Table of Contents

Chapter 2

Operators

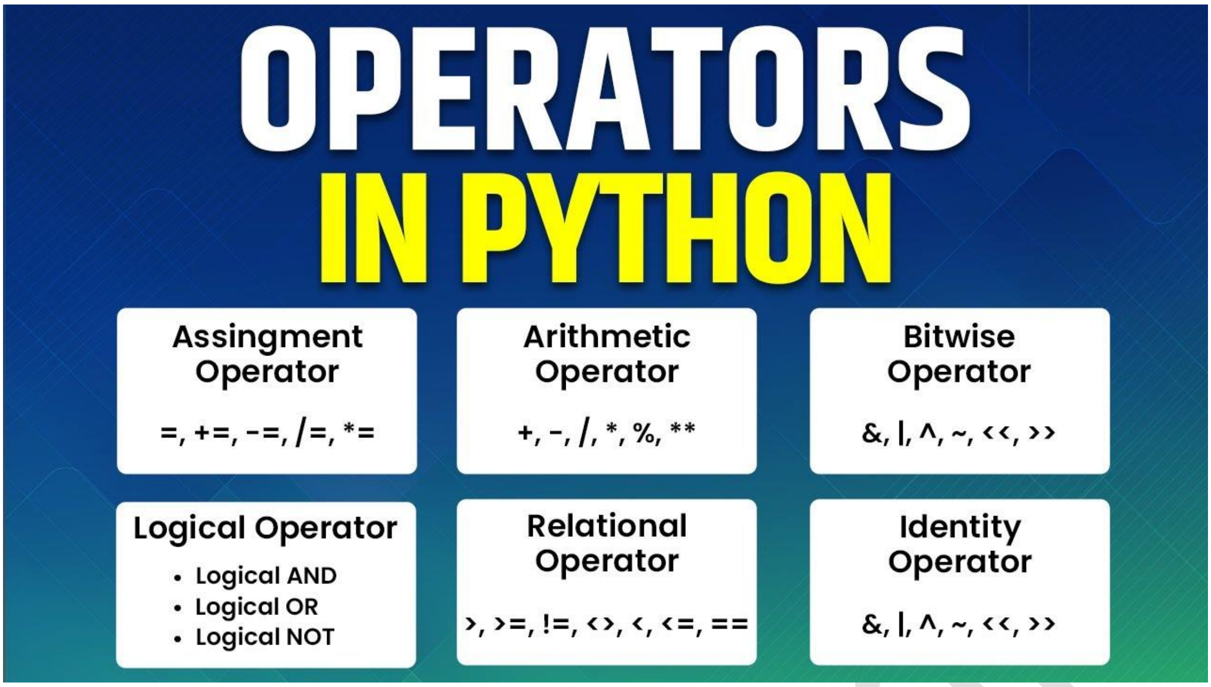
Operators:

In Python, operators are special symbols that designate that some sort of computation should be performed. The values that an operator acts on are called operands.

Following are the different types of operators in Python -
1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Logical operators
5. Identity operators
6. Membership operators
7. Bitwise operators

Ex.

```
a = 3
b = 4

# Arithmetic Operators
print("The value of 3+4 is ", 3+4)
print("The value of 3-4 is ", 3-4)
print("The value of 3*4 is ", 3*4)
print("The value of 3/4 is ", 3/4)

# Assignment Operators
a = 34
a -= 12
a *= 12
a /= 12
print(a)

# Comparison Operators
# b = (14<=7)
# b = (14>=7)
# b = (14<7)
# b = (14>7)
# b = (14==7)
b = (14!=7)
print(b)

# Logical Operators
bool1 = True
bool2 = False
print("The value of bool1 and bool2 is", (bool1 and bool2))
print("The value of bool1 or bool2 is", (bool1 or bool2))
print("The value of not bool2 is", (not bool2))

# Identity Operators : is , is not
print("a is b: ",a is b)
print("a is not b: ",a is not b)

# Membership Operators: in , not in
x = ["apple", "banana"]
print("banana in x: ","banana" in x)
print("pineapple not in x: ","pineapple" not in x)

# Bitwise Operator
a = 10
b = 4
# Print bitwise AND operation
print("a & b: ",a & b)
# Print bitwise OR operation
print("a | b: ",a | b)
# Print bitwise NOT operation
print("a ~ b: ",~a)
# print bitwise XOR operation
print("a ^ b: ",a ^ b)
# print bitwise right shift operation
print("a >> 2: ",a >> 2)
# print bitwise left shift operation
print("a << 2: ",a << 2)
```



OPERATORS

IN PYTHON

Assingment Operator

=, +=, -=, /=, *=

Arithmetic Operator

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

Bitwise Operator

&, |, ^, ~, <<, >>

Logical Operator

- Logical AND
- Logical OR
- Logical NOT

Relational Operator

>, >=, !=, <>, <, <=, ==

Identity Operator

&, |, ^, ~, <<, >>

Input Function:

Input function is used to take value from the user. Python takes every input as a string.

Ex.

```
a = input("Enter Your Name: ")
print("Hello ",a)

# Enter Your Name: Madi
# Hello Madi
```

Type Casting:

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)

float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Ex.

```
# Integers typecasting
a = int(1)   # x will be 1
b = int(2.8) # y will be 2
c = int("3") # z will be 3

# Float typecasting
d = float(1)     # x will be 1.0
e = float(2.8)   # y will be 2.8
f = float("3")   # z will be 3.0
g = float("4.2") # w will be 4.2

# String typecasting
h = str("s1") # x will be 's1'
i = str(2)    # y will be '2'
j = str(3.0)  # z will be '3.0'
```

```
#printing types:

print("type of a: ", type(a))
print("type of b: ", type(b))
print("type of c: ", type(c))
print("type of d: ", type(d))
print("type of e: ", type(e))
print("type of f: ", type(f))
print("type of g: ", type(g))
print("type of h: ", type(h))
print("type of i: ", type(i))
print("type of j: ", type(j))

# type of a:  <class 'int'>
# type of b:  <class 'int'>
# type of d:  <class 'float'>
# type of e:  <class 'float'>
# type of f:  <class 'float'>
# type of g:  <class 'float'>
# type of h:  <class 'str'>
# type of i:  <class 'str'>
# type of j:  <class 'str'>
```