Variables:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example −

x = 5  
y = "John"  
print(x)  
print(y)

x = 4 # x is of type int  
x = "Sally" # x is now of type str  
print(x)

String variables can be declared either by using single or double quotes:

### Example

x = "John"  
# is the same as  
x = 'John'

## Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)

### Example

#Legal variable names:  
myvar = "John"  
my\_var = "John"  
\_my\_var = "John"  
myVar = "John"  
MYVAR = "John"  
myvar2 = "John"  
  
#Illegal variable names:  
2myvar = "John"  
my-var = "John"  
my var = "John"

Remember that variable names are case-sensitive

## Assign Value to Multiple Variables

Python allows you to assign values to multiple variables in one line:

### Example

x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)

And you can assign the same value to multiple variables in one line:

### Example

x = y = z = "Orange"  
print(x)  
print(y)  
print(z)

## Output Variables

The Python print statement is often used to output variables.

To combine both text and a variable, Python uses the + character:

### Example

x = "awesome"  
print("Python is " + x)

You can also use the + character to add a variable to another variable:

### Example

x = "Python is "  
y = "awesome"  
z =  x + y  
print(z)

For numbers, the + character works as a mathematical operator:

### Example

x = 5  
y = 10  
print(x + y)

If you try to combine a string and a number, Python will give you an error:

### Example

x = 5  
y = "John"  
print(x + y)

## Global Variables

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

### Example

Create a variable outside of a function, and use it inside the function

x = "awesome"  
  
def myfunc():  
  print("Python is " + x)  
  
myfunc()

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

### Example

Create a variable inside a function, with the same name as the global variable

x = "awesome"  
  
def myfunc():  
  x = "fantastic"  
  print("Python is " + x)  
  
myfunc()  
  
print("Python is " + x)

## The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

### Example

If you use the global keyword, the variable belongs to the global scope:

def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

Also, use the global keyword if you want to change a global variable inside a function.

### Example

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = "awesome"  
  
def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | Str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | Dict |
| Set Types: | set, frozenset |
| Boolean Type: | Bool |
| Binary Types: | bytes, bytearray, memoryview |

## Getting the Data Type

You can get the data type of any object by using the type() function:

### Example

Print the data type of the variable x:

x = 5  
print(type(x))

## Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

|  |  |  |
| --- | --- | --- |
| **Example** | **Data Type** |  |
| x = "Hello World" | str |  |
| x = 20 | int |  |
| x = 20.5 | float |  |
| x = 1j | complex |  |
| x = ["apple", "banana", "cherry"] | list |  |
| x = ("apple", "banana", "cherry") | tuple |  |
| x = range(6) | range |  |
| x = {"name" : "John", "age" : 36} | dict |  |
| x = {"apple", "banana", "cherry"} | set |  |
| x = frozenset({"apple", "banana", "cherry"}) | frozenset |  |
| x = True | bool |  |
| x = b"Hello" | bytes |  |
| x = bytearray(5) | bytearray |  |
| x = memoryview(bytes(5)) | memoryview |  |

## Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions:

|  |  |  |
| --- | --- | --- |
| **Example** | **Data Type** |  |
| x = str("Hello World") | str |  |
| x = int(20) | int |  |
| x = float(20.5) | float |  |
| x = complex(1j) | complex |  |
| x = list(("apple", "banana", "cherry")) | list |  |
| x = tuple(("apple", "banana", "cherry")) | tuple |  |
| x = range(6) | range |  |
| x = dict(name="John", age=36) | dict |  |
| x = set(("apple", "banana", "cherry")) | set |  |
| x = frozenset(("apple", "banana", "cherry")) | frozenset |  |
| x = bool(5) | bool |  |
| x = bytes(5) | bytes |  |
| x = bytearray(5) | bytearray |  |
| x = memoryview(bytes(5)) | memoryview |  |