Python Scope

June 9, 2024

Dr.Labeed Al-Saad

**A variable is only available from inside the region it is created. This is called scope.

0.1 Local Scope

A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

Example:

A variable created inside a function is available inside that function:

[1]:
```
def myfunc():
  x = 300
  print(x)

myfunc()
```

```
300
```

0.2 Function Inside Function

As explained in the example above, the variable x is not available outside the function, but it is available for any function inside the function:

Example:

The local variable can be accessed from a function within the function:

[2]:
```
def myfunc():
  x = 300
  def myinnerfunc():
    print(x)
  myinnerfunc()

myfunc()
```

```
300
```

0.3 Global Scope

A variable created in the main body of the Python code is a global variable and belongs to the global scope.

Global variables are available from within any scope, global and local.

Example:

A variable created outside of a function is global and can be used by anyone:

[3]:
```
x = 300

def myfunc():
  print(x)  #print from inside function

myfunc()

print(x)   #print from min program
```

```
300
300
```

0.4 Naming Variables

If you operate with the same variable name inside and outside of a function, Python will treat them as two separate variables, one available in the global scope (outside the function) and one available in the local scope (inside the function):

Example:

The function will print the local x, and then the code will print the global x:

[4]:
```
x = 300

def myfunc():
  x = 200
  print(x)

myfunc()

print(x)
```

```
200
300
```

0.5 Global Keyword

If you need to create a global variable, but are stuck in the local scope, you can use the global keyword.

The global keyword makes the variable global.

Example:

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

[5]:
```
def myfunc():
  global x
  x = 300

myfunc()

print(x)
```

```
300
```

Also, use the global keyword if you want to make a change to 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:

[6]:
```
x = 300

def myfunc():
  global x
  x = 200

print(x) #prin x before changing value
myfunc()

print(x)  #prin x after changing value
```

```
300
200
```

0.6 Nonlocal Keyword

The nonlocal keyword is used to work with variables inside nested functions.

The nonlocal keyword makes the variable belong to the outer function.

Example:

If you use the nonlocal keyword, the variable will belong to the outer function:

[7]:
```
def myfunc1():
  x = "Jane"
  def myfunc2():
    nonlocal x
    x = "hello"
  myfunc2()
  return x
```

```
print(myfunc1())
```

```
hello
```

[ ]: