CS65: Introduction to Computer Science

File I/O

Md Alimoor Reza
Assistant Professor of Computer Science

Agenda

- Files and file I/O operations
  - Read
  - Write
  - Append

Three general steps for file handling

- Step 1: Open the file using a file variable
  - `file_variable = open(file_name, mode)`
- Step 2: Accomplish the operation using the file variable
  - `file_variable.read(string)`
  - `file_variable.readlines(string)`
  - `file_variable.write(string)`
- Step 3: Close the file using file variable
  - `file_variable.close()`

File Object

- **File object** is a variable associated with a specific file in the disk

```python
# reading 'test.txt' file from the current directory/
def read_file_v1():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.read()

    print('Reading the following:')
    print(my_str)

    file_handler.close()



# calling simple file reading operation
read_file_v1()
```

File on the disk (.txt file)

```
test.txt
100
100
3
100
9
9
```

Content after reading from our python program

```
>>> %Run lec15_demo.py

  Reading the following:
  100
  100
  3
  100
  9
  9
```

Open function

- *open() function* is used to open a file object and associates it with a file on the disk

- **File object** is a variable associated with a specific file in the disk

```
# reading 'test.txt' file from the current directory/
def read_file_v1():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.read()

    print('Reading the following:')
    print(my_str)

    file_handler.close()



# calling simple file reading operation
read_file_v1()
```

Mode of operation

- **File object** is a variable associated with a specific file in the disk
- *open() function* is used to open a file object and associates it with a file on the disk
- A string argument inside the *open()* specifies how the fill will be opened eg, 'r' —> reading, 'w' —> writing, 'a' —> appending

Mode

```
# reading 'test.txt' file from the current directory/
def read_file_v1():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.read()

    print('Reading the following:')
    print(my_str)

    file_handler.close()



# calling simple file reading operation
read_file_v1()
```

Name of the file

- **File object** is a variable associated with a specific file in the disk
- *open() function* is used to open a file object and associates it with a file on the disk
- A string argument inside the *open() function* specifies how the fill will be opened eg, 'r' —> reading, 'w' —> writing, 'a' —> appending
- Another string argument inside the *open() function* specifies the name of the file to be accessed

```
# reading 'test.txt' file from the current directory/folder
def read_file_v1():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.read()

    print('Reading the following:')
    print(my_str)

    file_handler.close()
```

Reading a file using *.readlines()* and for loop

```python
# reading 'test.txt' file from the current directory/
def read_file_v4():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.readlines()

    print('Reading the following:')
    print(my_str)
    print('Reading each line using for loop:')
    for each_line in my_str:

        print(each_line)

    file_handler.close()


# calling simple file reading operation
read_file_v4()
```

- Open an existing file on the disk in reading mode
- .readlines() method reads all lines as a list of strings
- Get each string from the list using value for loop
- File should be closed as a final operation

File on the disk (test.txt file)

test.txt

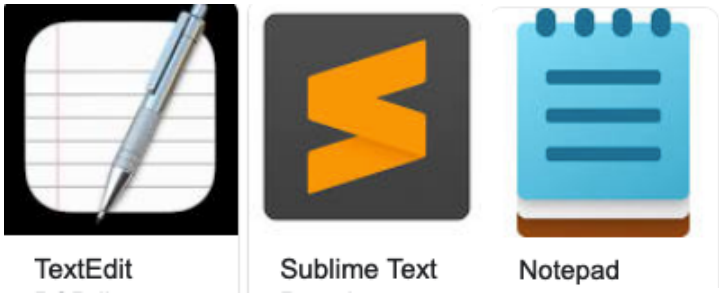
```
100
100
3
100
9
9
```

Content after reading from our python program

```
Reading the following:
['100\n', '100\n', '3\n', '100\n', '9\n', '9\n']
Reading each line using for loop:
100

100

3

100

9

9
```

Reading Data from a File

- Create a text file first using any editor of your choice
  - TextEdit (Mac OSX)
  - Notepad (Windows)



- Make sure you are in the same directory where your python file has been saved (or provide the correct path)

Reading Data from a File

- **Steps:**
  - Open the file with reading mode: **'r'** indicates that
  - Read the file with method ***.read()*** method
  - Close the file

