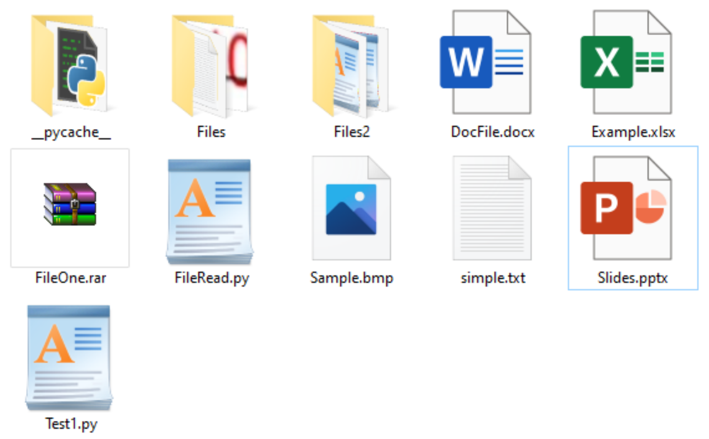
Reading and Writing Files

Computer Files

- When we turn off the computer anything stored in main memory is erased. All data used in our programs will be lost.

- Not to lose data, we need to save data as a file in secondary memory such as hard disk. Secondary memory is not erased when we turn of the computer.

- A *computer file* is a computer resource for recording data in a computer storage device, primarily identified by its file name (Wikipedia). Ex: test.py, sample.txt, lab.pdf, report.doc, song.mp3, etc.



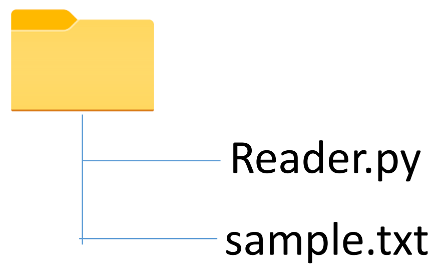
Handling Files

- In Python, we can read, write, update, and create files.

- The key function to work with files is the open() function.
  - open (*filename, mode*), where *filename* is a required parameter and it is the path to the file.
  - open() returns the file object.

- After using a file, it should be closed using the close() function.

```
f = open(‘sample.txt’)    # sample.txt is in the same directory
f.close()
```



Handling Files

open (*filename, mode*)

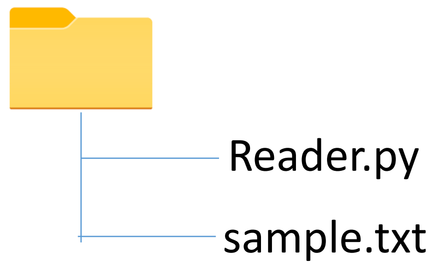
- There are four *modes* for opening a file:
  - r - to read - *Default value*. Opens a file for reading, error if the file does not exist.
  - a - to append - Opens a file for appending, creates the file if it does not exist.
  - w - to write - Opens a file for writing, creates the file if it does not exist.
  - x - to create - Creates the specified file, returns an error if the file exists.
- Also, we can specify if the file should be handled as binary or text mode
  - t - text - Text mode (*Default value*).
  - b - binary - Binary mode (e.g. images).

```
f = open(‘sample.txt’) is equivalent to
f = open(‘sample.txt’, ’rt’)   # because r and t are default values.
```

Reading an Entire File

- To read a file, we need to open it with mode ‘r’ as a second parameter
- The read(*n*) function returns *n* of bytes from the file. Default is -1 which means the whole file.

```
f = open(‘sample.txt’, ‘r’)      # error if the file does not exist
text =  f.read()
print(text)
f.close();
```



Python file methods: https://www.w3schools.com/python/python_ref_file.asp

Reading an Entire File

```
f = open('sample.txt', 'r')
print(f.read(15))   # read 15 bytes
print(f.read(12))   # read next 12 bytes
print(f.read(10))   # read next 12 bytes
rem = f.read()
print(len(rem)
print(rem[2:5])
f.close();
```

Python file methods: https://www.w3schools.com/python/python_ref_file.asp

Reading an Entire File

```
f = open(‘sample.txt’, ‘r’)
print(f.read(15))   # read 15 bytes
print(f.tell())     # returns the current file position
print(f.read(15))   # read next 15 bytes
f.seek(5)           # change the file stream position to 5
print(f.read(15))
f.close();
```

Python file methods: https://www.w3schools.com/python/python_ref_file.asp

Reading Line by Line

```
f = open(‘sample.txt’)
while True:
        line = f.readline()
        if line ==‘’:
                break
        print(line)
```

```
f = open(‘sample.txt’)
for line in f:
        print(line)
```

- We can use the rstrip() function to eliminate extra blank lines.

