



Session 10

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

1. Reading Files

There is 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

```
In [2]: 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

```
In [6]: 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 thenth line
```

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

```
In [5]: 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.

```
In [7]: 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 thenth 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.

```
In [8]: f = open('test.txt','r')

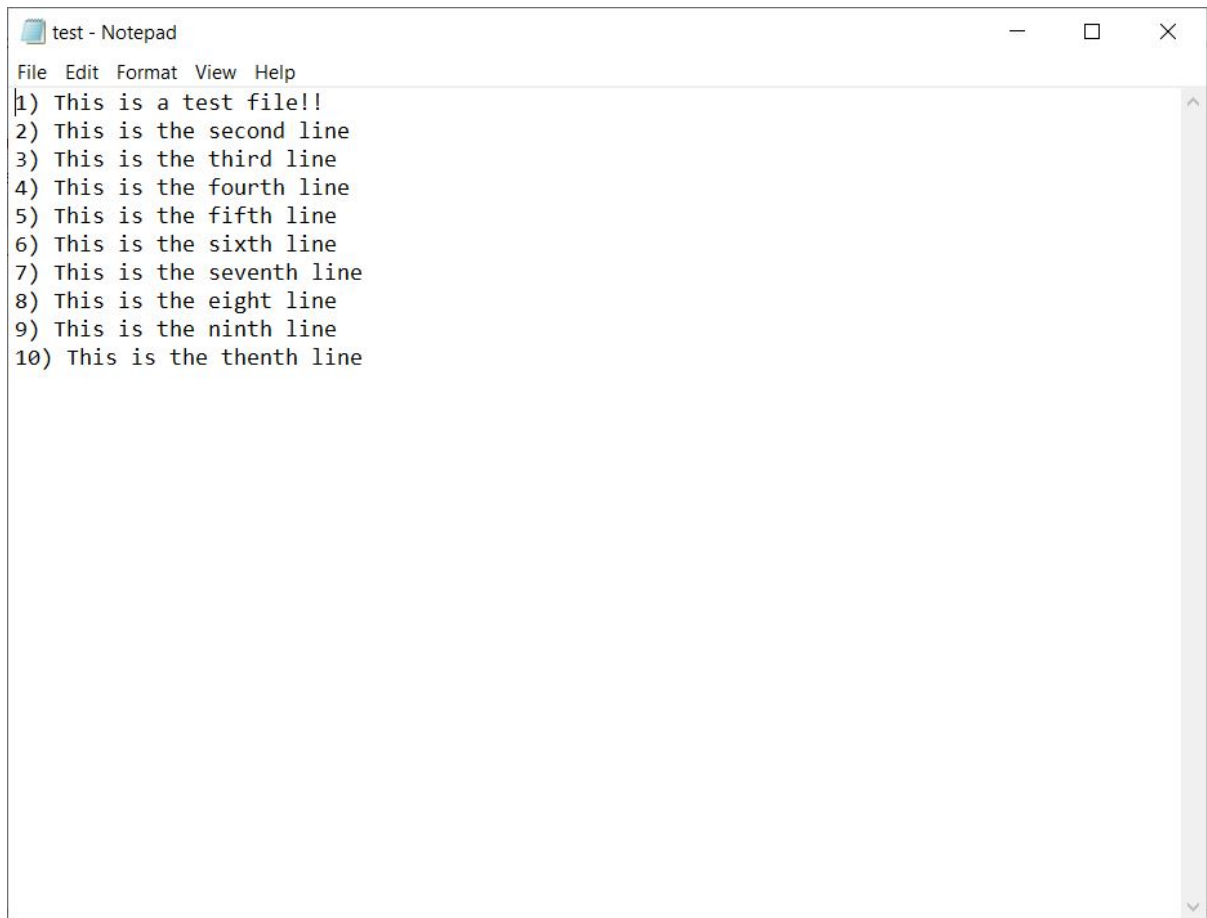
x = f.readlines()
print(x)

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 thenth line']
```

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

We cannot find these characters in the actual file.



```
test - Notepad
File Edit Format View Help
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 eighth line
9) This is the ninth line
10) This is the thenth line
```

- when you press 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 own `\n` and also the string itself has a `\n` this causes double enter key press. we can solve this issue by simply making the print statement to not put `\n` automatically.

In [9]:

```
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 thenth 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()

In [36]:

```
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 thenth 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 thenth line
273
```

2. Writing files

To write to a file we must open the file in write mode. The file name provide 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

```
In [1]: 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 using the `open()` function to read CVS files as well

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

	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

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

```
In [3]: 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 and 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 which might be required to be stripped.
-

In [4]:

```
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"
```

In [5]:

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

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

In [6]:

```
b = a[3]  
print(b)
```

```
"6691"
```

In [7]:

```
c = b.strip('"')  
print(c)
```

```
6691
```

In [8]:

```
d = int(c)  
print(d)
```

Home work

1. Write a python script to automatically send mail to all the mail ID's found in the cvs file that we just read. Use PyAutoGui to automate the process

In []: