

ENVIRONMENTAL DIAGRAMS AND HIGHER-ORDER FUNCTIONS Solutions

COMPUTER SCIENCE MENTORS 61A

January 30–February 3, 2023

1 Environment Diagrams

1. When do we make a new frame in an environment diagram?

We make a new frame in an environment diagram when calling a user-defined function, or when we are applying the operator to the operand(s). This occurs after both the operator and operand(s) are evaluated.

New frames are *not* created when a function is defined, only when it is called.

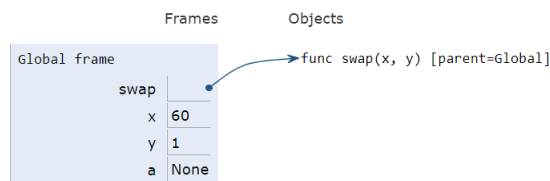
2. Give the environment diagram and console output that result from running the following code.

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

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

Print output (drag lower right corner to resize)

```
Swapped! 1 60  
Swapped! 1 None
```



```
f1: swap [parent=Global]  
    x | 1  
    y | 60  
    Return value | None
```

```
f2: swap [parent=Global]  
    x | 1  
    y | None  
    Return value | None
```

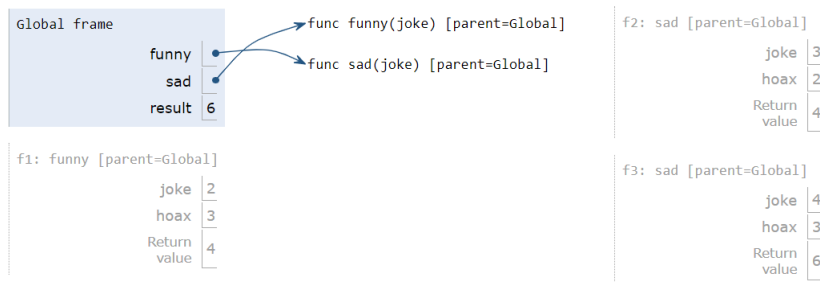
<https://tinyurl.com/y68m6qdj>

3. 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>

2 Higher-Order Functions

1. Why and where do we use lambda and higher-order functions? Can you give a practical example of where we would use a HOF?

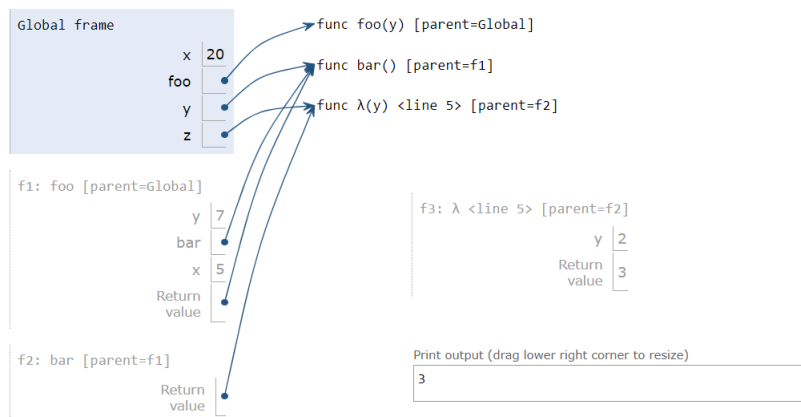
In practice, we use lambda functions to pass code as data in a concise manner. One specific example to illustrate the use of lambdas is the optional `key` parameter for `min` and `max` functions. Higher order functions serve as a tool of abstraction, allowing us to simplify repeated actions into one function that we can use over and over again, also referred to as currying. Students can have varying answers for practice uses of HOFs, though here are some suggestions for the average student coming across this worksheet:

- Our method signature is composed of one parameter, but we wish to use a higher order function with more parameters to abstract extra steps.
- When our function is long and complex; easier to read code when it's organized into several different higher order functions.
- Of course, the uses of higher order functions aren't limited to just these, but these are perhaps the most prominent uses. Feel free to email a CSM mentor about more uses!

2. 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>

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

4. 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
```

5. What would Python display?

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

```
>>> foo = mystery(lambda a, b: a(b), lambda c: 5 + square(c))  
>>> foo(-2)
```

9

6. 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