LGT3109
Introduction to Coding for Business with Python
(week 9)

Xiaoyu Wang

Dept. of LMS, The HK PolyU

Summary of Week 8

Dictionary

- Dictionary vs list: Index and keys.
- Dictionary basics: key-value pair, in operator, len() function, pop method, and del operator.
- Dictionary for counting: use get() function to handle couting.
- Traversing dictionary: use keys(), values(), and items().

Summary of Week 8

Tuples

- Tuples
  - ➢like list, but immutable.
  - ➢Used as items or keys for dictionaries
  - ➢Assignment: (x, y)=(‘a’, ‘b’)
- Comparing tuples: sort by keys or values

Task Automation: Organizing Files

- File and file paths
- Modules to operate files and folders
- Compressing files with the zipfile module

Motivation Case-Received Files

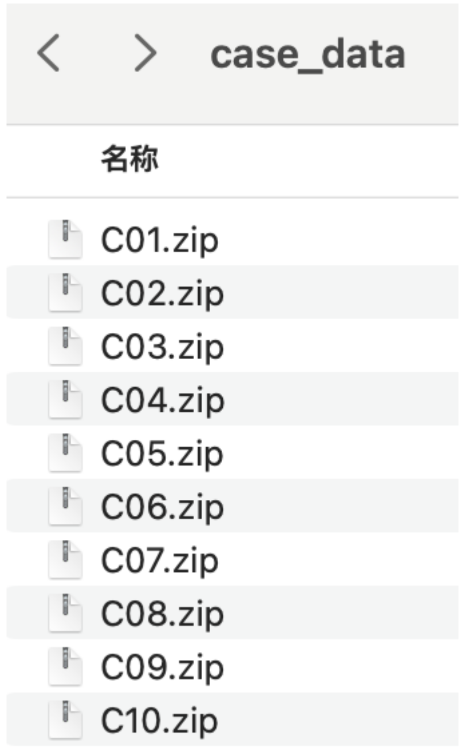
- Foxconn receives zip files from carriers.

- Each carrier's file name is {carrier_id}.zip.

- Each zip file has two files of shipping rates:

➢intra_asia.txt: shipping rates for Asia lanes.

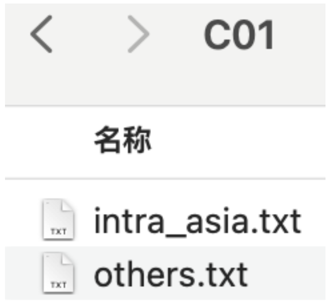
➢others.txt: shipping rates for other lanes.



case_data

名称

C01.zip
C02.zip
C03.zip
C04.zip
C05.zip
C06.zip
C07.zip
C08.zip
C09.zip
C10.zip



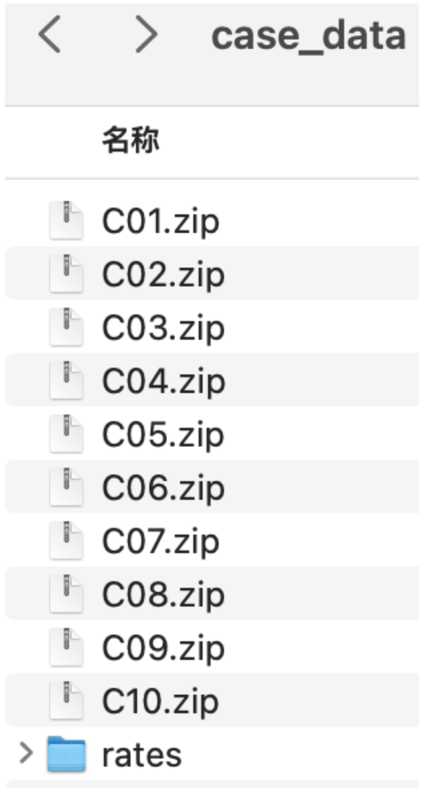
C01

名称

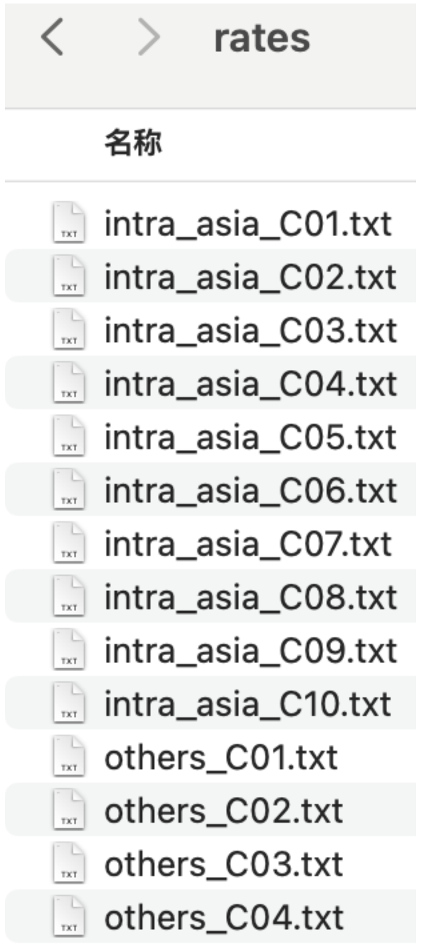
intra_asia.txt
others.txt

Motivation Case-Tasks

- Unpack all zip files in folder 'case_data'.
- Add {carrier_id} to the name of each shipping rate file as intra_asia_{carrier_id}.txt and others_{carrier_id}.txt.
- Move these files to the new folder 'rates'.



C01.zip
C02.zip
C03.zip
C04.zip
C05.zip
C06.zip
C07.zip
C08.zip
C09.zip
C10.zip
rates



intra_asia_C01.txt
intra_asia_C02.txt
intra_asia_C03.txt
intra_asia_C04.txt
intra_asia_C05.txt
intra_asia_C06.txt
intra_asia_C07.txt
intra_asia_C08.txt
intra_asia_C09.txt
intra_asia_C10.txt
others_C01.txt
others_C02.txt
others_C03.txt
others_C04.txt

Motivation Case-Algorithm

Algorithm

- Enter the working directory ‘case_data’
- Make a directory ‘rates’ if not exist
- For each ZIP file:
  - Extract the ZIP file
  - Extract carrier_id from its file name
  - Copy intra_asia.txt to intra_asia_{carrier_id}.txt in folder ‘rates’
  - Copy others.txt to others_{carrier_id}.txt in folder ‘rates’
  - Delete intra_asia.txt and others.txt

Data Structure

- Directory
  - Enter
  - Make
  - Exist
- ZIP file
  - Extract
- Files
  - Copy
  - Delete

Modules to Be Used

- pathlib: offering a data type (named Path) representing filesystem paths
- os: offering a way of using operating system dependent functionality (e.g., change dir, etc.)
- shutil: offering some high-level operations on files (e.g., copy, etc.)
- zipfile: offering operations on ZIP files (e.g., compress, extract, etc.)

