IE 509 Week 5

Python File Input/Output

Chapter 0:

- Data types (int, float, string, complex, etc) are implicitly declared by Python.
- **If, else** statements are used to execute a block of statements if the condition is true and another block of statements when the condition is false.
- **while, for** loops to repeatedly execute a set of statements
- **list, set** and **dictionary** are data types to store data

Handling files in Python

Most often we store data in files. Today we will learn how to access files, write data to files and read data from files.

Chapter I: Open and close a file

To open a file in writing mode, we use:

```
f1 = open('myfile.txt', 'w')
f1.close()
```

The first command opens a file in *write-only* mode. That is, we can only write to the file and not read from it.

- The first argument in the above open() function refers to the filename in the local directory. The second argument refers to the access modes(read, write, append),
- The close() function closes the file, and saves the data. Make sure to close the file whenever it is opened, else sometimes the data maybe lost.

A few popularly used access modes are as tabulated below.

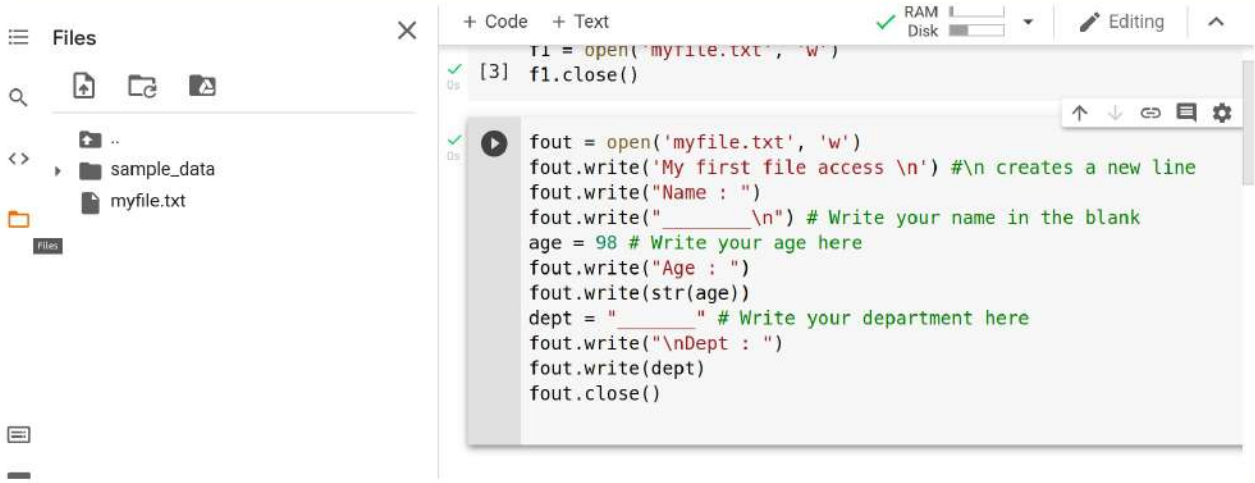
Mode	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

Chapter II: Writing to a file

Let's *write* something to the file:

```
fout = open('myfile.txt', 'w')
fout.write('My first file access \n') #\n creates a new line
fout.write("Name : ")
fout.write("_____\n") # Write your name in the blank
age = __ # Write your age here
fout.write("Age : ")
fout.write(str(age))
dept = "_____" # Write your department here
fout.write("\nDept : ")
fout.write(dept)
fout.close()
```

Now open the file by navigating to the files tab (click the folder icon located at left of the colab notebook) as shown in the image below.



For notebooks used on the local machine, one could go to the appropriate directory. If nothing regarding writing location is specified, it is the directory in which the notebook is saved.

You will see the following

My first file access
Name : _____
Age : __
Dept : _____

'\n' is used to add new lines. '\t' is used to add a tab space

Try yourself using myfile.txt created as above: Some hints from the later chapters of this sheet can be taken if the following can not be solved now.

Q1) Append another set of records to the file. The appended file should look like the following.

Name : <initial-name>
Age : <initial-age>
Dept : <initial-dept>

Name : <appended-name>
Age : <appended-age>

Dept : <appended-dept>

<initial-> refers to the records already written in myfile.txt before the new records are appended.

Q2) In the file created in Q1, write a new set of records and overwrite it so that only the created record exists in the file. The modified file should look like the following.

Name : <q2l-name>

Age : <q2-age>

Dept : <q2-dept>

Q3) What modes were used in Q1 and Q2? Can other modes be used for the same task?

