# Session 10

1. Reading Files
2. Writing files
3. Reading CSV files

## 1. Reading Files

There are no modules required as Python by default has the file input/output methods. when opening files we create an instance of the object as shown below

**NOTE: we must always close the file when we open it**

f = open('test. txt', mode = 'r')  
  
print(f.name)  
print(f.mode)  
  
f.close()

test.txt  
r

A file can be opened in multiple modes. read() and write() are most commonly used modes

#### There are 3 methods to read data from a file

1. read()
2. readline()
3. readlines()

The read() method reads the entire content of the file as a single string data

f = open('test. txt', 'r')  
  
x = f.read()  
print(x)  
  
f.close()

1) This is a test file!!  
2) This is the second line  
3) This is the third line  
4) This is the fourth line  
5) This is the fifth line  
6) This is the sixth line  
7) This is the seventh line  
8) This is the eight line  
9) This is the ninth line  
10) This is the tenth line

The readline() method reads a single line from the file at a time.

f = open('test.txt','r')  
  
x = f.readline()  
print(x)  
  
f.close()

1) This is a test file!!

This means that we can put the readline function inside a for loop to read the entire file.

f = open('test.txt','r')  
  
**for** i **in** range(0,10):  
 x = f.readline()  
 print(x)  
  
f.close()

1) This is a test file!!  
  
2) This is the second line  
  
3) This is the third line  
  
4) This is the fourth line  
  
5) This is the fifth line  
  
6) This is the sixth line  
  
7) This is the seventh line  
  
8) This is the eight line  
  
9) This is the ninth line  
  
10) This is the tenth line

**Note the added empty lines after every line read from the file. We will talk about this in a bit**

The readlines() method gives us a list with every element in the list as a line from the file.

f = open('test.txt','r')  
  
x = f.readlines()  
print(x)  
  
f.close()

['1) This is a test file!!\n', '2) This is the second line\n', '3) This is the third line\n', '4) This is the fourth line\n', '5) This is the fifth line\n', '6) This is the sixth line\n', '7) This is the seventh line\n', '8) This is the eight line\n', '9) This is the ninth line\n', '10) This is the tenth line']



#### Notice the \n at the end of every item in the list.

We cannot find these characters in the actual file.

* when you press the enter key it always inputs a new line character at the end. Hope you remember the new line character from the star pattern problem in session 5 (loops)
* This is the reason why we got that extra lines when using the readline() method.
* The print function puts its \n and also the string itself has a \n which causes a double-enter key press. we can solve this issue by simply making the print statement not put \n automatically.



f = open('test.txt','r')  
  
**for** i **in** range(0,10):  
 x = f.readline()  
 print(x,end = '')  
  
f.close()

1) This is a test file!!  
2) This is the second line  
3) This is the third line  
4) This is the fourth line  
5) This is the fifth line  
6) This is the sixth line  
7) This is the seventh line  
8) This is the eight-line  
9) This is the ninth line  
10) This is the tenth line



## Guess the output of the below code.💠

f = open('test.txt','r')  
  
x = f.read()  
print(x)  
y = f.read()  
print(y)  
  
f.close()



* seek()
* tell()

**with** open('test.txt') as f:  
 print(f.tell())  
 print(f.read())  
 print(f.tell())  
 f.seek(0)  
 print(f.tell())  
 print(f.read())  
 print(f.tell())

0  
1) This is a test file!!  
2) This is the second line  
3) This is the third line  
4) This is the fourth line  
5) This is the fifth line  
6) This is the sixth line  
7) This is the seventh line  
8) This is the eight line  
9) This is the ninth line  
10) This is the tenth line  
273  
0  
1) This is a test file!!  
2) This is the second line  
3) This is the third line  
4) This is the fourth line  
5) This is the fifth line  
6) This is the sixth line  
7) This is the seventh line  
8) This is the eight line  
9) This is the ninth line  
10) This is the tenth line  
273