- How to import modules

```
import module_name
```

- How to use objects in modules

```
module_name. member_name
```

- How to import objects (types, functions, etc) from modules
  - so that module name and dot can be omitted when using the objects

```
from module_name import member_name
member_name
```

Motivation Case-Code

```python
import shutil, os, zipfile
from pathlib import Path

os.chdir('case_data')

if not Path('rates').exists():
    os.makedirs('rates')

file_names = os.listdir('.')

for file_name in file_names:
    if file_name.endswith('.zip'):
        zip_file = zipfile.ZipFile(file_name)
        zip_file.extractall()
        zip_file.close()


        carrier_id, ext = file_name.split('.')

        shutil.copy('intra_asia.txt', f'rates/intra_asia_{carrier_id}.txt')
        shutil.copy('others.txt', f'rates/others_{carrier_id}.txt')

        os.unlink('intra_asia.txt')
        os.unlink('others.txt')

```

Task Automation: Organizing Files

- File and file paths
- Modules to operate files and folders
- Compressing files with the zipfile module

Path and Filename

- File has two key properties: path and filename:

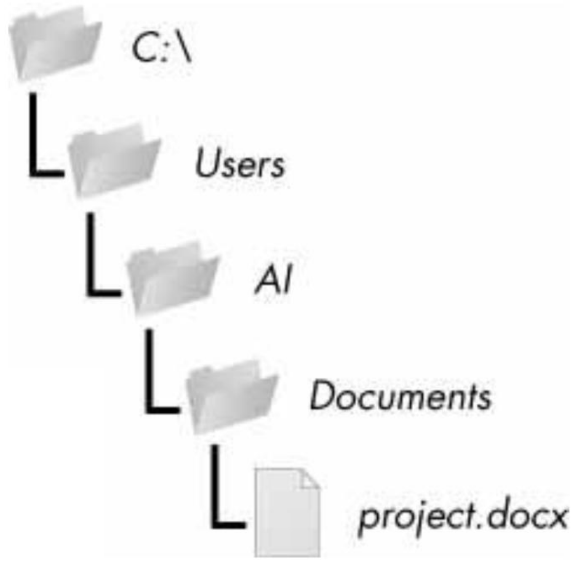
➢The path specifies the location (folder).

➢In filename, the extension shows file's type.

- The root folder contains all other folders.

➢C:\ is root folder in Windows system.

➢/. is root folder in macOS.



Backslash and Forward Slash

- In Windows, paths are written using backslashes (\) as the separator between folder names.

➢ e.g., C:\Users\name.txt

- In macOS, paths are written using forward slash (/) as the separator between folder names.

➢ e.g., /Users/name.txt

Modules to Be Used

- pathlib: offering a data type (named Path) representing filesystem paths
- os: offering a way of using operating system dependent functionality (e.g., change dir, etc.)
- shutil: offering some high-level operations on files (e.g., copy, etc.)
- zipfile: offering operations on ZIP files (e.g., compress, extract, etc.)

- How to import modules

```
import module_name
```

- How to use objects in modules

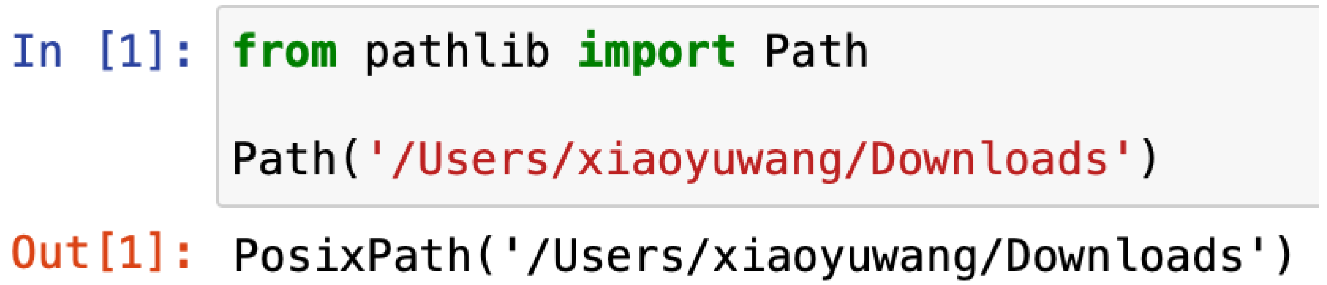
```
module_name. member_name
```

- How to import objects (types, functions, etc) from modules
  - so that module name and dot can be omitted when using the objects

```
from module_name import member_name
member_name
```

Path() Function in pathlib Module

- Use Path to represent a directory.

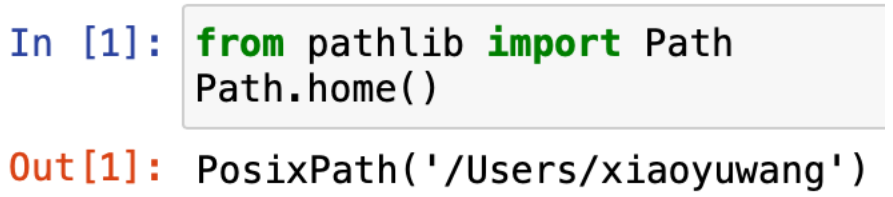


```
In [1]: from pathlib import Path

        Path('/Users/xiaoyuwang/Downloads')

Out[1]: PosixPath('/Users/xiaoyuwang/Downloads')
```

Home Directory

- A folder for user own files is called the home directory or home folder.

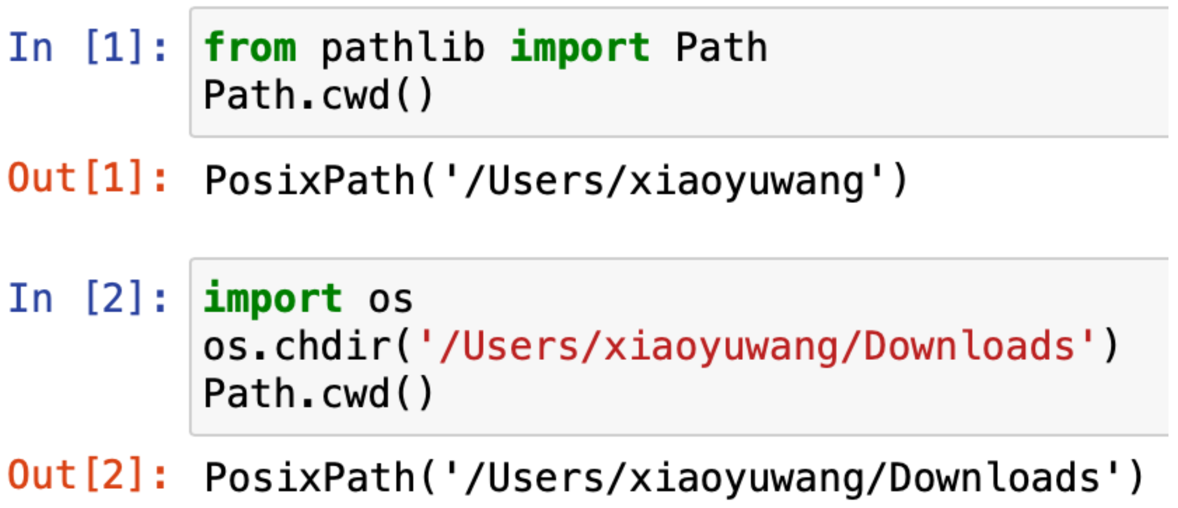
- Get home directory by calling: Path.home()



```
In [1]: from pathlib import Path
        Path.home()

Out[1]: PosixPath('/Users/xiaoyuwang')
```

