Scripting

WEEK 11 – FILE INPUT/OUTPUT

File Input/Output

Input refers to any data flowing into a Python program, and Output refers to any data flowing out of a Python program

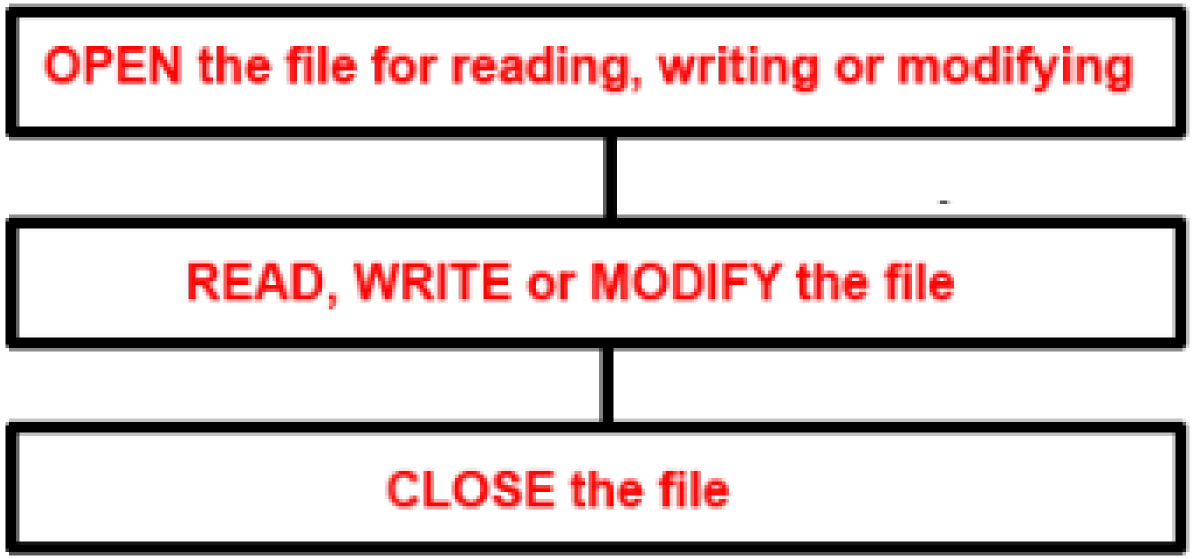
Up until now, all of our input and output has been via the console, which by default has been the keyboard and the screen

But Input and Output (I/O) can occur between multitudes of devices, both virtual and physical

File I/O means files can be opened, read, modified, copied, etc, as long as they have a name

File Input/Output

In Python, a file process needs to be undertaken in the following sequence:


OPEN the file for reading, writing or modifying
READ, WRITE or MODIFY the file
CLOSE the file

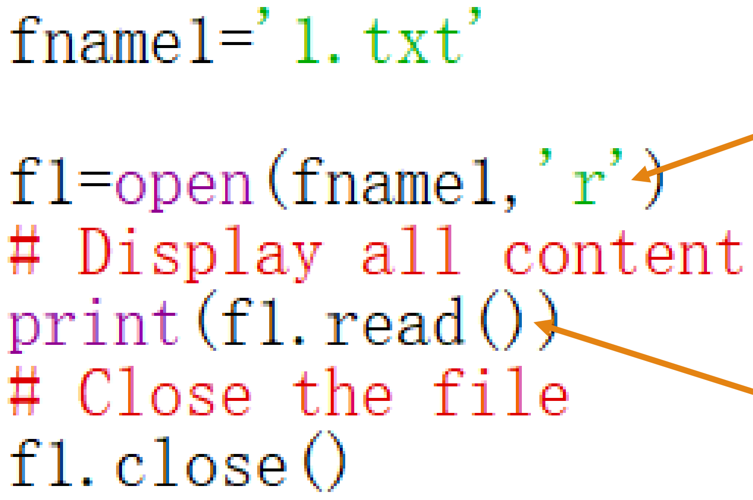
Open a File

A file can be opened in one of the following modes:

