Lecture 8: Diagnostic Review & Debugging

July 1st, 2021

Alex Kassil

Announcements

- The Diagnostic Quiz is 5PM-6PM PT this Friday, 7/2.
    - Format: a web-based exam of short answer, multiple choice, and free code-writing
    - Video proctoring will happen through Zoom.
    - A practice diagnostic exam is available here.
    - You may use as many pages of notes of your own creation as you'd like.
    - If you want to store your notes electronically, you must use a Google Doc and grant edit access to cs61a@berkeley.edu.
    - You will get an email with the encrypted exam pdf before your exam time
- Lab 03 is due tonight
- Hog Phase 1 is due tonight checkpoint is tonight
    - Just worth 1 point, you can submit phase 1 later and still get credit for it if you don’t finish the checkpoint tonight
- Hog Project is due Wednesday, July 7th
    - Submit by Tuesday, July 6th for an early submission bonus point!
- There will be Vitamin 04 released over the weekend due 8am Tuesday. We will email and make a post on edstem.org as soon as it is released
- No lecture monday, enjoy your long fourth of july weekend

Understanding and Differentiating between Iteration vs Recursion vs Higher Order Functions

Iteration vs Recursion vs Higher Order Functions

Let’s write a function that adds all the digits in a number in three different ways!

Let’s also practice using code.cs61a.org for the diagnostic quiz

Compare Iteration vs Recursion vs HOF

```
def sum_digits_iterative(n):
    result = 0
    while n:
        result += n % 10
        n //= 10
    return result

sum_digits_iterative(1234) == 10

def sum_digits_recursive(n):
    if n == 0:
        return 0
    return n % 10 + sum_digits_recursive(n // 10)

sum_digits_recursive(1234) == 10

def sum_digits_hof(n):
    def inner(total):
        if total == -1:
            return n
        return sum_digits_hof(n + total)
    return inner

sum_digits_hof(1)(2)(3)(4)(-1) == 10
```

Iterative

- Returns a number
- No function call in body
- Looping with while

Recursive

- Return with function call
- Base case
- Function calls itself

HOF

- Returns a function
- Defines an inner function
- Inner function calls outer function

Error Types

SyntaxError

- What it technically means
  - The file you ran isn’t valid python syntax
- What it practically means
  - You made a typo
- What you should look for
  - Extra or missing parenthesis
  - Missing colon at the end of an if, while, def statement, etc
  - You started writing a statement but forgot to put anything inside

IndentationError/TabError

- What it technically means
  - The file you ran isn’t valid python syntax, because of indentation inconsistency
- What it practically means
  - You used the wrong text editor
- What you should look for
  - You made a typo and misaligned something
  - You accidentally mixed tabs and spaces
    - Open your file in an editor that shows them
    - `cat -A filename.py` will show tabs and spaces

TypeError: … ‘X’ object is not callable ...

- What it technically means
  - Objects of type X cannot be treated as functions
- What it practically means
  - You accidentally called a non-function as if it were a function
- What you should look for
  - Variables that should be functions being assigned to non-functions
  - Local variables that do not contain functions having the same name as functions in the global frame

TypeError: … NoneType ...

- What it technically means
  - You used None in some operation it wasn’t meant for
- What it practically means
  - You forgot a return statement in a function
- What you should look for
  - Functions missing return statements
  - Printing instead of returning a value

NameError

- What it technically means
  - Python looked up a name but didn’t find it
- What it practically means
  - You made a typo
  - You are trying to access variables from the wrong frame
- What you should look for
  - A typo in the name
  - The variable either being defined in the current frame, or in one of the parent frames

UnboundLocalError

- What it technically means
  - A local to a frame variable is used before it is assigned
- What it practically means
  - You are trying to both use a variable from a parent frame, and have the same variable be a local variable in the current frame
- What you should look for
  - Assignment statements after using the variable

