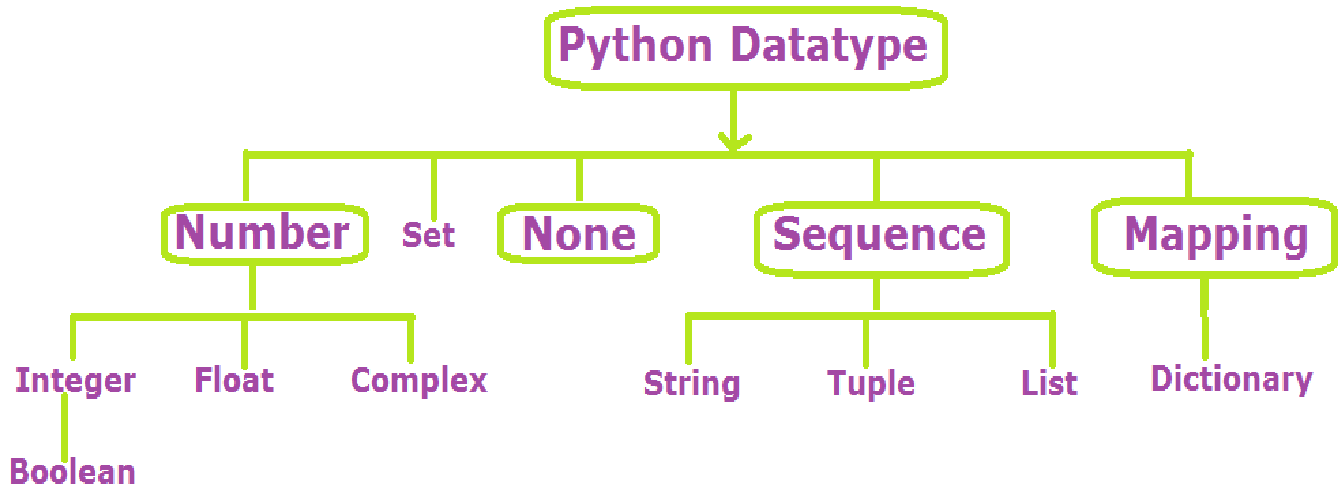
Python Fundamentals Part-2

In this chapter the following topics will be discussed in detail.

1. Datatype
2. Variable
3. Operator
4. Type Casting
5. Some Inbuilt functions
6. Types of Errors

Python Datatype

Datatype mean type of data that is being stored in memory during the program execution.



Integer: Stores only numbers without decimal point.
Example: 10, -20, 0

Float: Stores numbers with decimal point.
Example: 10.6, -20.4, 0.0

Complex: Stores integer or float type complex numbers
Example: 10j, -20j, 0j, 5+15j

Boolean: Stores only True or False Values.
Example: True, False

String: Stores sequence of characters that enclosed between single or double quotation marks.
Example: “hello”, ‘kv jjn’, ‘STD 01592’, “a@com”

List: Stores sequence of different types of Values. All values grouped together in [ ] brackets.
Example: [2,6,7,9], [4,”KVS”,18,”Raj”,2.5]

Tuple: Stores sequence of different types of Values. All values grouped together in ( ) brackets.
Example: (2,6,7), (4,”KVS”,18,”Raj”,2.5)

Dictionary: Stores sequence of different types of Values. All values grouped together in { } brackets.
Example:
{1:”Monday”, 2: “Tuesday”, 3: “Wednesday”},
{“name”:”Ajay”, “Age”:13, “Weight”:21.5}

Set: Stores sequence of different types of Immutable type of Values. All values grouped together in { } brackets.
Example: {2,6,7}, {4,”KVS”,18,”Raj”,2.5}

Immutable and Mutable Datatype

Immutable Datatype:

The immutable datatypes never allow to user to change their values. That mean the values once assign, cannot change in future.
Examples:

- Integer
- Float
- Boolean
- String
- Tuples

Mutable Datatype:

Mutable datatypes are those datatypes whose values can be changed. Or the values once assign, can change in future.
Examples:

- List
- Dictionary
- Set

Sets:

- Set created by using values separated by comma.
- Set elements are unordered and unindexed.
- Set is
- Does not allow to store duplicate values.
- Set cannot contain an element that is mutable.

