Announcements

• Homework 4 due Tuesday 10/8 @ 11:59pm.

• Project 2 due Thursday 10/10 @ 11:59pm.

• Guerrilla Section 2 this Saturday 10/5 & Sunday 10/6 10am–1pm in Soda.
  ▪ Topics: Data abstraction, sequences, and non-local assignment.
  ▪ Please RSVP on Piazza!

• Guest lecture on Wednesday 10/9, Peter Norvig on Natural Language Processing in Python.
  ▪ No video (except a screencast)! Come to Wheeler.
Mutable Functions
A Function with Behavior That Varies Over Time

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

Return value: remaining balance

>>> withdraw(25)
75

Argument: amount to withdraw

>>> withdraw(25)
50

Second withdrawal of the same amount

>>> withdraw(60)
'Insufficient funds'

Within the parent frame of the function!

>>> withdraw = make_withdraw(100)

Different return value!

Where's this balance stored?

A function has a body and a parent environment
Persistent Local State Using Environments

Example: http://goo.gl/cUC09s
Reminder: Local Assignment

### Execution rule for assignment statements:

1. Evaluate all expressions right of `=`, from left to right.

2. Bind the names on the left the resulting values in the **first frame** of the current environment.

Example: [http://goo.gl/Wxpg5Z](http://goo.gl/Wxpg5Z)
Non-Local Assignment & Persistent Local State

```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
```

(Demo)
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.

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/
### 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 name &quot;x&quot; to object 2 in the first frame of the current environment.</td>
</tr>
<tr>
<td>• &quot;x&quot; is not bound locally</td>
<td></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 env.</td>
</tr>
<tr>
<td>• &quot;x&quot; is bound locally</td>
<td></td>
</tr>
<tr>
<td>• nonlocal x</td>
<td>Re-bind &quot;x&quot; to 2 in the first non-local frame of the current environment in which it is bound.</td>
</tr>
<tr>
<td>• &quot;x&quot; is bound in a non-local frame</td>
<td></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 non-local frame</td>
<td></td>
</tr>
<tr>
<td>• nonlocal x</td>
<td>SyntaxError: name 'x' is parameter and nonlocal</td>
</tr>
<tr>
<td>• &quot;x&quot; also bound locally</td>
<td></td>
</tr>
</tbody>
</table>
Python Particulars

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

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

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

wd = make_withdraw(20)
wd(5)
```

`UnboundLocalError: local variable 'balance' referenced before assignment`

Example: [http://goo.gl/b0Vzc6](http://goo.gl/b0Vzc6)
Mutable Values & Persistent Local State

Mutable values can be changed *without* a nonlocal statement.

Example: [http://goo.gl/y4TyFZ](http://goo.gl/y4TyFZ)
Multiple Mutable Functions

(Demo)
Sameness and Change

*As long as we never modify objects, we can regard a compound object to be precisely the totality of its pieces.

*A rational number is just its numerator and denominator.

*This view is no longer valid in the presence of change.

*Now, a compound data object has an "identity" that is something more than the pieces of which it is composed.

*A bank account is still "the same" bank account even if we change the balance by making a withdrawal.

*Conversely, we could have two bank accounts that happen to have the same balance, but are different objects.

<table>
<thead>
<tr>
<th>John's Account</th>
<th>Steven's Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10</td>
<td>$10</td>
</tr>
</tbody>
</table>
Expressions are referentially transparent if substituting an expression with its value does not change the meaning of a program.

\[
\begin{align*}
\text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)) \\
\text{mul}(\text{add}(2, 24), \text{add}(3, 5)) \\
\text{mul}(26, \text{add}(3, 5))
\end{align*}
\]

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