Tracebacks

Parts of a Traceback

- Components
  - The error message itself
  - Lines #s on the way to the error
  - What’s on those lines
- Most recent call is at the bottom

```
def f(x):
    1 / 0
def g(x):
    f(x)
def h(x):
    g(x)
print(h(2))
```

```
Traceback (most recent call last):
  File "temp.py", line 7, in
<module>
    print(h(2))
  File "temp.py", line 6, in h
    g(x)
  File "temp.py", line 4, in g
    f(x)
  File "temp.py", line 2, in f
    1 / 0
ZeroDivisionError: division by zer
```

How to read a Traceback

1. Read the error message
   a. Remember what common error messages mean!
2. Look at each line, bottom to top, to see if you can find the error

```
def f(x):
    1 / 0
def g(x):
    f(x)
def h(x):
    g(x)
print(h(2))
```

```
Traceback (most recent call last):
  File "temp.py", line 7, in
<module>
    print(h(2))
  File "temp.py", line 6, in h
    g(x)
  File "temp.py", line 4, in g
    f(x)
  File "temp.py", line 2, in f
    1 / 0
ZeroDivisionError: division by zero
```

Diagnostic Review

Remove Digit

```
def remove_digit(n, digit):
    """Assuming N>=0, 0 <= DIGIT <= 9, return a number whose
    base-10 representation is the same as N, but with all instances
    of DIGIT removed.  If all digits removed, return 0.
    >>> remove_digit(123, 3)
    12
    >>> remove_digit(1234, 5)
    1234
    >>> remove_digit(1234, 1)
    234
    >>> remove_digit(111111, 1)
    0
    """
    if _______:
        return _______
    if _______:
        return _______
    return _______
```

Remove Digit

```
def remove_digit(n, digit):
    """Assuming N>=0, 0 <= DIGIT <= 9, return a number whose
    base-10 representation is the same as N, but with all instances
    of DIGIT removed.  If all digits removed, return 0.
    >>> remove_digit(123, 3)
    12
    >>> remove_digit(1234, 5)
    1234
    >>> remove_digit(1234, 1)
    234
    >>> remove_digit(111111, 1)
    0
    """
    if n == 0:
        return 0
    if n % 10 == digit:
        return remove_digit(n // 10, digit)
    return n % 10 + remove_digit(n // 10, digit) * 10
```

