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

Roadmap

Introduction

Functions

Data

Mutability

Objects

Interpretation

Paradigms

Applications

- 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?

```
>>> withdraw = make_withdraw(100)
>>> withdraw(25)
75
```

Return value: remaining balance

Argument: amount to withdraw

```
>>> withdraw(25)
50
```

Different return value!

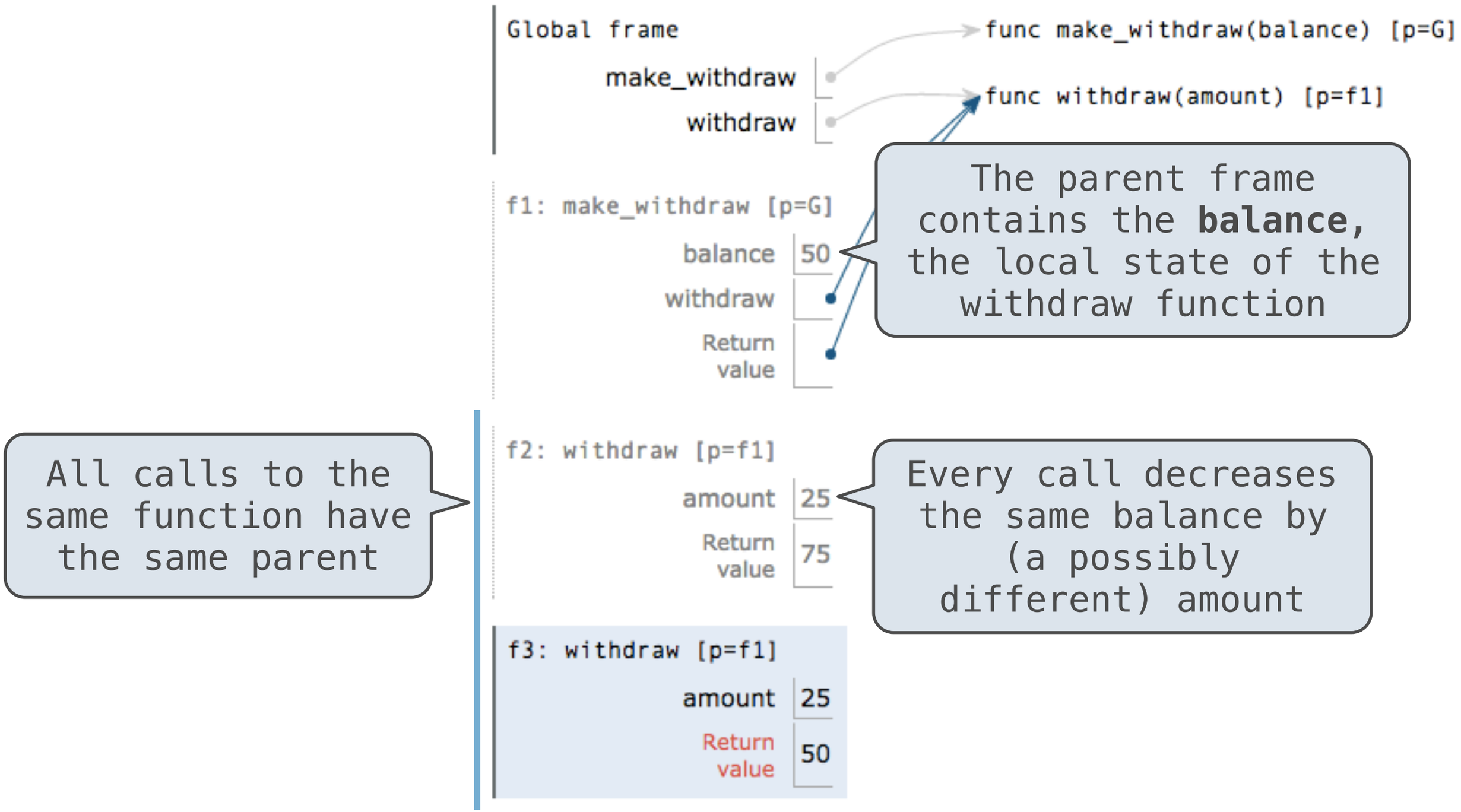
Second withdrawal of the same amount

```
>>> withdraw(60)
'Insufficient funds'

>>> withdraw(15)
35
```

Where is this balance stored?

