

CS65: Introduction to Computer Science

File I/O (continued)



Md Alimoor Reza
Assistant Professor of Computer Science

Agenda

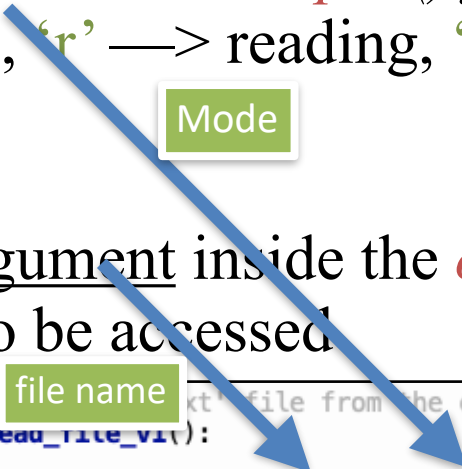
- Files and file I/O operations
 - **Read**
 - Write
 - Append

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()`

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 file 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



A diagram consisting of two blue arrows. The first arrow starts from the text 'Mode' in a green box and points to the second argument 'r' in the code snippet. The second arrow starts from the text 'file name' in a green box and points to the first argument 'test.txt' in the code snippet.

```
# read 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

- **Steps:**

- Open the file with reading mode: 'r' indicates that
- Read all lines using *.readlines()* method
- Iterate through using a *for loop*
- Close the file

- **Caution:**

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

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

```
# 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()
```

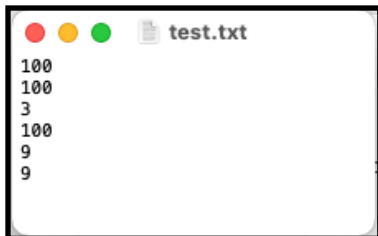
Open an existing file on the disk in reading mode

`.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)



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 a file from an arbitrary location in the disk

- **Steps:**
 - Provide the location of the file in your disk
 - as a function parameter of a *string* type
 - Open the file in reading mode: 'r' indicates that
 - Read using *.readlines()* method
 - Iterate through individual *string* with *for loop*
 - Close the file

Reading a file from an arbitrary location in the disk

```
# reading 'test.txt' file from a specific directory/folder
def read_file_v4(file_path):

    file_handler = open(file_path, 'r')

    my_list = file_handler.readlines()

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

        print(each_line)

    file_handler.close()

    return None

# calling simple file reading operation
file_path = '/Users/reza/Desktop/test.txt'
read_file_v4(file_path)
```

Open an existing file on the disk in reading mode

Location of an existing file in the disk

Demo: Reading a file from an arbitrary location in the disk

```
# reading 'test.txt' file from a specific directory/folder
def read_file_v4(file_path):

    file_handler = open(file_path, 'r')

    my_list = file_handler.readlines()

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

        print(each_line)

    file_handler.close()

    return None

# calling simple file reading operation
file_path = '/Users/reza/Desktop/test.txt'
read_file_v4(file_path)
```

Open an existing file on the disk in reading mode

Location of an existing file in the disk

File on the disk (test.txt file)



```
>>> %Run lec17_read_demo.py

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 and remove newline character using *.rstrip()* method

- **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

Reading a file and remove newline character using *.rstrip()* method

```
# reading & removing the newline characters with .rstrip()
def read_file_v5(file_path):

    file_handler = open(file_path, 'r')

    my_list = file_handler.readlines()

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

        print(each_line.rstrip('\n'))

    file_handler.close()

    return None

# calling simple file reading operation
file_path = '/Users/reza/Desktop/test.txt'
read_file_v5(file_path)
```

Open an existing file on the disk in reading mode

Removing newline character at the end

Location of an existing file in the disk

Demo: Reading a file and remove newline character using *.rstrip()* method

```
# reading & removing the newline characters with .rstrip()
def read_file_v5(file_path):

    file_handler = open(file_path, 'r')

    my_list = file_handler.readlines()

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

        print(each_line.rstrip('\n'))

    file_handler.close()

    return None

# calling simple file reading operation
file_path = '/Users/reza/Desktop/test.txt'
read_file_v5(file_path)
```

Open an existing file on the disk in reading mode

Removing newline character at the end

Location of an existing file in the disk

File on the disk (test.txt file)



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

```
>>> %Run lec17_read_demo.py

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
```

