PEP: 227 - Statically Nested Scopes

Louis Bouddhou, Alex Campbell, Josh Fermin

December 7, 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

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

Introduced changes in this PEP

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

Example - With Statically Nested Scopes

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

Namespaces

- Local
- Global
- Builtin

Whenever you run a simple Python script, the interpreter treats it as a module called **main**, which gets its own namespace. Also, the builtin functions that you would use live in another module called **builtin** and they have their own namespace.

Problems this PEP addresses

- Limited utility of nested functions.
- Confusion od fon among new users who are used to lexical scoping.

Discussion

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

1. Name in class scope is not acc sible
2. Global statement short-circuits the norma rules
3. Varibles are not declared.

Discussion - Name in Class Scope

- Names in a class scope:
- Resolved in the innermost (nested) f ction
- talk about why this is necessary

Discussion - Short Circuit

- Global statement is unaffected by change

```
myvariable = 5
def func():
    global myvariable
    myvariable = 6   #changes ‘myvariable‘ at the glo
    print myvariable #prints 6

func()
print myvariable  #prints 6 now because we were able
                  #to modify the reference in the fun
```

Discussion - Variables Not Declared

-

Problems - Backwards Compatibility

- Two kinds of compatibility problems used:

1. Code behav r
2. Syntax errors

Example

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