1) Suppose we have created a file with 500 lines of data and the file object reference is “f”. Illustrate what each of these following operations does:

In [1]:

```
f = open("lines.txt", "rb") # b is imp here
```

In [4]:

```
f.seek(0) # beg of the file
```

Out[4]:

```
0
```

In [5]:

```
f.seek(100, 1) # 100 bytes ahead from current pos
line = f.readline()
line
```

Out[5]:

```
b'ac viverra ligula, at rhoncus neque. Nam fringilla nunc id ex finibus eges
tas. Nunc faucibus est vel ligula luctus placerat. Nullam nibh sem, fermentu
m in facilisis sed, efficitur id est. Mauris quis nisi in urna volutpat ulla
mcorper. Pellentesque sodales at nibh eget ultricies. Interdum et malesuada
fames ac ante ipsum primis in faucibus. Nulla finibus lacus vel quam tempor,
convallis volutpat orci ullamcorper. Donec dignissim tempus justo vitae temp
or. Nam ac porta nunc. Donec porttitor tristique orci, sed porttitor sapien
imperdiet vitae. Maecenas vehicula semper varius. Interdum et malesuada fame
s ac ante ipsum primis in faucibus. Donec feugiat eget nibh sed fermentum. A
enean accumsan nulla sit amet leo varius, et pharetra est ultrices.\r\n'
```

In [6]:

```
f.seek(-10, 2) # 10bytes behind from end of file
line = f.readline()
line
```

Out[6]:

```
b'ultricies.'
```

In [7]:

```
f.seek(0, 2) # end of file
line = f.readline()
line
```

Out[7]:

```
b''
```

In [12]:

```
f.tell() # current pos
```

Out[12]:

```
3400
```

In [13]:

```
f.close()
```

2) Consider a file with 100 lines of data and the file object reference is “f”. Illustrate what each of these following operations does:

In [14]:

```
f = open("lines.txt", "rb") # b is imp here
```

In [15]:

```
f.seek(0) # beg of file
line = f.readline()
line
```

Out[15]:

```
b'orem ipsum dolor sit amet, consectetur adipiscing elit. Aenean porta facil
isis nisi et dapibus. Sed ac viverra ligula, at rhoncus neque. Nam fringilla
nunc id ex finibus egestas. Nunc faucibus est vel ligula luctus placerat. Nu
llam nibh sem, fermentum in facilisis sed, efficitur id est. Mauris quis nis
i in urna volutpat ullamcorper. Pellentesque sodales at nibh eget ultricies.
Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla finibus
lacus vel quam tempor, convallis volutpat orci ullamcorper. Donec dignissim
tempus justo vitae tempor. Nam ac porta nunc. Donec porttitor tristique orc
i, sed porttitor sapien imperdiet vitae. Maecenas vehicula semper varius. In
terdum et malesuada fames ac ante ipsum primis in faucibus. Donec feugiat eg
et nibh sed fermentum. Aenean accumsan nulla sit amet leo varius, et pharetr
a est ultrices.\r\n'
```

In [16]:

```
f.seek(50, 1) # 50 bytes ahead from current pos
line = f.readline()
line
```

Out[16]:

```
b't amet et leo. Donec aliquet risus a finibus viverra. Fusce consequat enim
diam, non sagittis risus aliquet vitae. Aenean ac orci commodo felis viverra
malesuada. Aliquam id mollis lectus. Mauris porta imperdiet massa ac malesua
da. Duis feugiat dictum risus congue posuere. Vestibulum ultricies iaculis n
ulla vel consectetur. Vestibulum eget tortor eu magna molestie blandit ac ut
quam. Pellentesque habitant morbi tristique senectus et netus et malesuada f
ames ac turpis egestas. Fusce lacus orci, blandit dignissim magna a, tempus
fermentum lacus. Donec volutpat magna ipsum, volutpat aliquam elit eleifend
quis. Etiam lectus sem, congue id elementum ut, bibendum a nisl. Sed malesua
da ex eros, ut condimentum dolor cursus in. Etiam sodales sapien at maximus
fermentum.\r\n'
```

In [17]:

```
f.seek(-5, 2) # 5 bytes behind from eof
line = f.readline()
line
```

Out[17]:

```
b'cies.'
```

In [19]:

```
f.seek(0, 2) # eof
line = f.readline()
line
```

Out[19]:

```
b''
```

In [21]:

```
f.tell() # current pos
```

Out[21]:

```
3400
```

In [22]:

```
f.close()
```

3) Demonstrate Object Serialization in python by creating a custom class called Employee. Employee will store Employee name, age, salary, married and having kid. Save it and load it up into a separate object and display the new object.

