3.7.1. Basics

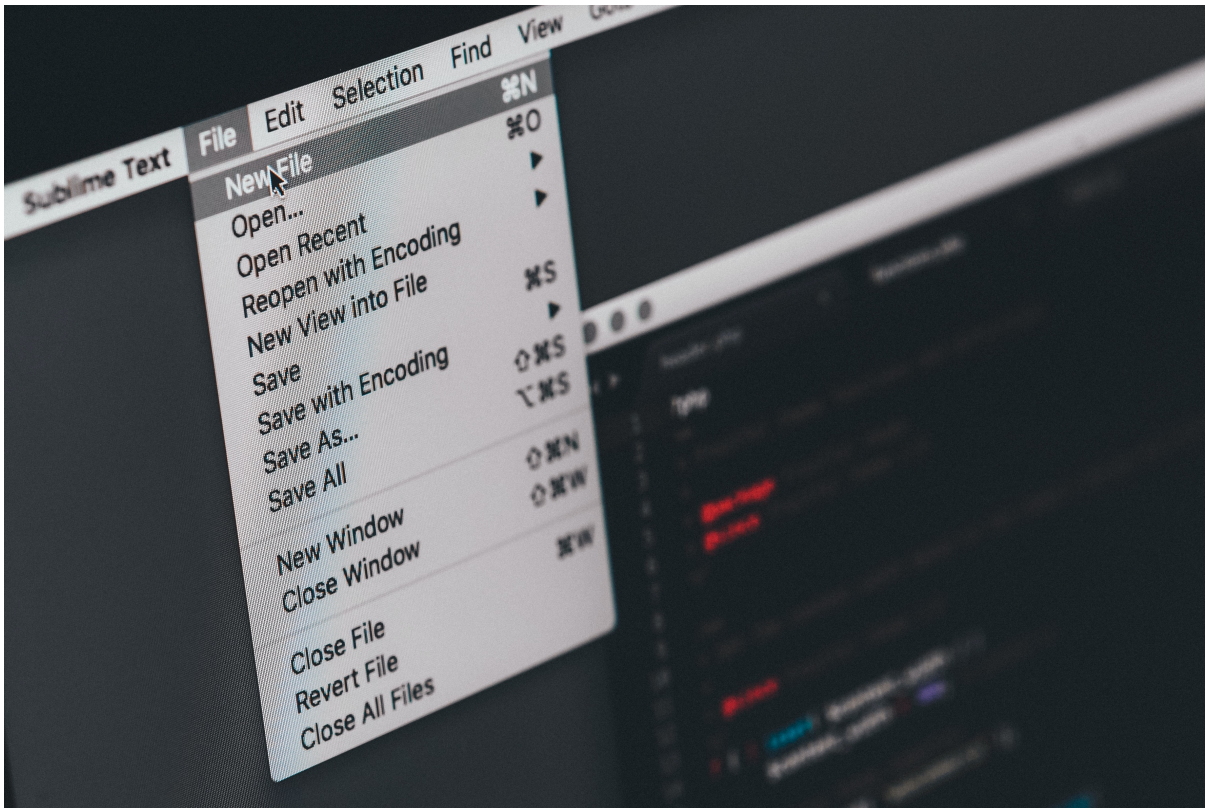


Fig. 3.7.1.1 Photo by Ilya Pavlov on Unsplash

Note

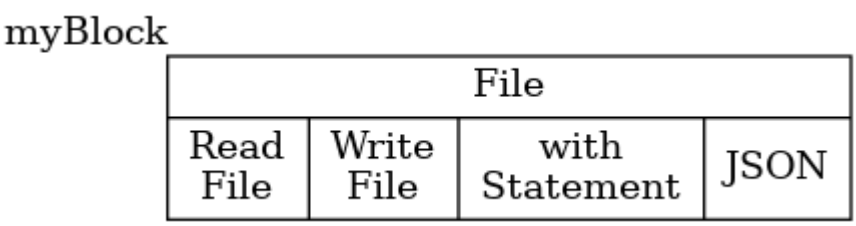
Outline

1. Introduction
    a. Syntax
    b. File Types
2. Examples
    a. Ex1: Read File Contents
    b. Ex2: Read One Line
    c. Ex3a: Read First 25 Characters
    d. Ex3b: Read Next Characters
    e. Ex4a: Write To File
    f. Ex4b: Write To Same File
    g. Ex5a: Delete File
    h. Ex5b: Check If File Existed
    i. Ex5c: Recover Deleted File
    j. Ex6a: Delete Folder
    k. Ex6b: Check If Folder Exist
    l. Ex6c: Recover Deleted Folder
    m. Ex6d: Safe Remove/Recover Folder

