Lecture 13: Mutable Functions

Brian Hou
July 12, 2016
Announcements

• Project 2 is due today (submit early and often)
  • Look at your Hog submission for composition feedback
• Midterm is on 7/14 from 5–8 PM in 2050 VLSB
  • TA-led review session during lecture tomorrow
  • Office hours after 3 PM on Thursday and on Friday have been rescheduled
• More information on Piazza
This short week (Mutability), the goals are:

- To explore the power of values that can mutate, or change
Mutable Functions
Functions That Change

How can we model a bank account that has a balance of $100?

```python
>>> withdraw = make_withdraw(100)

>>> withdraw(25)
75

>>> withdraw(25)
50

>>> withdraw(60)
'Insufficient funds'

>>> withdraw(15)
35
```

Where is this balance stored?
Persistent Local State in Environments

The parent frame contains the balance, the local state of the withdraw function.

All calls to the same function have the same parent.

Every call decreases the same balance by (a possibly different) amount.
```python
def make_withdraw(balance):
    """Return a withdraw function with a starting balance.""

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

**Nonlocal Assignment (demo)**

Declare the name `balance` nonlocal at the top of the function in which it is re-assigned

Re-bind `balance` in the first nonlocal frame in which it was bound previously
Nonlocal Assignment
Nonlocal Statements

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

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

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
http://www.python.org/dev/peps/pep-3104/
## Assignment Statements  

$x = 2$

<table>
<thead>
<tr>
<th>Status</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>• No nonlocal statement</td>
<td>Create a new binding from name &quot;x&quot; to value 2 in the first frame of the current environment</td>
</tr>
<tr>
<td>• &quot;x&quot; is not bound locally</td>
<td>Re-bind name &quot;x&quot; to value 2 in the first frame of the current environment</td>
</tr>
<tr>
<td>• No nonlocal statement</td>
<td>Re-bind &quot;x&quot; to 2 in the first nonlocal frame of the current environment in which &quot;x&quot; is bound</td>
</tr>
<tr>
<td>• nonlocal x</td>
<td>SyntaxError: no binding for nonlocal 'x' found</td>
</tr>
<tr>
<td>• &quot;x&quot; is not bound in a nonlocal frame</td>
<td>SyntaxError: name 'x' is parameter and nonlocal</td>
</tr>
<tr>
<td>• nonlocal x</td>
<td>SyntaxError: no binding for nonlocal 'x' found</td>
</tr>
<tr>
<td>• &quot;x&quot; is bound in a nonlocal frame</td>
<td>SyntaxError: no binding for nonlocal 'x' found</td>
</tr>
<tr>
<td>• &quot;x&quot; also bound locally</td>
<td>SyntaxError: no binding for nonlocal 'x' found</td>
</tr>
</tbody>
</table>
def make_withdraw(balance):
    def withdraw(amount):
        # nonlocal balance
        if amount > balance:
            return 'Insufficient funds'
        balance = balance - amount
        return balance
    return withdraw

UnboundLocalError: local variable 'balance' referenced before assignment

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.
Accounts
Mutable Sequences

```python
def make_withdraw(balance):
    b = [balance]
    def withdraw(amount):
        if amount > b[0]:
            return 'Insufficient funds'
        b[0] = b[0] - amount
        return b[0]
    return withdraw
```
Multiple Mutable Functions

```python
>>> brian = make_withdraw(100)
>>> marvin = make_withdraw(100000)
>>> brian(10)
90

>>> marvin(10000)
90000

>>> brian(100)
'Insufficient funds'

>>> marvin(100)
89900
```
Break!
Referential Transparency

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

\[
\text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5))
\]

\[
\text{mul}(\text{add}(2, 24), \text{add}(3, 5))
\]

\[
\text{mul}(26, \text{add}(3, 5))
\]

• Mutation operations violate the condition of referential transparency because they do more than just return a value; they change the environment.
Mutating Linked Lists
Summary

• The nonlocal statement allows us to mutate name–value bindings in a nonlocal frame

• Mutation is a powerful tool, but it also makes reasoning about programs more difficult

• The truth is: we don't usually use nonlocal to build our own objects with mutable state
  • We'll see another way next week

• Good luck on the midterm!