Working Directory

- Computer program (Python) has a current working directory.
- Use Path.cwd() to obtain current working directory.
- Use os.chdir() to change current working directory.



```
In [1]: from pathlib import Path
        Path.cwd()

Out[1]: PosixPath('/Users/xiaoyuwang')

In [2]: import os
        os.chdir('/Users/xiaoyuwang/Downloads')
        Path.cwd()

Out[2]: PosixPath('/Users/xiaoyuwang/Downloads')
```

Modules to Be Used

- pathlib: offering a data type (named Path) representing filesystem paths
- os: offering a way of using operating system dependent functionality (e.g., change dir, etc.)
- shutil: offering some high-level operations on files (e.g., copy, etc.)
- zipfile: offering operations on ZIP files (e.g., compress, extract, etc.)

- How to import modules

import module_name

- How to use objects in modules

module_name. member_name

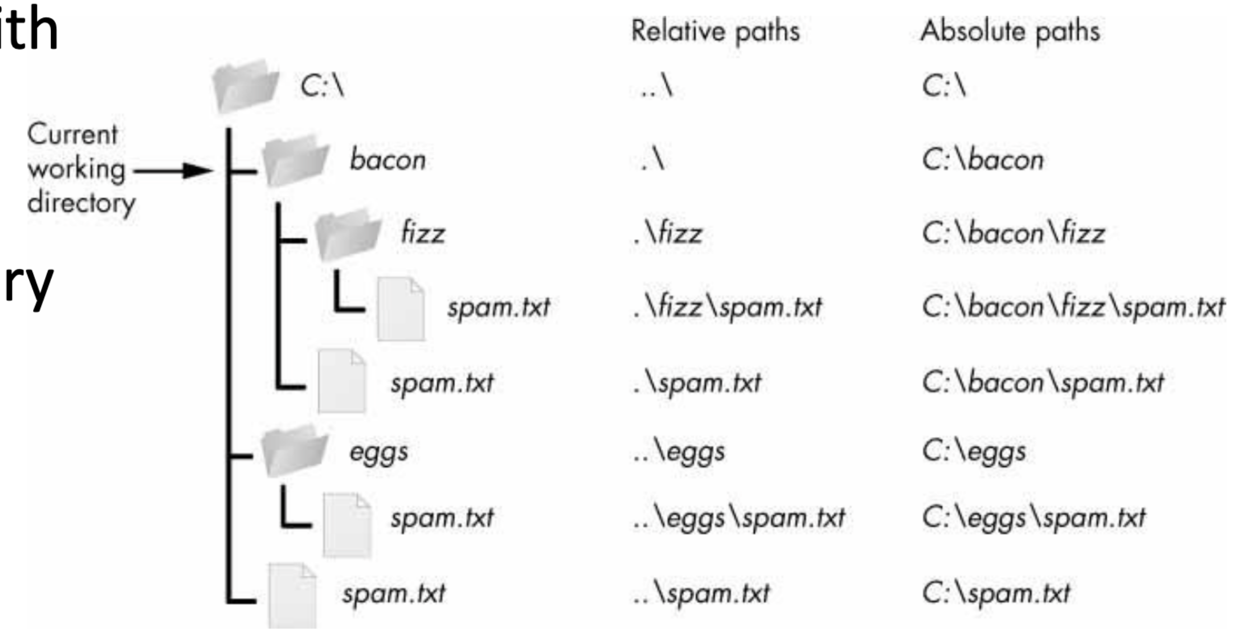
- How to import objects (types, functions, etc) from modules
  - so that module name and dot can be omitted when using the objects

from module_name import member_name

member_name

Absolute vs. Relative Paths

- An *absolute path* always begins with the root folder
- A *relative path* is relative to the program's current working directory
  - A dot folder (.) is for "this directory" (i.e., the current working directory)
  - A dot-dot folder ("..") means "the parent folder" of this directory
  - Use Path(relative_path).resolve() to obtain the absolute path of a given relative_path



	Relative paths	Absolute paths
C:\	..\	C:\
bacon	.\	C:\bacon
fizz	.\fizz	C:\bacon\fizz
spam.txt	.\fizz\spam.txt	C:\bacon\fizz\spam.txt
spam.txt	.\spam.txt	C:\bacon\spam.txt
eggs	..\eggs	C:\eggs
spam.txt	..\eggs\spam.txt	C:\eggs\spam.txt
spam.txt	..\spam.txt	C:\spam.txt

```
import os
from pathlib import Path
print(f'The current working path is: {Path.cwd()}.')
os.chdir('..')
print(f'The current working path is: {Path.cwd()}.')
print(Path('.').resolve())
```