Making a List of Lines from a File

```
filename = ‘sample.txt’
f = open(filename):
lines = f.readlines()
for line in lines:
        print(line.rstrip())
```

Writing to a File

- To write data to a file we need to open a file with mode ‘w’
- And use the write() function.
  - Creates the file if it does not exist

```
filename = ‘sample.txt’
f = open(filename, ‘w’):
        f.write(‘Hello World!’)
```

Writing to a File

- We can write only *strings* to a text file. If you want to store other variables in a text file, we have to convert them to string format.
- We can use the str() function.

```
x = 15
f.write(str(x))
```

- An alternative is to use the *format operator %.*

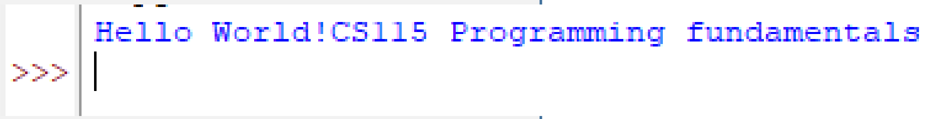
```
x = 15
f.write(‘%d’%x)
```

%d – integers
%f – floating point numbers
%s – strings
%.*n*f - floating point numbers with a *n* amount of digits to the right of the dot.

Writing Multiple Lines

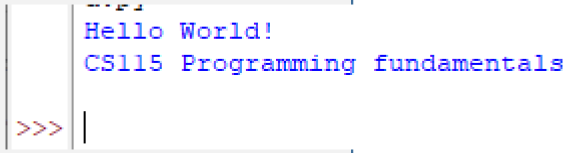
- The write() function doesn’t add any newlines to the text you write

```
f1 = open(‘sample.txt' , 'w')
f1.write(‘Hello World!’)
f1.write(‘CS115 Programming fundamentals’)
```



```
Hello World!CS115 Programming fundamentals
>>> |
```

```
f2 = open(‘sample.txt’, 'w'):
f2.write(‘Hello World!\n’)
f2.write(‘CS115 Programming fundamentals\n’)
```



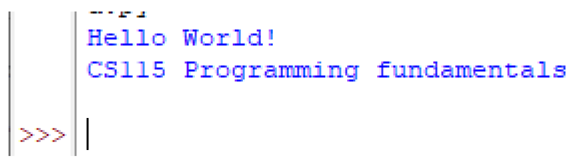
```
Hello World!
CS115 Programming fundamentals

>>> |
```

Writing Multiple Lines

```
f2 = open('sample.txt', 'w'):
f2.write('Hello World!\n')
f2.write('CS115 Programming fundamentals\n')
```

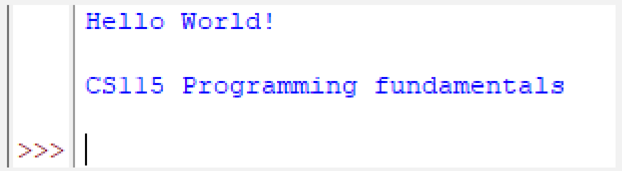
```
f = open('sample.txt','r')
print(f.read())
```



```
Hello World!
CS115 Programming fundamentals

>>> |
```

```
f = open('sample.txt', 'r'):
for x in f:
        print(x)
f.close()
```



```
Hello World!

CS115 Programming fundamentals

>>> |
```

As mentioned before, we can use the rstrip() function to eliminate extra blank lines.

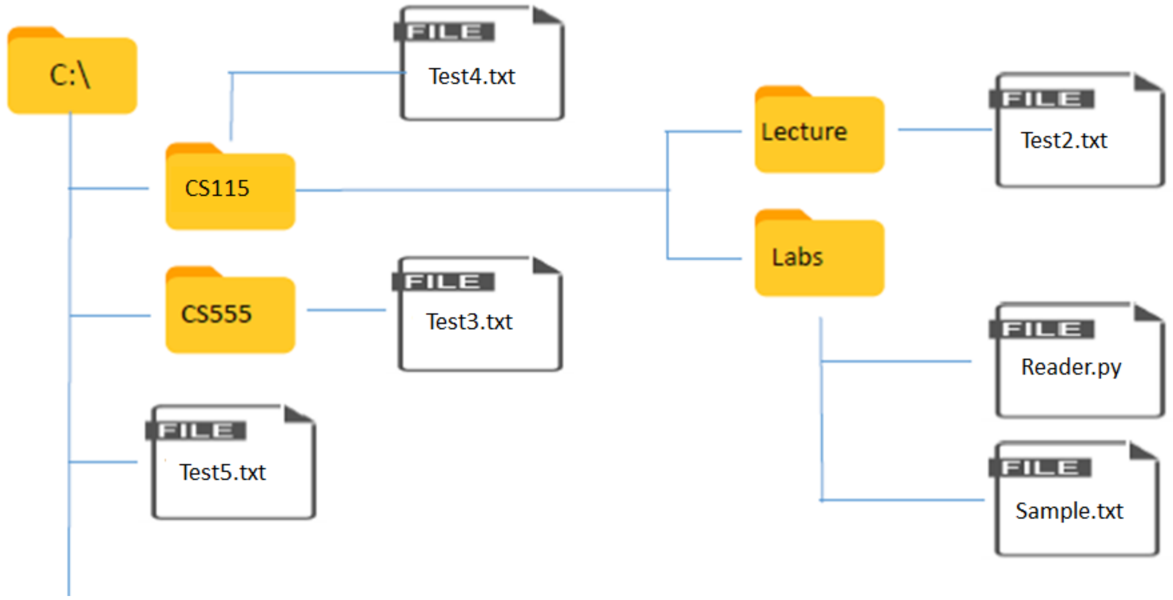
Appending to a File

- We can use ‘a’ mode to open a file for appending.
  - Creates the file if it does not exist