Note

Roadmap

1. This topic: File

> 
>

2. Course: Python 1
3. Subject: Programming
4. Field
    a. Software Engineering (SE)
    b. Computer Science and Information Engineering (CSIE)
    c. Electrical/Electronics Engineering (EE)

3.7.1.1. Introduction

1. 在Python裡面， 通常會使用open()函數來開啟電腦中已經存在的檔案， 以便能進行檔案內容的讀取、寫入或修改等動作。

3.7.1.1.1. Syntax

```
file_object = open(file_name [, access_mode][, buffering])
```

1. 如同上面的範例所示，[ ]裡面的參數都是可省略的。

    a. file_name 就是想開啟的檔案， 當然其中也包含了想要開起檔案的路徑。 如果沒有包含路徑的話， 則Python 會預設要開啟的檔案與目前在編輯的程式碼檔都位於同一個資料夾。

    b. access_mode 是想要存取的模式，存取模式的種類繁多，下面分別列表說明之。

    c. buffering 代表資料讀入的暫存空間， 0 代表沒有暫存空間， 大於或等於1 則是代表有暫存空間， 且一次可讀入多少資料量。 例如，3 就是表示一次讀入3 行的資料量。 至於-1，則表示會用預設的暫存區大小來暫存所讀入的資料內容。

3.7.1.1.2. File Types

模式	描述
r	以唯讀方式打開文件。文件游標將會放在文件的開頭。預設模式。
rb	以唯讀方式打開文件(二進制格式)。文件游標將會放在文件的開頭。 預設模式。
r+	打開一個文件用於讀寫。文件游標將會放在文件的開頭。
rb+	以二進制格式打開一個文件用於讀寫。 文件游標將會放在文件的開頭。
w	打開一個文件並且功能只限定於寫入。 如果該文件已存在則將其覆蓋。 如果該文件不存在，創建新文件。
wb	打開一個文件(以二進位格式)並且功能只限定於寫入。 如果該文件已存在則將其覆蓋。如果該文件不存在，創建新文件。
w+	打開一個文件並且能夠讀寫。如果該文件已存在則將其覆蓋。 如果該文件不存在，創建新文件。
wb+	打開一個文件(以二進位格式)並且能夠讀寫。 如果該文件已存在則將其覆蓋。如果該文件不存在，創建新文件。
a	打開一個文件並且功能限定為追加資料。 如果該文件已存在，文件游標將會放在文件的結尾。 也就是說，新的內容將會被寫入到已有內容之後。 如果該文件不存在，創建新文件進行寫入。
ab	打開一個文件(以二進位格式)並且功能限定為追加資料。 如果該文件已存在，文件游標將會放在文件的結尾。 也就是說，新的內容將會被寫入到已有內容之後。 如果該文件不存在，創建新文件進行寫入。

模式	描述
a+	打開一個文件並且能夠讀寫。 如果該文件已存在，文件游標將會放在文件的結尾。 文件打開時會是追加模式。 如果該文件不存在，創建新文件用於讀寫。
ab+	打開一個文件(以二進位格式)並且功能限定為追加資料。 如果該文件已存在，文件游標將會放在文件的結尾。 如果該文件不存在，創建新文件並且能夠讀寫。

1. 用open 開檔之後要執行的動作至少有以下三種： 讀檔，寫檔及關檔，分別以read()、write()、close() 為代表。
2. 而讀檔又有read()，readline()，readlines() 三種方法， 寫檔也是有write()、writeline()、writelines() 三種， 至於關檔就只有close() 一種。
3. read()一次可以讀取整份文件，通常會將整份文件的文字放在一個string 裡面， 對於文件中「行」的概念處理起來相當麻煩，因此在讀檔時並不常使用。
4. readline()跟readlines()的差異點在於readlines() 跟read()一樣會一次讀取整份文件。 然而，readlines()會將文件依行為單位存成list，接下來就可以使用for 迴圈進行處理。
5. readline() 比起readlines()來說就慢上許多，因為readline()一次只讀取一行。 所以，如果記憶體的空間足夠，基本上建議使用readlines() 一次性讀取完畢， 再用for迴圈逐行做進一步的處理。

