Python 程式設計：檔案輸入輸出與目錄管理

目錄

1. 絕對路徑 vs. 相對路徑

絕對路徑 (Absolute Path)：從檔案系統的「根目錄」開始的完整路徑。就像一張完整的地址，無論您從哪裡出發，都能準確找到目標。

例如：D:\github\python-tutor\Python 程式設計：檔案輸入輸出與目錄管理\test.py

相對路徑 (Relative Path)：從「目前所在的工作目錄」開始的路徑。就像您告訴朋友「隔壁那家店」，這個指令只有在您們倆都站在同一個地方時才有意義。

. 代表當前目錄。

.. 代表上一層目錄。

例如：Python 程式設計：檔案輸入輸出與目錄管理\test.py

2. OS 模組

2.1 os.getcwd(): 取得目前工作目錄 (Get Current Working Directory)

這個函式會告訴您，Python 目前「站」在哪個資料夾裡。

```
import os
```

```
print(os.getcwd()) # 印出目前的工作目錄
```

2.2 os.listdir(路徑): 獲得特定目錄的內容

這個函式會回傳一個串列 (list)，裡面包含了指定路徑下所有檔案和資料夾的名稱。

```
import os

# 列出指定路徑下的所有內容
print(os.listdir("D:\github\python-tutor\Python 程式設計：檔案輸入輸出與目錄管理"))
print(os.listdir(".")) # 也可以使用萬用字元 '*'，代表列出當前目錄的所有內容
```

2.3 os.walk(路徑): 深度遍歷目錄樹

os.walk() 是一個非常強大的工具，它可以深入一個資料夾，然後走遍裡面所有的子資料夾，並告訴您每個資料夾的內容。

```
import os

folder = 'D:/github/python-tutor/Python 程式設計：檔案輸入輸出與目錄管理'

# walk() 會為每一個它走到的資料夾回傳三個值：
# dirName: 目前資料夾的路徑
# subDirNames: 一個包含 dirName 底下所有「子資料夾名稱」的串列
# fileNames: 一個包含 dirName 底下所有「檔案名稱」的串列
for dirName, subDirNames, fileNames in os.walk(folder):
    print("目前目錄名稱 : ", dirName)
    print("此目錄的子目錄 : ", subDirNames)
    print("此目錄的檔案 : ", fileNames)
    print("-" * 20)
```

2.4 檢查路徑狀態：os.path.exists(路徑) / isdir(路徑) / isfile(路徑)

在對檔案或資料夾進行操作（如刪除、讀寫）前，先檢查它是否存在是一個非常重要的好習慣。

- exists(): 路徑是否存在？
- isdir(): 路徑是否為一個資料夾？
- isfile(): 路徑是否為一個檔案？
- os.path.getsize(路徑): 獲得檔案大小

2.5 目錄與檔案的建立、刪除、更名

- os.mkdir(路徑): 建立一個新資料夾。
- os.rmdir(路徑): 刪除一個空的資料夾。
- os.remove(路徑): 刪除一個檔案。
- os.rename(舊名稱, 新名稱): 重新命名檔案或資料夾。
- os.chdir(路徑): 變更目前工作目錄 (Change Directory)。

```
import os

# 要建立的資料夾名稱
mydir = 'test'

# 1. 檢查資料夾是否存在
if not os.path.exists(mydir):
    os.mkdir(mydir) # 如果不存在，就建立它
    print(f"資料夾 '{mydir}' 建立成功")
else:
    print(f"資料夾 '{mydir}' 已經存在")

# 2. 刪除資料夾
if os.path.exists(mydir):
    # 再次檢查，如果存在就刪除它
    os.rmdir(mydir)
    print(f"資料夾 '{mydir}' 刪除成功")
else:
    print(f"資料夾 '{mydir}' 不存在，無法刪除")
```

3. 寫入檔案

3.1 寫入文字檔案 (Text Files)

```
with open(filename, mode='r', encoding='utf-8') as ...
```

