Mutable Functions

A Function with Behavior That Varies Over Time

Let's model a bank account that has a balance of $100

<table>
<thead>
<tr>
<th>Argument:</th>
<th>amount to withdraw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return value:</td>
<td>remaining balance</td>
</tr>
</tbody>
</table>

Different return value!

Where's this balance stored?

Within the parent frame of the function!

A function has a body and a parent environment

Persistent Local State Using Environments

The parent frame contains the balance, the local state of the withdraw function
Every call decreases the same balance by (a possibly different) amount

Reminder: Local Assignment

1. Evaluate all expressions right of =, from left to right
2. Bind the names on the left to the resulting values in the current frame

Non-Local Assignment & Persistent Local State

Declare the name "balance" nonlocal at the top of the body of the function in which it is re-assigned
Re-bind balance in the first non-local frame in which it was bound previously

Non-Local Assignment
The Effect of Nonlocal Statements

nonlocal <name>, <name>, ...

Effect: Future assignments to that name change its pre-existing binding in the first non-local frame of the current environment in which that name is bound.

Python Docs: an "enclosing scope"

From the Python 3 language reference:
Names listed in a nonlocal statement must refer to pre-existing bindings in an enclosing scope.
Names listed in a nonlocal statement must not collide with pre-existing bindings in the local scope.

http://docs.python.org/release/3.1.3/reference/simple_stmts.html#the-nonlocal-statement

Effect: Future assignments to that name change its pre-existing binding in the first non-local frame of the current environment in which that name is bound.

nonlocal <name>, <name>, ...

Python Docs: an "enclosing scope"

From the Python 3 language reference:
Names listed in a nonlocal statement must refer to pre-existing bindings in an enclosing scope.
Names listed in a nonlocal statement must not collide with pre-existing bindings in the local scope.

The Many Meanings of Assignment Statements

Status	Effect

• No nonlocal statement
• "x" is not bound locally
Create a new binding from name "x" to object 2 in the first frame of the current environment

• No nonlocal statement
• "x" is bound locally
Re-bind name "x" to object 2 in the first frame of the current environment in which "x" is bound

• nonlocal x
• "x" is bound in a non-local frame
• "x" also bound locally
SyntaxError: no binding for nonlocal 'x' found

• nonlocal x
• "x" is not bound in a non-local frame
• "x" also bound locally
SyntaxError: name 'x' is parameter and nonlocal

Python Particulars

Python pre-computes which frame contains each name before executing the body of a function.

Within the body of a function, all instances of a name must refer to the same frame.

Multiple Mutable Functions

def make_withdraw(balance):
    def withdraw(amount):
        if amount > balance:
            return 'Insufficient funds'
        balance = balance - amount
        return balance
    return withdraw

withdraw = make_withdraw(20)
withdraw(5)

Referential Transparency, Lost

• Expressions are referentially transparent if substituting an expression with its value does not change the meaning of a program.

mutl(add(4, mul(6, 2)), add(3, 5))

mul(add(4, mul(6, 2)), add(3, 5))

mul(add(4, mul(6, 2)), add(3, 5))

• Mutation operations violate the condition of referential transparency because they do more than just return a value; they change the environment.
Go Bears!

```python
def oski(bear):
    def cal(berk):
        nonlocal bear
        if bear(berk) == 0:
            return [berk+1, berk-1]
        bear = lambda ley: berk-ley
        return [berk, cal(berk)]
    return cal(2)

oski(abs)
```

Environment Diagrams