Examples

```
Myset1={10,20,30,40}
Myset2={10,20,30,10,20,40}  # 10 and 20 are repeated
Myset3={10,[20,30],40}         # [20,30] is a List
print(Myset1)      #Output: {10,20,30,40}
print(Myset2)      #Output: {10,20,30,40}
print(Myset3)      #Output: Generate Error
```

Variables in Python

Variable:

The Variable is named location which is used to store values. The value of variable can be changed throughout the whole program execution. Characteristics of Variable are….

- Variable name should be declared before its use.
- Variable always reserve a memory location to store value in it.
- The datatype of variable will be as same the type of value (Literal) assigned to it.
- Value store at memory location can be read by using variable name.
- Variable name should be follow the rules to declare the identifiers.

Declaration of Variable:

A Variable should be declared before its use otherwise Python interpreter will generate error.

Syntax:

```
    Variable_Name = Value
```

Example:

```
    First_Name="Ajay"
    # Here First_Name is Variable name and "Ajay" is
      Value. The Value is of String type so the datatype
      of First_Name variable is also string.

    Age=15

    # Here Age is Variable name and 15 is Value. The
      Value is of integer type so the datatype of Age
      variable is also integer.
    City="Jaipur"              # Datatype is string
    Phone=9876543210           # Datatype is integer
    Principal=5000             # Datatype is integer
    Rate=2.5                   # Datatype is float
    Marks=[56,78,76,98,34]     # Datatype is List
    Boy= True                  # Datatype is Boolean
    Weight=None                # Datatype is None
```

Assignment of Variable:

Assignment mean to store the value at the location created by variable. Following are the methods to assignment of variable.

(A) Single Assignment

Syntax:

Variable_Name = Value / Variable / Expression /function

Where = is assignment operator.

At the left side of = operator, there will be variable always. At the right hand side, there may be any variable, value, expression or function.

Example:

```
A=10          # Value 10 assigned to Variable – A
B= A     #Value of Variable-A assigned to Variable – B
C=A+B # Result of expression A+B assigned to Variable – C
R=math.sqrt(25) # function’s value assigned to Variable – R
```

(B) Multiple Assignment

Syntax:

List of Variables separated by comma = List of Values separated by comma

Example:

A, B, C = 10, 20, 30

It means, A=10, B=20, C=30

NAME, AGE, CITY = “ALOK”, 15, “JAIPUR”

It means NAME=”ALOK”, AGE=15 and CITY=”JAIPUR”

(C) Multi Variable Assignment

Syntax:

Variable-1 = Variable-2 = Variable-3 = Value

Example:

A=B=C=100

Here Value 100 is assigned to of all three variables.

Global Variable Vs Local Variable

Global (Public) Variable is a variable that can access anywhere in whole program. In other words scope of a global variable is public.
The variable declared at the beginning of program is by default global in nature.
To declare a variable as global inside a local block, simply use the "global" keyword.
Example:

```
A=10
def local_area():
    global B
    B=100
    C=200
```

Variable A is Global by default

Variable B is now global

Variable C is Local

Operators in Python

The operators are symbols or words which can applied on variables/ values in an expression. Operators trigger some computation / action when they applied in expression

Types of Operators

- **(A) Arithmetical Operator**
- **(B) Relational Operator**
- **(C) Logical Operator**
- **(D) Assignment Operator**
- **(E) Short Hand Operator**
- **(F) Identity Operator**
- **(G) Membership Operator**
- **(H) Bitwise Operator**

(A) Arithmetical Operator

These operators used for mathematical operations. These operators always produce a numeric output.