In [23]:

```
import pickle
```

In [27]:

```
class Employee:

    def __init__(self, name, age, salary, married, having_kid):
        self.name = name
        self.age = age
        self.salary = salary
        self.married = married
        self.having_kid = having_kid

    def __str__(self):
        string = " ".join([
            self.name,
            str(self.age),
            str(self.salary),
            str(self.married),
            str(self.having_kid)
        ])
        return string
```

In [28]:

```
e = Employee("Deven", 20, 10000, False, False)
serial_e = pickle.dumps(e)
print(e, serial_e)
```

```
Deven 20 10000 False False b"\x80\x04\x95]\x00\x00\x00\x00\x00\x00\x00\x8c\x
08__main__\x94\x8c\x08Employee\x94\x93\x94)\x81\x94}\x94(\x8c\x04name\x94\x8
c\x05Deven\x94\x8c\x03age\x94K\x14\x8c\x06salary\x94M\x10'\x8c\x07married\x9
4\x89\x8c\nhaving_kid\x94\x89ub."
```

In [29]:

```
e2 = pickle.loads(serial_e)
print(e2)
```

```
Deven 20 10000 False False
```

4) Demonstrate Serialization in python by creating a custom class called Player. Player will store an id, name. Save it and load it up into a separate object and display the new object.

In [31]:

```
import pickle
```

In [34]:

```
class Player:

    def __init__(self, id_, name):
        self.name = name
        self.id = id_

    def __str__(self):
        string = " ".join([
            self.name,
            str(self.id)
        ])
        return string
```

In [36]:

```
p = Player(48, "Deven")
serial_p = pickle.dumps(p)
print(p, serial_p)
```

```
Deven 48 b'\x80\x04\x955\x00\x00\x00\x00\x00\x00\x00\x8c\x08__main__\x94\x8c
\x06Player\x94\x93\x94)\x81\x94}\x94(\x8c\x04name\x94\x8c\x05Deven\x94\x8c\x
02id\x94K0ub.'
```

In [37]:

```
p2 = pickle.loads(serial_p)
print(p2)
```

```
Deven 48
```

5) Implement a program to copy one python script into another by removing all the comment lines from the source file to destination file.

In [39]:

```
with open("read.txt", "r") as rf:
    with open("write.txt", "w") as wf: # or can put ", open('w.t', "w") in the above line"

        for line in rf:
            line_of_code = line.strip()
            if not line_of_code.startswith("#"):
                wf.write(line_of_code + '\n')

with open("read.txt", "r") as of:
    print("Old File: ", of.read(), sep='\n')

with open("write.txt", "r") as nf:
    print("New File: ", nf.read(), sep='\n')
```

```
Old File:
# welcome to this first program
print("Tony Stark created the Arc Reactor inside a cave")
a=5
print(f'The value of a is: {a}')
# program end

New File:
print("Tony Stark created the Arc Reactor inside a cave")
a=5
print(f'The value of a is: {a}')
```

6) Implement a program that generates a Quiz and uses two files-Questions.txt and Answers.txt. The program opens Questions.txt and reads a question and displays the question with options on the screen. The program then opens the Answers.txt file and displays the correct answers.

In [41]:

```
with open("questions.txt", "r") as qf:
    with open("answers.txt", "r") as af:

        for i in range(2):
            question = qf.readline()
            opt1 = qf.readline()
            opt2 = qf.readline()
            opt3 = qf.readline()
            opt4 = qf.readline()
            answer = af.readline()
            print(i+1, question)
            print("options: \n", opt1, opt2, opt3, opt4)
            print("answer: ", answer)
```

```
1 what is your name?

options:
 deven
 prakash
 paramaj
 omg

answer:  deven

2 what is your age?

options:
 39
 29
 20
 15

answer:  20
```

In [ ]: