Two Important Functions of File Handling

In [ ]:
```
Two important functions of file handling:
    1.tell()
    2.seek()
```

tell() Function

In [ ]:
```
--> tell function is used to return current position of the cursor or file pointer.

Syntax:
    f.tell()
```

Example

In [13]:
```
f=open("www.txt","r")
print(f.read(4))   #999A
print(f.tell())   #4
print(f.read(5))  #rnavp
print(f.tell())   #9

#999ArnavPrasad deepaliArnavPrasadShashankAbhishek
```

Out[13]:
```
9
```

seek(n) Function

In [ ]:
```
seek(n) -->seek function is used to change/move your cursor or file pointer in forward direction.

Syntax:
    f.seek(n)
```

In [16]:
```
f=open("www.txt","r")
f.seek(5)
print(f.read(4))   #999A
print(f.tell())   #4
print(f.read(5))  #rnavp
f.seek(15)
print(f.tell())   #9

#999ArnavPrasaddeepaliArnavPrasadShashankAbhishek
```

```
navP
9
rasad
15
```

In [22]:
```
with open("demo.txt","w") as f:
    f.write("Learning Python is Easy")
```

In [23]:
```
f=open("demo.txt","r")
print(f.read(5))
print(f.tell())
f.seek(10)
print(f.read())
print(f.tell())
```

```
Learn
5
ython is Easy
23
```

Ternary Operators or conditional operator

In [ ]:
```
It will simply check the condition and if the condition is true
then it will execute and return data otherwise else data will be return.
Syntax:
x=firsvalue if condition else second value

if the if condtion is true then firstvalue will ve returned otherwsie second value will be returned.
```

In [24]:
```
a,b=10,20
x=30 if a>b else 40
print(x)
```

```
40
```

In [26]:
```
#maximum of three number
a=10
b=60
c=30
x = a if a>b and a>c else b if b>c else c
print(x)
```

```
60
```

Bitwise Operators

In [ ]:
```
1 Byte --> 8 bits
```

In [ ]:
```
Steps:
    1.Convert the given integer into Binary
    2.Perform Bitwise Operator in it.
    3.Again convert the binary number into Decimal Format
```

In [ ]:
```
& --> and
| --> or
^ --> xor
>> --> left shift
<< --> right shift
```

In [ ]:
```
Bitwise operator are only applicable for integers and Boolean:
```

and or and Xor

In [ ]:
```
Bitwise  and --> &
if both bits are 1 then only result is 1 else result is 0

4 --> 00000100
5 --> 00000101

& --> 00000100   --> 4

Bitwise or --> |
if atleast 1 bit is 1 then answer is 1 else answer is 0

4 --> 00000100
5 --> 00000101

| --> 00000101   --> 5

Bitwise XOR --> ^
if bits are different then result is 1 else answer is 0


4 --> 00000100
5 --> 00000101

^ --> 00000001   --> 1
```

In [29]:
```
print(4&5)
print(4|5)
print(4^5)
```

```
4
5
1
```

left shift(<<)

In [ ]:
```
print(10<<2)
10 -->  00001010
        00101000

Steps:
    1.First convert the given number into 8-bit binary number
    2.In left shift remove left side bits based on the shifting value
    3.After removing the bits from left side add 00 bits at right side
    4.Convert binary bits into Decimal
```

In [30]:
```
10<<2
```

Out[30]:
```
40
```

In [31]:
```
20<<3
```

Out[31]:
```
160
```

In [ ]:
```
20<<3


20 - ->    1 0  1 0 0
           1 0  1 0 0 0 0  0 --> 160
```

Right shift (>>)

In [ ]:
```
10>>2

10 --> 00001010
       00000010



15>>3

15  --> 00001111
        00000001

Steps:
    1.First convert the given number into 8-bit binary number
    2.In right shift remove right side bits based on the shifting value
    3.After removing the bits from ride side add 00 bits at left side
    4.Convert binary bits into Decimal
```

In [34]:
```
15>>3
```

Out[34]:
```
1
```

In [1]:
```
10>>2
```

Out[1]:
```
2
```

In [ ]: