Hi! Welcome to 61A Discussion :)

We will begin at **5:10**!

Attendance: **go.cs61a.org/ben-disc**

Slides + Info: **cs61a.bencuan.me**

Secret Word ™: written on board

Announcements

- MT1 is a thing that exists
  - even more on this at the end of discussion
- Hog checkpoint and HW2 due tonight!
- No lab next week

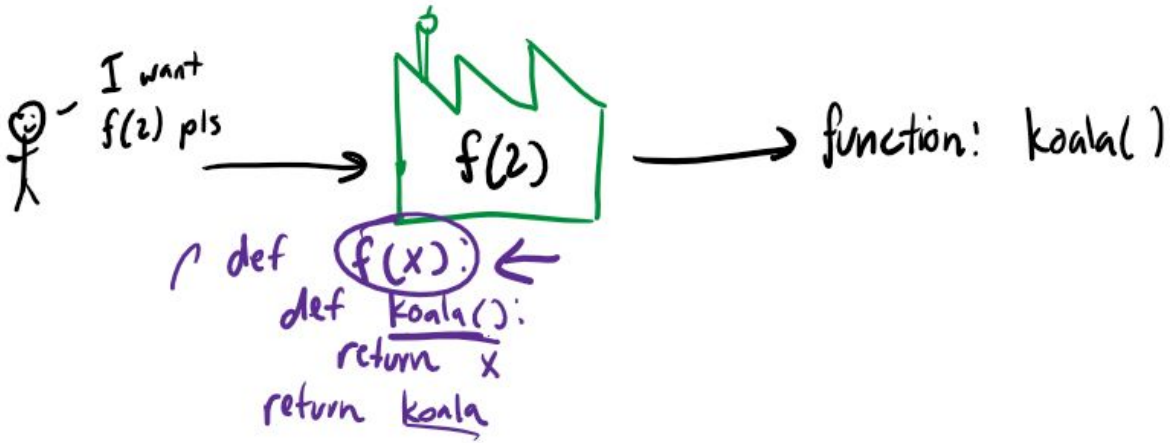
Agenda

- Attendance
- Environment diagrams
- Higher order functions
- Putting everything together
- Exam prep (?)

Lambda Warmup!

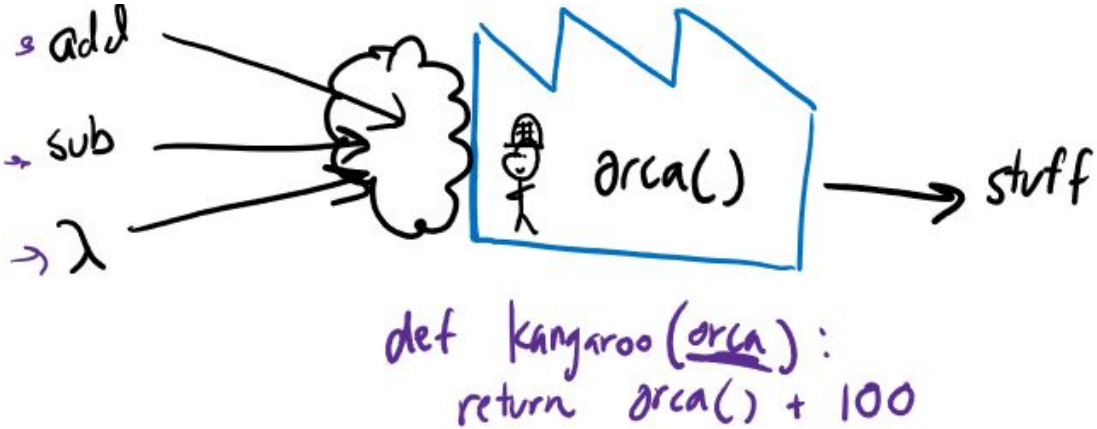
Lab review

HOF's are like factories that produce made-to-order functions



Lab review (2)

They can also take functions in as parameters!



Lab review (3)

Lambda too confusing? Turn them into normal def statements!!

```
f = lambda ox: ox + 1
```

```
def f(ox):
    return ox + 1
```

Lab review (4)

Lambdas can return functions too!

```
b = lambda ox: lambda turtle: ox + turtle
```

```
def b(ox):
    def anonymous(turtle):
        return ox + turtle
    return anonymous
```

Converting lambdas <-> defs

Try one for practice:

```
def sea(turtle):
    def goblin(shark):
        return shark + shark
    return goblin
```

Q5: Make Keeper

Write a function that takes in a number `n` and returns a function that can take in a single parameter `cond`. When we pass in some condition function `cond` into this returned function, it will print out numbers from 1 to `n` where calling `cond` on that number returns `True`.

```
def make_keeper(n):
    """Returns a function which takes one parameter cond and
    prints out all integers 1..i..n where calling cond(i) returns True.

    >>> def is_even(x):
    ...     # Even numbers have remainder 0 when divided by 2.
    ...     return x % 2 == 0
    >>> make_keeper(5)(is_even)
    2
    4
    """
```

Environment Diagrams

...why are we doing this?

