Lab 2 Notes: Higher order Functions

① Higher order Functions & currying

- Functions can return functions!

```
def outer():
    def inner():
        return "something"
    return inner
```

```
>>> outer()
Function
```

>>> $\underbrace{\texttt{outer()}}_{\text{inner}}\texttt{()}$

"something"

- currying: use higher order functions to convert function that takes multiple args into a chain of functions that each take a single arg

```
def curried_add(x):
    def helper(y):
        return x+y
    return helper
```

>>> $\underbrace{\underbrace{\texttt{curried\_add(2)}}_{f1}\texttt{(5)}}_{f2}$

7

- Env diagram:

<u>Global</u>

curried_add → func curried_add(x) [P=G]

<u>f1</u>: curried_add [P=G]

x 2

helper → func helper(y) [P=f1]

RV → func helper(y) [P=f1]

<u>f2</u>: helper [P=f1]

y 5

RV 7

② Lambda Expressions

- Lambda expr evaluate to functions. They specify:

  1) Parameters

  2) Return expr

- Syntax: lambda <parameters> : <return expr>

  EX: lambda x : x

  A function that takes x and returns x

- Properties of lambda ~~functions~~ expr:
  - An <u>expression</u> that evaluates to a value. Ev<u>aluating</u> the lambda expr does <u>NOT</u> create/modify variables
  - Creates an anonymous lambda function with no intrinsic name
  - Note that we can assign a variable to be equal to a lambda expression

```
f = lambda x, y : (x + y) * 2
```

```
>>> f(3,5)
    16
```

```
>>> (lambda x,y: (x+y))(3,5)
    16
```

Env. Diagram Example:

```
                                        r.v. f4
                                       ___________
1   def compose (f, g):                  r.v. f3
2                                          ____
3       return lambda x:  f (g (x))
                     r.v. f1
          ________________________________
4   f = compose (lambda x: x*x,
5                lambda y: y+1)
6   result = f (12)
             ______
             r.v. f2
```

Global

compose L → func compose (f, g) [P = G]

f L f1

result L169 f2

f1: compose [P=G]

f L → func λ(x) <line 4> [P=G]

g L → func λ(y) <line 5> [P=G]

RV L → func λ(x) <line 2> [P=f1]

f2: λ <line 2> [P=f1]

x L12

RV L169 f(r.v. f3) → f(13) → r.v. f4

f3: λ <line 5> [P= G]

y L12

RV L13

f4: λ < line 4> [P= G]

x L13

RV L169