CS61A
Discussion 1: Control and Higher Order Functions

TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Agenda

1. Attendance reminder

2. Check-in

3. Booleans/control:if (quick review)

4. Control: while (quick review)

5. Higher Order Functions

Reminder (from last week)

Help me do my job:

- **Ask questions**
- **Participate!**
- **Go to office hours (early) for help!**

Attendance

Form: **tinyurl.com/jerrydisc**

(Weekly question is not judged based on correctness)

Check-in

How was homework 1?

How was lab 1?

Started project 1 (Hog)?

Booleans

- There are “truthy” and “falsy” values:

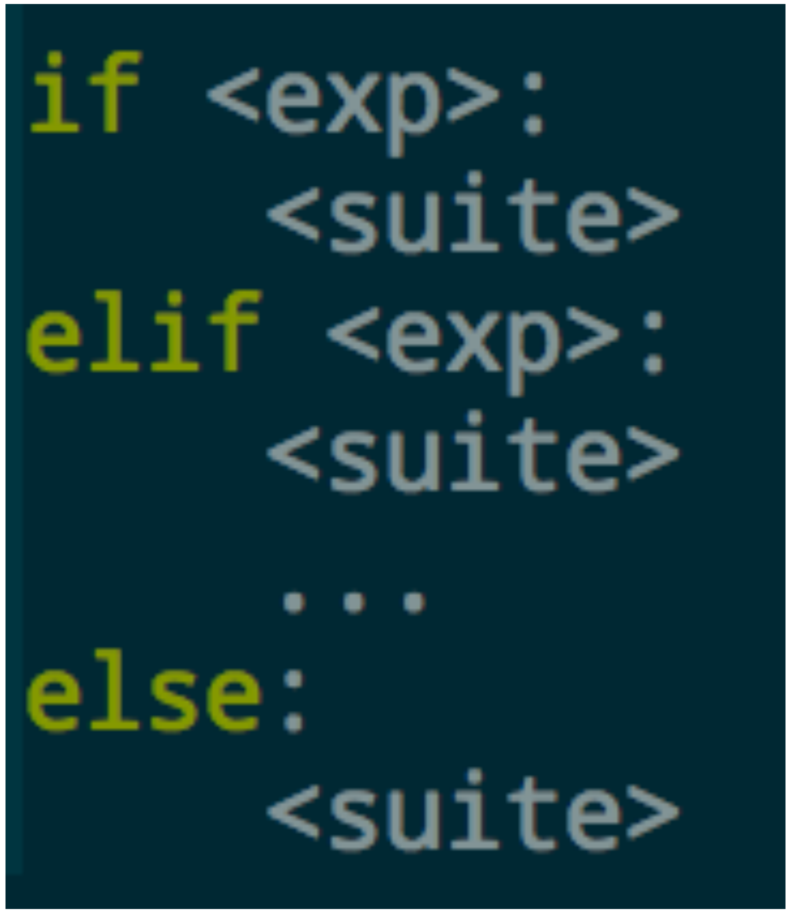
“Truthy”	“Falsy”	Notes
`True`	`False`	
`“banana”`	`‘’`	Empty string
`100, -12`	`0`	
`[1, 2, 3], {‘a’: 1, ‘b’: 2}`	`[], {}`	Will see later in the course

Boolean Operators

- **not** (negates)**,**
- **and** (true iff both are true)**,**
- **or** (false iff both are false)
- **Short circuit** and terminate early once the **result of a expression is known**