Persistent Local State in Environments



Nonlocal Assignment (demo)

```
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
```

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.

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.

Current frame

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
```

Status	Effect
• No nonlocal statement • "x" **is not** bound locally	Create a new binding from name "x" to value 2 in the first frame of the current environment
• No nonlocal statement • "x" **is** bound locally	Re-bind name "x" to value 2 in the first frame of the current environment
• nonlocal x • "x" **is** bound in a nonlocal frame	Re-bind "x" to 2 in the first nonlocal frame of the current environment in which "x" is bound
• nonlocal x • "x" **is not** bound in a nonlocal frame	SyntaxError: no binding for nonlocal 'x' found
• nonlocal x • "x" **is** bound in a nonlocal frame • "x" also bound locally	SyntaxError: name 'x' is parameter and nonlocal

Python Particulars (demo)

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

Local assignment

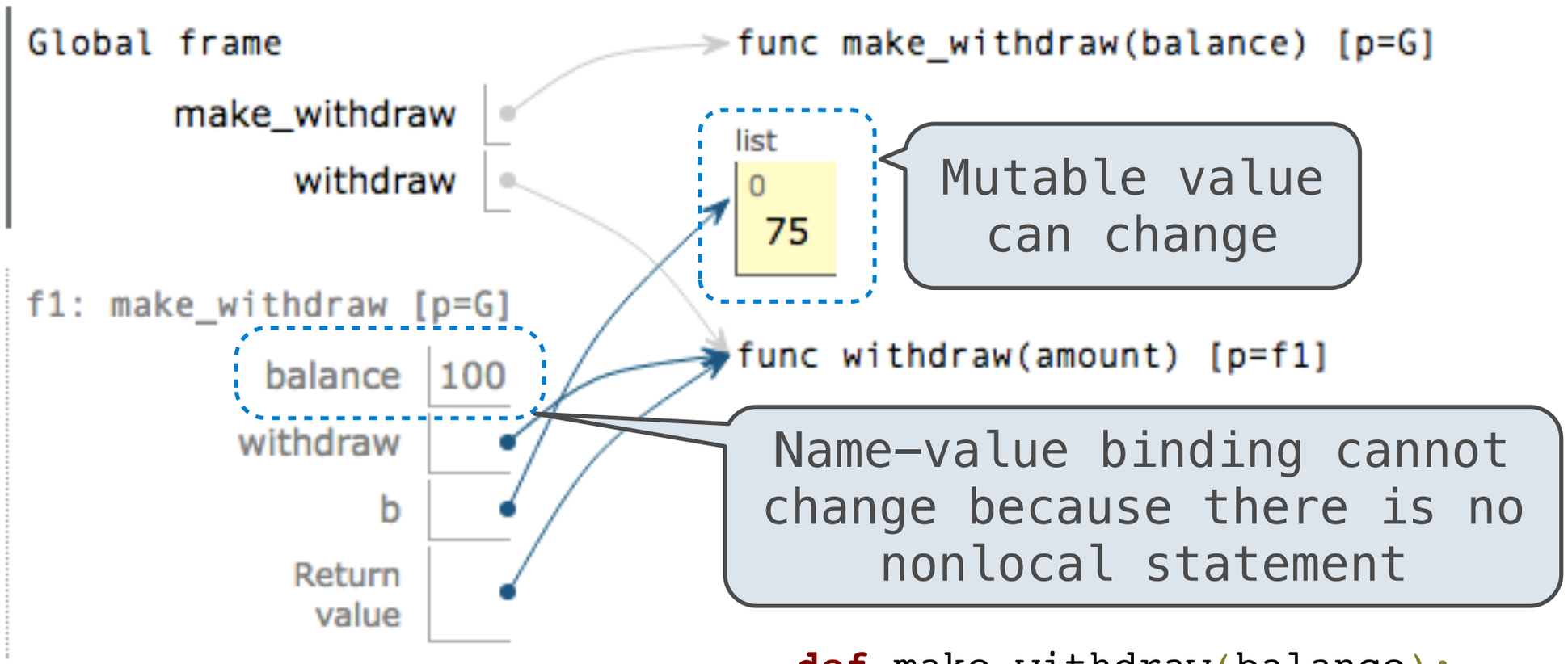
UnboundLocalError: local variable 'balance' referenced before assignment

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 (demo)




```
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 (demo)

```
>>> 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.

```
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**

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!