Mode	Meaning
r	Open a file for reading. Gives an error if it doesn’t exist
a	Open a file for appending. Creates the file if it doesn’t exist
w	Open a file for writing. Create the file, or overwrite an existing file
x	Create a file if it doesn’t exist. Gives an error if it does exist

Read a File

open() method is a method that opens a file and return a file object



```
fname1=’1.txt’

f1=open(fname1,’r’)
# Display all content
print(f1.read())
# Close the file
f1.close()
```

Read Mode

read() method reads the whole file and returns the content in the form of a string

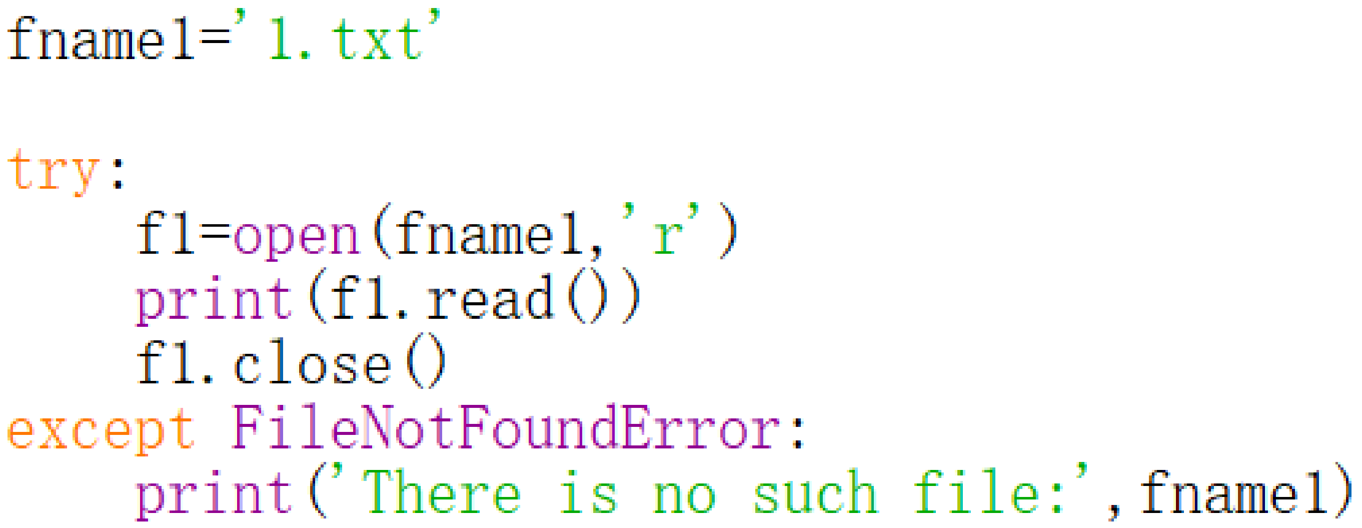
Read a File

```
fname1=’1.txt’

f1=open(fname1,’r’)
# Display first 5 characters
print(f1.read(5))
f1.close()
```

```
fname1=’1.txt’

f1=open(fname1,’r’)
# Read only one line
print(f1.readline())
f1.close()
```

Read a File

It is important to make your code robust and error tolerant. Use the**try** clause:

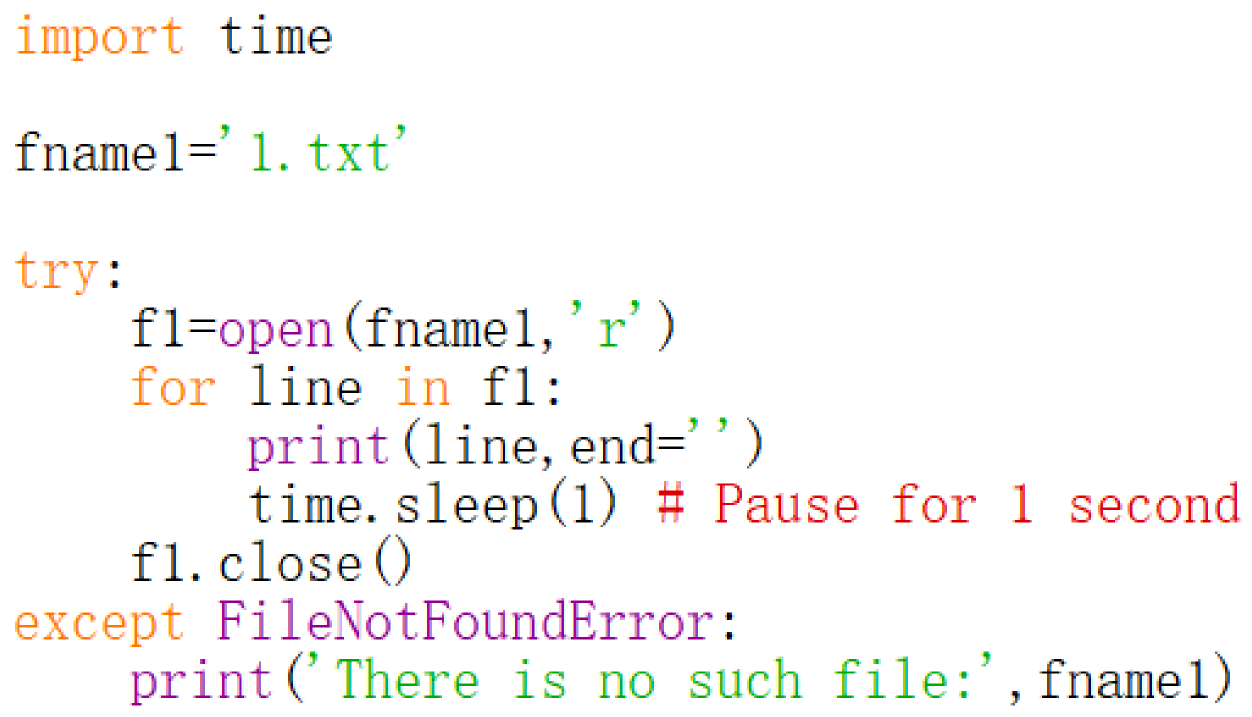


```
fname1=’1.txt’

try:
    f1=open(fname1,’r’)
    print(f1.read())
    f1.close()
except FileNotFoundError:
    print(’There is no such file:’,fname1)
```

In this case, if file **1.txt** doesn’t exist, it will display ‘*There is no such file: 1.txt*’, instead of crashing the program with an error

Read a File

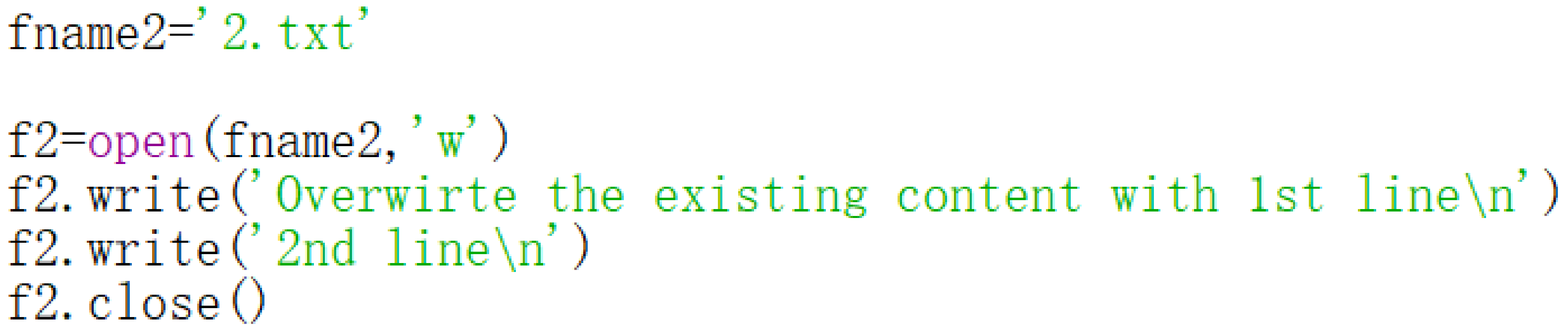
We can use loop to read lines from a file:



```
import time

fname1='1.txt'

try:
    f1=open(fname1,'r')
    for line in f1:
        print(line,end='')
        time.sleep(1) # Pause for 1 second
    f1.close()
except FileNotFoundError:
    print('There is no such file:',fname1)
```