Chapter III: Reading a file

1. **read(), seek(), tell()**

Let us now read from the file we have just created.

```
fin = open('myfile.txt', 'r')
string = fin.read(5) # Reads the first 5 characters from the file
print(string)
string = fin.read(4) # Reads the next 4 characters from the file
print(string)
pos = fin.tell()
print(pos)
fin.seek(20, 0) # Moves the current file position  to 20
string = fin.read(30) # Reads 30 characters from the current position
print(string)
fin.close()
```

Try yourself:

Q1) What will happen if seek() is used and there is nothing left to read?

Q2) Check what happens if seek() is used in an empty file?

Q3) Figure out what the second argument in seek() command does.

2. **next()**

next() is typically used in loops to return the next input line.

```
fin = open('myfile.txt')
for i in range(3): #range must be <= number of lines
      f = next(fin)
      print(f)
fin.close()
```

3. **readline()**

readline() returns the next line.

```
fin = open('myfile.txt')
print(fin.readline()) # returns and displays the first line
print(fin.readline(3)) # returns and displays first 3 characters in 2nd line
print(fin.readline()) # returns and displays the rest of the 2nd line
fin.close()
```

4. **readlines()**

readlines() returns the entire file.

```
fin = open('myfile.txt')
list_1 = fin.readlines() # returns the entire file as a list, each line is
an element
list_2 = fin.readline() # returns "" as we are at the end of file
fin.seek(0,0) # Moves current file position to start
print(list_1)
print(list_2)
```

Now let's read and print each line separately.

```
flines = fin.readlines()
for fline in flines:
      print(fline)
fin.close()
```

Chapter IV: Appending at end of file

Suppose we open the file in ‘a’ mode.

```
f1 = open('myfile.txt','a')
f1.write('\nTopics Learnt using a\n') # appends string to end of file.
f1.close()
```

Suppose we open the file in ‘w+’ mode:

```
f1 = open('myfile.txt','w+')
f1.write('\nTopics Learnt using w+\n') # appends string to end of file.
f1.close()
```

You can open the text file to see where the contents are written and how they change depending on the mode.

Chapter V: writelines()

Suppose we have a list of strings to write to a file. We do it in two ways.

```
mylist = ['PythonIntro\n', 'if-else\n', 'Loops\n', 'Strings\n']
f1 = open('writelines.txt', 'w+')
f1.write('Longway\n')
for string in mylist:
      f1.write(string)
f1.write('Shortcut\n')
f1.writelines(mylist)


f1.seek(0,0)
flines = f1.readlines()
f1.close()
for fline in flines:
      print(fline)
```

Try yourself: Write the following list of numbers to a new text file “numbers.txt”. Each number must be in a new line. [12, 23, 90, 45, 24, 24, 35, 56, 8, 22, 79, 15]. Save this file as it will be used in one of the submission exercises.

Chapter VI:

Now, let’s read the file numbers.txt as created in the previous exercise.

```
f1 = open('numbers.txt', 'r')
flines = f1.readlines()
f1.close()
numlist = []
for num in flines:
      numlist.append(int(num))
numlist
```

Exercises for submission
Create a notebook and solve the following. Try to solve each exercise in a single cell.

1. Write a program that creates a new file named lab7w.txt and writes the following contents(in the same format)
Course Name : IE509
Computer Programming and Algorithms
Language : Python

2. Padovan sequence is defined as follows. P(0)=P(1)=P(2) =1 and for n >= 3, P(n) = P(n-2) + P(n-3)
Write a program that generates the first n (taken as input) Padovan numbers and saves the numbers in a file named “padovan_n.txt”. For example in the input n is 500, the filename should be “padovan_500.txt”. Each Padovan number must be written in a new line.
Test the program with n = 500 and keep the resulting file for the next exercise.

3. Read the file created in exercise 2, and do the following:
a. Print the number of lines
b. Find how many numbers are a multiple of 3
c. Find how many numbers are prime

4. Consider an array A of n elements. The task is to sort the array in nondecreasing order. Consider the following algorithm.

```
for i ← 1 to n do
  x ← A[i];
  j ← i;
  while j > 0 and A[j-1] > x do
    A[j] ← A[j-1];
    j ← j – 1;
  end
  A[j] ← x
end
```

Write a program that accepts an array as input, first prints the original array, then prints the sorted array. Sort the array using the algorithm given above.

5. Modify the program in exercise 4 to read numbers from “numbers.txt” (which was created in Chapter V) and repeat the steps in exercise 4.