although practicality beats purity.
```





```
25 size = len(zenofPython)
26 nOffset = 0
27 nChunkSize = 100
28 with open(fileName, "w") as fileObj:
29     while True:
30         if nOffset > size:
31             break
32         print(fileObj.write(zenofPython[nOffset:nOffset+nChunkSize]))
33         nOffset += nChunkSize
```

14-4：讀取和寫入二進位檔案

- 拷貝二進位檔案
 - 一般圖檔、語音檔...等皆是二進位檔案，如果要開啟二進位檔案在open()檔案時需要使用"rb"，要寫入二進位檔案在open()檔案時需要使用"wb"。

```
1 fileName1 = "Lenna.jpg"
2 fileName2 = "LennaCopy.jpg"
3 with open(fileName1, "rb") as fileObj1:
4     srcData = fileObj1.read()
5     with open(fileName2, "wb") as fileObj2:
6         fileObj2.write(srcData)
```

名稱	修改日期	類型	大小
 Lenna.jpg	2021/11/22 下午 11:04	JPG 影像	157 KB
 LennaCopy.jpg	2021/11/22 下午 11:06	JPG 影像	157 KB



14-4：讀取和寫入二進位檔案

- 在不同檔案位置讀取二進位檔案：
 - 在二進位檔案中，Python 可透過 `tell()` 與 `seek()` 控制檔案指針的讀寫位置。
 - `tell()` 可傳回檔案指針目前的位置(從檔案開頭算起)，以 `byte` 為單位。
 - `seek()` 可讓檔案指針跳到指定位置，語法如下：
 - `offsetValue = seek(offset, origin)`
 - `origin`是0(預設)，讀寫指針移至開頭算起的第`offset`的`byte`位置。
 - `origin`是1，讀寫指針移至目前位置算起的第`offset`的`byte`位置。
 - `origin`是2，讀寫指針移至相對結尾的第`offset`的`byte`位置。

14-4：讀取和寫入二進位檔案

```
1  fileName = "Lenna.jpg"
2
3  with open(fileName, "rb") as fileObj:
4      print(f"目前位移: {fileObj.tell()}")
5      data = fileObj.read(30)
6      print(f"目前位移: {fileObj.tell()}")
7      fileObj.seek(100)
8      print(f"目前位移: {fileObj.tell()}")
9      data = fileObj.read(50)
10     print(f"目前位移: {fileObj.tell()}")
11     fileObj.seek(100, 1)
12     print(f"目前位移: {fileObj.tell()}")
```

目前位移: 0
目前位移: 30
目前位移: 100
目前位移: 150
目前位移: 250

14-5 : shutil模組

- 提供一些方法讓我們在 Python 的程式中執行檔案或目錄的複製、刪除、更動位置或更改名稱。
- import shutil

```
1  import shutil
2
3  shutil.copy("test.py", "testCopy.py")
4  shutil.copy("test.py", "D:\\PythonTest\\")
5  shutil.copy("E:\\test.py", "D:\\PythonTest\\")
```

14-5 : shutil模組

- 目錄的複製copytree()

```
1  import shutil
2
3  #目錄 test 複製成 testCopy
4  shutil.copytree("test", "testCopy")
5
6  #目錄 test 複製到 D:\PythonTest 中叫 testNew
7  shutil.copytree("test", "D:\\PythonTest\\testNew")
8
9  #目錄 E 槽的 test 複製到 D:\PythonTest 中叫 testNew
10 shutil.copytree("E:\\test", "D:\\PythonTest\\testNew")
```

14-5 : shutil模組

- 檔案的移動或更名move()
 - shutil.move(source, destination)

```
1  import shutil
2
3  shutil.move("test.py", "../test.py")
4
5  shutil.move("test.py", "testNew.py")
6
7  shutil.move("test.py", "../testNew.py")
```


14-5 : shutil模組

- 目錄的移動或更名move()

```
1  import shutil
2
3  shutil.move("test", "../test")
4
5  shutil.move("test", "testNew")
6
7  shutil.move("test", "../testNew")
```

14-5 : shutil模組

- 刪除底下有資料的目錄 `rmtree()`
 - `os` 模組的 `rmdir()` 只能刪除空的目錄，如果要刪除含資料檔案的目錄需使用 `rmtree()`。