Operator	Purpose	Example
+	Addition	10+3 Result: 13
-	Subtraction	10-3 Result: 7
*	Multiplication	10*3 Result: 30
/	Float Division	10/3 Result: 3.3333
//	Floor (Integer) Division	10//3 Result: 3
**	Exponential	10**3 Result: 1000
%	Modulus (Remainder after integer division)	10%3 Result: 1

(B) Relational Operator

These operators used between two values and always produce a Boolean output.
(Boolean Output: True, False)

Operator	Purpose	Example
<	Less than	10<5 Result: False
<=	Less than or equals to	10<=5 Result: False
>	Greater than	10>5 Result: True
>=	Greater than or equals to	10>=5 Result: True
==	Equals to	10==5 Result: False
!=	Not Equals to	10!=5 Result: True

(C) Logical Operators

These operators used between two relations and always produce a Boolean output.
(Boolean Output: True, False)

Types of Logical Operators
(i) Logical "and"
(ii) Logical "or"
(iii) Logical "not"

(i) Logical "and"

Realtion-1	Operator	Realtion-2	Result
True	and	True	True
True	and	False	False
False	and	True	False
False	and	False	False

Example

10>5 and 20<=15
True and False
Result: False

10>=5 and 20!=15
True and True
Result: True

(ii) Logical "or"

Realtion-1	Operator	Realtion-2	Result
True	or	True	True
True	or	False	True
False	or	True	True
False	or	False	False

Example

10>5 or 20<=15
True or False
Result: True

10<=5 or 20==15
False or False
Result: False

(i) Logical “not”

not operator is unary type of operator that requires only one operand to work on.

Operator	Realtion	Result
not	True	False
not	False	True

Example
not (10>5 or 20<=15)
not(True or False)
not(True)
Result: False

not(10<=5) or 20==15
not(False) or False
True or False
Result: True

(D) Assignment Operator (=)

This operator is used to assign the value to a variable.
Example:
City=”Jaipur”
School= “KV JJN”

(E) Short Hand Operator

These operators are derived from the arithmetical operators. It can reduce the length of expression and without affecting the result.
Rule: An expression can be shorter only if the same variable is available at the both side of assignment (=) Operator.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3

(F) Identity Operator

The identity operator used to check that both the operands reference to same memory location. It compare the memory location of two variables and return True or False accordingly.

- is operator
- not is operator

Operator	Example	Result
is	10 is 5	False
is	10 is 20//2	True
is	10 is 20/2	False
is not	10 is not 5	True
is not	10 is not 5*2	False

(G) Membership Operator (in, not in)

The membership operator used to check that one operand / value is available in sequence of values. It compare the operand with each value of sequence and return True or False accordingly.

- in operator
- not in operator

Operator	Example	Result
in	10 in [2, 4, 7, 12, 23]	False
in	10 in [12, 14, 10, 12, 23]	True
not in	10 not in [2, 4, 7, 12, 23]	True
not in	10 not in [12, 10, 12, 23]	False

(H) Bitwise Operator

The bitwise operators are similar to logical operators except that bitwise works upon binary digits.

Types of bitwise Operator

- **Bitwise and (&)**
- **Bitwise or (|)**
- **Bitwise eXclusive xor (^)**
- **Bitwise Complement (~)**

- **Bitwise and (&)**

Operand-1	Operator	Operand-2	Result
0	&	0	0
0	&	1	0
1	&	0	0
1	&	1	1

- **Bitwise or (\|)**

Operand-1	Operator	Operand-2	Result
0	|	0	0
0	|	1	1
1	|	0	1
1	|	1	1

- **Bitwise eXclusive xor (^)**
  It Return 1, if both bits are different else 0

Operand-1	Operator	Operand-2	Result
0	^	0	0
0	^	1	1
1	^	0	1
1	^	1	0

- **Bitwise Complement (~)**
  It unary operator and inverts the Return of given bit. If bit is 1 then it will convert into 0 and if bit is 0 then it will convert into 1

Operator	Operand	Result
~	0	1
~	1	0

Operator Priority

The operator priority (Precedence) is the order of evaluation of operators in an expression. The higher priority operator will evaluate first.

Operator	Description	Priority
( )	Parentheses	1 (Highest)
**	Exponentiation	2
*, /, //, %	Arithmetical Operators	3
+, -	Arithmetical Operators	4
<, <=, >, >=, !=, ==	All Relational Operators	5
is, is not	Identity operators	6
not	Logical not	7
and	Logical and	8
or	Logical or	9 (Lowest)

Note: if more than one same priority operators comes in an expression then they evaluate from left to right side as per associativity rule of operators (Except **).

Operator Associativity

Associativity is the order in which an expression (having multiple operators of same priority) is evaluated.

Rule-1: All the operators of same priority have the left to right associativity.

Rule-2: Exponential (**) operator has right to left associativity.

