LAMBDAS

COMPUTER SCIENCE MENTORS CS 88

February 22 to 26

1 Lambdas

A lambda expression evaluates to a function, called a lambda function. For example, `lambda x, y:  x + y` is a lambda expression, and can be read as "a function that takes in two parameters `x` and `y` returns `x + y`."

A lambda expression by itself evaluates to a function but does not bind it to a name. Also note that the return expression of this function is not evaluated until the lambda is called. This is similar to how defining a new function using a def statement does not execute the function's body until it is later called.

```
>>> what = lambda x : x + 5
>>> what
<function <lambda> at 0xf3f490>
```

Unlike `def` statements, lambda expressions can be used as an operator or an operand to a call expression. This is because they are simply one-line expressions that evaluate to functions.

```
>>> (lambda y: y + 5)(4)
9
>>> (lambda f, x: f(x))(lambda y: y + 1, 10)
11
```

1. What do lambda expressions do? Can we write all functions as lambda expressions? (Hint: think about the limitations of lambdas) In what cases are lambda expressions useful?

   Solution: Lambda expressions create functions. When a lambda expression is evaluated, it produces a function. We often use lambdas to create short anonymous functions that we won't need for too long.
   We can't write all functions as lambda expressions because lambda functions all have to have "return" statements. In addition, they can't contain very complex multi-line expressions.
   Lambda expressions are very useful. In HOF, we want to pass functions into another function or have a function return a function. Lambdas expressions serve as a short and convenient way to define functions in HOFs.

2. Determine if each of the following will error:

```
>>> 1/0
```

   Solution: Error

```
>>> boom = lambda: 1/0
```

   Solution: No error, since we don't evaluate the body of the lambda when we define it.

```
>>> boom()
```

   Solution: Error

3. Express the following lambda expression using a **def** statement, and the **def** statement using a lambda expression.
   `pow = lambda x, y: x**y`

   Solution:

```
def pow(x, y):
    return x**y
```

```
def foo(x):
    def f(y):
        def g(z):
            return x + y * z
```

```
        return g
    return f
```

Solution: foo = lambda x: lambda y: lambda z: x + y * z

4. For each of the following lines of code, determine what would be printed as the output.

```
>>> plus_one = lambda i: print(i+1)
>>> plus_one
```

Solution: function <lambda> at 0x61CSTUFF

```
>>> plus_one(6)
```

Solution: 7

```
>>> multiply = lambda x, y: x*y
>>> harder_lambda = lambda func: print(func(4, 5))
>>> harder_lambda(multiply)
```

Solution: 20

5. What would Python print?

```
>>> a = lambda: 5
>>> a()
```

Solution:
```
5
```

```
>>> a(5)
```

Solution:
```
TypeError: <lambda>() takes 0 positional arguments but 1
   was given
```

```
>>> b = lambda: lambda x: 3
>>> b()(15)
```

Solution:
```
3
```

```
>>> c = lambda x, y: x + y
>>> c(4, 5)
```

Solution:
```
9
```

```
>>> d = lambda x: lambda y: x * y
>>> d(3)
```

Solution:
```
<function ...>
```

```
>>> d(3)(3)
```

Solution:
```
9
```

```
>>> e = d(2)
>>> e(5)
```

Solution:
```
10
```

```
>>> f = lambda: print(1)
```

Solution:
```
# No output
```

```
>>> g = f()
```

Solution:
```
1
```

6. **Challenge Problem**: Draw Environment Diagrams for the following lines of code. Note: When working with lambdas in environment diagram problems, it is really helpful to write down which line the lambda was defined on.

```
square = lambda x: x * x
higher = lambda f: lambda y: f(f(y))
b = higher(square)(5)
a = (lambda f, a: f(a))(lambda b: b * b, 2)
```

Solution: Solution: https://tinyurl.com/y69u6hu3

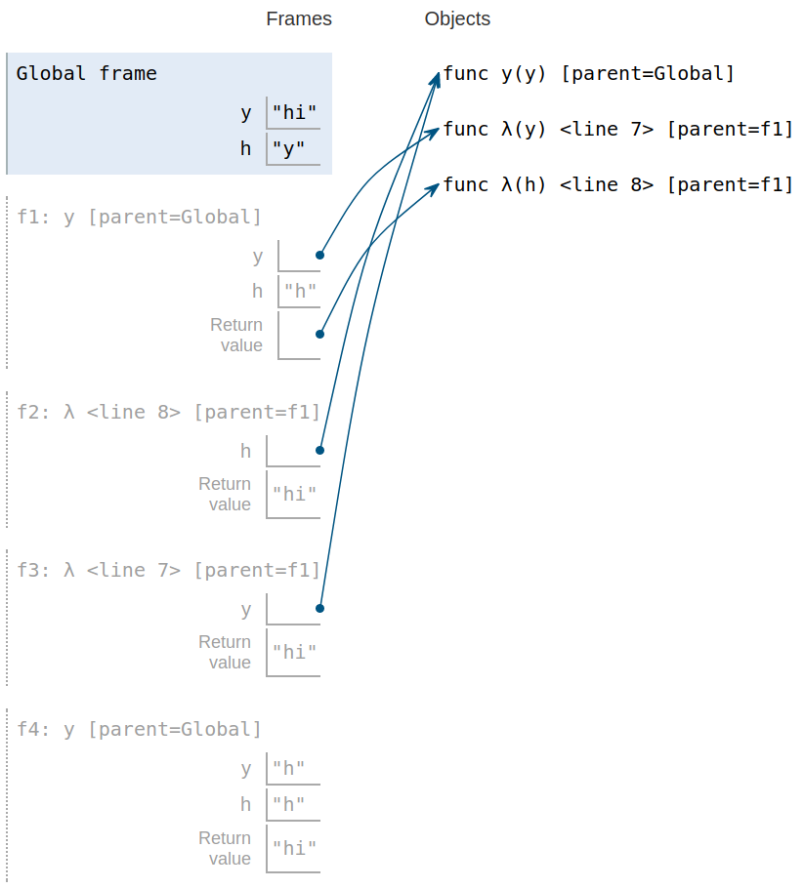
7. *The following question is extremely difficult. Something like this would not appear on the exam. Nonetheless, it's a fun problem to try.*

   Draw the environment diagram that results from executing the code below.

   Note that using the + operator with two strings results in the second string being appended to the first. For example `"C" + "S"` concatenates the two strings into one string `"CS"`

```
1 y = "y"
2 h = y
3 def y(y):
4     h = "h"
5     if y == h:
6         return y + "i"
7     y = lambda y: y(h)
8     return lambda h: y(h)
9 y = y(y)(y)
```

Solution:



Video walkthrough