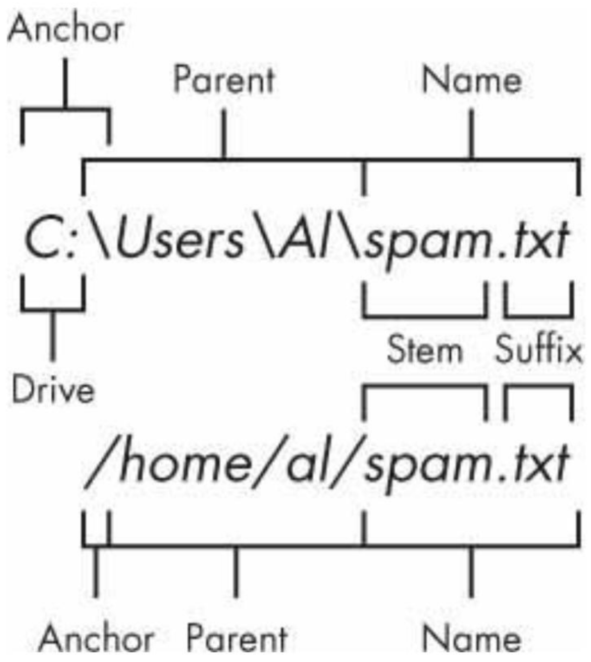
Getting Parts of a File Path

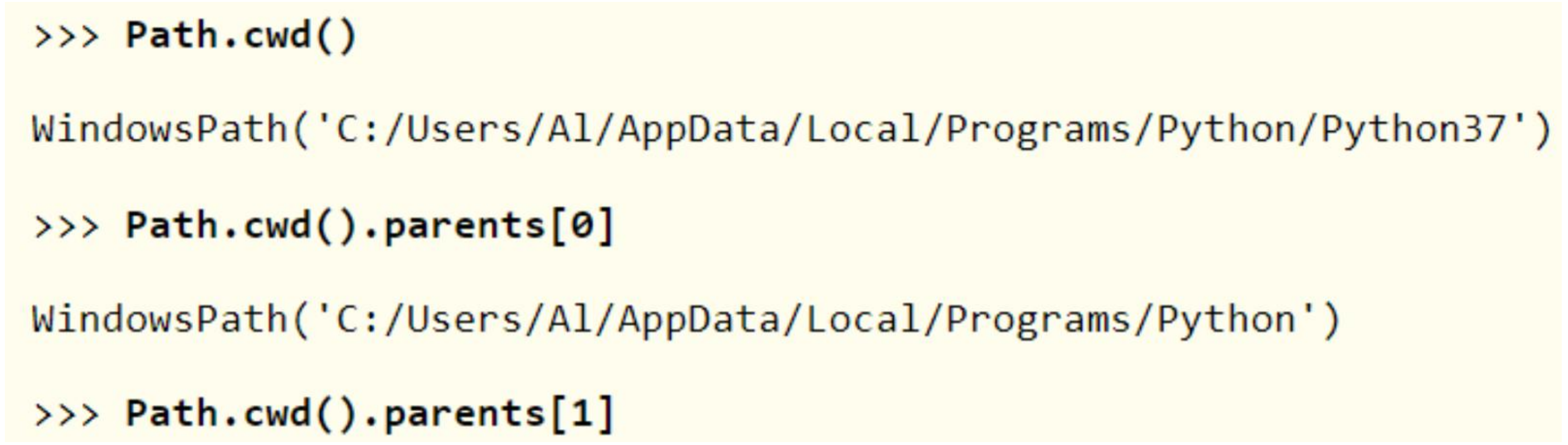
- Given a Path object p, you can extract the file parts as strings:

➢ p.anchor: the root folder.

➢ p.parent: the folder that contains the file.

➢ p.parents: a sequence of ancestor folders of with integer indices.





```
>>> Path.cwd()
WindowsPath('C:/Users/Al/AppData/Local/Programs/Python/Python37')
>>> Path.cwd().parents[0]
WindowsPath('C:/Users/Al/AppData/Local/Programs/Python')
>>> Path.cwd().parents[1]
```

Create New Folder

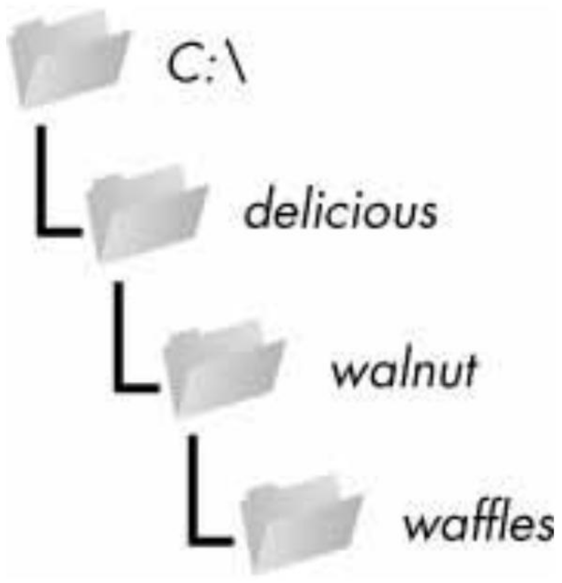
- The os.makedirs() function in os module can be used to create a new folder.

Making a folder by an absolute path

```
>>> import os

>>> os.makedirs('C:\\delicious\\walnut\\waffles')
```

Making a folder by a relative path

```
os.chdir('C:\\delicious\\walnut')
os.makedirs('waffles')
```



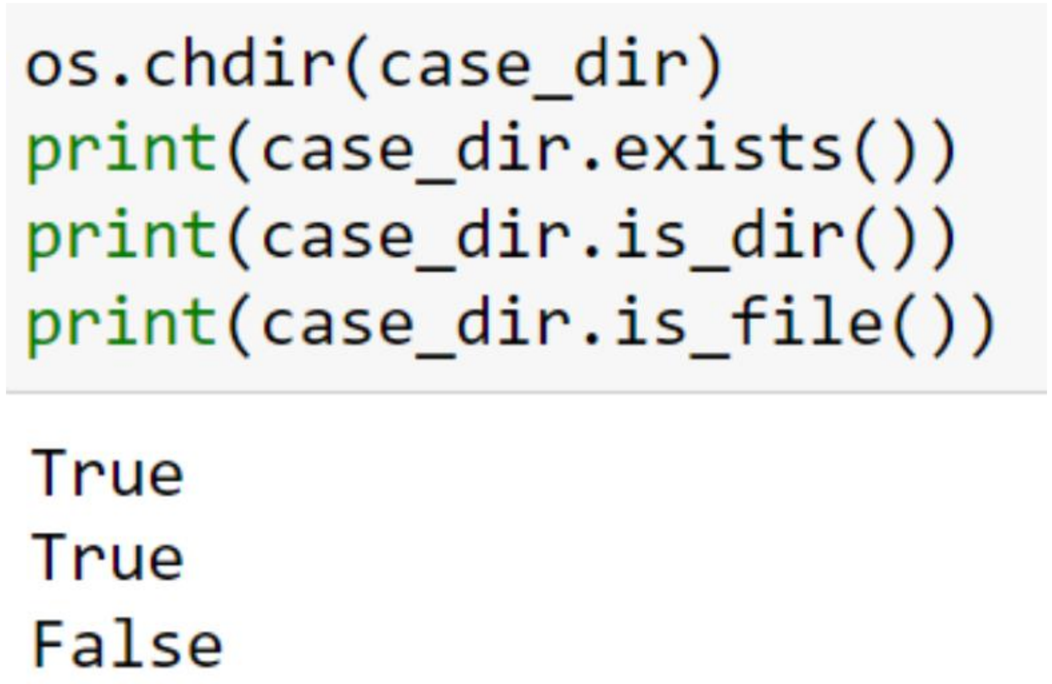
Check Path Validity

- To check whether a given path p exists and whether it is a file or folder.

➢ p.exists() returns True if the path exists.

➢ p.is_file() returns True if the path exists and is a file.