這是 Python 中處理檔案的黃金標準。open() 函式會回傳一個「檔案物件」，我們可以透過它來操作檔案。with 陳述式確保在程式碼區塊結束後，檔案會被自動、安全地關閉，即使中途發生錯誤也一樣。

- mode='w'：寫入模式(write)，**如果原檔案已存在，會覆蓋**
- mode='a'：添加模式(append)，**如果原檔案已存在，會往後添加**
- encoding='utf-8' ：指定檔案的文字編碼。'utf-8' 是最通用的國際標準，推薦使用。

```
# 設定檔名
fn = '大展鴻圖.txt'
# 寫入內容
string = """
別墅裡面唱 k　水池裡面銀龍魚
我送阿叔茶具　他研墨下筆直接給我四個字
大展鴻圖大師親手提筆字
大展鴻圖搬來放在辦公室 大展鴻圖關公都點頭　　有料
"""

# 使用 'w' 模式開啟檔案，如果檔案不存在會自動建立
```

```
with open(fn, 'w', encoding='utf-8') as fObj:
    # 將 string 的內容寫入檔案
    chars_written = fObj.write(string)
    print(chars_written) # 將回傳的寫入長度印出
```

3.2 寫入二進位檔案

到目前為止，我們處理的都是文字檔案。但電腦中還有很多檔案不是由單純的文字組成，例如圖片 (.jpg, .png)、音訊 (.mp3)、影片 (.mp4)、可執行檔 (.exe) 等。這些統稱為二進位檔案，它們由原始的位元組 (bytes) 資料構成。

要處理二進位檔案，我們只需要在 open() 的模式後面加上 'b'。

- 'rb': Read Binary - 讀取二進位檔案。
- 'wb': Write Binary - 寫入二進位檔案。

3.3 範例：複製一張圖片

```
src = './小姐姐.jpg'   # 來源檔案
dest = './複製的小姐姐.jpg' # 目標檔案

# 使用 'rb' (讀取二進位) 模式開啟來源檔案
with open(src, 'rb') as file_rd:
    img_data = file_rd.read() # 一次性讀取來源檔案的所有位元組內容
# 使用 'wb' (寫入二進位) 模式開啟目標檔案
with open(dest, 'wb') as file_wr:
    file_wr.write(img_data) # 將剛剛讀取的位元組內容寫入新檔案

print(f"檔案 {src} 已成功複製為 {dest}")
```

4. 讀取檔案

- mode='r'：檔案為讀取模式(read)

```
# 讀取檔案的全部內容
fn = './大展鴻圖.txt'

# 使用 with open，Python 會自動處理檔案的關閉
with open(fn, 'r', encoding='utf-8') as fObj:
    data = fObj.read() # 讀取檔案的全部內容
    print(data)
```

讀取檔案內容的幾種方法

- f.read():將整個檔案內容讀取成一個單一的字串。只適用於小檔案，如果檔案太大會耗盡記憶體。
- for line in f:: 這是處理文字檔最常用、最有效率的方式。它一次只讀取一行到記憶體中，非常適合處理大檔案。
- f.readlines(): 將檔案的每一行作為一個元素，全部讀取到一個串列中。同樣地，不適用於大檔案。

```
fn = './大展鴻圖.txt'
```

```
with open(fn, 'r', encoding='utf-8') as f:
    # for 迴圈會自動逐行讀取檔案
    for line in f:
        print(line)
        # print 預設會換行，而檔案中的行本身也帶有換行符，所以會多空一行
        # 使用 print(line.strip()) 可以移除多餘的空白和換行
```

5. Shutil 模組：安全的刪除方式

os 模組提供了基本的檔案操作，但當我們需要進行更複雜的操作，例如複製整個資料夾，或刪除一個非空的資料夾時，shutil (Shell Utilities 的縮寫) 模組就派上用場了。

```
import shutil

shutil.copy(來源, 目標): 複製一個檔案。
shutil.copytree(來源, 目標): 複製一整個資料夾，包含裡面的所有檔案和子資料夾。
shutil.move(來源, 目標): 移動一個檔案或資料夾。
shutil.rmtree(資料夾路徑): 刪除一整個資料夾，即使它不是空的。
```

```
import shutil

# 刪除 dir27 資料夾以及其下的所有內容
shutil.rmtree('dir27')
```

6. send2trash：安全的刪除方式

為了避免 rmtree() 帶來的風險，我們可以安裝一個第三方模組 send2trash。它的功能不是「永久刪除」，而是將檔案或資料夾「移至資源回收筒」，讓您還有機會可以還原。

```
pip install send2trash
```

```
import send2trash

# 將指定的檔案安全地移至資源回收筒
send2trash.send2trash('data14_28.txt')
print("檔案 data14_28.txt 已被移至資源回收筒。")
```

7. zipfile 模組：壓縮與解壓縮

zipfile 是 Python 內建的模組，讓您可以輕鬆地建立、讀取和解開 ZIP 壓縮檔。

7.1 壓縮檔案或目錄

這個程式會尋找 zipdir29 資料夾中所有的 .jpg 圖片檔，並將它們全部壓縮到一個名為 out29.zip 的新檔案中。

```
# 這個程式會尋找 zipdir29 資料夾中所有的 .jpg 圖片檔，並將它們全部壓縮到一個名為 out29.zip 的新檔案中。
import zipfile
import glob # 引入 glob 模組來尋找檔案

#    這會建立一個新的 out29.zip 檔案
fileZip = zipfile.ZipFile('out29.zip', 'w')

# 使用 glob 尋找 'zipdir29/' 目錄下所有 .jpg 結尾的檔案
for name in glob.glob('zipdir29/*.jpg'):
    # 將找到的檔案寫入 zip 檔
    # os.path.basename(name) 會取得不含路徑的檔名
    # ZIP_DEFLATED 是標準的壓縮演算法
    fileZip.write(name, compress_type=zipfile.ZIP_DEFLATED)

fileZip.close() # 完成後關閉 zip 檔案
```

7.2 讀取 ZIP 檔案內容

在解壓縮之前，我們可以先查看 ZIP 檔裡面包含了哪些檔案。

- zip 物件.namelist(): 回傳一個包含所有檔案名稱的串列 (list)。
- zip 物件.infolist(): 回傳一個包含 ZipInfo 物件的串列，提供更詳細的資訊，如檔案大小、壓縮後大小等。

```
import zipfile
# 這個程式會打開 out29.zip 並列出其內容。

# 1. 使用 'r' (讀取模式) 開啟 zip 檔
fileZip = zipfile.ZipFile('out29.zip', 'r')

# 2. 使用 namelist() 列出所有檔案名稱
print(fileZip.namelist())

# 3. 使用 infolist() 取得詳細資訊並逐一印出
for info in fileZip.infolist():
    print(info.filename, info.file_size, info.compress_size)

fileZip.close()
```

7.3 解壓縮 ZIP 檔案

extractall() 方法可以將壓縮檔裡的所有內容一次性解壓縮出來。

```
import zipfile

```

```
# 開啟要解壓縮的檔案
fileUnzip = zipfile.ZipFile('out29.zip')
# 指定解壓縮的目的地資料夾 'out31'，如果 'out31' 不存在，extractall 會自動建立它
fileUnzip.extractall('out31')
fileUnzip.close() # 關閉檔案
print("檔案已成功解壓縮至 out31 資料夾。")
```

8. pyperclip 模組：存取剪貼簿

pyperclip 讓您的程式能與系統剪貼簿互動，實現複製貼上的自動化（就是您按 Ctrl+C, Ctrl+V 時使用的那個）。

```
pip install pyperclip
```

- pyperclip.copy("要複製的文字"): 將文字複製到剪貼簿。
- pyperclip.paste(): 從剪貼簿貼上文字。

```
import pyperclip

# 將文字複製到系統剪貼簿
pyperclip.copy('知識就像內褲，看不見但很重要。')

# 從系統剪貼簿讀取（貼上）內容，並存到 string 變數中
string = pyperclip.paste()

print(string)
```