Mutable Functions

A Function with Behavior That Varies Over Time

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

Argument: amount to withdraw

Return value: remaining balance

Second withdrawal of the same amount

A function has a body and a parent environment

Persistent Local State Using Environments

Reminder: Local Assignment

Non-Local Assignment & Persistent Local State

Non-Local Assignment
The Effect of Nonlocal Statements

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.

https://docs.python.org/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"

The Many Meanings of Assignment Statements

<table>
<thead>
<tr>
<th>Status</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>No nonlocal statement</td>
<td>Create a new binding from &quot;x&quot; to object 2 in the first frame of the current environment</td>
</tr>
<tr>
<td>No nonlocal statement</td>
<td>Re-bind name &quot;x&quot; to object 2 in the first frame of the current environment</td>
</tr>
<tr>
<td>nonlocal x</td>
<td>*&quot;x&quot; is bound in a non-local frame</td>
</tr>
<tr>
<td>nonlocal x</td>
<td>*&quot;x&quot; is not bound in a non-local frame</td>
</tr>
<tr>
<td>nonlocal x</td>
<td>SyntaxError: no binding for nonlocal 'x' found</td>
</tr>
<tr>
<td>nonlocal x</td>
<td>*&quot;x&quot; is bound in a non-local frame</td>
</tr>
<tr>
<td>nonlocal x</td>
<td>*&quot;x&quot; also bound locally</td>
</tr>
<tr>
<td>nonlocal x</td>
<td>SyntaxError: name 'x' is parameter and nonlocal</td>
</tr>
</tbody>
</table>

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.

```
def make_withdraw(balance):
    def withdraw(amount):
        if amount + balance < balance:
            return "Insufficient funds"
        return balance - amount
    return withdraw
wd = make_withdraw(28)
wd(5)
```

UnboundLocalError: local variable 'balance' referenced before assignment

Multiple Mutable Functions

(Demo)

Mutable Values & Persistent Local State

Mutable values can be changed without a nonlocal statement.

```
make_withdraw_list(100): return withdraw
F1: withdraw(amount):
    if amount > balance:
        return withdraw
    return balance
make_withdraw_list(100): return withdraw
F2: withdraw(parent=FI):
    if amount > balance:
        return withdraw
    return balance
```

Interactive Diagram

Referential Transparency Lost

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

```
mul(add(2, mul(4, 6)), add(3, 5))
mul(add(2, 24), add(3, 5))
mul(26, add(3, 5))
```

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

Interactive Diagram