➢ p.is_dir() returns True if the path exists and is a directory.



```
os.chdir(case_dir)
print(case_dir.exists())
print(case_dir.is_dir())
print(case_dir.is_file())
```

```
True
True
False
```

Getting Folder Contents

- os.listdir(path) returns a list of filename strings for each file in the path.

```
print(os.listdir('.'))
```

```
['C01.zip', 'C02.zip', 'C03.zip', 'C04.zip', 'C05.zip', 'C06.zip', 'C07.zip', 'C08.zip', 'C09.zip',
'C10.zip', 'rates']
```

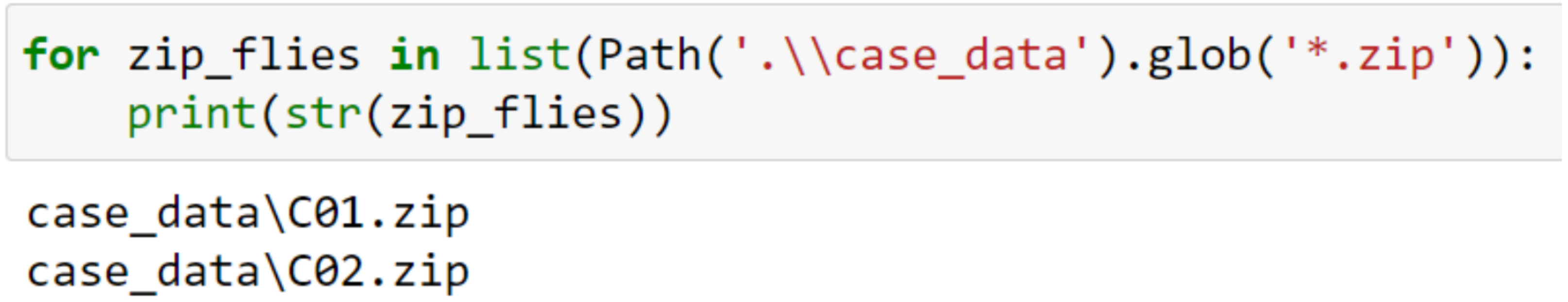
- How to get a list of names of zip files?

```
zip_file_names = []
for name in os.listdir('.'):
    if name.endswith('.zip'):
        zip_file_names.append(name)
print(zip_file_names)
```

```
['C01.zip', 'C02.zip', 'C03.zip', 'C04.zip', 'C05.zip', 'C06.zip', 'C07.zip', 'C08.zip', 'C09.zip',
'C10.zip']
```

Modifying List of Files Using Glob

- *: matches any characters, e.g., *.zip for all zip files.
- ?: matches any single character, e.g., ???.txt for all text files with three characters.
- [ranges]: match any single character in ranges, e.g., *[0-9].txt for all text files with a digit before their extensions.
- glob(pattern) returns a sequence of files as Path objects that match the pattern, which can be converted to a list by list().



```python
for zip_flies in list(Path('.\\case_data').glob('*.zip')):
    print(str(zip_flies))
```

```
case_data\C01.zip
case_data\C02.zip
```

What can we do now?

Algorithm

- Enter the working directory ‘case_data’
- Make a directory ‘rates’ if not exist
- For each ZIP file:
  - Extract the ZIP file
  - Extract carrier_id from its file name
  - Copy intra_asia.txt to intra_asia_{carrier_id}.txt in folder ‘rates’
  - Copy others.txt to others_{carrier_id}.txt in folder ‘rates’
  - Delete intra_asia.txt and others.txt

Data Structure

- Directory
  - Enter
  - Make
  - Exist
- ZIP file
  - Extract
- Files
  - Copy
  - Delete

Task Automation: Organizing Files

- File and file paths
- Modules to operate files and folders
- Compressing files with the zipfile module

Modules to Be Used

- pathlib: offering a data type (named Path) representing filesystem paths
- os: offering a way of using operating system dependent functionality (e.g., change dir, etc.)
- shutil: offering some high-level operations on files (e.g., copy, etc.)
- zipfile: offering operations on ZIP files (e.g., compress, extract, etc.)

- How to import modules

```
import module_name
```

- How to use objects in modules

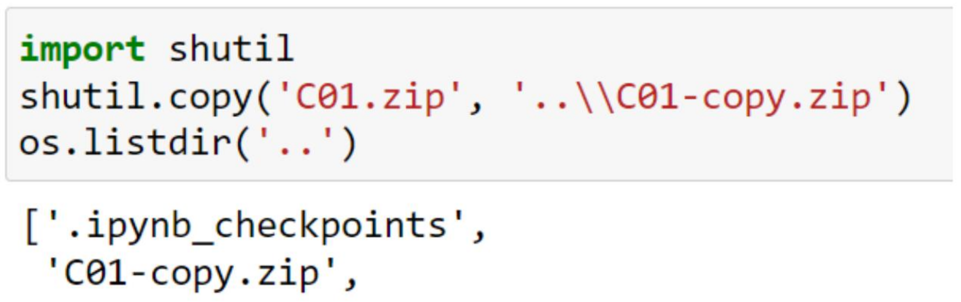
```
module_name. member_name
```

- How to import objects (types, functions, etc) from modules
  - so that module name and dot can be omitted when using the objects

```
from module_name import member_name
member_name
```

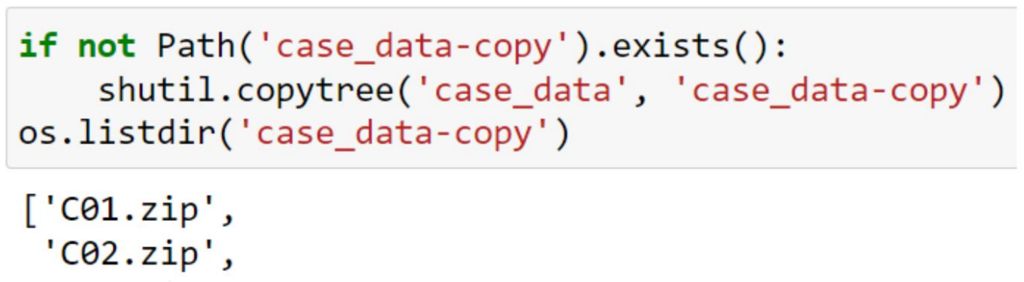
Copying Files and Folders

- shutil.copy(source, destination) copies the file at the path source to the folder at the path destination.
- shutil.copytree(source, destination) copies the folder at the path source, along with all of its files and subfolders, to the folder at the path destination.



```
import shutil
shutil.copy('C01.zip', '..\\C01-copy.zip')
os.listdir('..')
```

```
['.ipynb_checkpoints',
 'C01-copy.zip',
```



```
if not Path('case_data-copy').exists():
    shutil.copytree('case_data', 'case_data-copy')
os.listdir('case_data-copy')
```

```
['C01.zip',
 'C02.zip',
```

Moving and Renaming Files and Folders

- shutil.move(source, destination) will move the file or folder at the path source to the path destination.

