Module-3

Files &Regular Expressions

Module-3: **Files**

- Types of Files,
- Creating and Reading Text Data,
- File Methods to Read and Write Data,
- Reading and Writing Binary Files,
- The Pickle Module,
- Reading and Writing CSV Files

Files:

- File handling is an important part of any web application.
- Python has several functions for creating, reading, updating, and deleting files.
- 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).
- 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.

✔ Open a file

✔ Read or write (perform operation)

✔ Close the file

File Handling:

- The key function for working with files in Python is the open() function.

f=open(filename, mode)

- The open() function takes two parameters; filename, and mode.
- There are four different methods (modes) for opening a file:
  - "r" - Read - Default value. Opens a file for reading, error if the file does not exist
  - "a" - Append - Opens a file for appending, creates the file if it does not exist
  - "w" - Write - Opens a file for writing, creates the file if it does not exist
  - "x" - Create - Creates the specified file, returns an error if the file exists
- In addition you can specify if the file should be handled as binary or text mode
  - "t" - Text - Default value. Text mode
  - "b" - Binary - Binary mode (e.g. images)

Access Modes of the Files:

Access Modes of the Files

Mode	Description
"r"	Opens the file in read only mode and this is the default mode.
"w"	Opens the file for writing. If a file already exists, then it'll get overwritten. If the file does not exist, then it creates a new file.
"a"	Opens the file for appending data at the end of the file automatically. If the file does not exist it creates a new file.
"r+"	Opens the file for both reading and writing.
"w+"	Opens the file for reading and writing. If the file does not exist it creates a new file. If a file already exists then it will get overwritten.
"a+"	Opens the file for reading and appending. If a file already exists, the data is appended. If the file does not exist it creates a new file.
"x"	Creates a new file. If the file already exists, the operation fails.
"rb"	Opens the binary file in read-only mode.
"wb"	Opens the file for writing the data in binary format.
"rb+"	Opens the file for both reading and writing in binary format.

Open a File :

- To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
f = open("demofile.txt", "r")
```

Note: Make sure the file exists, or else you will get an error.

```
>>> f = open("demofile.txt")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    f = open("demofile.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'demofile.txt'
```

Open a File :

- Assume we have the following file, located in the same folder as Python:

```
Hello everyone
welcome to programmin in python
This elective subject for sixth semester
Happy Learning
```

- To open the file, use the built-in open() function.
- The open() function returns a file object, which has a read() method for reading the content of the file:

```
>>> f=open('example.txt','r')
>>> print(f.read())
Hello everyone
welcome to programmin in python
This elective subject for sixth semester
Happy Learning
```

Open a File :

- By default the read() method returns the whole text, but you can also specify how many characters you want to return:

```
Hello everyone
welcome to programmin in python
This elective subject for sixth semester
Happy Learning
```

- Return the 9 first characters of the file:

```
>>> f=open('example.txt','r')
>>> print(f.read(9))
Hello eve
```

File Object Attributes

- When the Python *open()* function is called, it returns a file object called a file handler. Using this file handler, you can retrieve information about various file attributes

List of File Attributes

Attribute	Description
file_handler.closed	It returns a Boolean True if the file is closed or False otherwise.
file_handler.mode	It returns the access mode with which the file was opened.
file_handler.name	It returns the name of the file.

File Object Attributes

For example,

```
>>> f=open('example.txt','r')
>>> print(f.name)
example.txt
>>> print(f.closed)
False
>>> print(f.mode)
r
```

File Methods to Read and Write Data

List of Methods Associated with the File Object

Method	Syntax	Description
read()	file_handler. read([size])	This method is used to read the contents of a file up to a size and return it as a string. The argument *size* is optional, and, if it is not specified, then the entire contents of the file will be read and returned.
readline()	file_handler.readline()	This method is used to read a single line in file.
readlines()	file_handler.readlines()	This method is used to read all the lines of a file as list items.
write()	file_handler. write(string)	This method will write the contents of the string to the file, returning the number of characters written. If you want to start a new line, you must include the new line character.
writelines()	file_handler. writelines(sequence)	This method will write a sequence of strings to the file.
tell()	file_handler.tell()	This method returns an integer giving the file handler's current position within the file, measured in bytes from the beginning of the file.
seek()	file_handler. seek(offset, from_what)	This method is used to change the file handler's position. The position is computed from adding *offset* to a reference point. The reference point is selected by the *from_what* argument. A *from_what* value of 0 measures from the beginning of the file, 1 uses the current file position, and 2 uses the end of the file as the reference point. If the *from_what* argument is omitted, then a default value of 0 is used, indicating that the beginning of the file itself is the reference point.

Read Lines:

- You can return one line by using the readline() method:

