PEP: 227 - Statically Nested Scopes

Louis Bouddhou, Alex Campbell, Josh Fermin

December 8, 2014

What is the problem?

- Cannot reference a variable in a higher order function (nested).
- Static scoping does not work within nested functions.

Example - Without Statically Nested Scopes

Louis Bouddhou, Alex Campbell, Josh Fermin

```
def bank_account(initial_balance):
    balance = [initial_balance]
    def deposit(amount):
        balance[0] = balance[0] + amount
        return balance
    return deposit
```

Introduced changes in this PEP

- Gives nested functions the scope of outer functions.
- This allows for variables within the parent function to be inherited by the nested function.

Problems this PEP addresses: Utility

- Limited utility of nested functions.

Example


```
from Tkinter import *
root = Tk()
Button(root, text="Click here",
       command=lambda root=root: root.test.configure(
```

Problems this PEP addresses: Non-lexical

- Confusion among new users who are used to lexical scoping.

Example


```
def make_adder(base):
  def adder(x):
    return base + x
  return adder
add5 = make_adder(5)
add5(6)
```

Namespaces

- Local
- Global
- Builtin

Local Namespace

Louis Bouddhou, Alex Campbell, Josh Fermin

- Local Namespace “‘python def local(num): foo = num * num return var

local(2) “‘

Global Namespace

Louis Bouddhou, Alex Campbell, Josh Fermin

```
bar = 10
def global(num)
    foo = bar * num
    return foo
global(2)
```

Builtin Namespace

Louis Bouddhou, Alex Campbell, Josh Fermin

```
def builtin(num)
    myList = list()
        for n in list:
            n = n * num
    return list

builtin(2)
```

Bounds


```
def bound(num)
    foo = bar * num
    return foo
function(2)

bar = 10
```

Name Search


```
myList = list([1,2,3])
print myList
def function(num):
  list = myList
  for x in range(0,len(list)):
    list[x] = list[x] * num
  return list # [2,4,6]
function(2)
```

Discussion

- The PEP works under all circumstances except for the following cases:

1. Name in class scope
2. Global statement short-circuits the normal rules

Discussion - Name in Class Scope

- Names in a class scope:
- Resolved in the innermost (nested) function

Discussion - Short Circuit

- Global statement is unaffected by change

```
myvariable = 5
def func():
    global myvariable
    myvariable = 6    #changes global var
    print myvariable #prints 6

func()
print myvariable  #prints 6
```

Problems - Backwards Compatibility

- Two kinds of compatibility problems caused:

1. Code behavior
2. Syntax errors

Example - Code Behavior

Louis Bouddhou, Alex Campbell, Josh Fermin

```
x = 1
def f1():
    x = 2
    def inner():
        print x
    inner()
```

Example - Syntax Errors


```
y = 1
def f():
    exec "y = 'gotcha'" # or from module import *
    def g():
        return y
```

Conclusion

- Changes in the pep are beneficial even though nested scopes aren't used that often.
- Only problems lie in backwards compatibility.