Global and Local Variables in Python

26 October 2020 09:40

Global and Local Variables in Python

Global variables are the one that are defined and declared outside a function and we need to use them inside a function.

```
# This function uses global variable s
def f():
    print(s)
# Global scope
s = "I love metaeducators"
f()
```

Output:

```

I love metaeducators
```

The variable s is defined as the string “I love `metaeducators`” before we call the function f(). The only statement in f() is the “print s” statement. As there is no local s, the value from the global s will be used.

```
# This function has a variable with
# name same as s.
def f():
    s = "Me too."
    print(s)

# Global scope
s = "I love metaeducators"
f()
print(s)
```

Output:

```
Me too.
I love metaeducators.
```

If a variable with the same name is defined inside the scope of function as well then it will print the value given inside the function only and not the global value.

The question is, what will happen if we change the value of s inside of the function f()? Will it affect the global s as well? We test it in the following piece of code:

```
def f():
    print(s)

    # This program will NOT show error
    # if we comment below line.
    s = "Me too."

    print(s)
```

```
# Global scope
s = "I love metaeductors"
f()
print(s)
```

Output:

```
Line 2: undefined: Error: local variable 's' referenced before assignment
```

To make the above program work, we need to use “global” keyword. We only need to use the global keyword in a function if we want to do assignments / change them. global is not needed for printing and accessing. Why? Python “assumes” that we want a local variable due to the assignment to s inside of f(), so the first print statement throws this error message. Any variable which is changed or created inside of a function is local if it hasn’t been declared as a global variable. To tell Python, that we want to use the global variable, we have to use the keyword **“global”**, as can be seen in the following example:

```
# This function modifies the global variable 's'
def f():
    global s
    print(s)
    s = "Look for metaeducators Python Section"
    print(s)

# Global Scope
s = "Python is great!"
f()
print(s)
```

Now there is no ambiguity.

Output:

```
Python is great!
Look for metaeducators Python Section.
Look for metaeducators Python Section.
```

A good Example

```
a = 1

# Uses global because there is no local 'a'
def f():
    print('Inside f() : ', a)

# Variable 'a' is redefined as a local
def g():
    a = 2
    print('Inside g() : ', a)

# Uses global keyword to modify global 'a'
def h():
    global a
    a = 3
    print('Inside h() : ', a)

# Global scope
print('global : ',a)
f()
print('global : ',a)
g()
print('global : ',a)
h()
print('global : ',a)
```

Output:

```
global :  1
Inside f() :  1
global :  1
Inside g() :  2
global :  1
Inside h() :  3
global :  3
```