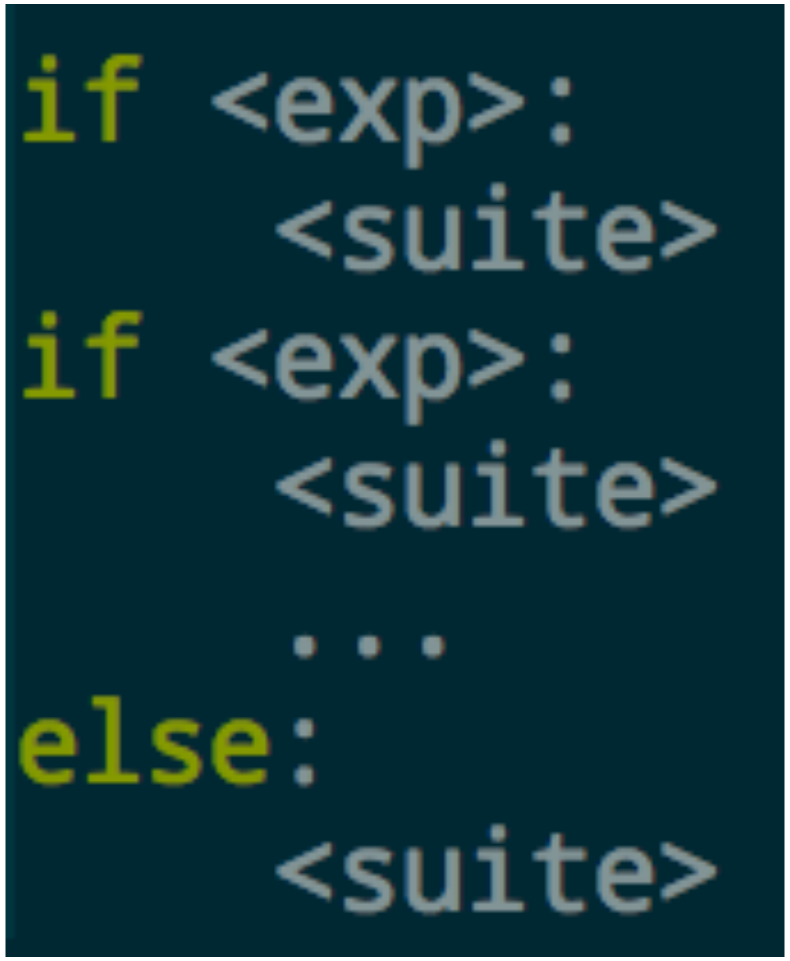
Control

If statements



```
if <exp>:
    <suite>
elif <exp>:
    <suite>
    ...
else:
    <suite>
```

Careful!



```
if <exp>:
    <suite>
if <exp>:
    <suite>
    ...
else:
    <suite>
```

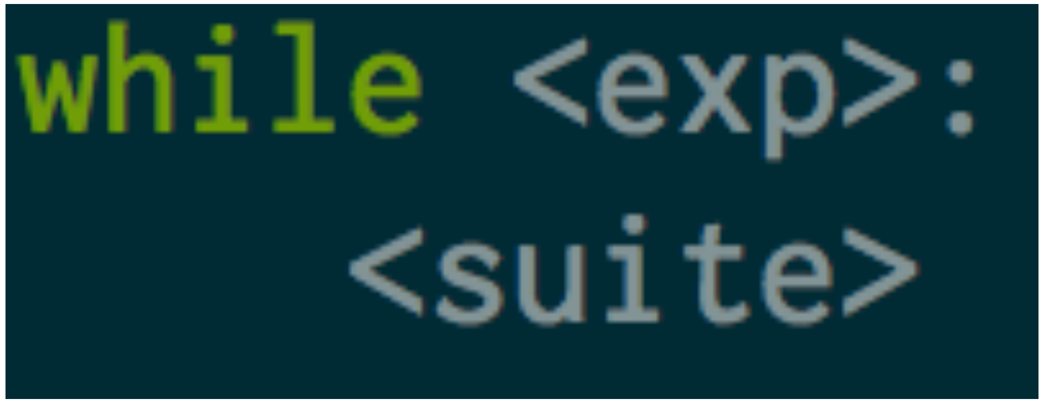
Booleans/If Practice

- Questions in section 1.3

Control

While statements

- <exp> is checked **before** executing the suite

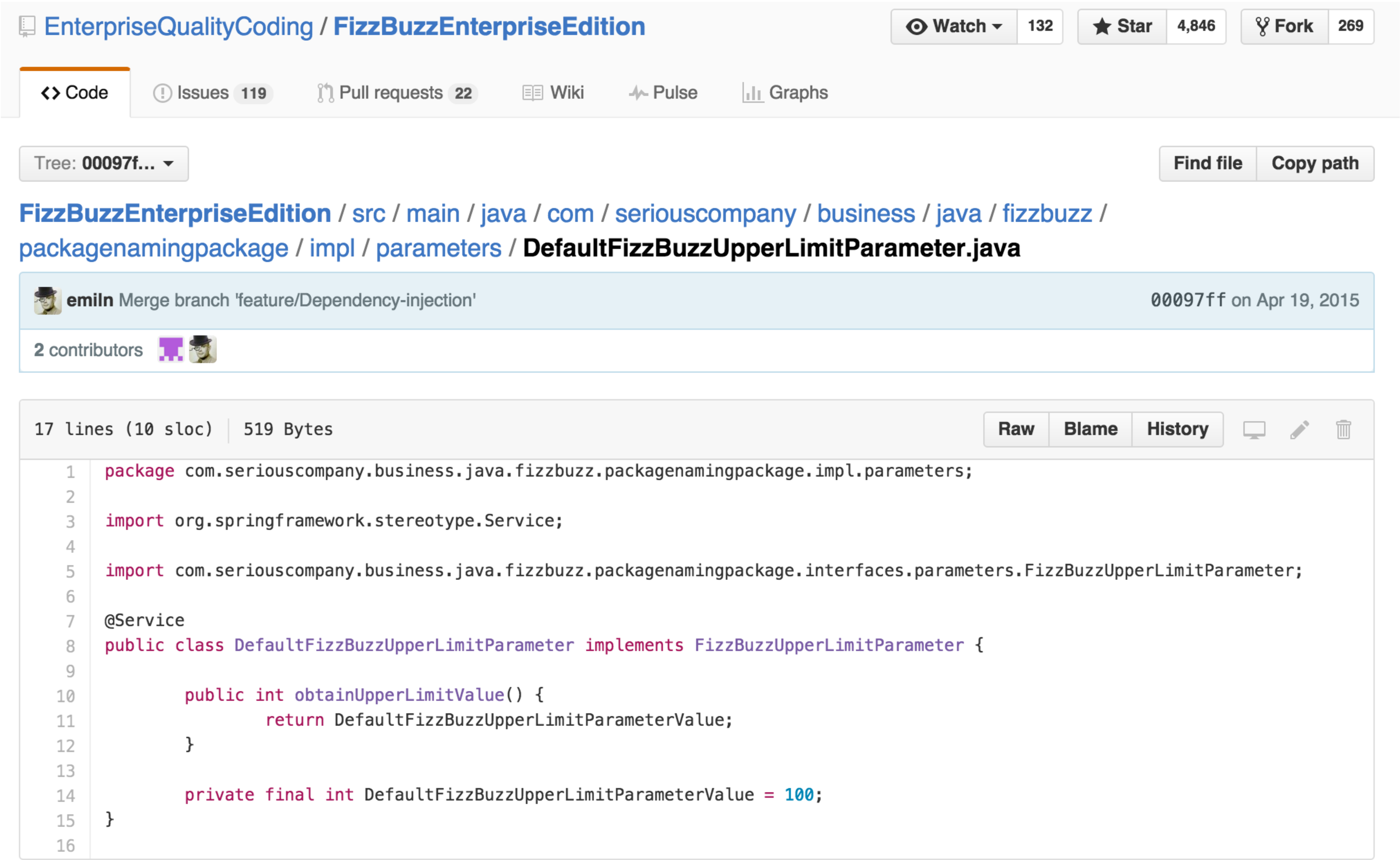


```
while <exp>:
    <suite>
```

While Practice

- Questions in section 1.4

FizzBuzz



FizzBuzzEnterpriseEdition / src / main / java / com / seriouscompany / business / java / fizzbuzz / packagenamingpackage / impl / parameters / **DefaultFizzBuzzUpperLimitParameter.java**

