Python Variables

 variables are used to store data that can be referenced and manipulated during program execution. A variable is essentially a name that is assigned to a value.

Variables act as placeholders for data. They allow us to store and reuse values in our program.

*# Variable 'x' stores the integer value 10*

x = 5

*# Variable 'name' stores the string "Samantha"*

name = "Samantha"

print(x)

print(name)

**Rules for Naming Variables**

* Variable names can only contain letters, digits and underscores (\_).
* A variable name cannot start with a digit.
* Variable names are case-sensitive (myVar and myvar are different).
* Avoid using [Python keywords](https://www.geeksforgeeks.org/python-keywords/) (e.g., if, else, for) as variable names.

Valid example

age = 21

\_colour = "lilac"

total\_score = 90

Invalid

1name = "Error" *# Starts with a digit*

**class** = 10 # '**class**' is a reserved keyword

user-name = "Doe" *# Contains a hyphen*

**Assigning Values to Variables**

**Basic Assignment**

**Variables in Python are assigned values using the =**[**operator.**](https://www.geeksforgeeks.org/python-operators/)

x = 5

y = 3.14

z = "Hi"

**Dynamic Typing**

Python variables are dynamically typed, meaning the same variable can hold different types of values during execution.

x = 10

x = "Now a string"

**Multiple Assignments**

Python allows multiple variables to be assigned values in a single line.

a = b = c = 100

print(a, b, c)

x, y, z = 1, 2.5, "Python"

print(x, y, z)

**Type Casting a Variable**

[Type casting](https://www.geeksforgeeks.org/type-casting-in-python/) refers to the process of converting the value of one data type into another. Python provides several built-in functions to facilitate casting, including int(), float() and str() among others.

**Basic Casting Functions**

* [**int()**](https://www.geeksforgeeks.org/python-int-function/)– Converts compatible values to an integer.
* [**float()**](https://www.geeksforgeeks.org/float-in-python/)– Transforms values into floating-point numbers.
* [**str()**](https://www.geeksforgeeks.org/python-str-function/)– Converts any data type into a string.

s = "10"

n = int(s)

cnt = 5

f = float(cnt)

age = 25

s2 = str(age)

**Getting the Type of Variable**

In Python, we can determine the type of a variable using the type() function. This built-in function returns the type of the object passed to it.

n = 42

print(type(n))

**Scope of a Variable**

There are two methods how we define scope of a variable in python which are [local and global](https://www.geeksforgeeks.org/global-local-variables-python/).

**Local Variables:**

Variables defined inside a function are local to that function.

**def** f():

a = "I am local"

print(a)

f()

**Global Variables:**

Variables defined outside any function are global and can be accessed inside functions using the [global keyword](https://www.geeksforgeeks.org/global-keyword-in-python/).

a = "I am global"

**def** f():

**global** a

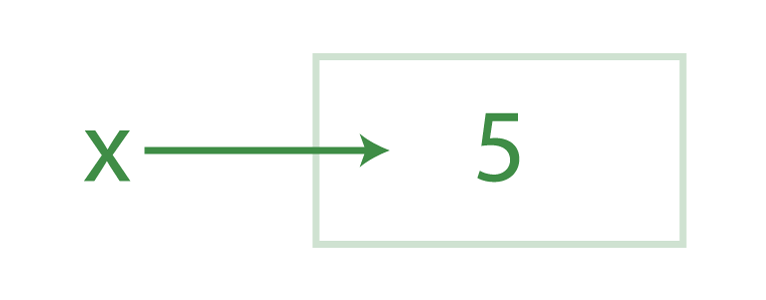
a = "Modified globally"

print(a)

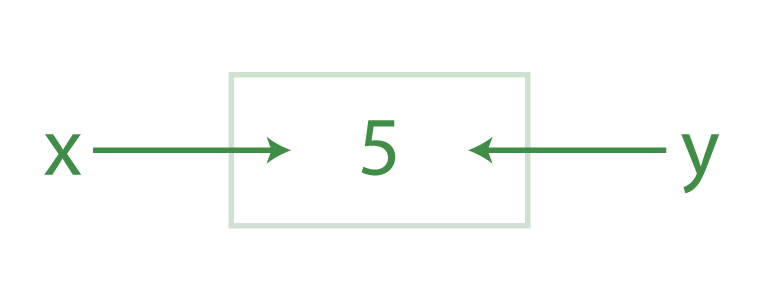
f()

print(a)

