Scope Of Variables

All variables in a program may not be accessible at all locations in that program. Part(s) of the program within which the variable name is legal and accessible, is called the scope of the variable. A variable will only be visible to and accessible by the code blocks in its scope.

There are broadly two kinds of scopes in Python −

- Global scope
- Local scope

Global Scope

A variable/name declared in the top-level segment **(`__main__`)** of a program is said to have a global scope and is usable inside the whole program (Can be accessed from anywhere in the program).

In Python, a variable declared outside a function is known as a global variable. This means that a global variable can be accessed from inside or outside of the function.

Local Scope

Variables that are defined inside a function body have a local scope. This means that local variables can be accessed only inside the function in which they are declared.

The Lifetime of a Variable

The lifetime of a variable is the time for which the variable exists in the memory.

- The lifetime of a Global variable is the entire program run (i.e. they live in the memory as long as the program is being executed).

- The lifetime of a Local variable is their function's run (i.e. as long as their function is being executed).

Creating a Global Variable

Consider the given code snippet:

```
x = "Global Variable"
def foo():
    print("Value of x: ", x)
foo()
```

Here, we created a global variable `x = "Global Variable"`. Then, we created a function `foo` to print the value of the global variable from inside the function. We get the output as:

```
Global Variable
```

Thus we can conclude that we can access a global variable from inside any function.

What if you want to change the value of a Global Variable from inside a function?

Consider the code snippet:

```
x = "Global Variable"
def foo():
    x = x -1
    print(x)
foo()
```

In this code block, we tried to update the value of the global variable **x**. We get an output as:

```
UnboundLocalError: local variable 'x' referenced before assignment
```

This happens because, when the command `x=x-1`, is interpreted, Python treats this **x** as a local variable and we have not defined any local variable **x** inside the function `foo()`.

Creating a Local Variable

We declare a local variable inside a function. Consider the given function definition:

```
def foo():
    y = "Local Variable"
    print(y)
foo()
```

We get the output as:

```
Local Variable
```

Accessing A Local Variable Outside The Scope

```
def foo():
    y = "local"
foo()
print(y)
```

In the above code, we declared a local variable **y** inside the function `foo()`, and then we tried to access it from outside the function. We get the output as:

```
NameError: name 'y' is not defined
```

We get an error because the lifetime of a local variable is the function it is defined in. Outside the function, the variable does not exist and cannot be accessed. In other words, a variable cannot be accessed outside its scope.

Global Variable And Local Variable With The Same Name

Consider the code given:

```
x = 5
def foo():
    x = 10
    print("Local:", x)
foo()
print("Global:", x)
```

In this, we have declared a global variable `x = 5` outside the function `foo()`. Now, inside the function `foo()`, we re-declared a local variable with the same name, **x**. Now, we try to print the values of **x**, inside, and outside the function. We observe the following output:

```
Local: 10
Global: 5
```

In the above code, we used the same name **x** for both global and local variables. We get a different result when we print the value of **x** because the variables have been declared in different scopes, i.e. the local scope inside `foo()` and global scope outside `foo()`.

When we print the value of the variable inside `foo()` it outputs `Local: 10`. This is called the local scope of the variable. In the local scope, it prints the value that it has been assigned inside the function.

Similarly, when we print the variable outside `foo()`, it outputs global `Global: 5`. This is called the global scope of the variable and the value of the global variable **x** is printed.