3.7.1.2. Examples

3.7.1.2.1. Ex1: Read File Contents

1. Source

Listing 3.7.1.2.1.1 /src/File/OpenReadlines/__init__.py

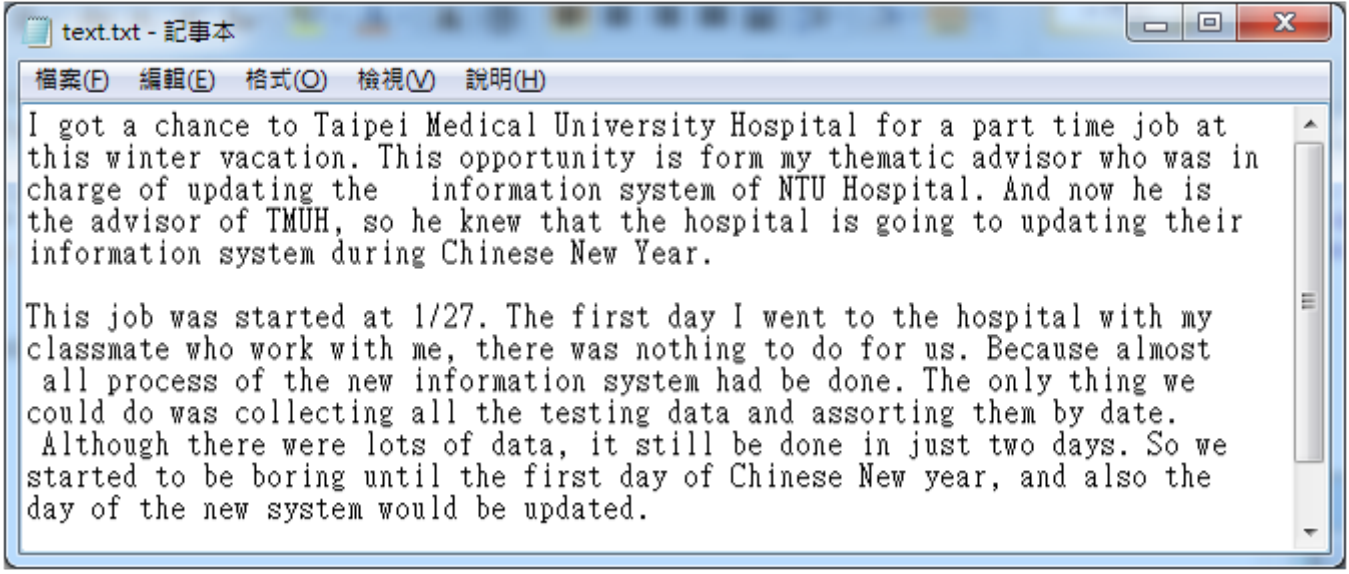
```
f = open("text.txt", "r")    # 以讀方式打開檔
for line in f.readlines():   # 依次讀取每行
    line = line.strip()      # 去掉每行頭尾空白
    print(line)
f.close()
```

2. Output

```
I got a chance to Taipei Medical University Hospital for a part time job at this winter
vacation.
This opportunity is form my thematic advisor who was in charge of updating the information
system of NTU Hospital.
And now he is the advisor of TMUH,
so he knew that the hospital is going to updating their information system during Chinese
New Year.

This job was started at 1/27.
The first day I went to the hospital with my classmate who work with me,
there was nothing to do for us.
Because almost all process of the new information system had be done.
The only thing we could do was collecting all the testing data and assorting them by date.
Although there were lots of data, it still be done in just two days.
So we started to be boring until the first day of Chinese New year,
and also the day of the new system would be updated.
```

3. 文字檔案內容



Important

a. If we meet an error to read Chinese characters as shown as below, please use the following code to skip it.

```
UnicodeDecodeError: 'cp950' codec can't decode byte 0xe5 in position 13: illegal multibyte sequence
```

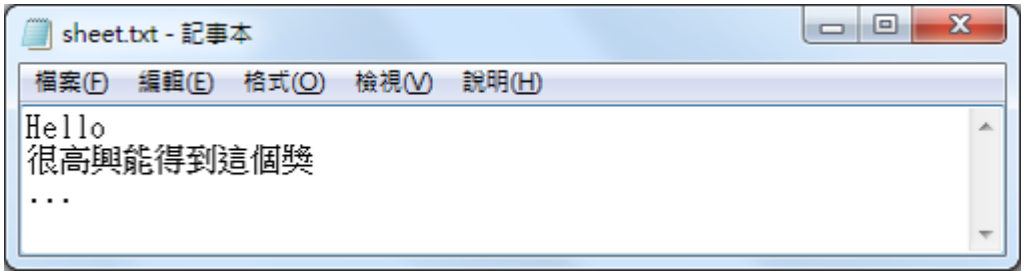
b. Solution

```
f = open("text.txt", "r", encoding="utf-8")
```

4. write() 方法可將任何字串寫入目前已經被打開的檔案。值得注意的是，Python 字串可以是二進位資料，而不僅僅是文字內容。
5. 通常，write() 方法不會自動在字串的結尾中添加分行符號（'\n'），因此如果要換行則必須自行加上分行的符號。

3.7.1.2.2. Ex2: Read One Line

1. 普羅在走上台後，從容不迫地用魔法召喚出準備好的小抄，自在地大聲朗誦。



2. Source

Listing 3.7.1.2.2.1 /src/File/OpenReadline/__init__.py

```
f = open("sheet.txt", "r")  # 以唯獨模式開啟檔案
print(f.readline())         # 單行讀入
print(f.readline())
print(f.readline())
f.close()                   # 關閉檔案
```

3. Output

```
Hello

很高興能得到這個獎

...
```

4. Note that we found that every line append one more '\n', hence it has one more new line after every line.

3.7.1.2.3. Ex3a: Read First 25 Characters

1. Source

Listing 3.7.1.2.3.1 /src/File/OpenReadFirst25Char/__init__.py

```
f = open("text.txt", "r")    # 以讀方式打開檔
print(f.read(25))
f.close()
```

2. Output

```
I got a chance to Taipei
```

3.7.1.2.4. Ex3b: Read Next Characters

1. Source

Listing 3.7.1.2.4.1 /src/File/OpenReadFirst25Char2/__init__.py

```
print('[1234567890123456789012345678901234567890]')
f = open("text.txt", "r")    # 以讀方式打開檔
print('[' + f.read(25) + ']')
print('[' + f.read(30) + ']')
print('[' + f.read(40) + ']')
f.close()
```

2. Output

```
[1234567890123456789012345678901234567890]
[中文字串測試。I got a chance to ]
[Taipei Medical University Hosp]
[ital for a part time job at this winter ]
```

3.7.1.2.5. Ex4a: Write To File

1. Source

Listing 3.7.1.2.5.1 /src/File/OpenWriteFile/__init__.py

```
# 打開一個檔
fo = open("write.txt", "w+")
fo.write("Python is a great language.\nYeah its great!!\n")
fo.close()      # 關閉打開的文件

# 執行完上面的程式碼，會在這個Python 檔案的路徑中新建一個write.txt
#===========
```

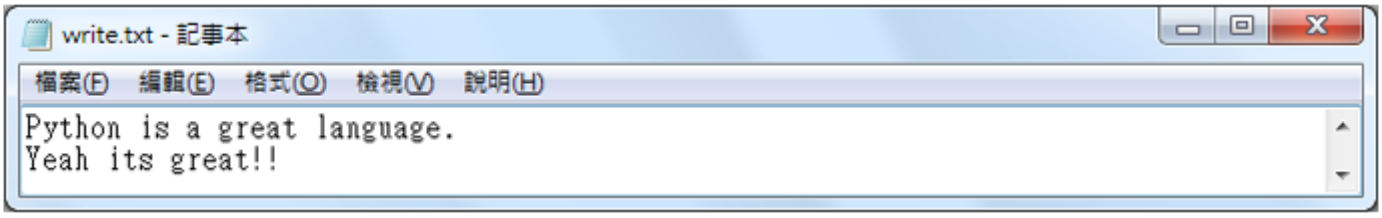
```
# 下面要再開啟這個檔，並且輸出這個檔的內容

fo = open("write.txt", "r")
for line in fo.readlines():
    line = line.strip()
    print(line)
fo.close()
```