```
Hello everyone
welcome to programmin in python
This elective subject for sixth semester
Happy Learning
```

- Read one line of the file:

```
f=open('example.txt','r')
print(f.readline())
print(f.readline())
```

By calling readline() two times, you can read the two first lines:

```
Hello everyone

welcome to programmin in python
```

Read Lines:

- You can return one line by using the readline() method:

```
>>> f=open('example.txt','r')
>>> print(f.readline())
Hello everyone

>>> print(f.readlines())
['welcome to programmin in python\n', 'This elective subject for sixth semester\n', 'Happy Learning\n']

>>> f=open('example.txt','r')

>>> for i in f:
        print(i)
```

- By looping through the lines of the file, you can read the whole file, line by line:

Close Files:

- It is a good practice to always close the file when you are done with it.
- Close the file when you are finish with it:

```
f=open('example.txt','r')
for i in f
   print(i)
f.close()
```

To check ,file is closed

- print(f.closed)

Note: You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

Write to an Existing Files:

- To write to an existing file, you must add a parameter to the open() function:
  - "a" - Append - will append to the end of the file
  - "w" - Write - will overwrite any existing content
- Open the file “example.txt" and append content to the file:

```
f=open('example.txt','a')
f.write(' Learning python is fun')
f.close()

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

Write to an Existing Files:

- Open the file “example.txt" and overwrite the content:

```
f=open('example.txt','w')
f.write(' Learning python is funny')
f.close()

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

Note: the "w" method will overwrite the entire file.

Create a New Files:

- To create a new file in Python, use the open() method, with one of the following parameters:
- "x" - Create - will create a file, returns an error if the file exist
- "a" - Append - will create a file if the specified file does not exist
- "w" - Write - will create a file if the specified file does not exist
- Create a file called "myfile.txt":

```
f=open('demofile.txt','x')
```

- Create a new file if it does not exist:

```
f=open('example.txt','w')
f.write(' Learning python is fun')
f=open('example.txt','r')
print(f.read())
```

File Positions:

- The tell() method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.
- The seek(offset[, from]) method changes the current file position. The offset argument indicates the number of bytes to be moved. The from argument specifies the reference position from where the bytes are to be moved.
- f.tell() # get the current file position
- f.seek(0) # bring file cursor to initial position

File Positions:

```
f=open('example.txt','r+')
str=f.read(10)
print('Reading..: ',str)

position=f.tell()
print('Current postion ',position)

position=f.seek(0,0)
str=f.read(10)
print('Again reading.. ',str)
f.close()
```

Renaming and Deleting Files:

- Python os module provides methods that help you perform file-processing operations, such as renaming and deleting files.
- To use this module you need to import it first and then you can call any related functions.

• The rename() Method

- The rename() method takes two arguments, the current filename and the new filename.
- Syntax: os.rename(current_file_name, new_file_name)

```
import os

# Rename a file from test1.txt to test2.txt
os.rename( "test1.txt", "test2.txt" )
```

Delete a File:

- To delete a file, you must import the OS module, and run its os.remove() function:
- Remove the file “text2..txt":
- Check if File exist:
- Check if file exists, then delete it:

```
#Deleting the file
import os
if os.path.exists('text2.txt'):
    os.remove('text2.txt')
else:
    print('The file doesnot exsits')
```

Delete Folder:

- To delete an entire folder, use the os.rmdir() method: Remove the file “text2.txt":
- Remove the folder "myfolder":

```
import os
os.rmdir("myfolder")
```

Note: You can only remove *empty* folders.

Reading and Writing Binary Files:

- We can usually tell whether a file is binary or text based on its file extension.
- This is because by convention the extension reflects the file format, and it is ultimately the file format that dictates whether the file data is binary or text.
- The string 'b' appended to the mode opens the file in binary mode and now the data is read and written in the form of bytes objects.
- **Write Python Program to Create a New Image from an Existing Image**

```
f1=open("flowers.jpg","rb")
f2=open("newflower.jpg","wb")
pic=f1.read()
f2.write(pic)
#f2.write(f1.read())
print(f"New Image is available with the name\n{f2}")
```

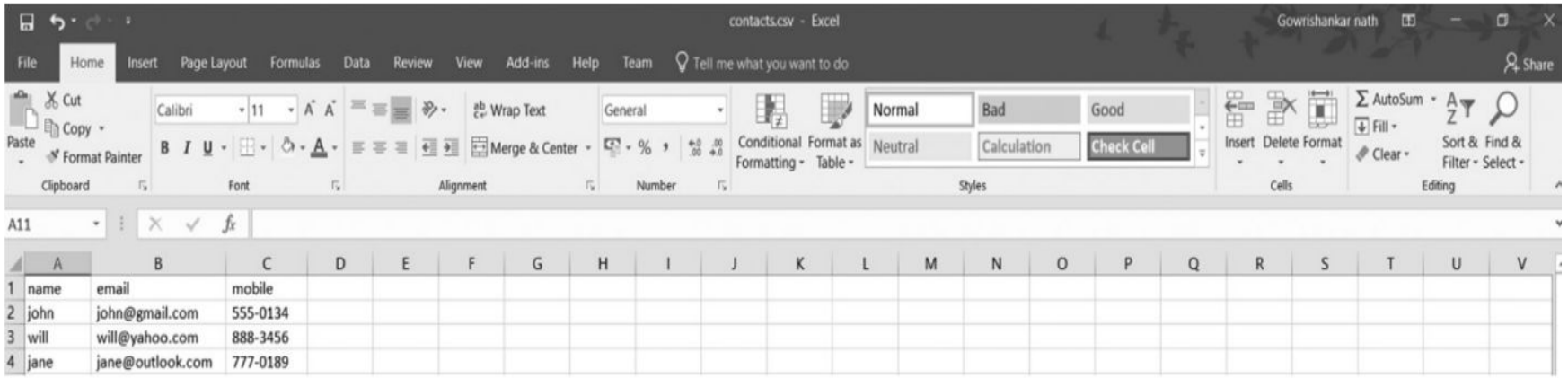
CSV Files:

- CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases.
- Since a comma is used to separate the values, this file format is aptly named Comma Separated Values. CSV files have .csv extensions
- Consider the "contacts.csv" file, which when opened in a text editor, the CSV file looks like this

contacts.csv

```
1  name, email, mobile
2  john, john@gmail.com, 555-0134
3  will, will@yahoo.com, 888-3456
4  jane, jane@outlook.com, 777-0189
```

- Opened in Excel, our example CSV file "contacts.csv" looks like this



Reading and Writing CSV Files:

- To read from a CSV file use csv.reader() method. The syntax is,

  csv.reader(csvfile)

- where csv is the module name and csvfile is the file object.
- To write to a CSV file, use the csv.writer() method. The syntax is,

  csv.writer(csvfile)

- where csv is the module name and csvfile is the file object.
- The syntax for writerow() method is,

  csvwriter.writerow(row)

Reading and Writing CSV Files:

- where the csvwriter is the object returned by the writer() method and writerow() method will write the row argument to the writer() method's file object.
- The syntax for writerows() method is,

```
csvwriter.writerows(rows)
```

- Here, the writerows() method will write all the rows argument (a list of row objects) to the writer() method's file object.
- Programmers can also read and write data in dictionary format using the DictReader and DictWriter classes, which makes the data much easier to work with.
- The syntax for DictWriter is,

```
class csv. DictWriter(f, fieldnames, extrasaction='raise')
```

Reading and Writing CSV Files:

- The syntax for DictReader is,

  class csv. DictReader(f, fieldnames=None, restkey = None)

- The syntax for DictWriter is,

  class csv. DictWriter(f, fieldnames, extrasaction='raise')

Program:

- Consider "Sample_Program.py" Python file. Write Python program to remove the comment character from all the lines in a given Python source file. Sample content of "Sample_Program.py" Python file is given below.

Sample_Program.py

