**27-02-2025**

**Literals**

**=========**

-> Literal is a value

-> Different types of literals are:

1) Integral Literals

2) Floating-point Literals

3) Complex Literals

4) Boolean Literals --> True, False

5) String Literals --> '', "", ''' '''

1) Integral Literals

---------------------

-> in 4-ways:

1) Decimal Literals

2) Binary Literals

3) Octal Literals

4) Hexadecimal Literals

-> Decimal Literal is base-10 literal.

we can able to define the decimal data using the total of 10 letters only.

those are: 0 to 9

Ex: 1, -1, 123456 etc.

-> The PVM (Python Virtual Machine) can understand any number/integer as decimal by default.

That is the reason, print() can print any integer literal in decimal format only.

-> Binary Literal is base-2 literal.

we can allow to define the number using only 2 letters.

Those are: 0 and 1

Ex: 10110001101, 101 ==> Invalid binary literals.

-> That we need to process the binary, that number must be prefixed with '0b' or '0B'

Ex: 0b10110001101, 0B101 ==> Valid Binary Literals

-> Octal literal is base-8 literal.

we can allow to define the octal literal with the total of 8 letters.

those are: 0 to 7

-> The Octal literal can always be prefix with '0o' or '0O'.

Ex: 0o1276, 0O123 ==> valid literals

-> Hexadecimal literal is the base-16 literal.

we can allow to define the hexadecimal with the total 16 letters

Those are: 0 to 9 (10) and alphabets (a/A to f/F) (6)

Here:

a ==> 10, b ==> 11, c ==> 12, d ==> 13, e ==> 14, f ==> 15

-> This is also called as "Alpha-numerical number".

-> Can always be prefixed with "0x" or "0X".

Ex: 0x1234, 0XAF123 ==> valid hexadecimals

a = 1023

b = 1010101

c = 0B1010101 # binary

d = 1276

e = 0O1276 # octal

f = 0x1234

g = 0XAF12

print(a)

print(b)

print(c) # binary number internally convert into decimal automatically

print(d)

print(e) # octal value internally convert into decimal automatically

print(f)

print(g) # hexadecimals can convert into decimal and then print by the print()

print(0xa,0xb,0xc,0xd,0xe,0xf)

# print(0xg)

2) Floating-point Literals

---------------------------

-> Floating point literals can be defined in two ways:

1) using decimal point

2) exponential/scientific format.

Note:

====

-> binary, octal and hexadecimal literals never use to define float.

Ex: 0b1011.101 ==> error

a = 123.234

b = 1e5

c = 1.2e-7

# d = 0b1101.1101

# d = 0o123.321

# d = 0x123.af

# d = 0b1101e-3

print(a)

print(type(b))

print(type(c))

3) Complex Literals

----------------------

Real part:

----------

real part can be with either binary or octal or hexadecimal or decimal or float or boolean

Imaginary part:

---------------

imaginary part can be with only decimal or float.

a = 0b11001 + 12.3j

b = 0o123 - 12j

c = 0x1af2 + 1.23j

# d = 123 - 0b11001j

# d = 123 + 0o123j

# d = 321 - 0Xaf12j

d = True + 123j

# e = 12 - Falsej

print(type(a))

print(type(b))

print(type(c))

print(type(d))