2. Output

```
Python is a great language.
Yeah its great!!
```

3. Proof



4. 至於writeline()跟writelines()的差異乃在於writeline()是輸入完之後會換行，而writelines()則是將list當作輸入參數，以list 的資料型態將資料寫入檔案。

5. 不過值得注意的是，當list 中每個元素被寫入後，並不會自動換行。

3.7.1.2.6. Ex4b: Write To Same File

1. Source

Listing 3.7.1.2.6.1 /src/File/OpenWriteFile2/__init__.py

```
from datetime import datetime

# 打開一個檔
fo = open("write.txt", "a+")
sDateTime = datetime.now().strftime("%Y%m%d %H:%M:%S")
fo.write(sDateTime + '\n')
fo.write("Python is a great language.\nYeah its great!!\n")
fo.close()      # 關閉打開的文件

# 執行完上面的程式碼，會在這個Python 檔案的路徑中新建一個write.txt
#===========
# 下面要再開啟這個檔，並且輸出這個檔的內容

fo = open("write.txt", "r")
for line in fo.readlines():
    line = line.strip()
    print(line)
fo.close()
```

2. Output

```
20220515 14:24:25
Python is a great language.
Yeah its great!!
20220515 14:24:50
Python is a great language.
Yeah its great!!
20220515 14:25:21
```

```
Python is a great language.
Yeah its great!!
```

3.7.1.2.7. Ex5a: Delete File

1. We prepare a file which will be deleted by this Python program.
2. This file, [deleteMe.txt], is cloned at same folder for backup reason.
3. Please do not delete the backup file, [deleteMe-20220515-cph.txt], any more.
4. If we need to recover the deleted file, just need to clone the backup file.
5. Source

Listing 3.7.1.2.7.1 /src/File/DeleteFile/__init__.py

```
import os

os.remove('deleteMe.txt')
```

6. Output 1

    a. Delete [deleteMe.txt] file at first time and it is deleted.

```

```

7. Output 2

    a. We delete the same file again, then there is no such file for deletion.

    b. Hence, an error is raised and shown as below.

```
Traceback (most recent call last):
  File
"W:\_D\Data.cph\Workspace\Python3\Py7.WhoNotes.Py\src\File\DeleteFile\__init__.py",
line 3, in <module>
    os.remove('deleteMe.txt')
FileNotFoundError: [WinError 2] 系統找不到指定的檔案。: 'deleteMe.txt'
```

3.7.1.2.8. Ex5b: Check If File Exist

1. Source

Listing 3.7.1.2.8.1 /src/File/CheckIfFileExist/__init__.py

```
import os

if os.path.exists("deleteMe.txt"):
    os.remove("deleteMe.txt")
    print("The [deleteMe.txt] file is existed and deleted...")
else:
    print("The [deleteMe.txt] file is not exist...")
```

2. Output 1

a. If [deleteMe.txt] is existed, a success message is printed.

```
The [deleteMe.txt] file is existed and deleted...
```

3. Output 2

a. If [deleteMe.txt] is NOT existed, a failure message is printed.

```
The [deleteMe.txt] file is not exist...
```

3.7.1.2.9. Ex5c: Recover Deleted File

1. Source

Listing 3.7.1.2.9.1 /src/File/RecoverDeletedFile/__init__.py

```
import os
import shutil
import subprocess

if os.path.exists("deleteMe.txt"):
    os.remove("deleteMe.txt")
    print("The [deleteMe.txt] file is existed and deleted...")
else:
    print("The [deleteMe.txt] file is not exist...")
    sChoice = input('Choose one recover method (1,2-5, 6-7): ')
    # os
    if (sChoice == '1'):    # os.system()
        print("The [deleteMe.txt] file is recovered by os.system()...")
        os.system('copy deleteMe-20220515-cph.txt deleteMe.txt')

    # shutil
    elif (sChoice == '2'):  # shutil.copyfile()
        print("The [deleteMe.txt] file is recovered by shutil.copyfile()...")
        shutil.copyfile('deleteMe-20220515-cph.txt', 'deleteMe.txt')
    elif (sChoice == '3'):  # shutil.copy()
        print("The [deleteMe.txt] file is recovered by shutil.copy()...")
        shutil.copy('deleteMe-20220515-cph.txt', 'deleteMe.txt')
    elif (sChoice == '4'):  # shutil.copy2()
        print("The [deleteMe.txt] file is recovered by shutil.copy2()...")
        shutil.copy2('deleteMe-20220515-cph.txt', 'deleteMe.txt')
    elif (sChoice == '5'):  # shutil.copyfileobj()
        print("The [deleteMe.txt] file is recovered by shutil.copyfileobj()...")
        shutil.copyfileobj(open('deleteMe-20220515-cph.txt', 'rb'),
                           open('deleteMe.txt', 'wb'))

    # subprocess
    elif (sChoice == '6'):  # subprocess.call()
        print("The [deleteMe.txt] file is recovered by subprocess.call()...")
        sStatus = subprocess.call('copy deleteMe-20220515-cph.txt deleteMe.txt',
shell=True)
        print(sStatus)
    elif (sChoice == '7'):  # subprocess.check_output()
        print("The [deleteMe.txt] file is recovered by subprocess.check_output()...")
        sStatus = subprocess.check_output('copy deleteMe-20220515-cph.txt deleteMe.txt',
shell=True)
        print(sStatus)

```

```
        print("The [deleteMe.txt] file is recovered now...")
```

2. Output 1

    a. If [deleteMe.txt] is existed, a success message is printed.

```
1  The [deleteMe.txt] file is existed and deleted...
```

3. If [deleteMe.txt] is NOT existed, then we can use one of the methods to recover deleted file.

4. Output 2: Use os.system()

```
1  The [deleteMe.txt] file is not exist...
2  Choose one recover method (1,2-5, 6-7): 1
3  The [deleteMe.txt] file is recovered by os.system()...
4  複製了         1 個檔案。
5  The [deleteMe.txt] file is recovered now...
```

5. Output 3a: Use shutil.copyfile()

```
1  The [deleteMe.txt] file is not exist...
2  Choose one recover method (1,2-5, 6-7): 2
3  The [deleteMe.txt] file is recovered by shutil.copyfile()...
4  The [deleteMe.txt] file is recovered now...
```

6. Output 3b: Use shutil.copy()

```
1  The [deleteMe.txt] file is not exist...
2  Choose one recover method (1,2-5, 6-7): 3
3  The [deleteMe.txt] file is recovered by shutil.copy()...
4  The [deleteMe.txt] file is recovered now...
```

7. Output 3c: Use shutil.copy2()

```
1  The [deleteMe.txt] file is not exist...
2  Choose one recover method (1,2-5, 6-7): 4
3  The [deleteMe.txt] file is recovered by shutil.copy2()...
4  The [deleteMe.txt] file is recovered now...
```

8. Output 3d: Use shutil.copyfileobj()

```
1  The [deleteMe.txt] file is not exist...
2  Choose one recover method (1,2-5, 6-7): 5
3  The [deleteMe.txt] file is recovered by shutil.copyfileobj()...
4  The [deleteMe.txt] file is recovered now...
```

9. Output 4a: Use subprocess.call()

```
1  The [deleteMe.txt] file is not exist...
2  Choose one recover method (1,2-5, 6-7): 6
3  The [deleteMe.txt] file is recovered by subprocess.call()...
4  複製了         1 個檔案。
5  0
6  The [deleteMe.txt] file is recovered now...
```

10. Output 4b: Use subprocess.check_output()

```
The [deleteMe.txt] file is not exist...
Choose one recover method (1,2-5, 6-7): 7
The [deleteMe.txt] file is recovered by subprocess.check_output()...
b'\xbd\xc6\xbbs\xa4F         1 \xad\xd3\xc0\xc9\xae\xd7\xa1C\r\n'
The [deleteMe.txt] file is recovered now...
```

3.7.1.2.10. Ex6a: Delete Folder

1. Source

Listing 3.7.1.2.10.1 /src/File/DeleteFolder/__init__.py

```
import os

os.rmdir('deleteThisFolder')
```

2. Output

```

```

3.7.1.2.11. Ex6b: Check If Folder Exist

1. Source

Listing 3.7.1.2.11.1 /src/File/CheckIfFolderExist/__init__.py

```
import os

if os.path.exists('deleteThisFolder'):
    os.rmdir('deleteThisFolder')
    print("The [deleteThisFolder] folder is existed and deleted...")
else:
    print("The [deleteThisFolder] file is not exist...")
```

2. Output 1

    a. If [deleteMe.txt] is existed, a success message is printed.