```
1  print("This is a sample program")
2  #print("Python is a very versatile language")
```

```
1  def main():
2      with open("Sample_Program.py") as file_handler:
3          for each_row in file_handler:
4              each_row = each_row.replace("#", "")
5              print(each_row, end="")
6  if __name__ == "__main__":
7      main()
```

```
print("This is a sample program")
print("Python is a very versatile language")
```

Program:

- Write Python Program to Reverse Each Word in "secret_societies.txt" file. Sample Content of "secret_societies.txt" is Given Below.

secret_societies.txt

```
1  Secret Societies
2  Freemasons Illuminati
3  Rosicrucians Bilderberg Knights Templar
```

- Write a program to get a following result

- Expected result

```
terceS seiteicoS
snosameerF itanimullI
snaicurcisoR grebredliB sthginK ralpmeT
```

Program:

- Write Python Program to Count the Occurrences of Each Word and Also Count the Number of Words in a "quotes.txt" File. Sample Content of "quotes.txt" File is Given Below

quotes.txt

```
1  Happiness is the longing for repetition.
2  Artificial intelligence is no match for natural stupidity.
```

- Write a program to get a following result
- Expected result

```
The number of times each word appears in a sentence is
{'Happiness': 1, 'is': 2, 'the': 1, 'longing': 1, 'for': 2, 'repetition.': 1, 'Artificial': 1, 'intelligence': 1, 'no': 1, 'mat
ch': 1, 'natural': 1, 'stupidity.': 1}
```

Program:

- Write Python Program to Find the Longest Word in a File. Get the File Name from User. (Assume User Enters the File Name as "animals.txt" and its Sample Contents are as Below)

```
def read_file(file_name):
    with open(file_name) as file_handler:
        longest_word = ""
        for each_row in file_handler:
            word_list = each_row.rstrip().split()
            for each_word in word_list:
                if len(each_word) > len(longest_word):
                    longest_word = each_word
    print(f"The longest word in the file is {longest_word}")


def main():
    file_name = input("Enter file name: ")
    read_file(file_name)


if __name__ == "__main__":
    main()
```

```
Enter file name: animals.txt
The longest word in the file is Rhinocerose
```

Program:

- Consider a File Called "workfile". Write Python Program to Read and Print Each Byte in the Binary File.

```
1  def main():
2      with open("workfile", "wb") as f:
3          f.write(b"abcdef")
4      with open("workfile", "rb") as f:
5          byte = f.read(1)
6          print("Print each byte in the file")
7          while byte:
8              print(byte)
9              byte = f.read(1)
10
11
12 if __name__ == "__main__":
13     main()
```

```
Print each byte in the file
b'a'
b'b'
b'c'
b'd'
b'e'
b'f'
```

Program:

- Write Python program to read and display each row in "biostats.csv" CSV file. Sample content of "biostats.csv" is given below.

biostats.csv

```
"Name",      "Sex", "Age", "Height (in)", "Weight (lbs)"
"Alex",        "M",    41,          74,          170
"Bert",        "M",    42,          68,          166
"Elly",        "F",    30,          66,          124
"Fran",        "F",    33,          66,          115
```

```python
import csv


def main():
    with open("biostats.csv", newline="") as csvfile:
        csv_reader = csv.reader(csvfile)
        print("Print each row in CSV file")
        for each_row in csv_reader:
            print(",".join(each_row))


if __name__ == "__main__":
    main()
```

If csvfile is a file object, it should be opened with newline = "".

```
Print each row in CSV file
Name,      "Sex", "Age", "Height (in)", "Weight (lbs)"
Alex,        "M",    41,          74,          170
Bert,        "M",    42,          68,          166
Elly,        "F",    30,          66,          124
Fran,        "F",    33,          66,          115
```

Program:

- Write Python program to read and display rows in "employees.csv" CSV file that start with employee name "Jerry". Sample content of "employees.csv" is given below.

employees.csv

