**Day-01**

**18-02-2025**

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

**Datatypes & Literals**

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

-> Python is dynamically typed programming language.

Because, the Python Virtual Machine (PVM) or Interpreter can understand the type of value/data based on the assignment.

a = 12

print(type(a))

-> Datatypes are used to describe the type of data/value which can be stored in variable.

-> Datatypes are classified into two types:

 1) Primitive types

 2) Non-primitive types (or) Reference types

-> The datatypes are based on the value can be categorized into two types:

 1) Immutable datatypes

 2) Mutable datatypes

id()

===

-> identity function

-> id() can be used to know address of the data (heap memory location)

**Syntax:**

 **id(variable/value)**

-> When a variable not allowed for the modification/change after the definition within the same address location. If you can try to modify the value of the variable, the new memory location can be get created by destroying the older address location. This kind of datatypes or variables are called as "Immutable Datatypes".

a = 121 # variable assignment

print("Before to change, the a = ")

print(a)

print(type(a))

print(id(a))

a = 12.1 # variable re-assignment

print("After the change, the a = ")

print(a); print(type(a))

print(id(a))

-> When a variable can be allowed for the modification within the same address location, those are called as "Mutable datatypes".

a = [10,20,30,40,50] # reference type

print("Before change:")

print(a)

print(id(a))

a[1] = 21

print("After change:")

print(a)

print(id(a))

**1) Primitive types**

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

-> Any fundamental or basic data representation is called as "primitive datatypes".

-> All primitive datatypes are "immutable types".

-> The primitive datatypes are:

 i) Integral Types

 ii) Floating-point Type

 iii) Complex Type

 iv) Boolean Type

 v) Text Type

**i) Integral Types**

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

-> any value without decimal point is called as "Integral data".

 -> The integral datatype in python is "integer". (No variants like other languages)

Integral Literals:

===========

-> Literal is called as "Value".

-> four types of integral literals:

 i) Decimal literal

 ii) Binary literal

 iii) Octal literal

 iv) Hexadecimal Literal

1) Decimal literal:

============

-> base-10 value, because to define any decimal value we can use only 10 letters. Those are 0 to 9.

Ex: 101, 123 etc.

2) Binary literal:

===========

-> base-2 value, because we can define a binary value with allowed letters. Those are: 0 and 1.

-> The binary number always be prefixed with '0b' '0B'.

Ex: 0b101010

3) Octal literal:

==========

-> base-8 value, because the octal value can be allowed to define with 8 letters only those are: 0 to 7

-> Octal values are always be prefixed with '0o' or '0O'.

Ex: 0o1723

4) Hexadecimal literal:

===============

-> base-16 value

-> alphanumeric value

-> allowed letters to define hexadecimal value are:

 0 to 9 ==> digits

 a to f/A to F ==> 10 to 15

-> The hexadecimal value can always be prefix with '0x' or '0X'

Ex: 0xaf123, 0x1234, 0XAFDE etc.

Note:

=====

1) Any number by default can understand by the PVM as "Decimal".

Ex: 123, 102, etc.

2) print() can allow to print any value (binary/octal/hexadecimal/decimal) in only the decimal format.

a = 121

b = 123456

c = 9876543210987654321

d = 1010101 # decimal

e = 0b1010101 # binary

f = 7123 # decimal

g = 0o7123 # octal

h = 0xaf123

print(type(a),type(b),type(c))

print(d)

print(e)

print(f)

print(g)

print(h)

**Day-02**

**19-02-2025**

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

**2) Floating-point Datatypes:**

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

float

=====

 -> a value/literal can define with the number including the decimal point

 Ex: 0.001, 123.234 etc.

float literals

========

 1) using decimal point

 ===============

 -> float literal can possible to define with only decimal literals.

 -> When we can consider the binary/octal/hexadecimal to define the float value, we can get "syntax error".

a = 123.234

# b = 0b11001.101

# b = 0o1723.23

# b = 0Xaf.12

print(type(a))

# print(type(b))

 2) exponential format/scientific format

 ========================

 Ex: exponential values ==> 123e5, 90.89e-2 etc.

 -> 123e5 ==> 123 X 10^5

 -> 90.89e-2 ==> 90.89 X 10^-2

 -> In the given exponential number,

 before the 'e' we have a value called as "mantissa".

 after the 'e' we have a value called as "exponent".

Note:

====

-> Any exponent value (if mantissa is either with decimal or float) can be a float by default.

a = 123e6

b = 321E7

c = 1.009e-2

print(type(a))

print(type(b))

print(type(c))

**3) Complex Type:**

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

-> pre-defined class for complex type is "complex".

-> complex data is the combination of real data and imaginary data.

-> Here: real data can be any integer or any float and imaginary data can be either decimal or float only.

-> Imaginary value in the complex number can always define with 'j' (imaginary value can be suffixed with 'j')

a = 123 + 123j

b = 100 - 200j

# c = 0b11001 - 0b101j

c = 0b11001 + 1.23j

d = 1.23 - 1.23j

e = 1e5 + 2e-3j

print(type(a))

print(type(b))

print(type(c))

print(type(d))

print(type(e))

**4) Boolean Type:**

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

-> pre-defined class ==> bool

-> boolean type has only two literals:

 1) True

 2) False

-> In python, the PVM can understand "True" as 1 and "False" as 0

 because, the python is dynamically typed programming language.

a = True

b = False

print(type(a))

print(type(b))

print(a+b)

print(a-b)

print(a\*b)

print(a/b)

**Text Type:**

**=======**

-> We have only one text based datatype i.e., "string".

-> String is the collection/group of characters

-> String in python can define with:

 single quotes

 double quotes

 triple quotes

Note:

=====

In python, there is no character data representation.

-> String type is a pre-define datatype

 we have a pre-defined class "str".

a = 'R'

b = 'Ravi'

c = "B"

d = "Ball"

e = '''Python'''

print(type(a))

print(type(b))

print(type(c))

print(type(d))

print(type(e))

Note:

====

-> In python, all datatypes built-in/pre-defined datatypes

That every datatype in python already included with a pre-defined class.

Q: How we can say the python is an Object oriented programming language?

========================================

Ans: That every data definition in python can understood as an object by PVM. Because every datatype is a pre-defined that means, for every datatype we have one pre-defined class.

Ex: Integer ---> class 'int'

Float ---> class 'float'