- Really, really good for understanding how computers interpret code
- Helpful for debugging programs
- Project 4: you'll make your own interpreter

make your own ED's

tutor.cs61a.org

Basic ED rules

- Boxes hold **values**
  - Evaluate all expressions fully!
- Arrows point to **functions/objects**

- **Function definition** vs **Function call**
  - Creating **new arrow** vs creating **new frame**

Disc1 Q7/8 (example)

Q7: Assignment Diagram

Use these rules to draw an environment diagram for the assignment statements below:

```
x = 11 % 4
y = x
x **= 2
```

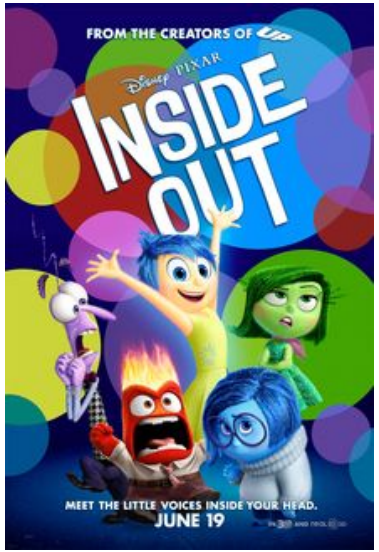
Q8: def Diagram

Use these rules for defining functions and the rules for assignment statements to draw a diagram for the code below.

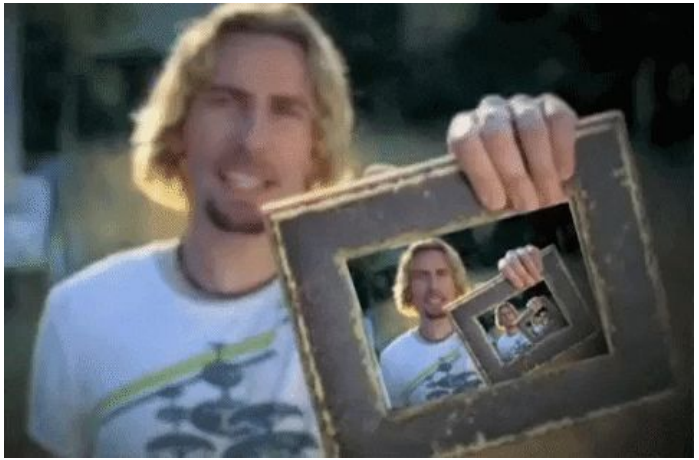
```
def double(x):
    return x * 2

def triple(x):
    return x * 3

hat = double
double = triple
```

The 6 steps of a Function Call

1. **Evaluate the operator**
2. **Evaluate the operands**
    a. Inside out, left to right
3. **Create a new frame**
4. **Copy parameters**
5. **Evaluate body**
6. **Return value**



Symbol lookup



Trying to find variable x?

1. Look in the current frame
2. Look in the parent frame
3. Look in the parent's parent frame
4. Look in the parent's parent's parent frame
5. ...
6. If there are no more frames, then error

Example

```
1  x = 4
2  def add_num(x):
3      return lambda y: x + y
4
5  add_two = add_num(2)
6  add_two(3)
```

Q1: Call Diagram

Let's put it all together! Draw an environment diagram for the following code. You may not have to use all of the blanks provided to you.

```
def double(x):
    return x * 2

hmmm = double
wow = double(3)
hmmm(wow)
```

Q2: Nested Calls Diagrams

Draw the environment diagram that results from executing the code below. You may not need to use all of the frames and blanks provided to you.

```
def f(x):
    return x

def g(x, y):
    if x(y):
        return not y
    return y

x = 3
x = g(f, x)
f = g(f, 0)
```

Q4: Make Adder

Draw the environment diagram for the following code:

```
n = 9
def make_adder(n):
    return lambda k: k + n
add_ten = make_adder(n+1)
result = add_ten(n)
```

There are 3 frames total (including the Global frame). In addition, consider the following questions:

1. In the Global frame, the name `add_ten` points to a function object. What is the intrinsic name of that function object, and what frame is its parent?
2. What name is frame `f2` labeled with (`add_ten` or λ)? Which frame is the parent of `f2`?
3. What value is the variable `result` bound to in the Global frame?

Midterm Info

General info

When: 8-10pm Monday the 13th

Where: You'll know by Sunday! Will be in-person for most people

What: Control, HOFs, WWPD, (reverse) ED's

Also, **no lab next week!**

Some tips & tricks

- I have some posted on cs61a.bencuan.me
- https://cs61a.org/articles/studying/

TL;DR:

- Catch up on all your lectures+assignments
- Do 2-3 practice tests
- Get sleep
- **Remember to check your data types!!!!!!**

Stuff to add to your cheat sheet

- Check data types!
- Short circuiting examples
- Environment diagram rules (6 steps of function call, lookup rules...)
- Example of an environment diagram problem
- Lambda to def conversion
- Examples of python syntax (def, while, if, return)
- A smiley face or some variant :)