Write a File

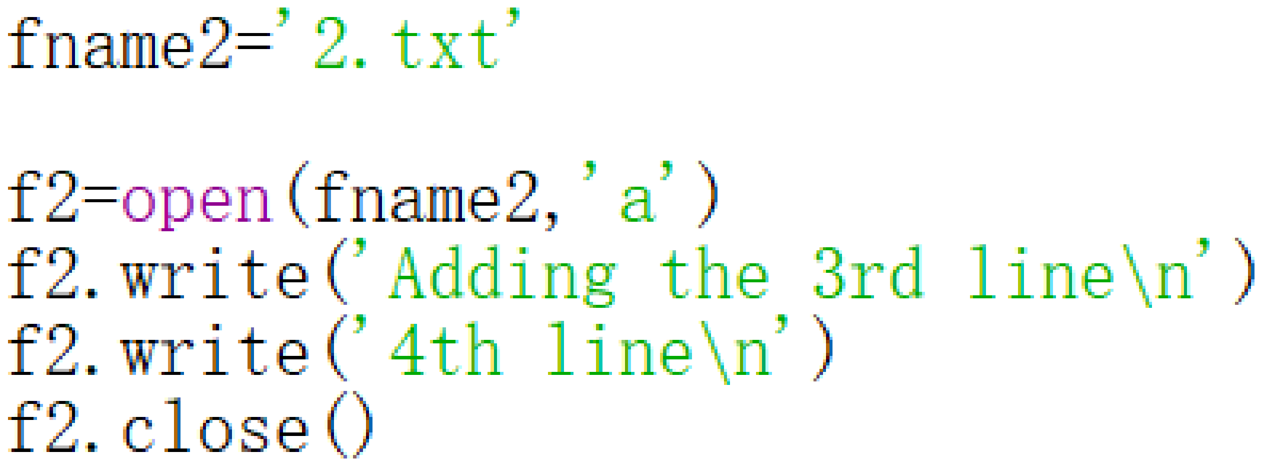
We can also modify the content of a file by using w or a mode. Remember, w overwrites the existing content, while a appends new lines into a file



```
fname2=’2.txt’

f2=open(fname2,’w’)
f2.write(’Overwirte the existing content with 1st line\n’)
f2.write(’2nd line\n’)
f2.close()
```

Append to a File

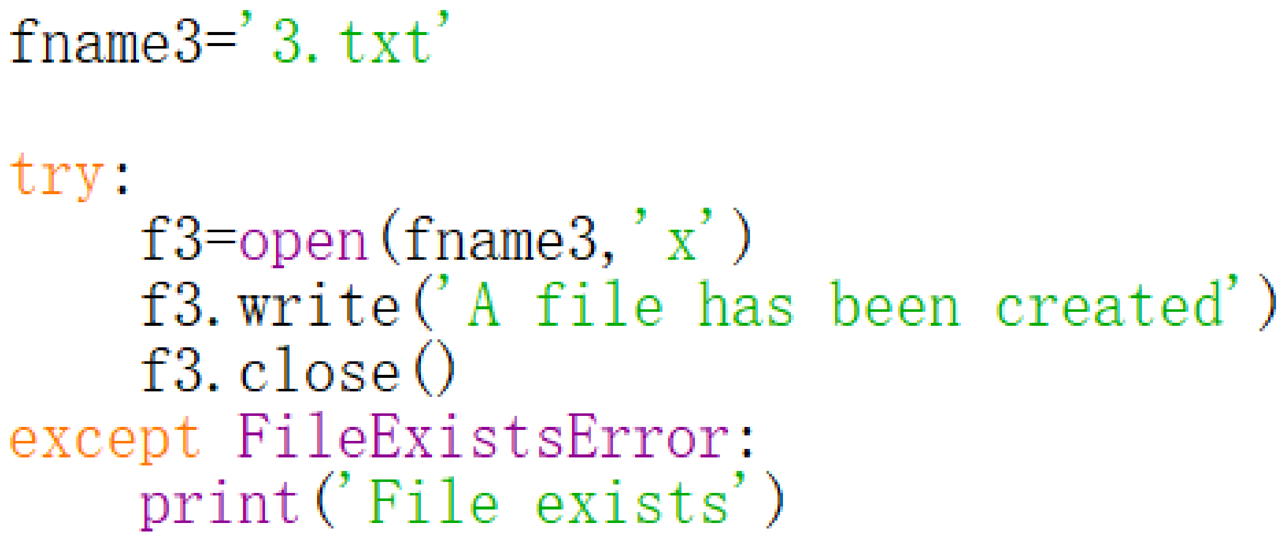
Add new content to a file:



```
fname2='2.txt'

f2=open(fname2,'a')
f2.write('Adding the 3rd line\n')
f2.write('4th line\n')
f2.close()
```

Create a File

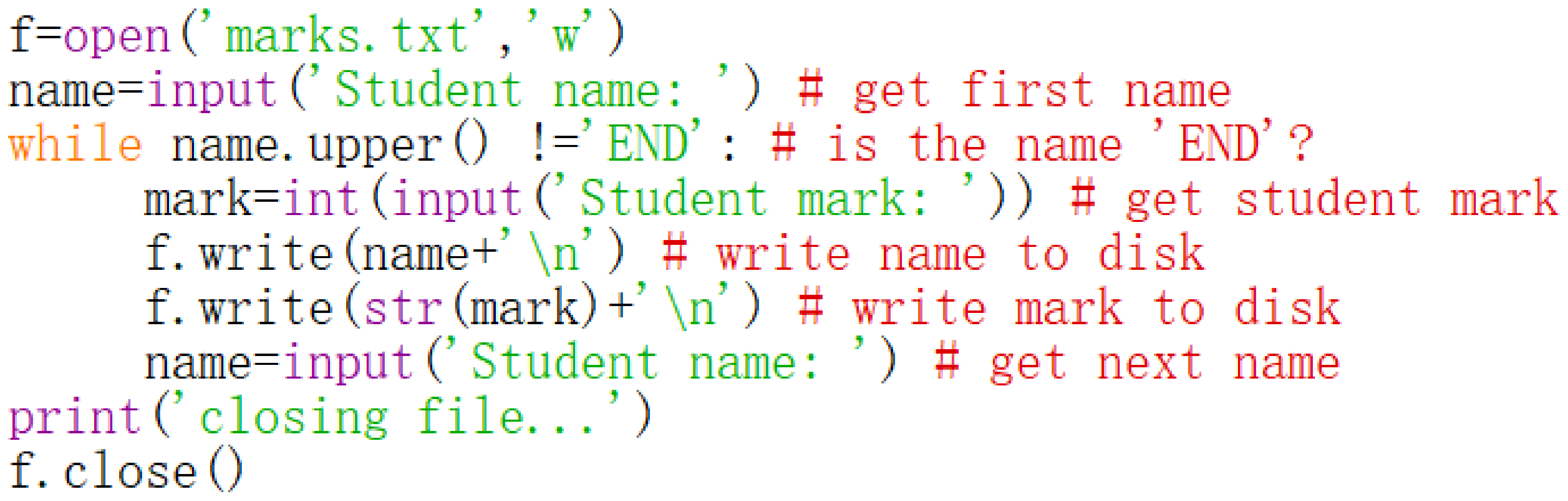
The x mode creates a file if it doesn’t exist; if the file exist, it will generate an error:



```python
fname3=’3.txt’

try:
    f3=open(fname3,’x’)
    f3.write(’A file has been created’)
    f3.close()
except FileExistsError:
    print(’File exists’)
```

