Python is a general-purpose, dynamically typed, high-level, compiled and interpreted, garbage-collected, and purely object-oriented programming language that supports procedural, object-oriented, and functional programming.

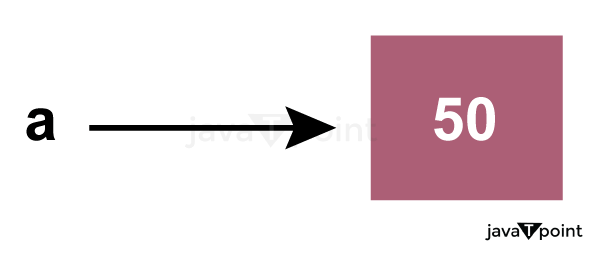
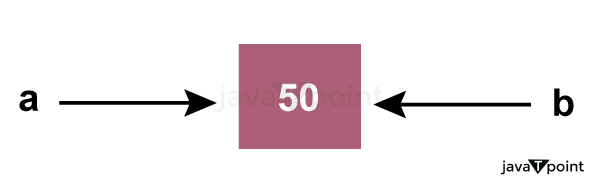
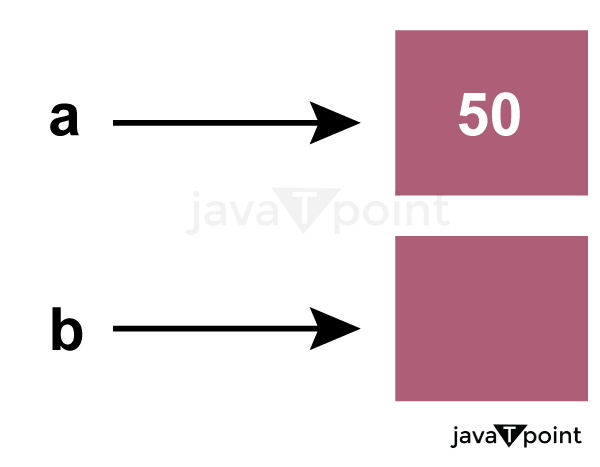
Python uses indentation to define a block of code. Indentation is nothing but adding whitespace before the statement when it is needed. Instead of Semicolon as used in other languages, Python ends its statements with a NewLine character. Python is a case-sensitive language, which means that uppercase and lowercase letters are treated differently. In Python, comments can be added using the '#' symbol. Any text written after the '#' symbol is considered a comment and is ignored by the interpreter.

**Python Variables**A variable is the name given to a memory location. A value-holding Python variable is also known as an identifier. Since Python is an infer language that is smart enough to determine the type of a variable, we do not need to specify its type in Python.

* Variable names must begin with a letter or an underscore, but they can be a group of both letters and digits. Eg: \_var, my\_Variables, my\_v1
* White space and special characters (!, @, #, %, etc.) are not allowed in the identifier name. ^, &, \*).
* The equal (=) operator is utilized to assign worth to a variable.

The multi-word keywords can be created by the following method.

* Camel Case - In the camel case, each word or abbreviation in the middle of begins with a capital letter. There is no intervention of whitespace. For example - nameOfStudent, valueOfVaraible, etc.
* Pascal Case - It is the same as the Camel Case, but here the first word is also capital. For example - NameOfStudent, etc.
* Snake Case - In the snake case, Words are separated by the underscore. For example - name\_of\_student, etc.

**Object References**  

In the above image, the variable a refers to an integer object. Suppose we assign the integer value 50 to a new variable b. The variable b refers to the same object that a points to because Python does not create another object.

**Assigning Variables:** We can use”,” for concatenation operation  
x=y=z=20 #same value to different variables  
print(x," is same as",y,z) # 20 is same as 20 20  
a,b=20,30 #multiple values to multiple variables  
name="Aadhya"  
print("My name is",name,"numbers are ",a,b) # My name is Aadhya numbers are 20 30

**In- built functions**  
1) Print(“Aadhya”) –Aadhya  
*print("Aadhya") #Aadhya  
print ("Aadhya's life") #Aadhya's life   
print('Aadhya "life"') #Aadhya "life"  
print('Aadhya\'s "life"') #Aadhya's "life" –using backslash to ignore ‘  
print("Aadhya's \"life") #Aadhya's "life*

print('navin'+'navin') #navinnavin   
print('navin','navin') #navin navin  
print(3\*'navin') #navinnavinnavin  
print("c:\docs\navin")   
# c:\docs  
# avin  
print(r"c:\docs\navin") #c:\docs\navin

--r means raw string i.e. print as it is(avoiding \n new character)  
2) type("Aadhya") -- <class 'str'>  
3) id(--object) Every object created in Python has a unique identifier. Python gives the dependability that no two items will have a similar identifier. The object identifier is identified using the built-in id() function.  
*a=50  
b=a  
print(id(a)) #* *140731432914904  
print(id(b)) #* *140731432914904  
#Reassigning variable  
a=100  
print(id(a)) #* *140731432916504*4)del <variable\_name> --deleting a variable  
*a=20  
print(a) #20  
del a  
print(a) #NameError: name 'a' is not defined*

**Python Data Types**Every value has a datatype, and variables can hold values. Python is a powerfully composed language; consequently, we don't have to characterize the sort of variable while announcing it. The interpreter binds the value implicitly to its type.

*type("john") #<class 'str'>  
type(3.456) #<class 'float'>  
type(12) #<class 'int'>  
type('a') #<class 'str'>*



Python supports three kinds of numerical data.

* Int: Whole number worth can be any length, like numbers 10, 2, 29, - 20, - 150, and so on. An integer can be any length you want in Python. Its worth has a place with int.
* Float: Float stores drifting point numbers like 1.9, 9.902, 15.2, etc. It can be accurate to within 15 decimal places.
* Complex: An intricate number contains an arranged pair, i.e., x + iy, where x and y signify the genuine and non-existent parts separately. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

The instance () capability is utilized to check whether an item has a place with a specific class.  
*a=20  
b=1+2j  
print(isinstance(a,int)) #true  
print(isinstance(b,complex)) #true*

## **Python Operators** The operator is a symbol that performs a specific operation between two operands 1) Arithmetic Operators - Arithmetic operators used between two operands for a particular operation. There are many arithmetic operators. It includes the exponent (\*\*) operator as well as the + (addition), - (subtraction), \* (multiplication), / (divide), % (reminder), and // (floor division) operators. Division operator always gives floating answers eg 4/2 = 2.0

a,b=20,6  
print(a+b) #26  
print(a-b) #14  
print(a\*b) #120  
print(a/b) #3.33333333333  
print(a//b) #3 --Doesn't round off  
print(b\*\*2) #36   
print(a%b) #2