CMPE 30032
Module 2
File Handling in Python

Learning Goals/Objectives

Be able to read, comprehend, trace, adapt and create Python code that:

- Opens a file
- Reads data from a file into a program
- Writes data from a program into a file
- Appends data from a program into a file
- Closes a file

What Is File?

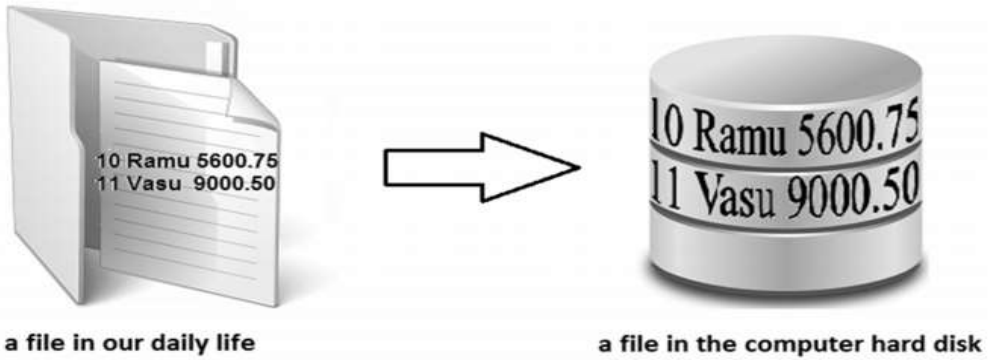
File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g., hard disk).

- Since Random Access Memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.

- When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order.

1. Open a file
2. Read or write (perform operation)
3. Close the file



a file in our daily life

a file in the computer hard disk

File Types?

In Python, there are two types of files. They are:

- Text files
- Binary files

Text files store the data in the form of characters. For example, if we store employee name “Jerald”, it will be stored as 6 characters and the employee salary 8900.75 is stored as 7 characters.

Text files are used to store characters or strings.

File Types?

Binary files store entire data in the form of bytes, i.e., a group of 8 bits each. For example, a character is stored as a byte and an integer is stored in the form of 8 bytes (on a 64-bit machine). When the data is retrieved from the binary file, the programmer can retrieve the data as bytes.

Binary files can be used to store text, images, audio and video. Image files are generally available in .jpg, .gif or .png formats.

We cannot use text files to store images as the images do not contain characters.

What Is File Handling?

Computer programs can import data from and export data to files outside the code.

Your code

Read - Get data from a file into your program

Write/Append - send data from your program into a file

External file

File Permissions

r	*Read* - File can be ‘looked at’ by the program but not changed. Error if the file doesn’t already exist.
a	*Append* - File can be added to. File is created if it doesn’t already exist.
w	*Write* - Overwrites existing data, starting from the beginning of the file. File is created if it doesn’t already exist.

Read From a File

Read From A File - The Algorithm

1. Connect to and open the file
   a. Give the file name and path
   b. Set the permissions for opening
2. Read the contents into a variable
3. Output the variable
4. Close the file.

Read All From A File - How To Cod

1. Create a new variable to store the contents of the file.

2. ‘open’ tells the program to open the file.

4. ‘r’ means **read only.** The program can look at data from the file but not change it.

```
myFile = open("test.txt", "r")


for line in myFile:
 print(line)
myFile.close()
```

3. Put the **whole filename**

Read From A File - How To Code

5. ‘for’ is another type of loop. It has a fixed length so does not need a condition.

6. ‘line in myFile’ sets the length of the loop to the number of lines in the external file.

```
yFile =      ("test.txt", "r")

for line in myFile:
 print(line)
myFile.close()
```

7. Outputs each line from the file one by one. The loop moves through each one.

8. **ALWAYS** close the file once you have finished with it.

Read Functions

read(n)

– Read at most n characters form the file. Reads till end of file if it is negative or None.

For example:

```
f = open("text.txt")
print (f.read())     # Reads the entire file
f.close()

f = open("text.txt")
print(f.read(5))   # Reads the first 5 characters of the file
f.close()
```

- **readline(n=-1)**

– Read and return one line from the file. Reads in at most n bytes if specified.

For example:

```
f = open("text.txt")
print(f.readline())#Reads the first line of the file
f.close()
```

Read Functions

- **readlines(n=-1)**
  - Read and return a list of lines from the file. Reads in at most n bytes/characters if specified.

For example:

```
f = open("text.txt")
data = f.readlines()
for line in data:
    print(line)
```

Write & Append to a File

Write To A File - How To Code

1. Create a new variable to store the contents of the file.

2. 'open' tells the program to open the file.

4. 'w' means **write.** If there's an existing file with this name Python will open it. If not it will create it.

```
myFile = open("test.txt", "w")


myFile.write("Andy")


myFile.close()
```

3. Put the **whole filename**

5. Put the data or variable to be written

Append To A File - How To Co

1. Create a new variable to store the contents of the file.

2. ‘open’ tells the program to open the file.

4. ‘a’ means **append.** This means ‘add to the end’ of the file.

```
myFile = open("test.txt", "a")

myFile.write(“Andy”)

myFile.close()
```

3. Put the **whole filename**

5. Put the data or variable to be written

Programming Exercises:

P-1(25 points). Write a Python program that reads a text file named numbers.txt that contains 20 integers. The program will create two other text file; the first text file will be named even.txt that will contains all even numbers extracted from the numbers.txt. The second text file will be named odd.txt that will contains all odd numbers extracted from the numbers.txt.

P-2 (25 points). Write a Python program that read a file containing the name of 20 students together with their GWA. The program will outputs the name of the student who got the highest GWA (including the GWA).

Programming Exercises:

P-3. (25 points). Write a method in python to write multiple line of text contents into a text file mylife.txt. See sample output:

```
Enter line: Beautiful is better than ugly.
Are there more lines y/n? y
Enter line: Explicit is better than implicit.
Are there more lines y/n? y
Enter line: Simple is better than complex.
Are there more lines y/n? n
```

P-4. (25 points). Write a method in python that will create two separate text files after reading the source text file named integers.txt that contains 20 integers. The first output file will be named double.txt containing the square of all even integers found in integers.txt and the second file will be named triple.txt containing the cube of all odd numbers found in the integers.txt.