Write Multiple Lines into a File

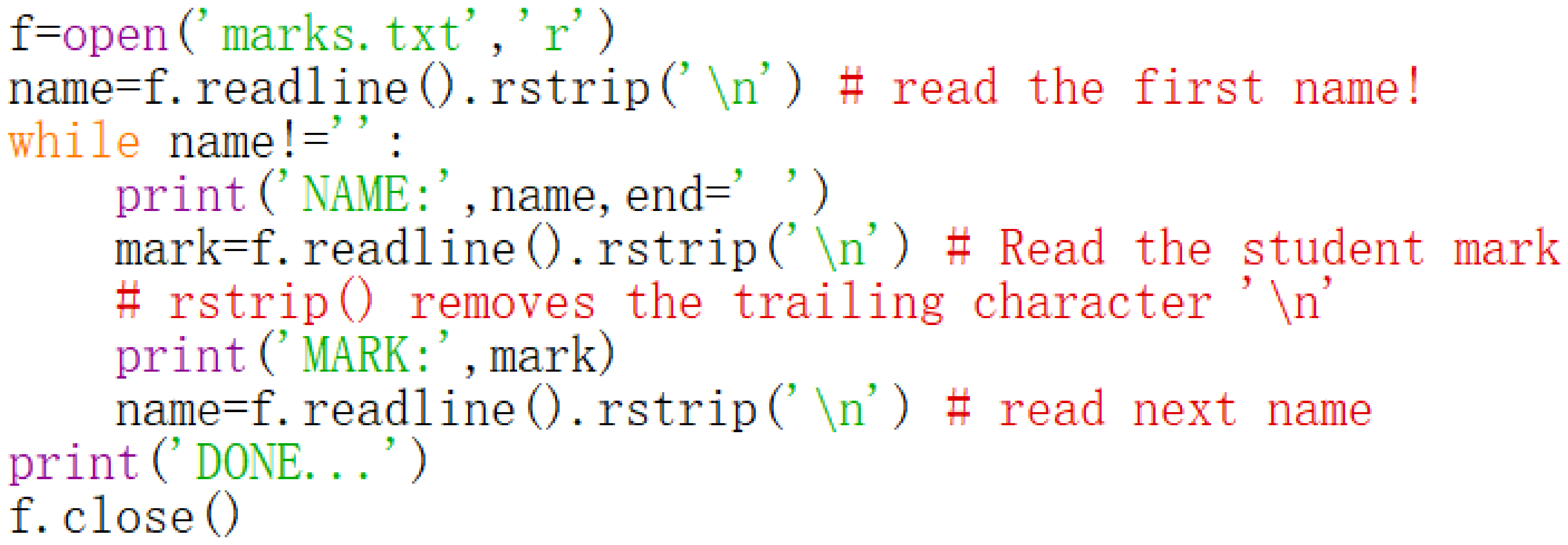
Writing student names and marks to a file named **marks.txt**



```
f=open(’marks.txt’,’w’)
name=input(’Student name: ’) # get first name
while name.upper() !=’END’: # is the name ’END’?
    mark=int(input(’Student mark: ’)) # get student mark
    f.write(name+’\n’) # write name to disk
    f.write(str(mark)+’\n’) # write mark to disk
    name=input(’Student name: ’) # get next name
print(’closing file...’)
f.close()
```

Read Multiple Lines from a File

Reading and displaying lines from a file:



```
f=open(’marks.txt’,’r’)
name=f.readline().rstrip(’\n’) # read the first name!
while name!=’’ :
    print(’NAME:’,name,end=’  ’)
    mark=f.readline().rstrip(’\n’) # Read the student mark
    # rstrip() removes the trailing character ’\n’
    print(’MARK:’,mark)
    name=f.readline().rstrip(’\n’) # read next name
print(’DONE...’)
f.close()
```

Use Lists for File I/O

There are two methods that allow file I/O directly to/from lists: **writelines()** & **readlines()**

```
cities = ['Sydney\n', 'Melbourne\n', 'Brisbane\n', 'Canberra\n']

# Open a file for writing.
f = open('cities.txt', 'w')
# Write the list to the file.
f.writelines(cities)
f.close()

f = open('cities.txt', 'r')
# Read the list to the file.
citylist=f.readlines()
#display the list
for city in citylist:
    print(city,end='')
f.close()
```