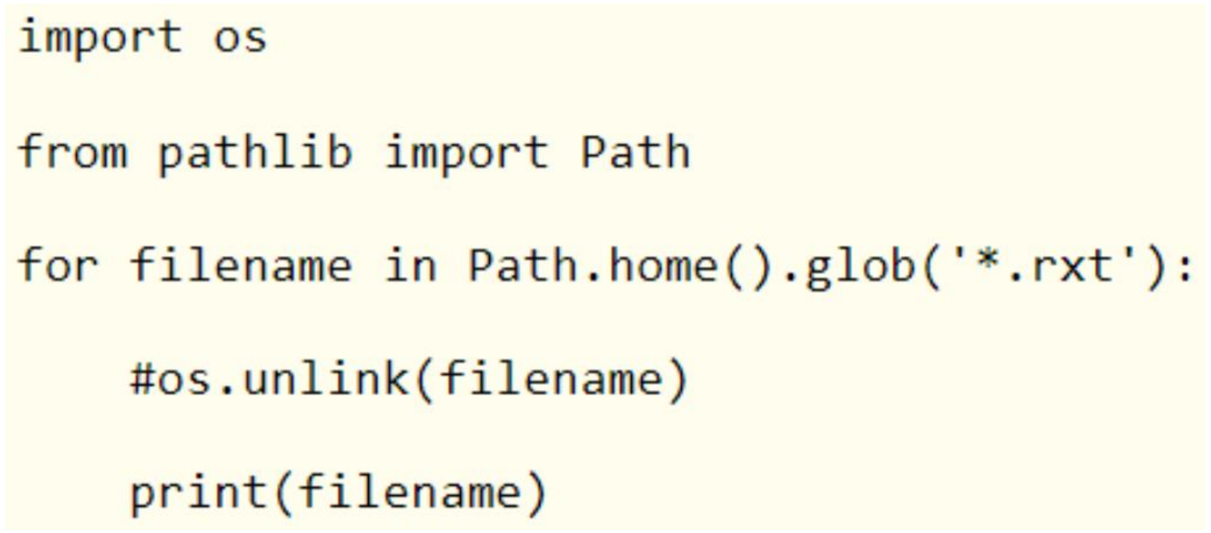
```
>>> import shutil

>>> shutil.move('C:\\bacon.txt', 'C:\\eggs')

'C:\\eggs\\bacon.txt'
```

- When the destination path also specify a filename, the source file is moved and renamed.

```
>>> shutil.move('C:\\bacon.txt', 'C:\\eggs\\new_bacon.txt')

'C:\\eggs\\new_bacon.txt'
```

Permanently Deleting Files and Folders

- Delete a file at a given path by os.unlink(path)
- Delete empty folder at a given path by os.rmdir(path)
- Delete folder and all of its content by shutil.rmtree(path)
- Use comment to avoid files and folders accidentally being deleted.



```
import os

from pathlib import Path

for filename in Path.home().glob('*.rxt'):

    #os.unlink(filename)

    print(filename)
```

Walking a Directory Tree

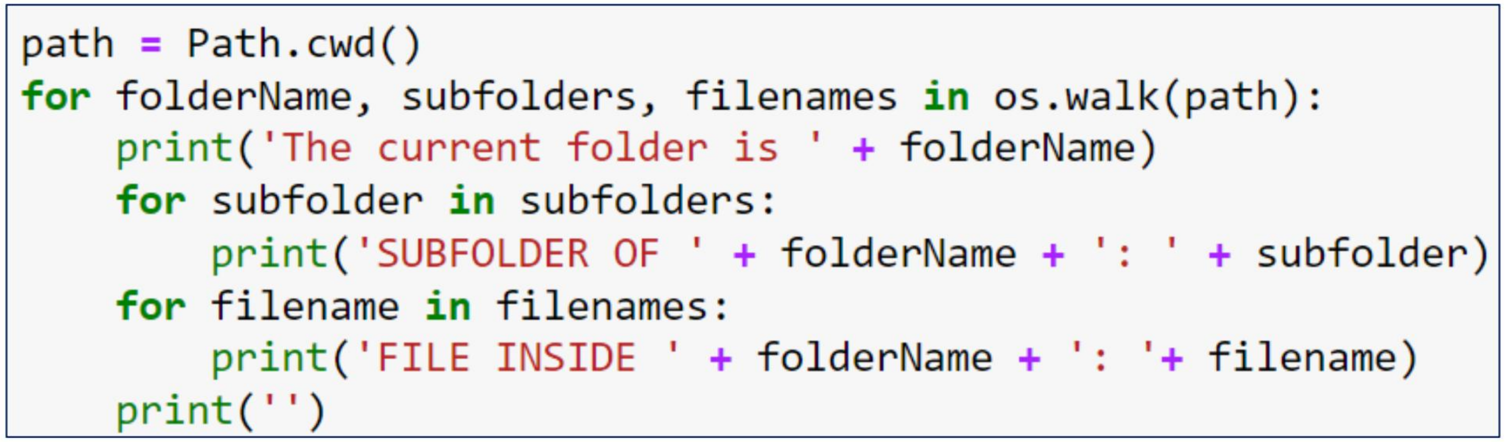
- To get each file at a given Path and its descendant subfolders, use os.walk(Path), which returns following three values:

➢A string of the current folder’s name.

➢A list of strings of the folders in the current folder.

➢A list of strings of the files in the current folder.

- Use for-loop to get each file from the returned tuple sequence.



```
path = Path.cwd()
for folderName, subfolders, filenames in os.walk(path):
    print('The current folder is ' + folderName)
    for subfolder in subfolders:
        print('SUBFOLDER OF ' + folderName + ': ' + subfolder)
    for filename in filenames:
        print('FILE INSIDE ' + folderName + ': '+ filename)
    print('')
```

What can we do now?

Algorithm

- Enter the working directory ‘case_data’
- Make a directory ‘rates’ if not exist
- For each ZIP file:
  - Extract the ZIP file
  - Extract carrier_id from its file name
  - Copy intra_asia.txt to intra_asia_{carrier_id}.txt in folder ‘rates’
  - Copy others.txt to others_{carrier_id}.txt in folder ‘rates’
  - Delete intra_asia.txt and others.txt

Data Structure

- Directory
  - Enter
  - Make
  - Exist
- ZIP file
  - Extract
- Files
  - Copy
  - Delete

Task Automation: Organizing Files

- File and file paths
- Modules to operate files and folders
- Compressing files with the zipfile module

ZIP Files

- ZIP files (with .zip extension) can compress many files in one.
- Compressing a file reduces size.
- It’s handy to package several files into one.
- Create and extract ZIP files using functions in the zipfile module.

Modules to Be Used

- pathlib: offering a data type (named Path) representing filesystem paths
- os: offering a way of using operating system dependent functionality (e.g., change dir, etc.)
- shutil: offering some high-level operations on files (e.g., copy, etc.)
- zipfile: offering operations on ZIP files (e.g., compress, extract, etc.)

