Scope Rules in Python
LEGB

June 17, 2015

Brian A. Malloy

1. Overview

- All computer languages have scope rules
- Scope rules specify which variables can be "seen"
- These rules can be summarized as LEGB:
  1. Local
  2. Enclosing
  3. Global
  4. Built-in
- The search order matters: first search Local, then Enclosing, Global, and Built-in

2. Local Scope

- Always search Local Scope first
- Local Scope refers to names assigned in any way within a function, that are not declared as global
- That is, all names local to a function
- However, it's difficult to study Local Scope w/out studying Global Scope
- So we will look at Global first, and then Local and Enclosed

3. Global Scope

- Global scope is searched after Local, and Enclosed
- Global scope is simplest to understand
- A name declared at Global scope, is not enclosed in a function
- In the following valid Python program, x is declared in Global scope:

```
x = 99
print x
```


3.1. Order of Global Name Declarations

- However, a name must be defined before it is used
- The following Python program is not valid:

```
print x       # use of x
x = 99        # definition of x
```

```
Traceback (most recent call last):
  File "below.py", line 1, in <module>
    print x
NameError: name 'x' is not defined
```


4. Local Revisited

- Local Scope: names assigned in a function
- Local is searched first
- Consider the output of the following valid Python program:

```
1 x = 99
2 y = 17
3 def fun(x):
4   y = 100
5   print x, y
6 fun(77)
7 print x, y
********************
77 100
99 17
```

4.1. Explanation

- Local (to a function) scope is searched first
- **x**: for the print statement on line #5, the parameter to the function, line #3, is the value of x that will be found first.
- **y**: for the print statement on line #5, the definition of y on line #4 is found first, because that definition of y is Local to fun
- **print on line #7**: Python cannot look inside of functions, so there is no Local scope at line #7, only Global; thus, the definitions on line #1–2 will be used for x & y.

5. Keyword global

- A global variable can be declared in a function using the keyword global
- Line #2 contains a global declaration of x
- That is why x can be seen on line #6
- **Caution**: Must call the function (line #5)

```
1 def fun():
2   global x
3   x = 100
4
5 fun()
6 print x
```

5.1. Must Enter Function Scope

- In the following program, fun is never called.
- Thus, the variable x is never defined.

```
1 def fun():
2   global x
3   x = 100
4
5 print x
********************
File "global.py", line 5, in <module>
  print x
NameError: global name 'x' is not defined
```

5.2. Two Global Variables

- The program below has 2 global variables, and
- No local variables

```
1 count = 99
2 def fun():
3   global x
4   x = 100
5
6 fun()
7 print count, x
```

6. Enclosing Scope

- Python permits functions to be nested
- When searching for a variable, first search Local, and then search Enclosing scopes
- To find x on line #5: first search bar, then fun, where the parameter is found
- To find y on line #5: first search bar, then fun; y found on line #3

```
1 x = 99
2 def fun(x):
3   y = 100
4   def bar():
5     print x, y
6   bar()
7 fun(77)
```


7. Builtin Scope

- If a name is not found using Local, Enclosing, or Global, then the builtin names are searched
- Each line in the program below uses two builtin names
- You can find all builtin names: dir(__builtins__)

```
1 print pow(2,3)
2 print __name__
3 print len('cat')
```

7.1. Builtin Scope Searched Last

- Only 1 builtin name is used below: print on line # 4
- The builtin name pow is not found because a user-defined instance of pow is found at Global scope on line #1.

```
1 def pow(base, exponent):
2   return base**exponent
3
4 print pow(2,3)
```

8. No New Scope

- if, elif, or else blocks do **not** declare a new scope.
- variables defined in if, elif, or else block are in Global scope,
- only if the block is entered.
- The print on line #4 finds y only because the condition on line #2 is True

```
1 x = 99
2 if x > 0:
3   y = 17
4 print y
```



8.1. If Not Entered ⇒ Not Defined

- If an if, elif, or else block is not entered, then a variable in the block will not be defined.
- Variable y will not be found on line #4 because the condition on line #2 is False
- This rule applies to while and for blocks also

```
1 x = 0
2 if x > 0:
3    y = 17
4 print y
*****************************************
Traceback (most recent call last):
  File "notfound.py", line 4, in <module>
    print y
NameError: name 'y' is not defined
```