    ```
    The [deleteThisFolder] folder is existed and deleted...
    ```

3. Output 2

    a. If [deleteMe.txt] is NOT existed, a failure message is printed.

    ```
    The [deleteThisFolder] file is not exist...
    ```

3.7.1.2.12. Ex6c: Recover Deleted Folder

1. Source

Listing 3.7.1.2.12.1 /src/File/RecoverDeletedFolder/__init__.py

```
import os
import psutil
import subprocess

if os.path.exists('deleteThisFolder'):
    try:
        os.remove('deleteThisFolder')
    except Exception as e:
        print('Delete [deleteThisFolder] folder exception...')
        print('Exception Message: ', e)
        # PermissionError: [WinError 32] The process cannot access the file
        # because it is being used by another process
        print('Try to unlock the process that is using [deleteThisFolder] folder...')
        for proc in psutil.process_iter():
            if (proc.name() == 'python.exe') or (proc.name() == 'python3.exe'):
                proc.kill()
        os.remove('deleteThisFolder')

    print('The [deleteThisFolder] folder is exist and deleted...')
else:
    print('The [deleteThisFolder] folder is not exist...')
    print('The [deleteThisFolder] folder is recovered by os.mkdir() and os.system()...')
    os.mkdir('deleteThisFolder')
    os.system('copy deleteThisFolder2 deleteThisFolder')
    print('The [deleteThisFolder] folder is recovered now...')

```

2. Output 1

    a. If [deleteThisFolder] folder is existed, a success message is printed.

    b. It might be followed by a exception message which is shown as below.

    c. However, it sometimes cannot be executed and stalled there.

    d. Hence, this kind of the method is not good for usage.

```
Delete [deleteThisFolder] folder exception...
Exception Message:  [WinError 5] 存取被拒。: 'deleteThisFolder'
Try to unlock the process that is using [deleteThisFolder] folder...
The [deleteThisFolder] file is existed and deleted...
```

3. If [deleteThisFolder] folder is NOT existed, then we can use the following method to recover deleted folder.

4. Output 2: Use os.mkdir() and os.system()

```
The [deleteThisFolder] folder is not exist...
The [deleteThisFolder] folder is recovered by os.mkdir() and os.system()...
deleteThisFolder2\deleteMe-20220515-cph.txt
deleteThisFolder2\deleteMe.txt
複製了         2 個檔案。
The [deleteThisFolder] folder is recovered now...
```

3.7.1.2.13. Ex6d: Safe Remove/Recover Folder

1. Source

Listing 3.7.1.2.13.1 /src/File/RecoverDeletedFolder2/__init__.py

```
import os
import psutil
import subprocess

def deleteFolder():
    prog = subprocess.Popen(['runas',
                             '/user:Administrator',
                             'rmdir deleteThisFolder'],
                             stdin=subprocess.PIPE)
    sPwd = getAdminPassword()
    print(sPwd)
    #prog.stdin.write(sPwd.encode(encoding = 'UTF-8'))
    sData = prog.communicate(input=sPwd.encode(encoding = 'UTF-8'))[0]
    print(sData)
    return

def getAdminPassword():
    f = open('password.txt', 'r')
    sPwd = f.readline()
    f.close()
    return(sPwd)

if os.path.exists('deleteThisFolder'):
    try:
        print('The [deleteThisFolder] folder is removed by subprocess.call...')
        deleteFolder()
    except Exception as e:
        print('Delete [deleteThisFolder] folder exception...')
        print('Exception Message: ', e)
        # PermissionError: [WinError 32] The process cannot access the file
        # because it is being used by another process
        print('Try to unlock the process that is using [deleteThisFolder] folder...')
        for proc in psutil.process_iter():
            if (proc.name() == 'python.exe') or (proc.name() == 'python3.exe'):
                proc.kill()
        deleteFolder()

    print('The [deleteThisFolder] folder is exist and deleted...')
else:
    print('The [deleteThisFolder] folder is not exist...')
    print('The [deleteThisFolder] folder is recovered by os.mkdir() and os.system()...')
    os.mkdir('deleteThisFolder')
    os.system('copy deleteThisFolder2 deleteThisFolder')
    print('The [deleteThisFolder] folder is recovered now...')