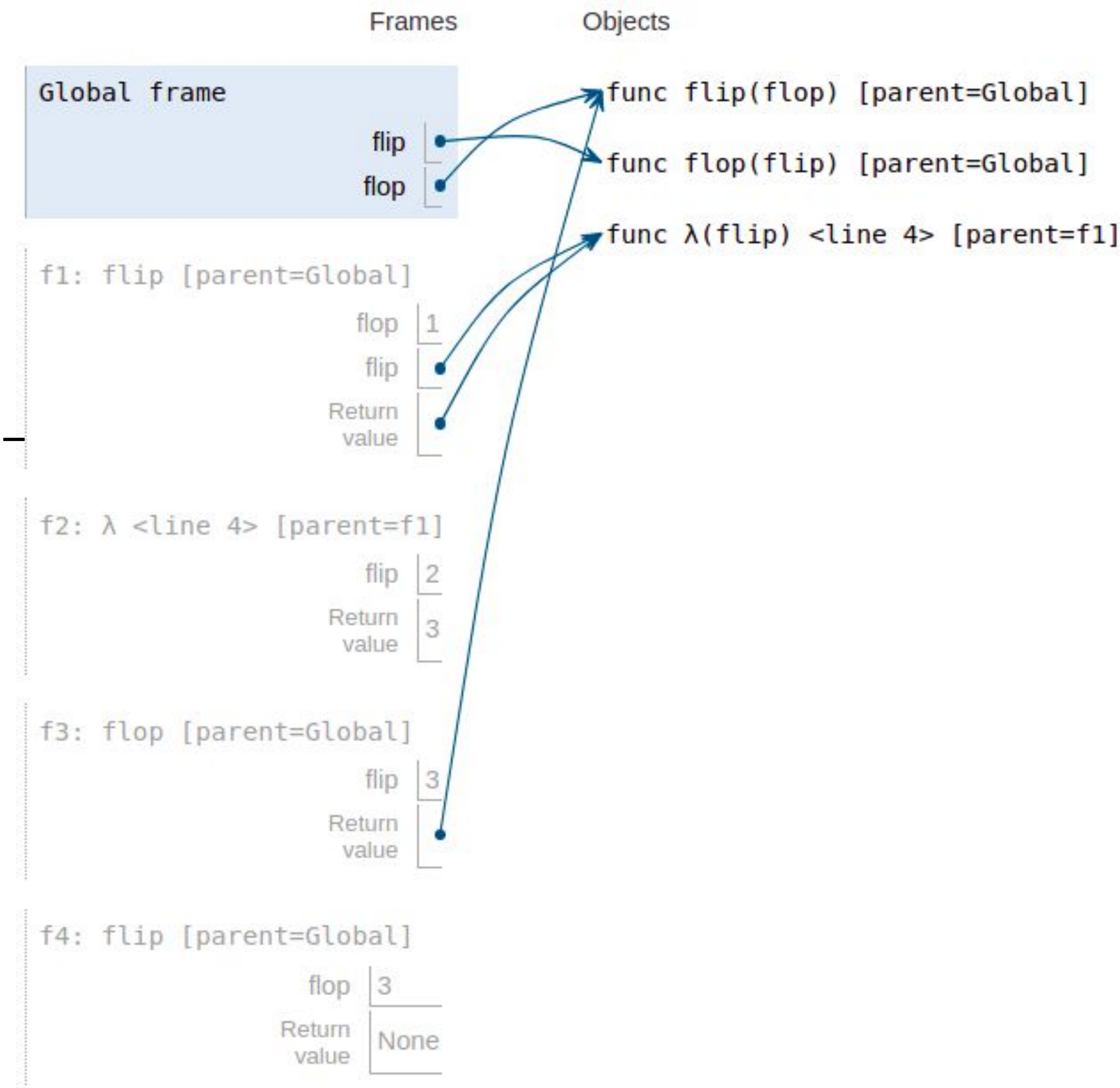
A Day at the Beach

```
def flip(flop):
    if ______:
        ______
    flip = ______
    return flip

def flop(flip):
    return flop

______

flip(____)(3)
```



Frames

Objects

Global frame

flip

flop

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) <line 4> [parent=f1]

f1: flip [parent=Global]

flop 1

flip

Return value

f2: λ <line 4> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value

f4: flip [parent=Global]