## 2. Writing files

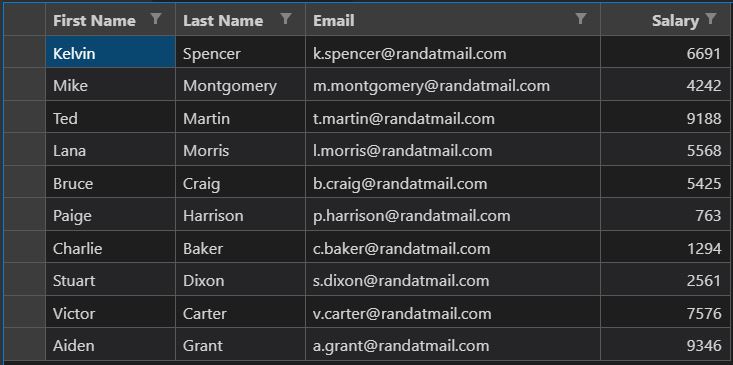
To write to a file we must open the file in write mode. The file name provided will be created automatically. If we provide a file name that already exists then everything in the file will be deleted.

**To edit a file we must use the append mode.**

Lets we want to create a copy of the test.txt file

r = open('test. txt', mode = 'r')   
  
w = open('test\_copy.txt',mode = 'w')  
  
x = r.read()  
  
w.write(x)  
  
r.close()  
w.close()

## 3. READING CSV FILE



CSV files are similar to Excel sheets which allows us to store data as rows and columns. As the name suggests (CVS - Comma Separated Values) are just simple text files with every column separated using commas and every row separated using new lines.

**This means that we can be using the open() function to read CVS files as well**

## TASK1: calculate the sum of salaries of all employees in the below CSV file

We Can first read the content of the file using the open() method

f = open(‘data. csv’)  
x = f.read()  
  
print(x)

"First Name", "Last Name", "Email", "Salary"  
"Kelvin", "Spencer", "k.spencer@randatmail.com", "6691"  
"Mike", "Montgomery", "m.montgomery@randatmail.com", "4242"  
"Ted", "Martin", "t.martin@randatmail.com", "9188"  
"Lana", "Morris", "l.morris@randatmail.com","5568"  
"Bruce", "Craig", "b.craig@randatmail.com", "5425"  
"Paige", "Harrison", "p.harrison@randatmail.com", "763"  
"Charlie", "Baker", "c.baker@randatmail.com", "1294"  
"Stuart", "Dixon", "s.dixon@randatmail.com", "2561"  
"Victor", "Carter", "v.carter@randatmail.com", "7576"  
"Aiden", "Grant", "a.grant@randatmail.com", "9346"



* As we can see all the columns are separated using commas so we can simply read a line split at, and select the salary column save the elements into a new list, and later do a sum of the elements.
* Also, notice the inverted quotes around the digits in the salary column which denotes that the salary is a string and requires to be converted to integers before conversion
* NOTE: Sometimes there is an extra pair of inverted quotes that might be required to be stripped.



f = open('data.csv')  
  
*# discarding the first line as it is the columns*  
x = f.readline()  
  
y = f.readline()  
  
print(y)

"Kelvin", "Spencer", "k.spencer@randatmail.com", "6691"

a = y.split(',')  
print(a)

['"Kelvin"', '"Spencer"', '"k.spencer@randatmail.com"', '"6691"\n']

b =a[3]  
print(b)

"6691"

c = b.strip('"\n')  
print(c)

6691

d = int(c)  
print(d)

6691

# Homework

### 1. Write a Python script to automatically send mail to all the mail IDs found in the CVS file that we just read. Use PyAutoGui to automate the process

#Task1: Write a program to read all the characters from the file(“data.txt”) and display in uppercase.

#TAsk2: Write a program to read all the characters from the file(“data.txt”) and display in uppercase.

# Task 3: Write a program in Python to find the number of characters in the first line of a text file (“data.txt”) using readlines().

# Task 5: Write a program to count the number of vowels in a file.

#Task6: Write a program to write the following data in a file “data.txt”.

#Task7: Write a program to read data from “data.txt” and write in another file “dest.txt”

#Task8: Write a program to read all data from “data.txt” and write in another file “dest.txt” except the vowels.

#Task9: Write a program to read data from the file “data.txt” and count the frequency of word input from the user.