Methods to Read Line(s)

readline() reads one entire line from the file

```
# Call readline() twice to return both the
# first and the second line
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

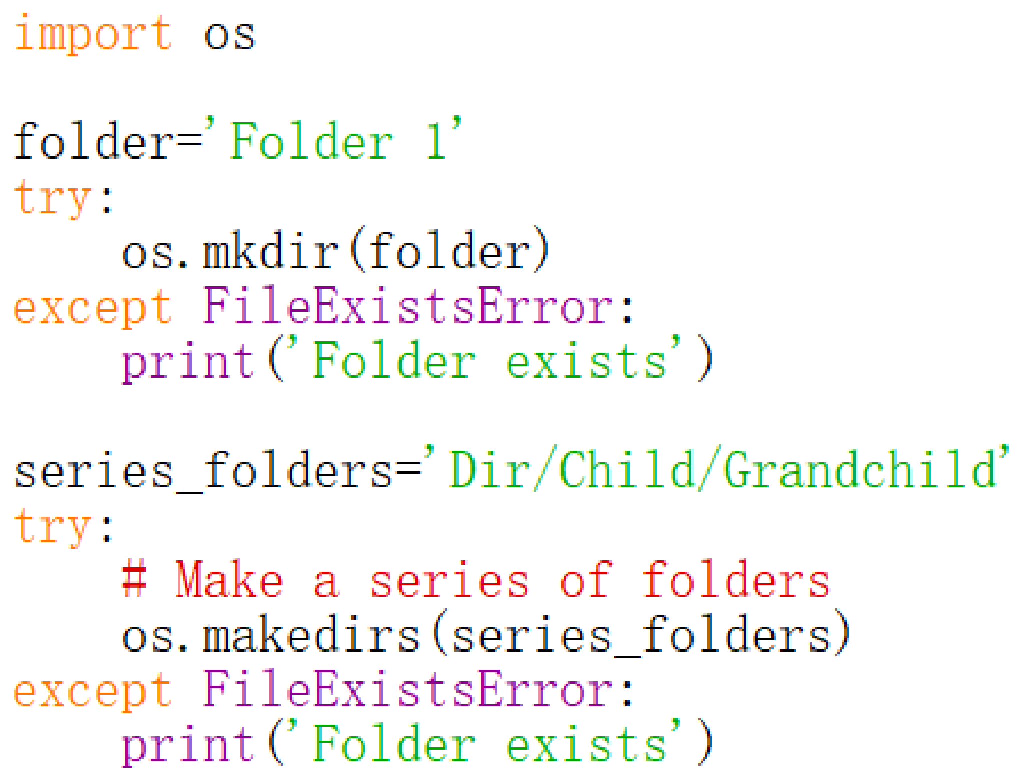
readlines() return all lines in the file, as a list where each line is an item in the list object

```
# line gets an item from a list
for line in f.readlines():
    print(line)
```

Use os Module

The **os** module contains many methods and properties to do with files and directories (folders).

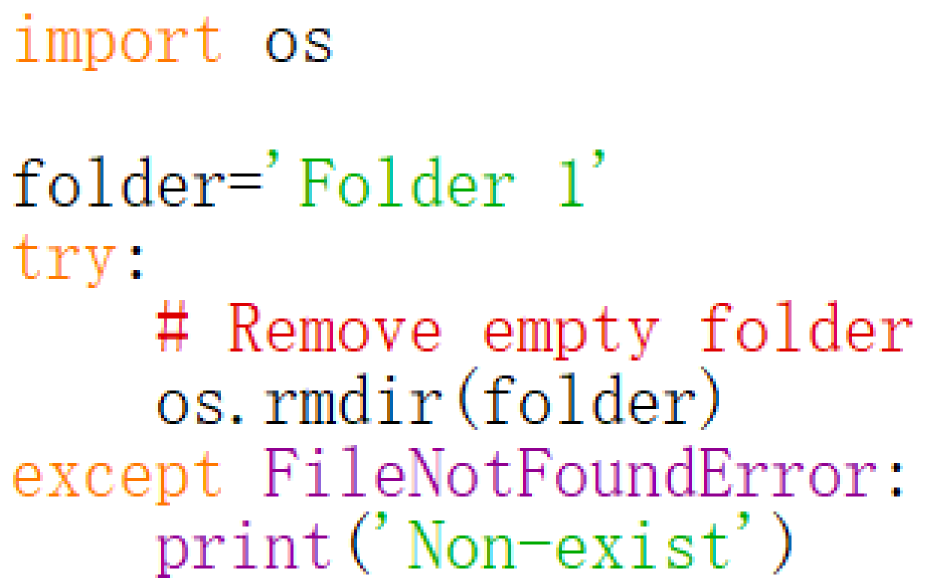
Create folder(s):