Read: different methods

Method with syntax	What it returns
<code>f1.read()</code>	Returns the entire file content as a string
<code>f1.readline()</code>	Returns one line of the file content as a string
<code>f1.readlines()</code>	Returns a list of strings where 1st list item is the content of the 1st line, 2nd list item is the content of the 2nd line, ... Last list item is the content of the last line

Opening file using **with** statement

- A **with** statement can be used to open a file, execute a block of statements, and automatically close the file at the end

```
# reading 'test_rstrip.txt' using with keyword
def read_file_v6():

    with open('test_rstrip.txt', 'r') as file_handler:

        str_list = file_handler.readlines()
        print(str_list)

        for each_line in str_list:

            # 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(each_line)

# calling simple file reading operation
read_file_v6()
```

Indentation (one tab here)

Different syntax now, ie, 'with' and 'as' keywords

No need to close the file

```
>>> %Run lec15_read_demo.py
['hello world\n', 'how are you\n', 'reza loves python programming\n']
hello world
how are you
reza loves python programming
```

Agenda

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

Writing data to a file

- **Steps:**
 - Open the file with writing mode: 'w' indicates that
 - Write into file with method `.write()`
 - Close the file
- **Caution:** Need to do *`file_handler.close()`* for Python to see the written content and free the reserved resources

Writing data to a file

```
# writing texts into a file 'test_write.txt' in the current dir
def write_file_v1():

    file_handler = open('test_write.txt', 'w')

    for i in range(3):

        my_str = input('Enter a new string: ')
        file_handler.write(my_str)
        file_handler.write("\n")

    # caution: do not forget to do this
    file_handler.close()

# calling simple file writing operation
write_file_v1()
```

Open a new file named 'test_write.txt' in the current directory (on the disk) in writing mode

.write() method adds content (as a string variable) in the file

File should be closed as a final operation to see the written content and free the reserved resources

Contents as received from user input

```
Enter a new string: hi there
Enter a new string: i am taking cs65
Enter a new string: i am learning to code in python
```

Newly created file ('test_write.txt') on the disk

```
test_write.txt
hi there
i am taking cs65
i am learning to code in python
```

Write: Alternate way of writing into a file

- Provide the **file object** (the variable associated with a specific file in the disk) inside the *print()* function as an argument

```
f1_var = open('test_write.txt', 'w')
print("Hi there!", file=f1_var)
print("How are you?", file=f1_var)
print("Python programming.", file=f1_var)
f1_var.close()
```

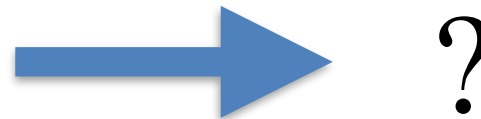
Exercise: Alternative way of writing into a file

- Convert the program using this alternative method

```
print(string, file=f1_var)
```

```
# writing texts into a file 'test_write.txt' in
def write_file_v1():
    file_handler = open('test_write.txt', 'w')
    for i in range(3):
        my_str = input('Enter a new string: ')
        file_handler.write(my_str)
        file_handler.write("\n")
    # caution: do not forget to do this
    file_handler.close()

# calling simple file writing operation
write_file_v1()
```



Agenda

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

Appending data to a file

- **Steps:**
 - Open the file with writing mode: 'a' indicates that
 - Write into file with method `.write()`
 - Close the file
- **Caution:** Need to do *`file_handler.close()`* for Python to see the written content and free the reserved resources

Appending data to a file

```
# appending the contents if a file exists with
# the same name as 'test_write.txt' (caution)
def append_file_v1():

    file_handler = open('test_write.txt', 'a')

    file_handler.write("\n-----\n")
    file_handler.write("this part along with the previous line \n")
    file_handler.write("is appended into the temp_write.txt file")
    file_handler.write("\n-----\n")
    # caution: do not forget to do this
    file_handler.close()

# calling simple file writing operation
append_file_v1()
```

Open an existing file named 'test_write.txt' in the current directory (on the disk) in appending mode

.write() method adds content (as a string variable) in the file

File should be closed as a final operation to see the written content and free the reserved resources

Previous content (before)

```
test_write.txt
hi there
i am taking cs65
i am learning to code in python
```

Current content (after)

```
test_write.txt
hi there
i am taking cs65
i am learning to code in python
-----
this part along with the previous line
is appended into the temp_write.txt file
-----
```