- How to import modules

```
import module_name
```

- How to use objects in modules

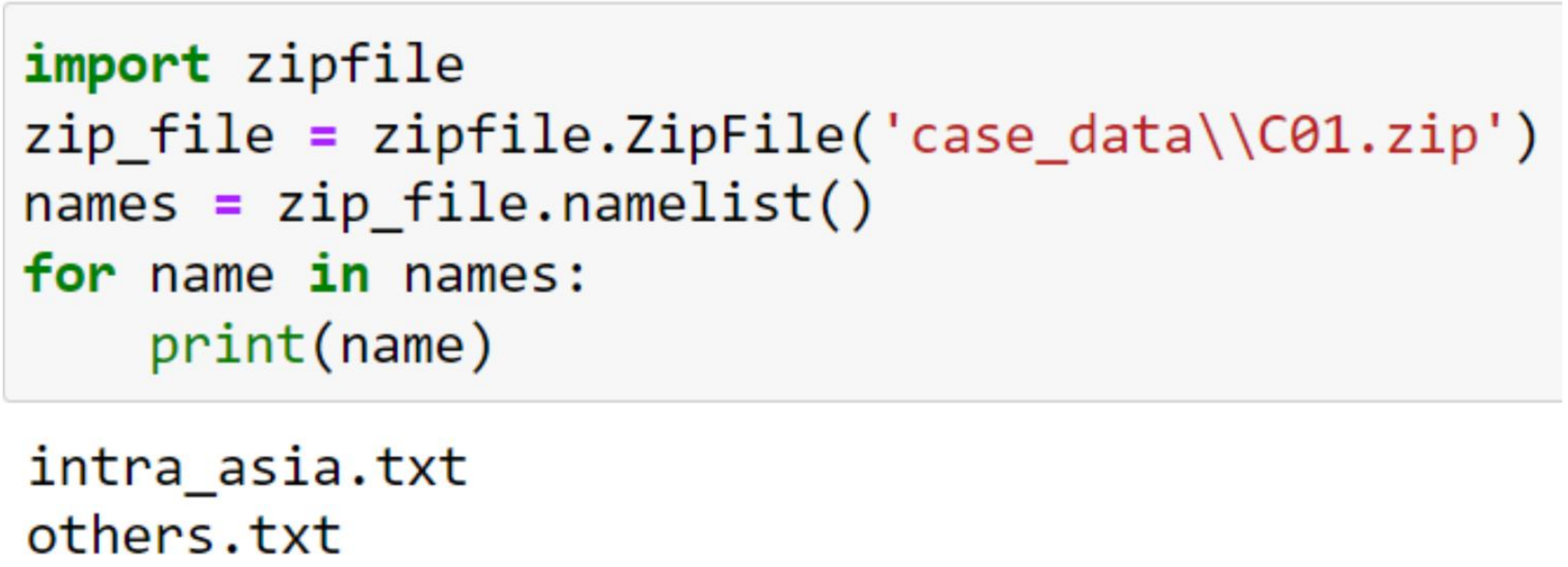
```
module_name. member_name
```

- How to import objects (types, functions, etc) from modules
  - so that module name and dot can be omitted when using the objects

```
from module_name import member_name
member_name
```

Reading ZIP Files

- zipfile.ZipFile(file_name) to create a ZipFile.
- A ZipFile namelist() returns a list of (strings) names of files and folders contained in the ZIP file.



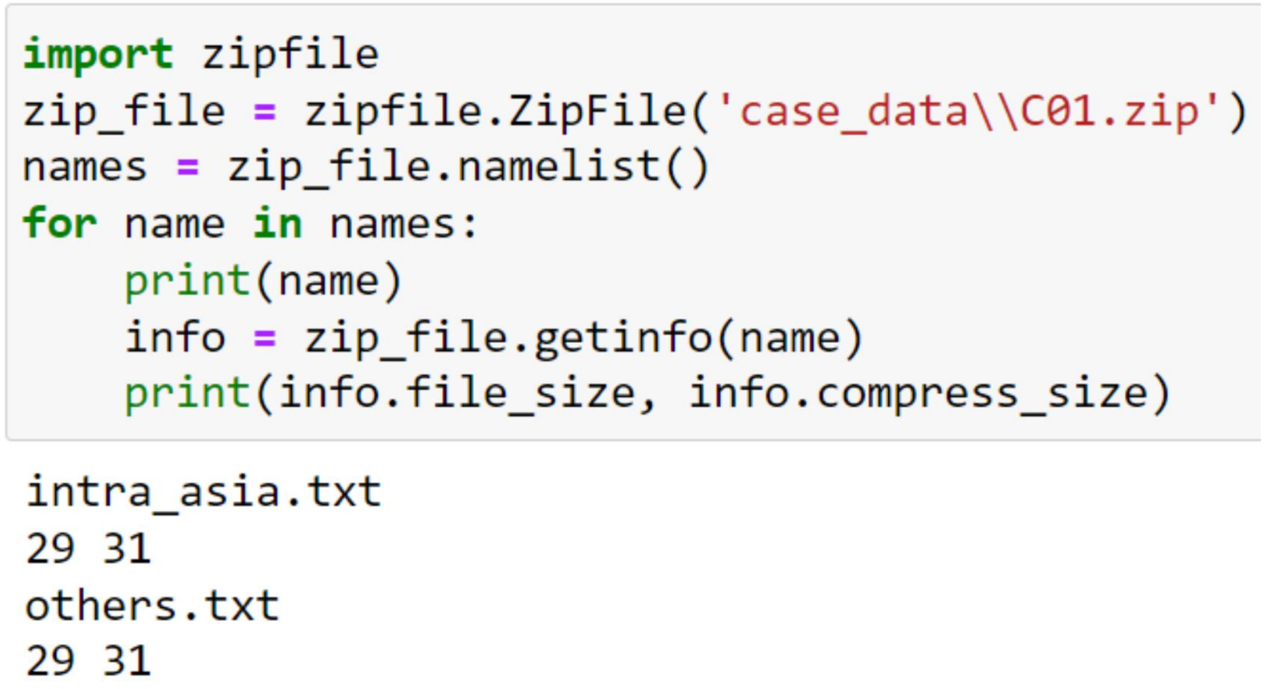
```python
import zipfile
zip_file = zipfile.ZipFile('case_data\\C01.zip')
names = zip_file.namelist()
for name in names:
    print(name)
```

```
intra_asia.txt
others.txt
```

Reading ZIP Files

- The names of files and folders in the ZIP file.
- Use getinfo() to obtain file attributes:

➢such as file_size and compress_size.

