CS61A Discussion 2

http://tinyurl.com/61adisc2

Announcements

- Hog due Today!
- HW3 due next Tues 9/12
- Lab due tomorrow

Some FAQs from last discussion

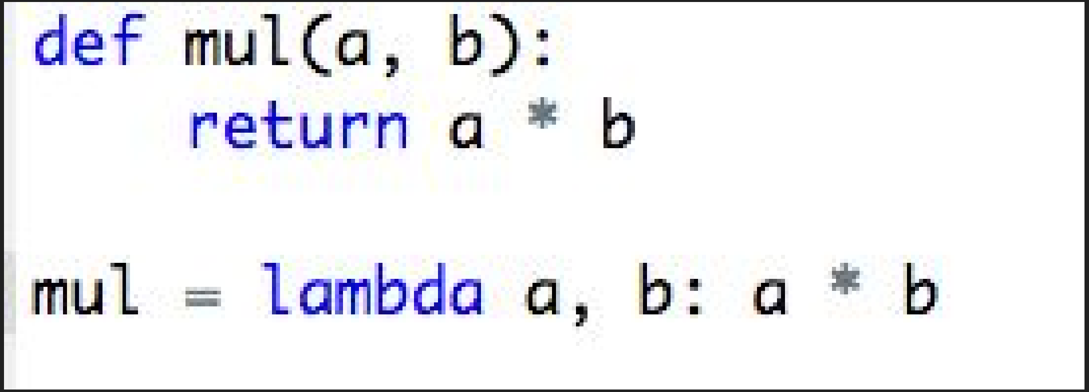
- If a name is defined in frame, and the same name is assigned to something else in a child frame
  - Leaves the value in the parent frame unaffected
  - Creates a binding in the child frame
  - Example https://goo.gl/BWpRr8

Some Common Errors from Quiz 1

- When returning a number, Python outputs without quotes
  - Ex. return 3 => 3 (not ‘3’)
  - Returning a string will have quotes, printing a string will not
- print(square(x))
  - Python first evaluates square(x), then print will display that value
- The interactive output will display
  - Printed values
  - The return value of the expression (does not include nested function calls)

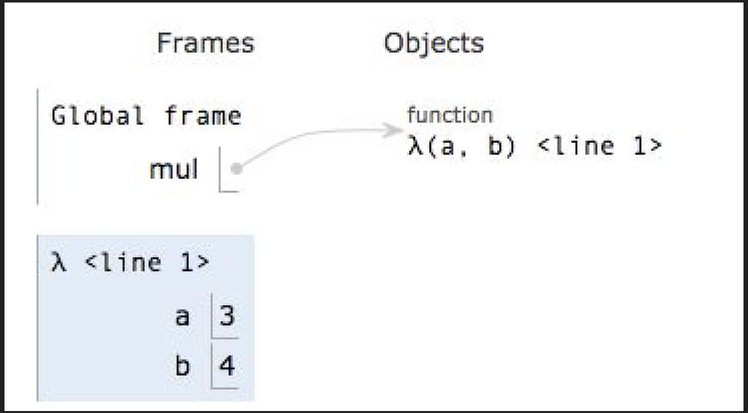
Lambda Expressions

- Shorthand for a def expression
- Useful for simple functions
    - Only one line
    - Body is just the return (no pre-processing!)
    - Parent frame is the frame in which it was defined



```
def mul(a, b):
    return a * b

mul = lambda a, b: a * b
```



Worksheet Time (Env Diag 1.1, 1.2)

Recursion

What is Recursion?

- Defining something in reference to itself
  - In this case, we define the return value of a function in terms of itself

Steps to Recursion

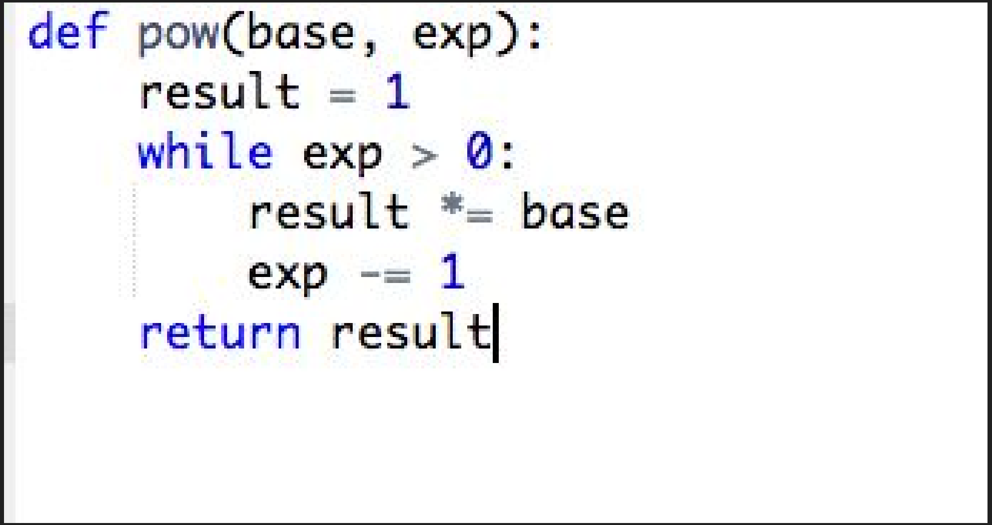
1. Define a base case
2. Work towards the base case
3. Make a recursive call (requires a leap of faith!)

Something familiar (maybe)

- Arithmetic sequences!
  - $a_n = a_{n-1} + 2$
  - $a_0 = 1$
- Try in python
  - http://pythontutor.com/live.html#mode=edit

Pow revisited

- http://pythontutor.com/live.html#mode=edit



```
def pow(base, exp):
    result = 1
    while exp > 0:
        result *= base
        exp -= 1
    return result
```

Worksheet Time (Recursion 2.2, 2.3, 2.4)

Tree Recursion

- Recursion sequence which makes multiple recursive calls in its recursive case
- $a_n = a_{n-1} + a_{n-2} + 1$
  - $a_0 = 3$
  - $a_1 = 6$
- Try in python
  - https://goo.gl/p3UWbP

Worksheet Time
(Tree Recursion 3.1, 3.2)