**Object Reference in Python**

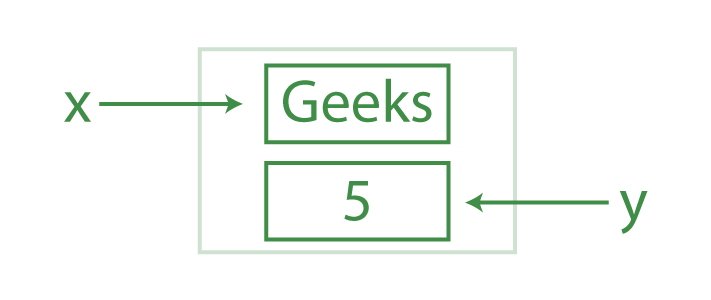


y = x

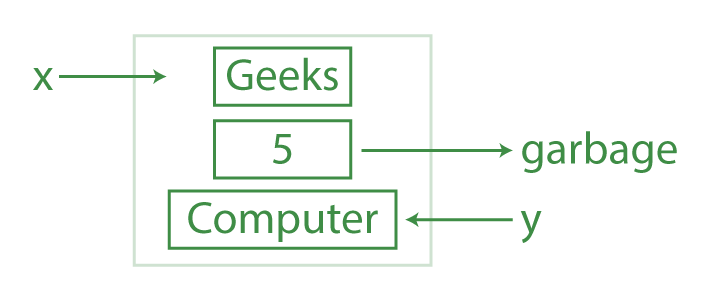


* Python encounters the first statement, it creates an object for the value 5 and makes x reference it. The second statement creates y and references the same object as x, not x itself. This is called a [*Shared Reference*](https://www.geeksforgeeks.org/shared-reference-in-python/), where multiple variables reference the same object.

X=”geeks”



y = "Computer"



* Python creates yet another object for "Computer" and updates y to reference it.
* The original object 5 no longer has any references and becomes eligible for garbage collection.

**Delete a Variable Using del Keyword**

X=10

Print(x)

Del x ----raise NameError

**Explanation:**

* del x removes the variable x from memory.
* After deletion, trying to access the variable x results in a NameError, indicating that the variable no longer exists.

Operators

Arithmetic Operators

Relational Operators

Logical Operators

Bitwise Operators

Assignment Operators

Comparison operators

a = 13

b = 33

print(a > b)

print(a < b)

print(a == b)

print(a != b)

print(a >= b)

print(a <= b)

Logical Operators

a = **True**

b = **False**

print(a **and** b)

print(a **or** b)

print(**not** a

Bitwise Operator

1. Bitwise NOT
2. Bitwise Shift
3. Bitwise AND
4. Bitwise XOR
5. Bitwise OR

a = 10

b = 4

print(a & b)

print(a | b)

print(~a)

print(a ^ b)

print(a >> 2)

print(a << 2)

**Identity Operators in Python**

**is and is not are the**[**identity operators**](https://www.geeksforgeeks.org/python-membership-identity-operators-not-not/)**both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.**

a = 10

b = 20

c = a

print(a **is** **not** b)

print(a **is** c)

**Membership Operators in Python**

In Python, **in** and **not in** are the[membership operators](https://www.geeksforgeeks.org/python-membership-identity-operators-not-not/) that are used to test whether a value or variable is in a sequence.

x = 24

y = 20

list = [10, 20, 30, 40, 50]

**if** (x **not** **in** list):

print("x is NOT present in given list")

**else**:

print("x is present in given list")

**if** (y **in** list):

print("y is present in given list")

**else**:

print("y is NOT present in given list")

**Ternary Operator in Python**

a, b = 10, 20

min = a **if** a < b **else** b

print(min)

**Operator Precedence in Python**

expr = 10 + 20 \* 30

print(expr)

Operator Associativity in python

print(100 / 10 \* 10)

print(5 - 2 + 3)

print(5 - (2 + 3))

print(2 \*\* 3 \*\* 2)

Python Keywords

**import** **keyword**

print("The list of keywords is : ")

print(keyword.kwlist)