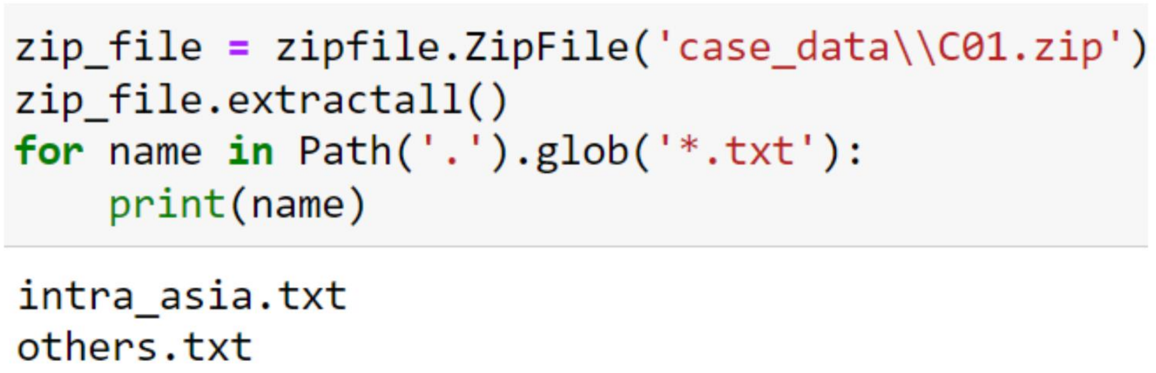


```python
import zipfile
zip_file = zipfile.ZipFile('case_data\\C01.zip')
names = zip_file.namelist()
for name in names:
    print(name)
    info = zip_file.getinfo(name)
    print(info.file_size, info.compress_size)
```

```
intra_asia.txt
29 31
others.txt
29 31
```

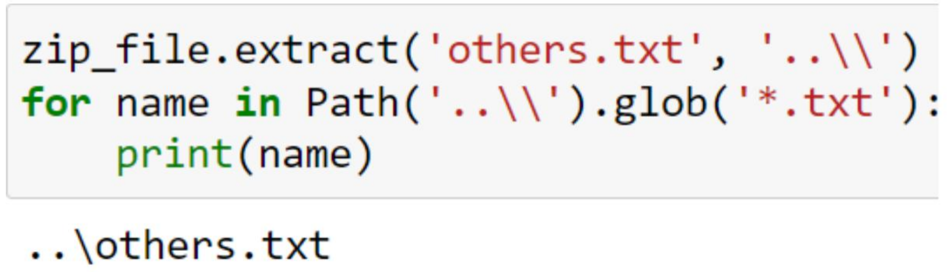
Extracting from ZIP Files

- The extractall(Path) extracts all the files and folders from a ZIP file into the folder at Path.
- The extract(file, Path) extracts a single file from the ZIP file into the folder at Path.
- If Path is not provided, the working directory is adopted.



```
zip_file = zipfile.ZipFile('case_data\\C01.zip')
zip_file.extractall()
for name in Path('.').glob('*.txt'):
    print(name)
```

```
intra_asia.txt
others.txt
```



```
zip_file.extract('others.txt', '..\\')
for name in Path('..\\').glob('*.txt'):
    print(name)
```

```
..\others.txt
```

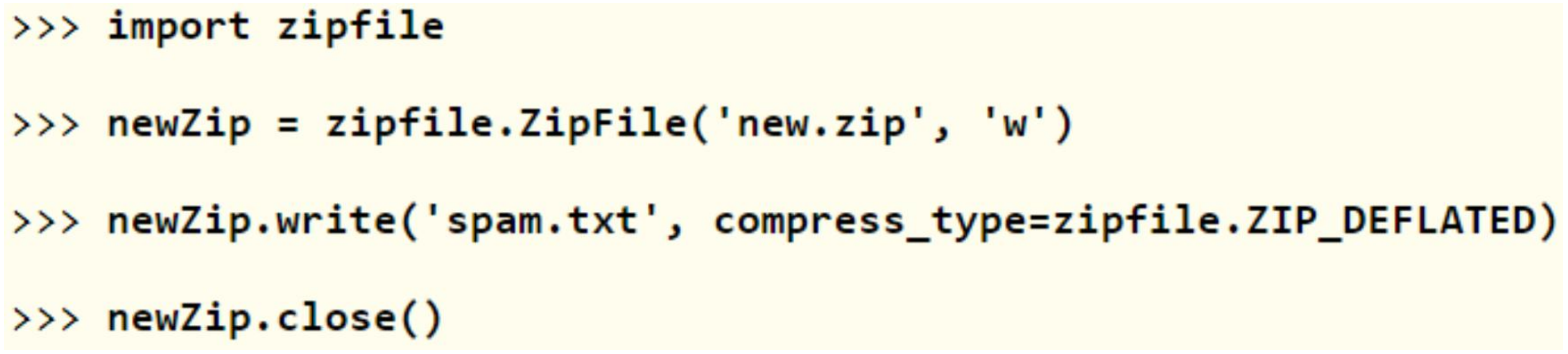
Creating and Adding to ZIP Files

- Open ZipFile object in write mode by passing 'w'.
- write(file_name, compress_type) can compress a file and add it into the ZIP file. The write() method has two parameters:

➢ file_name: file name to add.

➢ compress_type: method to compress (e.g., zipfile.ZIP_DEFLATED).

- Use close() to close ZipFile.



```
>>> import zipfile

>>> newZip = zipfile.ZipFile('new.zip', 'w')

>>> newZip.write('spam.txt', compress_type=zipfile.ZIP_DEFLATED)

>>> newZip.close()
```

What can we do now?

Algorithm

- Enter the working directory ‘case_data’
- Make a directory ‘rates’ if not exist
- For each ZIP file:
  - Extract the ZIP file
  - Extract carrier_id from its file name
  - Copy intra_asia.txt to intra_asia_{carrier_id}.txt in folder ‘rates’
  - Copy others.txt to others_{carrier_id}.txt in folder ‘rates’
  - Delete intra_asia.txt and others.txt

Data Structure

- Directory
  - Enter
  - Make
  - Exist
- ZIP file
  - Extract
- Files
  - Copy
  - Delete

Motivation Case-Revisit

```python
import shutil, os, zipfile
from pathlib import Path

os.chdir('case_data')

if not Path('rates').exists():
    os.makedirs('rates')

file_names = os.listdir('.')

for file_name in file_names:
    if file_name.endswith('.zip'):
        zip_file = zipfile.ZipFile(file_name)
        zip_file.extractall()
        zip_file.close()


        carrier_id, ext = file_name.split('.')

        shutil.copy('intra_asia.txt', f'rates/intra_asia_{carrier_id}.txt')
        shutil.copy('others.txt', f'rates/others_{carrier_id}.txt')

        os.unlink('intra_asia.txt')
        os.unlink('others.txt')
```

Motivation Case-Revisit

```python
import shutil, os, zipfile
from pathlib import Path

os.chdir('case_data')

if not Path('rates').exists():
    os.makedirs('rates')

file_names = os.listdir('.')

for file_name in file_names:
    if file_name.endswith('.zip'):
        zip_file = zipfile.ZipFile(file_name)
        zip_file.extractall()
        zip_file.close()


        carrier_id, ext = file_name.split('.')

        text_file_names = Path('.').glob('*.txt')

        for text_file_name in text_file_names:
            base_name, extension = text_file_name.name.split('.')
            shutil.move(f'{text_file_name}', f'rates/{base_name}_{carrier_id}.txt')

```

Acknowledgement

- Acknowledgements / Contributions

- Initial Development: Charles Severance, University of Michigan School of Information
- Further Development: Zhou Xu, Hong Kong Polytechnic University
- Continuous development: Xiaoyu Wang, Hong Kong Polytechnic University