Module 15

File Handling in Python

- A file is a container in a computer system for storing data
- Data is permanently stored
- Types of files
  - Text (Human readable) Eg. Text files
  - Binary (Machine readable) Eg. JPG, MP4, ...

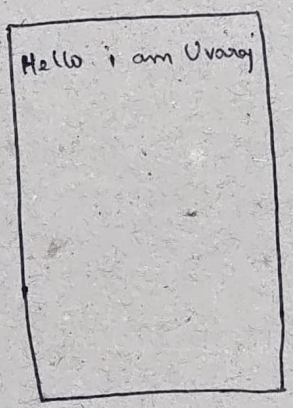
Why Files ?

Variables store the data temporarily, whereas file stores the data permanently.

Opening a file

- Open : returns a file handle
- In order to perform any kind of operations on file, first we open it.

```
f = open("name.txt")
type(f)
- io.TextIOWrapper
f.closed
False
f.read()
'Hello i am Uvaraj'
f.close()
f.closed
True
```

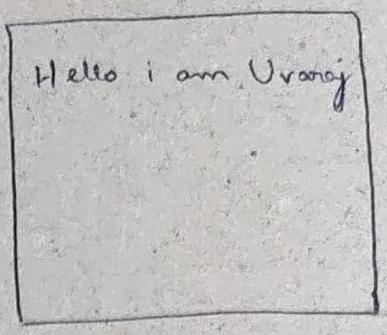


name.txt

Modes of opening a file

- 'r' : read only
- 'w' : write only
- 'a' : appending data at the end of file
- 'wt' : write text
- 'wb' : write binary
- 'rb' : read binary
- 'rt' : read text

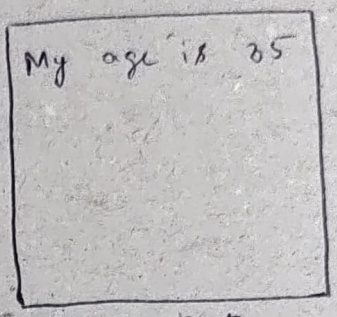
```
f = open ("name.txt")
f.read()
'Hello i am Uvaraj'
f.write ("My age is 35")
Unsupported operation : not writable
f.close()
```



name.txt

```
f = open ("name.txt","w")
f.write ("My age is 35")
12
f.read()
Unsupported operation : not readable
f.close()
```

Absence of f.close() will cause memory leak.



name.txt

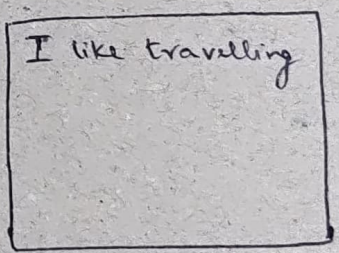
---

with open

- Most common way to perform operations on files
- It closes the file after performing operations

```
with open ("name.txt") as f :
    print (f.read())
My age is 35
f.closed
True
```

```
with open ("name.txt","w") as f :
    f.write ("I like travelling")
```

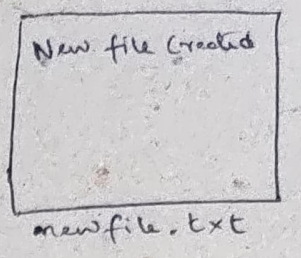


name.txt

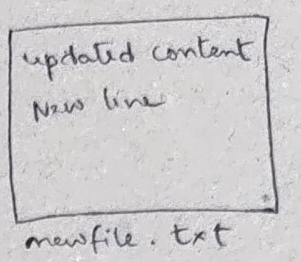
Writing in a file.

- You can write both text and binary file
- You can either write or append in a file

```
# Creating new file and write in it
with open("newfile.txt", "w") as f:
    f.write("New file created")
```




```
# Overwrite the existing file
with open("newfile.txt", "w") as f:
    f.write("Updated content \n")
    f.write("New line")
```



Reading a file

- using read() method, we can read a file
- It opens in read only format

```
with open("newfile.txt", "r") as f:
    data = f.read()
    print(data)
```

```
Updated content
New line
```

Some Methods

- read() : reads the whole data
- read(l) : reads data of length l
- tell() : tells you about the position of file handle
- seek() : it helps to reposition your file handle
- readlines() : reads data line-by-line

```
with open("newfile.txt", "r") as f:
    # reading first 7 characters
    data = f.read(7)
    print(data)
    # tell the position of file handle
    print(f.tell())
    # reading next 7 characters
    data = f.read(7)
    print(data)
```

Updated content
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

```
Updated
7
 content
```

```
with open("newfile.txt", "r") as f:
    # reading first 7 characters
    data = f.read(7)
    print(data)

    # Resetting the file handler to desired position
    f.seek(2)
    print(f.read(1))
```

Updated
dated

```
with open("newfile.txt", "r") as f:
    data = f.readlines()
    print(data)
```

['Updated content \n', 'New line']

```
with open("newfile.txt", "r") as f:
    print(f.readline())
    print(f.readline())
```

Updated content
New line

```
with open("newfile.txt", "r") as f:
    data = f.readlines()
    for i in data:
        print(i)
```

Updated content
New line

Binary Data

- To read binary data, we open file in "rt" mode
- To write binary data, we open file in "wt" mode

```
with open ("uvaraj.jpeg") as f :
    data = f.read()
    print (data)
```

UnicodeDecodeError : 'utf-8' codec can't decode byte 0xff in position 0 : invalid start byte

---

```
with open ("uvaraj.jpeg", "rb") as f :
    data = f.read()
    print (data)
```

Read Binary data

b '\xff\xd8\xff.....

.....

.....

.....

.....

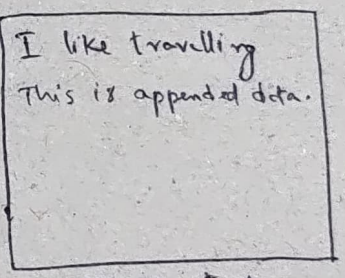
```
with open ("uvaraj.jpeg", "rb") as f :
    data = f.read()
    with open ("uvarajNew.jpeg", "wb") as d :
        d.write (data)
```

Write Binary data

Append

- This mode helps us to append in a file.

```
with open ("name.txt", "a") as f :
    f.write ("This is appended data.")
```



name.txt