```
import os

folder=’Folder 1’
try:
    os.mkdir(folder)
except FileExistsError:
    print(’Folder exists’)

series_folders=’Dir/Child/Grandchild’
try:
    # Make a series of folders
    os.makedirs(series_folders)
except FileExistsError:
    print(’Folder exists’)
```

Use os Module

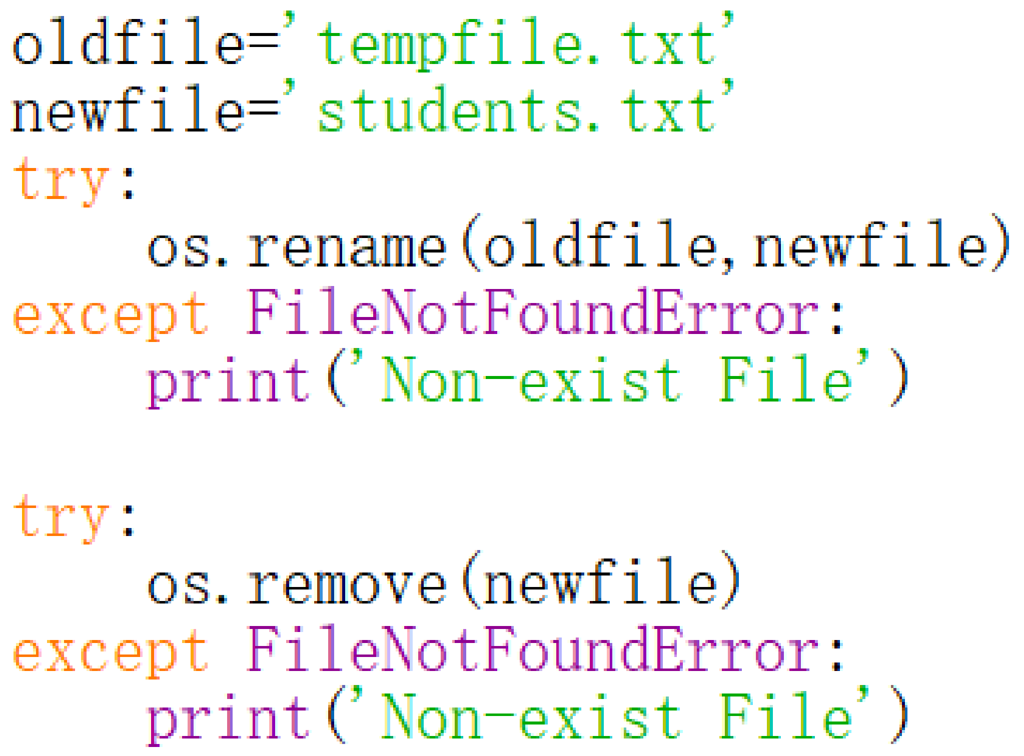
Remove an empty folder:



```
import os

folder='Folder 1'
try:
    # Remove empty folder
    os.rmdir(folder)
except FileNotFoundError:
    print('Non-exist')
```

Use os Module

Manipulate a file:



```
oldfile=’tempfile.txt’
newfile=’students.txt’
try:
    os.rename(oldfile,newfile)
except FileNotFoundError:
    print(’Non-exist File’)

try:
    os.remove(newfile)
except FileNotFoundError:
    print(’Non-exist File’)
```