Type Casting

Python provides the various numeric data types like integer, float and complex. To convert one type of data into other type of data is term as type casting. It is possible by using some python functions.

- **int()**
- **float()**
- **complex()**

Example

```
x = 1 # int
y = 2.8 # float
z = 1j # complex

#convert from int to float:
a = float(x)
#convert from float to int:
b = int(y)
#convert from int to complex:
c = complex(x)
print(a)            # Output: 1.0
print(b)            # Output: 2
print(c)            # Output: (1+0j)
print(type(a))      # Output: <class 'int'>
print(type(b))      # Output: <class 'float'>
print(type(c))      # Output: <class 'complex'>
```

Note: cannot convert complex numbers into another type.

Python Built-in Functions

Python is very rich in built in functions and libraries.
Built in functions are already created and stored in Python Interpreter.

1. input() function
   The input() used to read the string value from keyboard. This function works at the program execution time.
   Example:
   ```
   Name=input("Enter Your Name: ")
   City=input()
   ```

2. print() function
   The print() used to write/ display any type of value on screen.
   Example:
   ```
   print("Hello Students")   # Display only String
   print(name)   # Display only String value
   print(2+5)   # Display integer value
   print(2.5+5)   # Display float value
   print("Your name is:", name)
   # Display String and integer type values
   ```

3. int() function
   int() used to convert the convertible string value in integer value.
   Example:
   ```
   A="25"
   # "25" is in double quotation so it is a string value.
   ```

```
A=int(A)   # “25” will convert into integer value
print(A)  # Output will be 25 (Without Quotation)
A=”C-25”     #”C-25” is string value
A=int(A)
# Show error, because “C-25” is not a convertible string value
```

4. float() function

float() used to convert the convertible string value in float value

Example:

```
A=”-25.5”
A=float(A)
print(A)      # Output: -25.5
A=”25.6f”
A=float(A)
print(A)      # Output: Error
```

5. type() function

type() display the datatype of given value or variable.

Example:

```
A=100
print(type(“25”))        # Output: <class ‘str’>
print(type(“KVS”))       # Output: <class ‘str’>
print(type(25))          # Output: <class ‘int’>
print(type(25.5))        # Output: <class ‘float’>
print(type(1+0j))        # Output: <class ‘complex’>
print(type(A))           # Output: <class ‘int’>
print(type(True))        # Output: <class ‘bool’>
print(type(None))        # Output: <class ‘NoneType’>
```

6. id() function

This function used to find the memory address of iven variable or value.
Example:

```
A=10
print(id(A))            # 5619710255  (Assumed)
print(id(True))         # 9849702213  (Assumed)
print(id(False))        # 1254770225  (Assumed)
print(id(None))         # 6288732549  (Assumed)
print(id(2.678+6j))     # 2549935457  (Assumed)
print(id(5))            # 5493147895  (Assumed)
```

7. ord() function

In Python ord(character Value) will return the equivalent ASCII Value of given Character value.

Ex. ord(‘65’) => output: A

ord('क')   => output: 2325

8. chr() function

In Python chr(ASCII Value) will return the equivalent Character value of given ASCII.

Ex. chr(42)   => Output: *

chr(38)   => Output: &

chr(2326)   => Output: ख

9. format() function

- the format() used to display the output in different formatted forms.
- Example:
- Name="Ajay"

- age=16
- city="Jhunjhunu"
- print("Hello '{}' live in city '{}' and hi age is {} year".format(Name,city,age))

- Output:

Hello 'Ajay' live in city 'Jhunjhunu' and hi age is 16 year

random module

import random

random()

it return random floating point number from 0 to 1.
Where 0 is inclusive and 1 is exclusive.
print(random.random())   # output: 0.236486 (Assumed)

randint(a,b)

it return random integer number between a and b.
Where both a and b are inclusive.
print(random.randint(2,5))   # output: 2/3/4/5 (Assumed)

randrange(start, stop, step)

This function gets 3 parameters as
Start: Optional, by default is 0. (Integer)
Stop: Mandatory, Exclusive in result. (Integer)
Step: Optional, by default is 1 (integer)

print(random.randrange(2,20,2))
it will generate an Even random number between 2 and 20 (where 20 is exclusive)

****Finish****