```
1 import shutil
2
3 shutil.rmtree("test")
```

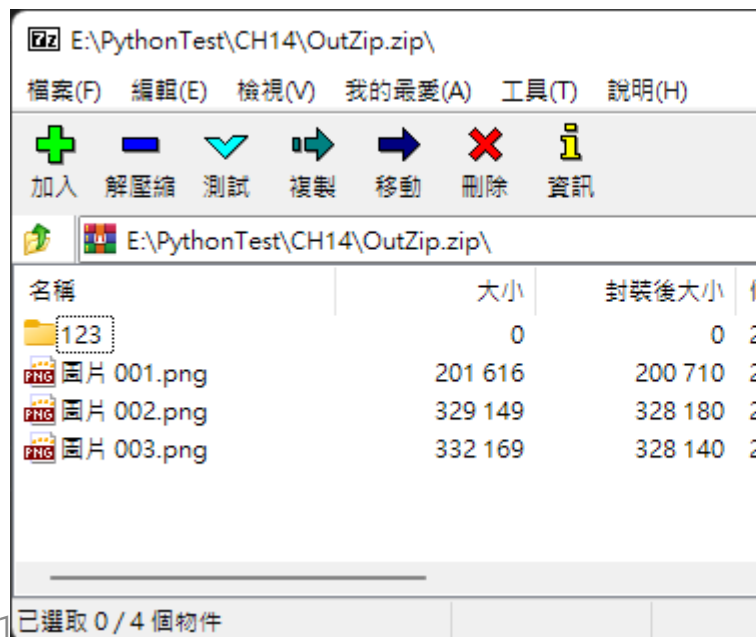
14-5-1：安全刪除檔案或目錄send2trash()

- 內建的shutil模組在刪除資料夾或檔案後就無法復原了。
- 第三方模組 send2trash 可以將檔案或資料夾放入資源回收筒。
- 需要先安裝第三方模組
 - pip install send2trash
 - 使用方法如下：

```
import send2trash  
send2trash.send2trash(檔案或資料夾名稱)
```

14-6：檔案壓縮與解壓縮zipFile

- import zipfile
- 執行檔案或目錄的壓縮
 - fileZip = zipfile.ZipFile("out.zip", "w") # out.zip是未來儲存壓縮結果
 - fileZip.write(要縮的檔案完整路徑, 檔名, 壓縮方法)



```
1 import zipfile
2 import glob, os
3
4 fileZip = zipfile.ZipFile("OutZip.zip", "w")
5 for name in glob.glob("E:\\Downloads\\*"):
6     print(os.path.basename(name))
7     fileZip.write(name, os.path.basename(name), zipfile.ZIP_DEFLATED)
8
9 fileZip.close()
```

14-6：檔案壓縮與解壓縮zipFile

- 讀取zip檔案

```
1 import zipfile
2
3 listZipInfo = zipfile.ZipFile("OutZip.zip", "r")
4 print(listZipInfo.namelist()) #列出所有的壓縮檔案
5 print()
6
7 for info in listZipInfo.infolist():
8     print(info.filename, info.file_size, info.compress_size)
```

```
['123/', '圖片 001.png', '圖片 002.png', '圖片 003.png']
```

```
123/ 0 0
```

```
圖片 001.png 201616 200710
```

```
圖片 002.png 329149 328180
```

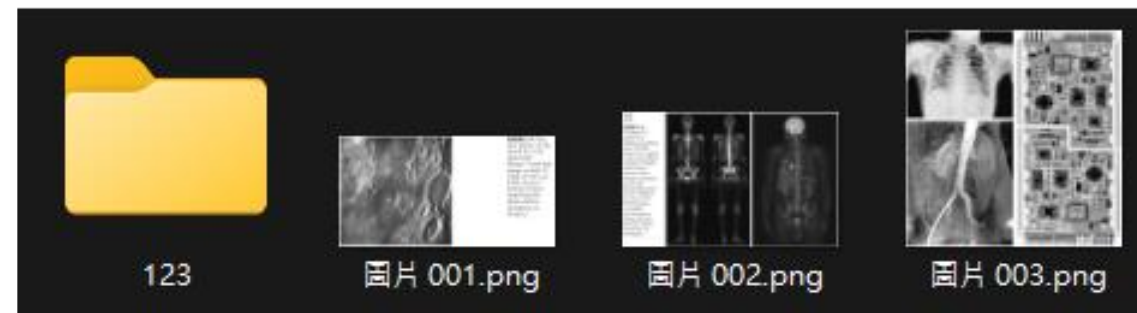
```
圖片 003.png 332169 328140
```

14-6：檔案壓縮與解壓縮zipFile

- 解壓縮zip檔案

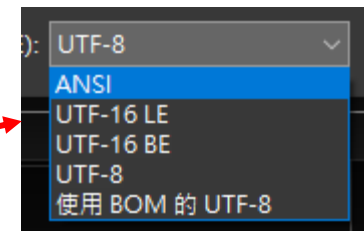
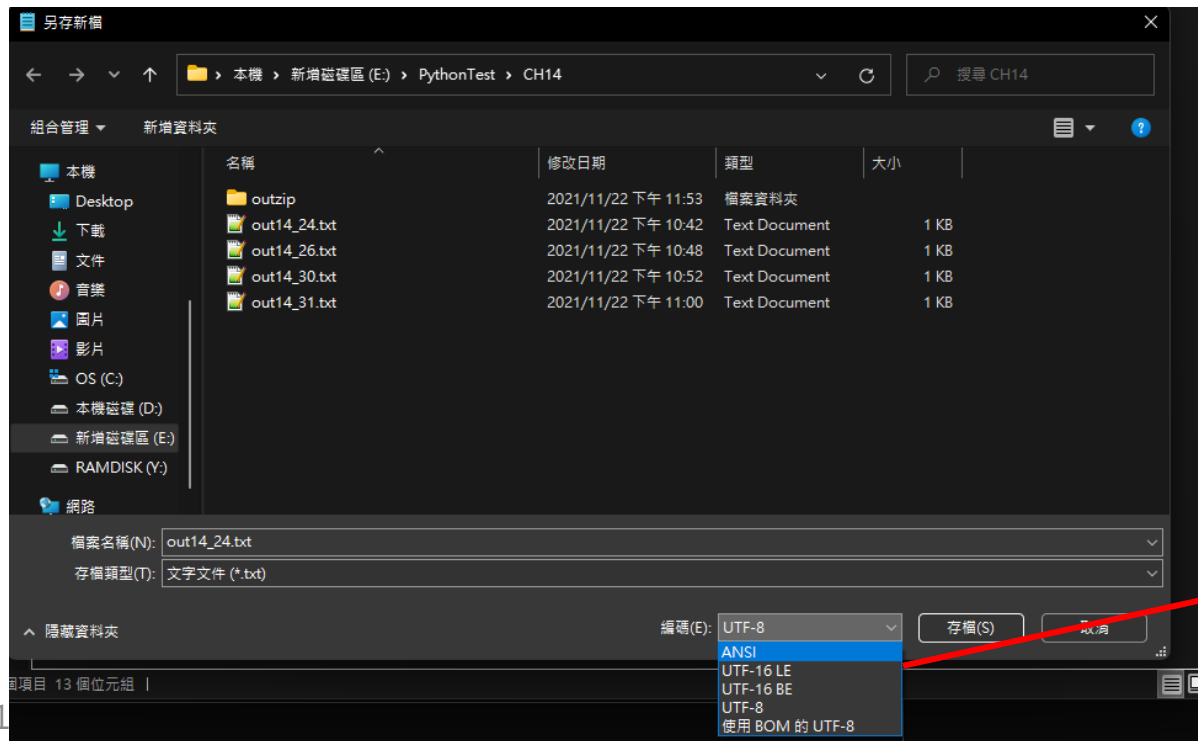
```
1 import zipfile
2
3 listZipInfo = zipfile.ZipFile("OutZip.zip", "r")
4 listZipInfo.extractall("outzip")
5 listZipInfo.close()
```

名稱	修改日期	類
outzip	2021/11/22 下午 11:53	檔
Lenna.jpg	2021/11/22 下午 11:04	JF
LennaCopy.jpg	2021/11/22 下午 11:06	JF



14-7：認識編碼格式encode

- 目前為止所談到的文字檔 (.txt) 的檔案開啟，編碼都是使用 windows 作業系統預設方式 (cp950)。
 - 常用的有 utf-8 與 cp950。
 - `file_Obj = open(file, mode="r", encoding="cp950")`



14-7：認識編碼格式encode

```
1 fileName = "E:\\PythonTest\\exercise\\ch14_20.txt"
2
3 with open(fileName, encoding = "cp950") as fileObj:
4     data = fileObj.read()
5
6 print(data)
```

明志工專
台北工專
我愛明志工專

14-7：認識編碼格式encode

- utf-8: 英文全名是8-bit Unicode Transformation Format

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20-utf8.txt"
2
3  with open(fileName, encoding = "cp950") as fileObj:
4      data = fileObj.read()
5
6  print(data)
```

```
Traceback (most recent call last):
  File "e:\\PythonTest\\CH14\\test.py", line 4, in <module>
    data = fileObj.read()
UnicodeDecodeError: 'cp950' codec can't decode byte 0xe6 in position 0: illegal multibyte sequence
```

```
3  with open(fileName, encoding = "utf-8") as fileObj:
4      data = fileObj.read()
```

明志工專
台北工專
我愛明志工專

14-7：認識編碼格式encode

- 認識utf-8編碼的BOM

- 使用中文Windows作業系統的記事本以utf-8執行編碼時，作業系統會在文件前端增加位元組順序記號(Byte Order Mark, 簡稱BOM)，俗稱文件前端代碼，主要功能是判斷文字以Unicode表示時，位元組的排序方式。

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20-utf8.txt"
2
3  with open(fileName, encoding = "utf-8") as fileObj:
4      data = fileObj.readlines()
5
6  print(data)
```

```
['\ufeff明志工專\n', '台北工專\n', '我愛明志工專\n']
```

14-7：認識編碼格式encode

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20-utf8.txt"
2
3  with open(fileName, encoding = "utf-8-sig") as fileObj:
4      data = fileObj.readlines()
5
6  print(data)
```

```
['明志工專\n', '台北工專\n', '我愛明志工專\n']
```

```
1  fileName = "E:\\PythonTest\\exercise\\ch14_20-utf8-noBOM.txt"
2
3  with open(fileName, encoding = "utf-8") as fileObj:
4      data = fileObj.readlines()
5
6  print(data)
```

14-8：剪貼簿的應用

- pip install pyperclip
- import pyperclip
- 安裝完成後就可以使用下列2個方法：
- copy()：可將串列資料拷貝至剪貼簿。
- paste()：將剪貼簿資料拷貝回字串變數

```
1  import pyperclip
2
3  pyperclip.copy("NCKU-Miin Wu SoC")
4  string = pyperclip.paste()
5  print(string)
```

NCKU-Miin Wu SoC