Python - Variables

Python variables are the reserved memory locations used to store values with in a Python Program. This means that when you create a variable you reserve some space in the memory.

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

Creating Python Variables

Python variables do not need explicit declaration to reserve memory space or you can say to create a variable. A Python variable is created automatically when you assign a value to it. 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 –

```
counter = 100          # Creates an integer variable
miles   = 1000.0       # Creates a floating point variable
name    = "Zara Ali"   # Creates a string variable
```

Printing Python Variables

Once we create a Python variable and assign a value to it, we can print it using **print()** function. Following is the extension of previous example and shows how to print different variables in Python:

```
counter = 100          # Creates an integer variable
miles   = 1000.0       # Creates a floating point variable
name    = "Zara Ali"   # Creates a string variable

print (counter)
print (miles)
print (name)
```

Edit & Run

Here, 100, 1000.0 and "Zara Ali" are the values assigned to *counter*, *miles*, and *name* variables, respectively. When running the above Python program, this produces the following result –

```
100
1000.0
Zara Ali
```

Delete a Variable

You can delete the reference to a number object by using the del statement. The syntax of the del statement is –

```
del var1[,var2[,var3[....,varN]]]
```

You can delete a single object or multiple objects by using the del statement. For example –

```
del var
del var_a, var_b
```

Example

Following examples shows how we can delete a variable and if we try to use a deleted variable then Python interpreter will throw an error:

```
counter = 100
print (counter)

del counter
print (counter)
```

Edit & Run

This will produce the following result:

```
100
Traceback (most recent call last):
  File "main.py", line 7, in <module>
    print (counter)
NameError: name 'counter' is not defined
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously which means you can create multiple variables at a time. For example –

```
a = b = c = 100

print (a)
print (b)
print (c)
```

Edit & Run

This produces the following result:

```
100
100
100
```

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables. For example –

```
a,b,c = 1,2,"Zara Ali"

print (a)
print (b)
print (c)
```

Edit & Run

This produces the following result:

```
1
2
Zara Ali
```

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "Zara Ali" is assigned to the variable c.

Python Variable Names

Every Python variable should have a unique name like a, b, c. A variable name can be meaningful like color, age, name etc. There are certain rules which should be taken care while naming a Python variable:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number or any special character like $, (, * % etc.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Python variable names are case-sensitive which means Name and NAME are two different variables in Python.
- Python reserved keywords cannot be used naming the variable.

Example

Following are valid Python variable names:

```
counter = 100
_count  = 100
name1 = "Zara"
name2 = "Nuha"
Age  = 20
zara_salary = 100000

print (counter)
print (_count)
print (name1)
print (name2)
print (Age)
print (zara_salary)
```

Edit & Run

This will produce the following result:

```
100
100
Zara
Nuha
20
100000
```

Example

Following are invalid Python variable names:

```
1counter = 100
$_count  = 100
zara-salary = 100000

print (1counter)
print ($count)
print (zara-salary)
```

Edit & Run

This will produce the following result:

```
File "main.py", line 3
    1counter = 100
           ^
SyntaxError: invalid syntax
```

Python Local Variable

Python Local Variables are defined inside a function. We can not access variable outside the function.

> A Python functions is a piece of reusable code and you will learn more about function in Python - Functions tutorial.

Following is an example to show the usage of local variables:

```
def sum(x,y):
   sum = x + y
   return sum
print(sum(5, 10))
```

Edit & Run

```
15
```

Python Global Variable

Any variable created outside a function can be accessed within any function and so they have global scope. Following is an example of global variables:

```
x = 5
y = 10
def sum():
   sum = x + y
   return sum
print(sum())
```

Edit & Run

This will produce the following result:

```
15
```