Python Programming [Instructor 1]

Fundamental Data Types Operators

* Fundamental data Types are :-

→ integer
→ floats
→ complex
→ Boolean
→ None
→ String

* Numbers are of 3 Types

→ int
→ float
→ complex

* <u>integer</u> [any real number without the decimal point]

Ex:- 10, 0, -10 [Decimal format]

* integers can be represented in binary, octal & hexa decimal values.

* Binary conversion automatic

10 Binary value = 1010

$\boxed{0b}$ 1010

↓ represents automatic

Ex:- a = 0b1010
print (a) ↲

output : 10

* print (bin(765)) ↲

output : ob1011111101

⇒ bin - should be proceded by ob, oB, allowed digits are 0 & 1, Any format to Binary - bin( )

* Octal representation

⇒ oct - should be proceded by 0O, Oo 0 to 7, Any format to Octal - oct( )

Ex:- * print (oct(9875)) ↲

output: 4012

*

* hexadecimal

=) hex - should be procedded by 0X, 0x, allowed digits are 0 to 15, Any formal to Hexa - hex()

* print (hexa (64641)) ↲

output: 0xfc98

* Only integers values are represented in these format
* Using eval function convertion can be done from one format to other

Ex:-
```
a = eval (hex(0b101001001001))
print (a, type (a))
```

```
output :  64664
          10377
```

Float :-

* Float can be represented only in decimal format
* Any value with a decimal point is float

Ex:- 10.0, 10.20, -10.20, 0.001

* There will be no short (or) long type only int

int - 8

short - 16

long - 32, 64

Ex:-
```
a = 10
Print (a, type (a))
b = 108764687100
Print (b, type (b))
```

```
C = 108 108 108 108 110 110 108 110 108 110124
print (c , type (c)
```

```
output:   10 <'class 'int'>
               < class 'int'>
               < class 'int'>
```

* sys.getsize of (a)
sys.get size of (b)
sys.getsize of (c)

Ex:-

```
importsys
a= 12345
print(a, type(a))
print (size =', sys.getsizeof(a)),
```

```
output: 12345, int
            size = w
```

Complex :-

Any number with format of a+bj is a is real & imaginary

Ex:-

```
a= 10 + 20j
print (a, type(a))
print (a. real)
print (a.imag)
```

```
output:-   10 +20j , complex

           10.0

           20.0
```

Boolean :-

* Values are "True" & "False".

* Boolean to int

  True = 1

  False = 0

* Boolean to Float

  True = 1.0

  False = 0.0

* Boolean to complex

  True = 1 + 0j

  False = 0 + 0j

Ex:- * print (complex(True, False)) ↵

```
output: (1+0j)
```

print (complex (True, True)) ↵

```
output : (1+1j)
```

Operators in Python

* Arthemetic
* Comparsion

- * Equality
- * logical
- * Bitwise
- * Assignment
- * Compound Assignment
- * Membership
- * Identity
- * Ternary

* Arthematic operator :-

+ , - , * , / , // , % , **

/ → Float division

// → Floor division

* True division - Floating point after division [Float division]

Ex:-

```
print (10/2)
print (10/3)
```

```
output :-    5.0
             3.3333333
```

*

```
print (10//2)
print (10//3)
```

```
output :-   5
            3
```

Comparsion Operator :-

Ex:- print ('Python' < 'python') ↲

```
output : True
```

* print (ord ('P')) ↲

```
output : 80
```

Equality Operator :-

Ex:- print ('A' == 65)
print ('A' == 'A') ↲

```
output : False
         True
```

* print (ord ('A' == 65)) ↲

```
output : True
```

Logical operator :- [&, or, not)

True and True = True
True and False = False
False and True = False
False and False = False

} and

True or True = True
True or False = True
False or True = True
False or False = False

or

not True = False
not False = True

not

* logical operator - works on the logic - logic should always be evaluated as True (or) False

Ex:- (10<20) and ('True' != 'True')

output: False

Bitwise Operator:-

* This operators works only on bits - 0[False] and 1[True]
* These operator works only on integer.
* All integer values are converted into bits representation & then bit by bit operator happens.
* Truth Tables are given below.

* Bitwise and Truth Tables

1 & 1 = 1

1 & 0 = 0

0 & 1 = 0

0 & 0 = 0

* Bitwise or Truth Tables

1 or[|]1 = 1

1 or / 0 = 1

0 or / 1 = 1

0 or / 0 = 0

* Bitwise xor Truth Tables

1 xor ^ 1 = 0

1 xor ^ 0 = 1

0 xor ^ 1 = 1

0 xor ^ 0 = 0

* Bitwise negation :-

= - (num + 1)

Ex:- print(-10)

= - 11

* print(- - 9) ↲

= -8

* Bitwise left shift "<<"
  Bitwise right shift ">>"

* This operator takes two operands
  * first one the number which you want to shift
  * second one is the number by how many bit's you want to shift

Ex:- 230 [LHS] 5 bit's
left hand side

① 230 : binary value

11100110

② Write 32/64 bit value

00000000000 00 0 00 0 0 0 0 0 11100110

③ Remove 5 bit's from ~~right~~ left & add right

print(230 << 5) ↲

7360

1hr 33 min