Environment Diagrams & Higher-Order Functions Meta

Computer Science Mentors 61A

September 9 – September 13, 2024

Example Timeline

- Environment Diagrams Mini Lecture + Q1 [15 Mins]

  1. You can use Q1 as an example for the mini-lecture

  2. A lot of students (even those with significant programming background) get confused by env. diagrams. It's probably worth doing the minilecture even if you have advanced students.

  3. You should address when we need to make a new frame in an environment diagram [This was the old Q1].

- Problems: Environment Diagrams [7 Mins]

  Q2 - Joke

- Higher Order Functions Overview + Reasoning [3 Mins]

  Make sure you explain why & where to use lambda functions & HOF functions. Give students a practical example if needed.

- Problems (You pick which ones): HOF [25 Mins]

  1. Q3 - Foobar
  2. Q4 - xyz
  3. Q5 - whole_sum
  4. Q6 - mystery
  5. Q7 - lambda-wwpd

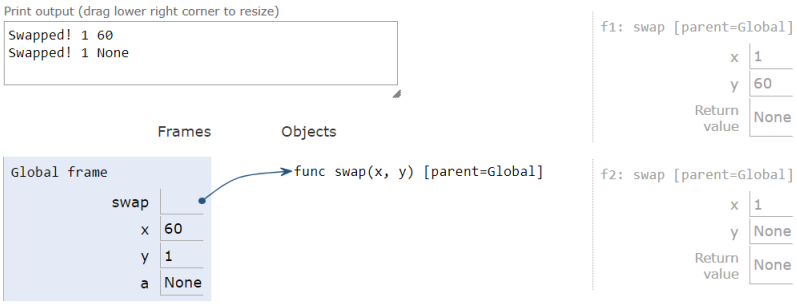
Created by Jim Fang, Bryan Adolfo Ventura Benitez. Worksheet feedback: links.jimfang.dev/feedback

1 Environment Diagrams

1. Give the environment diagram and console output that result from running the following code.

```
def swap(x, y):
    x, y = y, x
    return print("Swapped!", x, y)

x, y = 60, 1
a = swap(x, y)
swap(a, y)
```



https://tinyurl.com/y68m6qdj

Suggested Time: 5 min; Difficulty: Medium

- This question stresses variables in different scopes.

  Show difference between `x` and `y` in both global and local frames.

  Also note to students that a call to `swap(x, y)` will not actually swap the values of `x` and `y` in the frame where it is called.

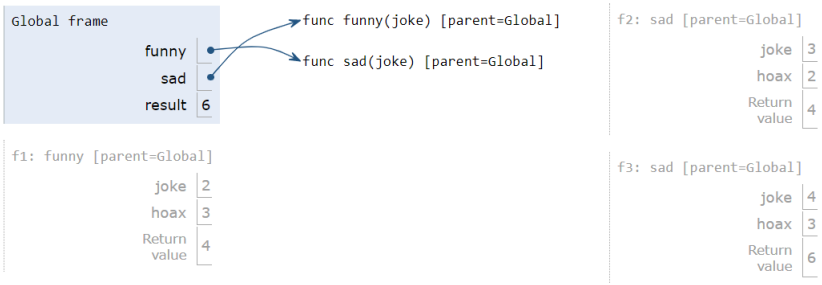
- It might also be good to recap what `x, y = y, x` does in python – ensure students know that this is a special feature of python and that switching happens in 1 line, by order of how the values are listed.

2. Draw the environment diagram that results from running the following code.

```
def funny(joke):
    hoax = joke + 1
    return funny(hoax)

def sad(joke):
    hoax = joke - 1
    return hoax + hoax

funny, sad = sad, funny
result = funny(sad(2))
```



https://tinyurl.com/y5lc4fez

Suggested Time: 7 min; Difficulty: Medium

- Make sure that the students understand how Python looks for a value of a variable, from local (to parent(s)) to global.
- Make sure your students understand the difference between an intrinsic name and a bound name

  Intrinsic: For user defined functions, this intrinsic name is the name used in the **`def`** statement

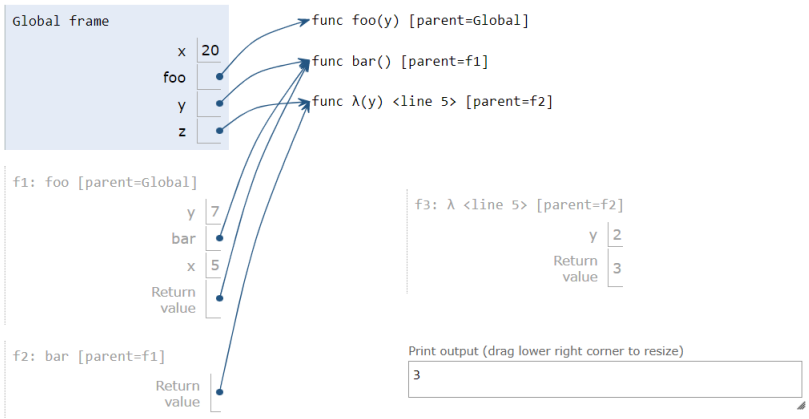
  Bound: Names of variables that point to the function object. A function can have many bound names, and the bound names of a function can often change.
- It may be good to remind your students to evaluate the functions on the right hand side first, then assign to variables on the left hand side.

2 Higher-Order functions

1. Give the environment diagram and console output that result from running the following code.

```
x = 20
def foo(y):
    x = 5
    def bar():
        return lambda y: x - y
    return bar

y = foo(7)
z = y()
print(z(2))
```



https://tinyurl.com/yxfcvxxa

Suggested Time: 10 min; Difficulty: Easy

2. Fill in the blanks (*without using any numbers in the first blank*) such that the entire expression evaluates to 9.

```
(lambda x: lambda y: lambda: y(x))(3)(lambda z: z*z)()
```

Suggested Time: 6 min; Difficulty: Medium

- Give your students advice on how to break down these nested fill in the blank/skeleton questions.

  You can tell them about the typical make an educated guess based on intuition, then testing it by plugging it in and running the function manually.

3. Write a function, `whole_sum`, which takes in an integer, `n`. It returns another function which takes in an integer, and returns `True` if the digits of that integer sum to `n` and `False` otherwise.

```
def whole_sum(n):
    """
    >>> whole_sum(21)(777)
    True
    >>> whole_sum(142)(10010101010)
    False
    """
    def check(x):

        ____________________

        while _______________:

            last = ____________________

            ____________________________

            ____________________________

        return ____________________

    return ____________________
```

```
def whole_sum(n):
    def check(x):
        total = 0
        while x > 0:
            last = x % 10
            x = x // 10
            total += last
        return total == n
    return check
```

Suggested Time: 8 Mins; Difficulty: Medium

- Remind your students that for HOFs, you must **`return`** the inner function (ie we must **`return`** `check` to use it).
- Also depending on the skill level of students in your section, a recap of digit manipulation may be needed (ie x // 10, x % 10, etc.)

4. Write a higher-order function that passes the following doctests.

   Challenge: Write the function body in one line.

```
def mystery(f, x):
    """
    >>> from operator import add, mul
    >>> a = mystery(add, 3)
    >>> a(4) # add(3, 4)
    7
    >>> a(12)
    15
    >>> b = mystery(mul, 5)
    >>> b(7) # mul(5, 7)
    35
    >>> b(1)
    5
    >>> c = mystery(lambda x, y: x * x + y, 4)
    >>> c(5)
    21
    >>> c(7)
    23
    """


    def helper(y):
        return f(x, y)
    return helper
```

   Challenge solution:

```
    return lambda y : f(x, y)
```

   Suggested Time: 5 Min - This problem is probably optional; Difficulty: Medium;

   Using doctests to understand how a function should work is a fundamental part of CS 61A. The goal of this question is to force students to exercise that muscle by removing any other description of `mystery`.

5. What would Python display?

   (a)
```
> (lambda x: x(x))(lambda y: 4)
4
```

   (b)
```
> (lambda x, y: y(x))(mul, lambda a: a(3, 5))
15
```

   Suggested Time: 5 - 7 Min; Difficulty: Medium - This problem is probably also optional unless your students want extra lambda or HOF practice.