Demo: Reading Data from a File

```python
# reading 'test.txt' file from the current directory/folder
def read_file_v1():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.read()

    print('Reading the following:')
    print(my_str)

    file_handler.close()



# calling simple file reading operation
read_file_v1()
```

test.txt

```
100
100
3
100
9
9
```

```
>>> %Run lec15_demo.py

  Reading the following:
  100
  100
  3
  100
  9
  9
```

Demo: Reading Data from a File

```python
# reading a file from a given directory/folder and file_name
def read_file_v2(file_path, file_name):

    file_handler = open(file_path + file_name, 'r')

    my_str = file_handler.read()

    print('Reading the following:')
    print(my_str)

    file_handler.close()


# callling file reading operation with file_path, file_name
file_path = '/Users/reza/Desktop/'
file_name = 'test.txt'

read_file_v2(file_path, file_name)
```

test.txt

```
100
100
3
100
9
9
```

```
>>> %Run lec15_demo.py

  Reading the following:
  100
  100
  3
  100
  9
  9
```

Reading a File using .readlines()

- **Steps:**
  - Open the file with reading mode: **'r'** indicates that
  - Read the file with ***.readlines()*** method
  - Close the file

- **Caution:**
  - make sure you are in the same directory or provide the correct path

Demo: Reading a File using .readlines()

```python
# reading 'test.txt' file from the current directory/folder
def read_file_v3():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.readlines()

    print('Reading the following:')
    print(my_str)

    file_handler.close()



# calling simple file reading operation
read_file_v3()
```

test.txt

```
100
100
3
100
9
9
```

```
>>> %Run lec15_demo.py

  Reading the following:
  ['100\n', '100\n', '3\n', '100\n', '9\n', '9\n']
```

Reading a File using .readlines() + for loop

- **Steps:**
  - Open the file with reading mode: **'r'** indicates that
  - Read the file and iterate through using a ***for loop***
  - Close the file

- **Caution:**
  - make sure you are in the same directory or provide the correct path

Demo: Reading a File using .readlines() + for loop

```python
# reading 'test.txt' file from the current directory/folder
def read_file_v4():

    file_handler = open('test.txt', 'r')

    my_str = file_handler.readlines()

    print('Reading the following:')
    print(my_str)
    print('Reading each line using for loop:')
    for each_line in my_str:

        print(each_line)

    file_handler.close()


# calling simple file reading operation
read_file_v4()
```

test.txt

```
100
100
3
100
9
9
```

```
Reading the following:
['100\n', '100\n', '3\n', '100\n', '9\n', '9\n']
Reading each line using for loop:
100

100

3

100

9

9
```

Reading a File using .readlines() + for loop

- **Steps:**
  - Open the file with reading mode: **'r'** indicates that
  - Read the file and iterate through using a ***for loop***
    - Eliminate ***'\n'*** (newline character) using ***.rstrip()*** method
  - Close the file

- **Caution:**
  - make sure you are in the same directory or provide the correct path

Demo: Reading a File using *.readlines()* + for loop + *.rstrip()* method

```python
def read_file_v5():

    file_handler = open('test_rstrip.txt', 'r')

    str_list = file_handler.readlines()

    print(str_list)
    print('-----------------------------------------------------------------')

    for each_line in str_list:

        print('Before: size = {} and string = {} '.format(len(each_line), each_line))

        # use string .rstrip() method to strip specific character
        # (in our case newline character '\n') from the end of the string
        each_line = each_line.rstrip('\n')

        print('After: size = {} and string = {} '.format(len(each_line), each_line))
        print('-----------------------------------------------------------------')

    file_handler.close()


# calling simple file reading operation
read_file_v5()
```

```
hello world
how are you
reza loves python programming
```

```
['hello world\n', 'how are you\n', 'reza loves python programming\n']
-----------------------------------------------------------------
Before: size = 12 and string = hello world

After: size = 11 and string = hello world
-----------------------------------------------------------------
Before: size = 12 and string = how are you

After: size = 11 and string = how are you
-----------------------------------------------------------------
Before: size = 30 and string = reza loves python programming

After: size = 29 and string = reza loves python programming
-----------------------------------------------------------------
```