Scope of Variables in Python

The existence of a variable in a region is called its scope, or the accessibility of a variable defines it scope. Scope of Variables are of utmost importance while accessing variables . We will discuss global and local variables below. At first, we hope you know how to create variables in Python.

Let's begin with Local Scope:

Local Scope

Whenever a variable is defined in a function, it can be used only in that function. The value of that variable cannot be accessed from outside the function. Let us see an example:

```
## local scope

# Function Definition
def demo( ):
  a = 100
  print (a)

# call the demo function
demo()
```

The output is as follows:

```
100
```

In the above program, let us do some changes and print the variable outside the function. What will happen? Yes, an error, **NameError**, will get generated because the scope of a local variable is inside the function:

```
## local scope
```

```
# Function Definition
def demo( ):
  a = 100

# error
print (a)

# call the demo function
demo()
```

The output is as follows displaying **NameError** since the variable is local to the function and we are trying to access it from outside the function:

```
Traceback (most recent call last): File "./prog.py", line 8, in
```

Global Scope

A variable has Global Scope, if it's created outside the function i.e. the code body and are accessible from anywhere. Let us see an example, wherein we will create a global variable and try to access the variable from outside as well as inside the function:

```
## global scope
a = 100

# Function Definition
def demo( ):
   print (a)

print (a)

# call the demo function
demo()

print (a)
```

The output is as follows:

```
100
100
100
```

Let us see an example, wherein we will use the same variable name inside and outside the function. The values will be assigned twice i.e. from outside the function and inside the function. In this case, both these variables with the same name would be considered different i.e. One of them inside and another outside the function. We will see the above example again and understand the same:

```
# global scope
# outer
a = 5

# Function Definition
def demo( ):

   #inner
   a = 10
   print ("Inner = ",a)

# call the demo function
demo()

#outer
print ("Outer = ",a)
```

The output is as follows:

```
Inner = 10
Outer = 5
```

GLOBAL keyword: Change the value of a global variable

To change the value of a global variable inside the function (global variable, we saw above), we can also use the GLOBAL keyword provide by Python. This would allow us change the variable value inside the function. Let us see the syntax:

```
global variable_name
```

Above, *variable_name* is the name of the variable.

Let us see an example wherein we are changing the global variable inside a function using the GLOBAL keyword:

```
## GLOBAL keyword

# global scope
# outer
a = 5
```

```python
# Function Definition
def demo( ):

   #inner
   global a
   a = 10
   print ("Inner = ",a)

# call the demo function
demo()

#outer
print ("Outer = ",a)
```

The output is as follows. We updated the global variable value from 5 to 10 inside the function using GLOBAL keyword:

```
Inner = 10
Outer = 10
```

In this post, we discussed Scope of Variables in Python.

Recommended Posts

- Python Operators
- Python Numbers
- Type Conversion
- Python Strings
- Loops in Python
- Decision Making Statements
- Functions in Python with Examples
- Python Tuples with Examples
- Dictionary Tutorial in Python
- Python Lists