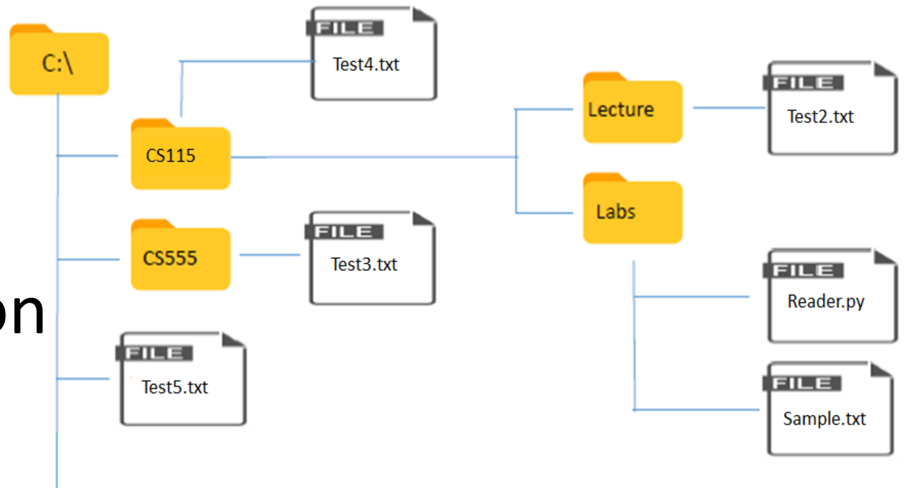
```
filename = ‘sample.txt’
f = open(filename, ‘a’):
        f.write(‘Hello World!’)
```

File Paths

- Files on disk are organized by a set of rules known as a file system. File systems are made up of files and directories, which are containers for both files and other directories.



File Paths



- When you call *open('Sample.txt')* in Reader.py, Python looks that file in the current directory, i.e., where *Reader.py* program is stored.

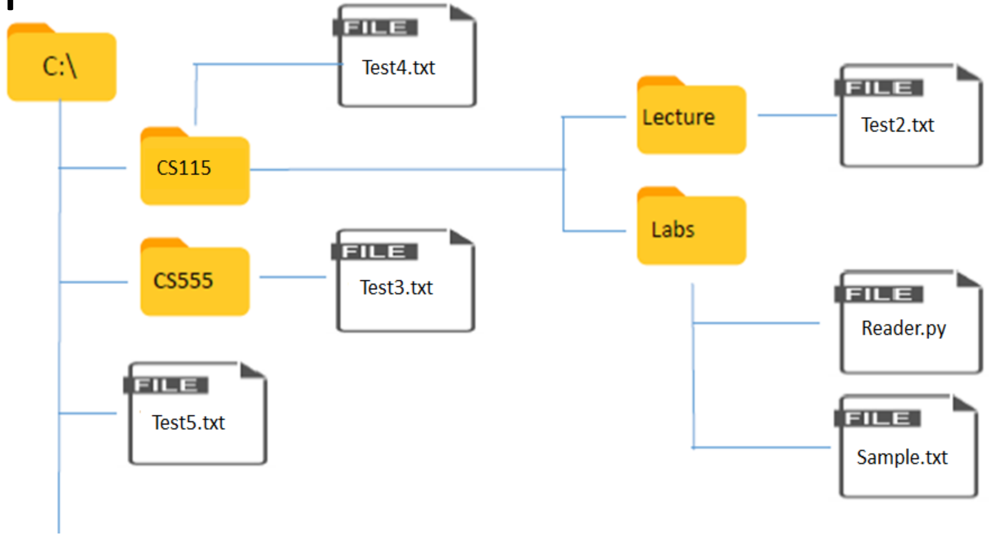
- Sometimes, the file you want to open will not be in the same directory as your program file. Therefore, you need to provide *a file path, a string that represents the location of a file.* A file path tells your program to look in a specific location on your system.
  - A *relative file path* tells Python to look for a given location relative to the directory where the currently running program file is stored.

  - An *absolute file path* tells Python exactly where the file is on your computer regardless of where the program that's being executed is stored.

File Paths - Example

- For example, to open Test2.txt as shown in the figure (on Windows).
  - C:\\CS115\\Lecture\\Test2.txt‘ (two backslashes as \ is escape character) or
  - ‘C:/CS115/Lecture/Test2.txt’ (Absolute paths) or
  - ‘/CS115/Lecture/Test2.txt’ (Absolute paths) or
  - ‘../Lecture/Test2.txt’ (Relative path)
    - double-dot (..) is used to move one directory up.

f = open(‘../../CS555/Test3.txt’, ’r’)



Exercise

- Write a function that takes two file objects A and B as parameters, and reads lines of A and writes to B with the lines in reversed order (i.e. the first line in file A becomes the last one in file B)