flop 3

Return value None

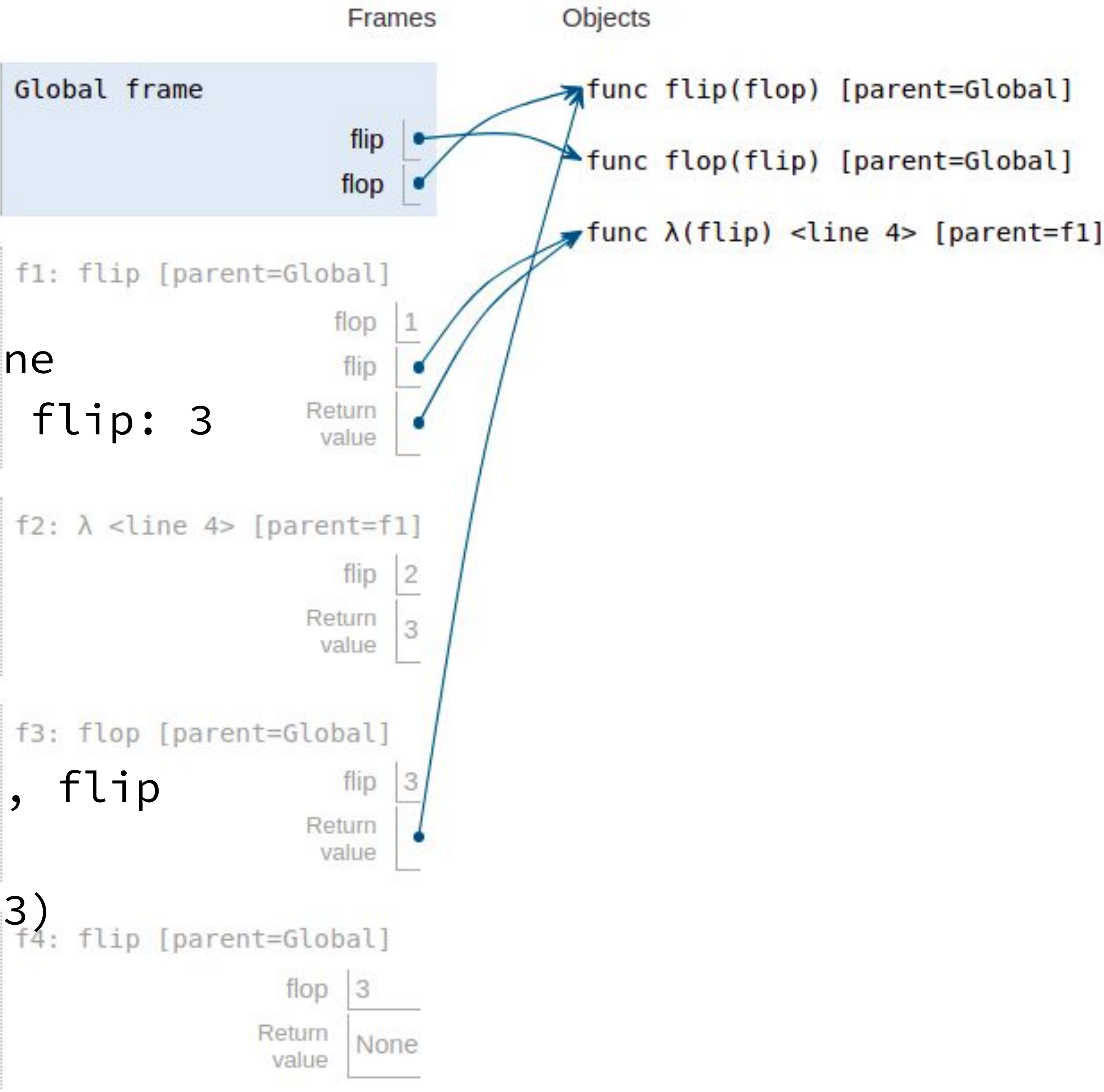
A Day at the Beach

```
def flip(flop):
    if flop > 2:
        return None
    flip = lambda flip: 3
    return flip

def flop(flip):
    return flop

flip, flop = flop, flip

flip(flop(1)(2))(3)
```



Frames

Objects

Global frame

flip ●

flop ●

func flip(flop) [parent=Global]

func flop(flip) [parent=Global]

func λ(flip) <line 4> [parent=f1]

f1: flip [parent=Global]

flop 1

flip ●

Return value ●

f2: λ <line 4> [parent=f1]

flip 2

Return value 3

f3: flop [parent=Global]

flip 3

Return value ●

f4: flip [parent=Global]

flop 3

Return value None

Interleave Digits

Given two numbers, A and B, containing the same number of digits, returns the result of interleaving the digits of A and B, starting with the first digit A, then the first digit of B, then the second digit of A, etc.

```
def interleave(a, b):
    """
    >>> interleave(1, 2)
    12
    >>> interleave(0, 1)
    1
    >>> interleave(1, 0)
    10
    >>> interleave(123,456)
    142536
    """
    if _______________:
        _______________
    return _______________
```

Interleave Digits

Given two numbers, A and B, containing the same number of digits, returns the result of interleaving the digits of A and B, starting with the first digit A, then the first digit of B, then the second digit of A, etc.

```
def interleave(a, b):
    """
    >>> interleave(1, 2)
    12
    >>> interleave(0, 1)
    1
    >>> interleave(123,456)
    142536
    """
    if a < 10 and b < 10:
        return a * 10 + b
    return interleave(a // 10, b // 10) * 100 + (a %
10) * 10 + b % 10
```

Sp21 Midterm 1 Problems

https://cs61a.org/exam/sp21/mt1/61a-sp21-mt1.pdf