```

2. Output 1

a. If [deleteThisFolder] folder is existed, a success message is printed.

b. It might be followed by a exception message which is shown as below.

c. However, it sometimes cannot be executed and stalled there.

d. Hence, this kind of the method is not good for usage.

```
Delete [deleteThisFolder] folder exception...
Exception Message:  [WinError 5] 存取被拒。: 'deleteThisFolder'
Try to unlock the process that is using [deleteThisFolder] folder...
The [deleteThisFolder] file is existed and deleted...
```

3. If [deleteThisFolder] folder is NOT existed, then we can use the following method to recover deleted folder.

4. Output 2: Use os.mkdir() and os.system()

```
The [deleteThisFolder] folder is not exist...
The [deleteThisFolder] folder is recovered by os.mkdir() and os.system()...
deleteThisFolder2\deleteMe-20220515-cph.txt
deleteThisFolder2\deleteMe.txt
複製了         2 個檔案。
The [deleteThisFolder] folder is recovered now...
```

Important

Summary

1. Here is a summary of built-in Python functions for working with files.

Build-in	Description
open()	Opens a file and returns a file object. Modes include 'r' for read, 'w' for write, 'a' for append.
close()	Closes an open file. Flushes buffered data to disk.
read()	Reads contents of a file. Can take a max number of bytes to read as parameter.
readline()	Reads a line of text from the file.
readlines()	Reads all lines of a file into a list.
write()	Writes text data to a file. Accepts a string.
with	Can include 2+ resources for processing.
seek()	Changes stream position in a file. Offset 0 starts at beginning.
tell()	Returns current position in a file.
truncate()	Resizes a file to given size.
flush()	Flushes buffered data to disk.

2. These built-in functions allow working with files in Python.
3. The open() function opens a file and returns a file object, which then contains methods like read(), write(), seek(), close() etc. to manipulate the file. Using with open() is best practice to ensure files are closed properly.

Note

1. Start: 20170719
2. System Environment:

Listing 3.7.1.2.13.2 requirements.txt

```
sphinx==7.1.2                           # Sphinx
graphviz>=0.20.1                        # Graphviz
sphinxbootstrap4theme>=0.6.0            # Theme: Bootstrap
sphinx-material>=0.0.35                 # Theme: Material
sphinxcontrib-plantuml>=0.25            # PlantUML
sphinxcontrib.bibtex>=2.5.0             # Bibliography
sphinx-autorun>=1.1.1                   # ExecCode: pycon
sphinx-execute-code-python3>=0.3        # ExecCode
btd.sphinx.inheritance-diagram>=2.3.1   # Diagram
sphinx-copybutton>=0.5.1                # Copy button
sphinx_code_tabs>=0.5.3                 # Tabs
sphinx-immaterial>=0.11.3               # Tabs

#-----------------------------------------------------
#-- Library Upgrade Error by Library Itself
#   >> It needs to fix by library owner
#   >> After fixed, we need to try it later
#-----------------------------------------------------
pydantic==1.10.10                       # 2.0: sphinx compiler error, 20230701

#-----------------------------------------------------
#-- Minor Extension
#-----------------------------------------------------
sphinxcontrib.httpdomain>=1.8.1         # HTTP API

#sphinxcontrib-blockdiag>=3.0.0         # Diagram: block
#sphinxcontrib-actdiag>=3.0.0           # Diagram: activity
#sphinxcontrib-nwdiag>=2.0.0            # Diagram: network
#sphinxcontrib-seqdiag>=3.0.0           # Diagram: sequence

#-----------------------------------------------------
#-- Still Wait For Upgrading Version
#-----------------------------------------------------

#-----------------------------------------------------
#-- Still Under Testing
#-----------------------------------------------------
#numpy>=1.24.2                          # Figure: numpy

#-----------------------------------------------------
#-- NOT Workable
#-----------------------------------------------------
#sphinxcontrib.jsdemo==0.1.4    # ExecCode: Need replace add_js_file()
#jupyter-sphinx==0.4.0          # ExecCode: Need gcc compiler
#sphinxcontrib.slide==1.0.0     # Slide: Slideshare
#hieroglyph==2.1.0              # Slide: make slides
#matplotlib>=3.7.1              # Plot: Need Python >= v3.8
#manim==0.17.2                  # Diagram: scipy, numpy need gcc
#sphinx_diagrams==0.4.0         # Diagram: Need GKE access
#sphinx-tabs>=3.4.1                     # Tabs: Conflict w/ sphinx-material
```