```
First Name,Gender,Start Date,Last Login Time,Salary,Bonus %,Senior Management,Team
Douglas,Male,8/6/1993,12:42 PM,97308,6.945,true,Marketing
Jerry,Male,1/10/2004,12:56 PM,95734,19.096,false,Client Services
Thomas,Male,3/31/1996,6:53 AM,61933,4.17,true,
Donna,Female,11/27/1991,1:59 PM,64088,6.155,true,Legal
Maria,Female,4/23/1993,11:17 AM,130590,11.858,false,Finance
Donna,Female,7/22/2010,3:48 AM,81014,1.894,false,Product
Jerry,Male,3/4/2005,1:00 PM,138705,9.34,true,Finance
```

- Write a program to get a following result
- Expected result

Program:

- Write Python Program to Read Data from ''pokemon.csv'' csv File Using DictReader. Sample Content of ''pokemon.csv'' is Given Below

pokemon.csv

```
1  Pokemon,Type
2  Bulbasaur,Grass
3  Charizard,Fire
4  Squirtle,Water
5  Pikachu,Electric
6  Rapidash,Fire
```

- Write a program to get a following result

- Expected result

```
Bulbasaur, Grass
Charizard, Fire
Squirtle, Water
Pikachu, Electric
Rapidash, Fire
```

Program:

- Write Python program to demonstrate the writing of data to a CSV file using DictWriter class

```
1  import csv
2  def main():
3      with open('names.csv', 'w', newline='') as csvfile:
4          field_names = ['first_name', 'last_name']
5          writer = csv.DictWriter(csvfile, fieldnames=field_names)
6          writer.writeheader()
7          writer.writerow({'first_name': 'Baked', 'last_name': 'Beans'})
8          writer.writerow({'first_name': 'Lovely', 'last_name': 'Spam'})
9          writer.writerow({'first_name': 'Wonderful', 'last_name': 'Spam'})
10 if __name__ == "__main__":
11     main()
```

Pickle module: Pickling and Unpickling of Objects

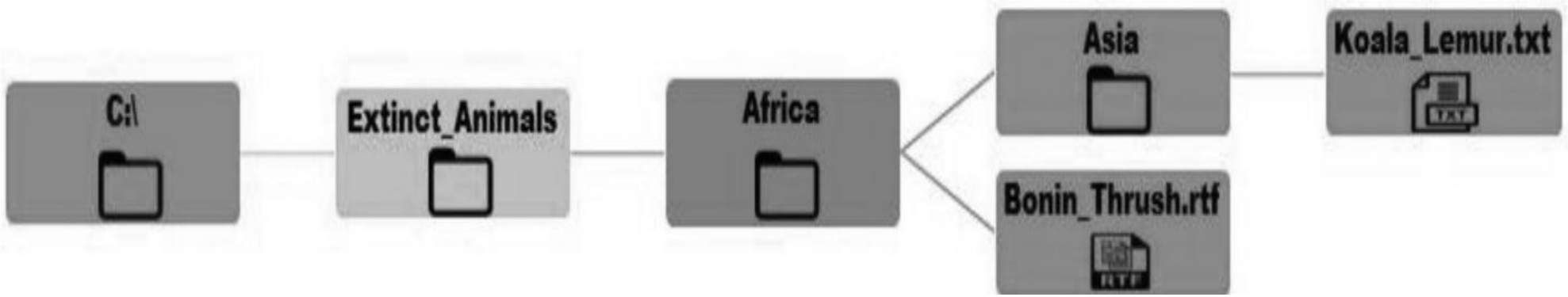
- Sometimes we have to write total state of object to the file and we have to read total object from the file.
- The process of writing state of object to the file is called **pickling** and the process of reading state of an object from the file is called **unpickling**.
- We can implement pickling and unpickling by using pickle module of Python.
- pickle module contains **dump()** function to perform pickling.

  pickle.dump(object,file)
- pickle module contains **load()** function to perform unpickling

  obj=pickle.load(file)

Program:

- Consider the File Structure Given Below. Write Python Program to Delete All the Files and Subdirectories from the Extinct_Animals Directory



- Write a program to delete all the files

Online Materials:

- https://www.w3schools.com/python/python_file_handling.asp
- https://www.programiz.com/python-programming/file-operation
- https://www.geeksforgeeks.org/file-handling-python/
- https://www.tutorialspoint.com/python/python_files_io.htm
- https://www.guru99.com/reading-and-writing-files-in-python.html