emiln Merge branch 'feature/Dependency-injection' 00097ff on Apr 19, 2015

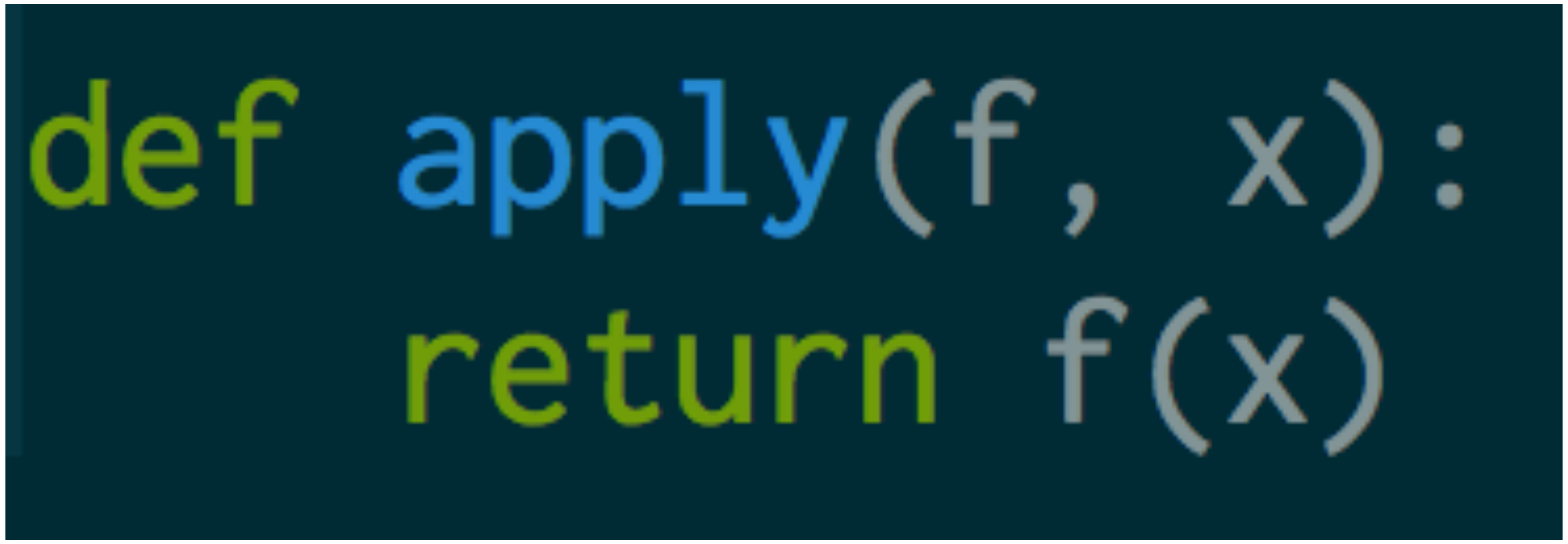
2 contributors

17 lines (10 sloc) | 519 Bytes

```java
package com.seriouscompany.business.java.fizzbuzz.packagenamingpackage.impl.parameters;

import org.springframework.stereotype.Service;

import com.seriouscompany.business.java.fizzbuzz.packagenamingpackage.interfaces.parameters.FizzBuzzUpperLimitParameter;

@Service
public class DefaultFizzBuzzUpperLimitParameter implements FizzBuzzUpperLimitParameter {

        public int obtainUpperLimitValue() {
                return DefaultFizzBuzzUpperLimitParameterValue;
        }

        private final int DefaultFizzBuzzUpperLimitParameterValue = 100;
}

```

Higher Order Functions

Functions as arguments



```
def apply(f, x):
    return f(x)
```

Higher Order Functions

Big idea: **Functions can be treated as “variables” — a powerful tool for abstraction!**

- Can pass as arguments or returned
- Analogy is a bit limited, can’t necessarily “add” two functions

Functions that manipulate other functions are **higher order**

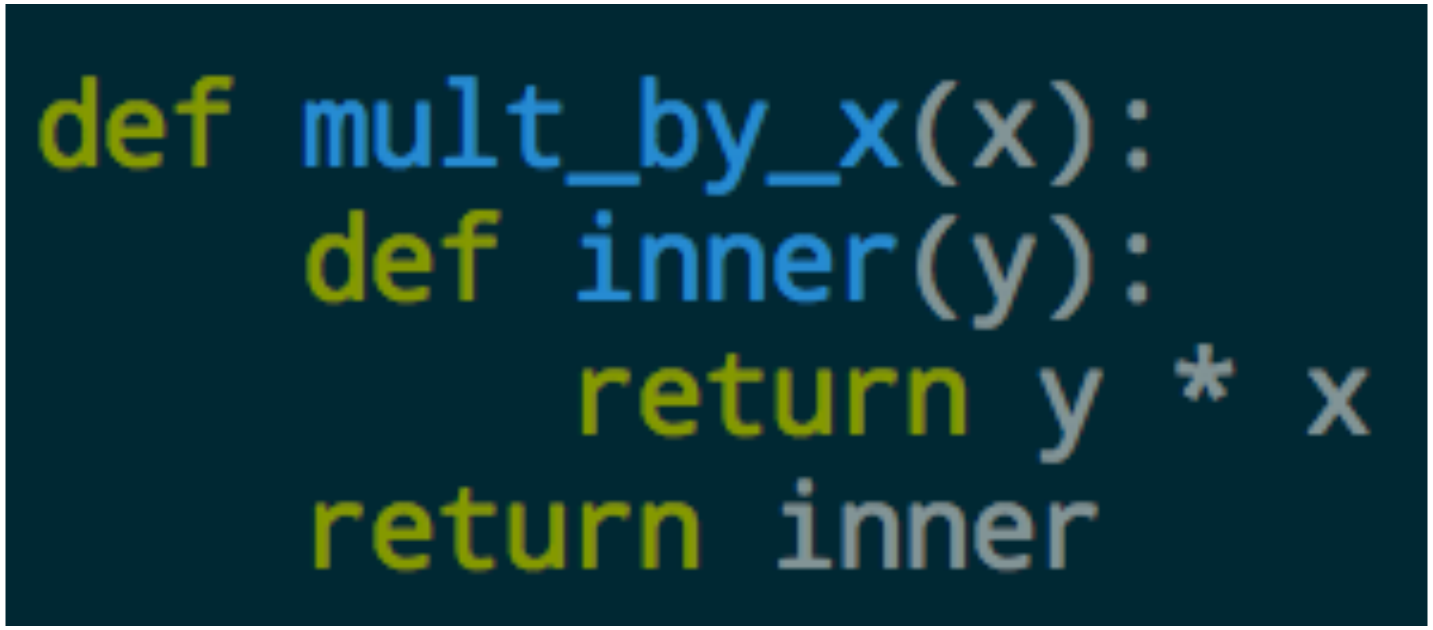
HOF Practice

- Questions in section 2.2

Higher Order Functions

Functions as return values

- Might need to define a function within another function

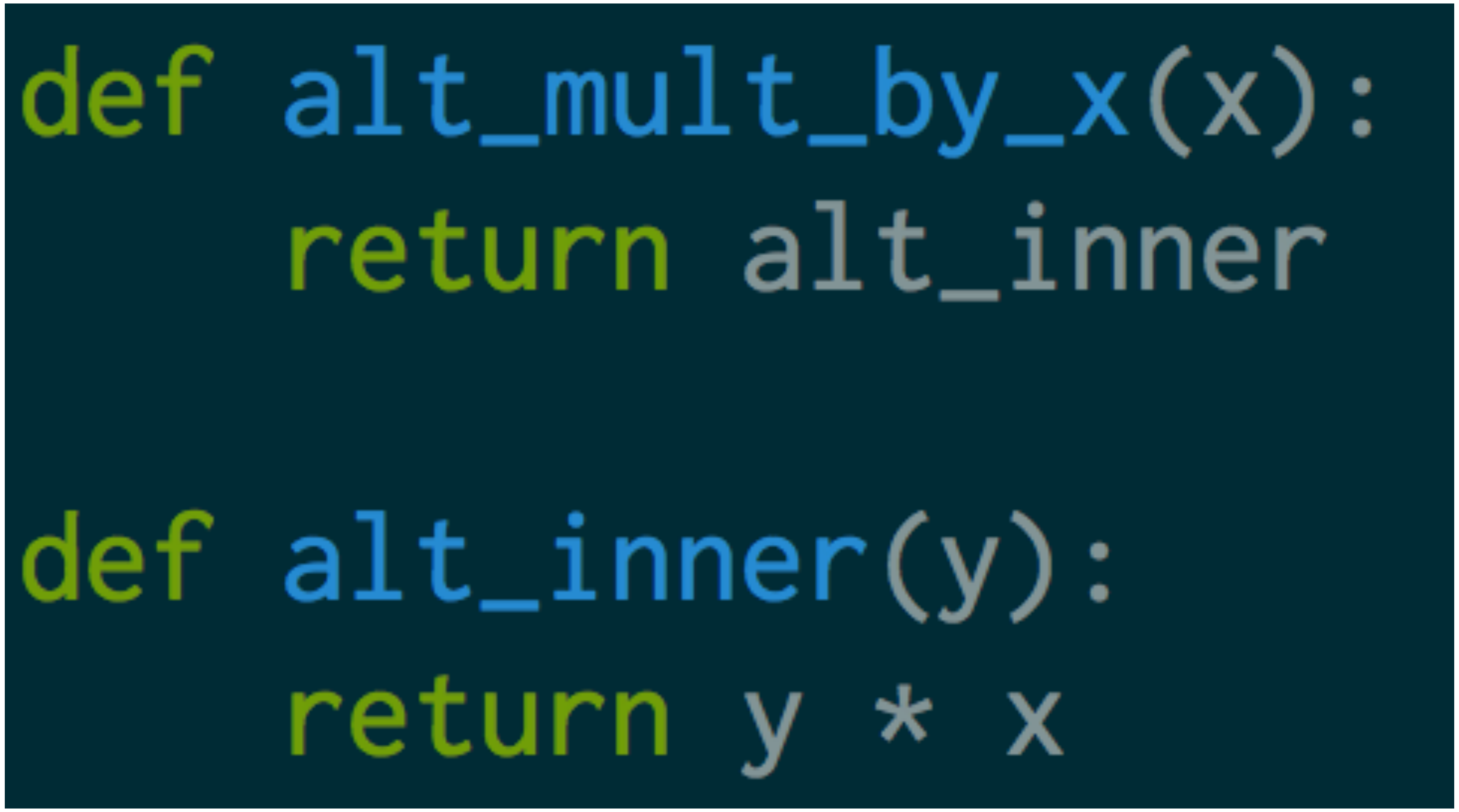


```
def mult_by_x(x):
    def inner(y):
        return y * x
    return inner
```

Higher Order Functions

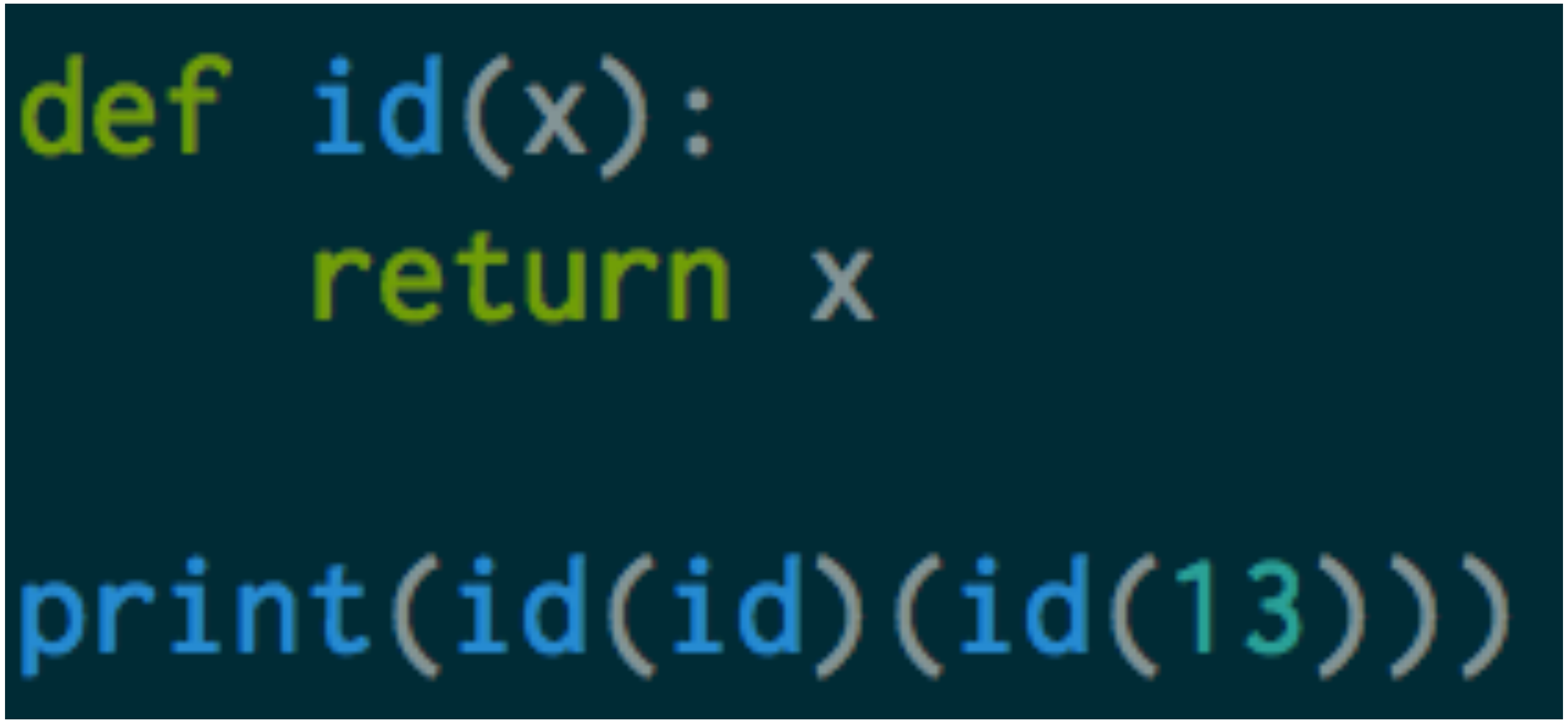
Q: Why not put inner in global?

A: x is in the wrong frame! (what’s the parent of `alt_inner`?)



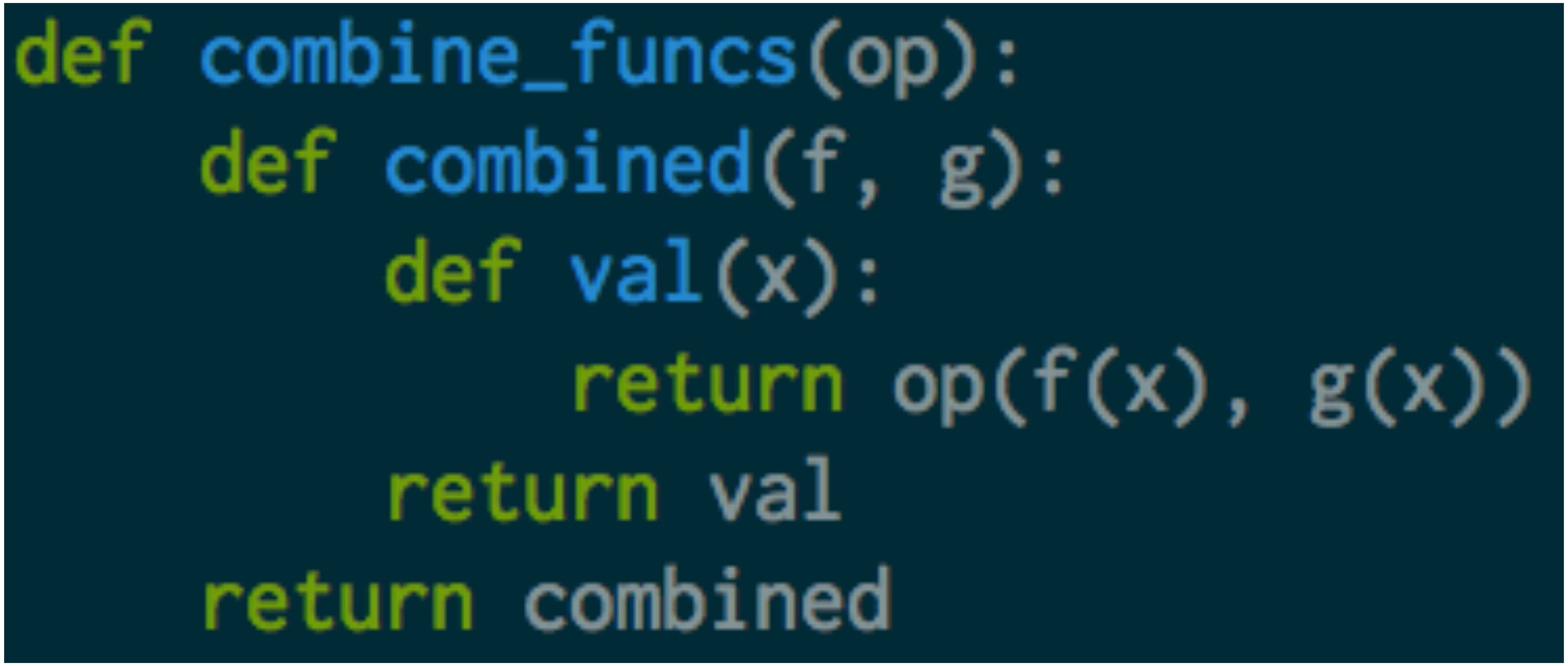
```
def alt_mult_by_x(x):
    return alt_inner

def alt_inner(y):
    return y * x
```

“Fun” example from lecture



```python
def id(x):
    return x

print(id(id)(id(13)))
```

Another Example



```
def combine_funcs(op):
    def combined(f, g):
        def val(x):
            return op(f(x), g(x))
        return val
    return combined
```

HOF Practice

- Questions in section 2.4