Welcome to 6.101!

Do Now:

- **Grab a handout at the front! → 2 sheets**
- Taking notes via a tablet / pen+paper highly encouraged
- **Digital handout:**

Announcements:

- Lisp part 1 lab due Friday 5/3 at 5pm
  - Highly recommended to start early, this is a two part lab that requires all functionality from part 1!
  - Lab hours are less busy early in the week!
- Final Exam: Tuesday, 5/21 at 1:30 in Johnson Track

Resources:

- py.mit.edu (course website), 6.101-help@mit.edu, 6.101-personal@mit.edu
- My office hours: Wed. 12-1 pm in 56-191
- If you want to meet outside of office hours, please reach out!

Today's Agenda

- Why LISP?
- Tokenize / Parse / Evaluate overview
- Environment diagrams and scoping
  - global + nonlocal keywords
- Interpreters, Scheme, and evaluate

~ be brave! ask and answer questions + tell me your name!~

Why LISP?

Why write interpreters?

- (Hopefully) it's cool/fun!
- It can help you understand languages you already know
- Profound idea: the interpreter is just another program

Why LISP?

- "A language that doesn't affect the way you think about programming, is not worth knowing"
- Similar semantics to Python in some ways (help us understand Python)
- MIT and LISP have a long history
- Minimal syntax: less time on tokenizing/parsing, more on evaluation

What is an interpreter?

“a program which converts the high-level language to machine code and then executes it on the go”

Related: What is a compiler?
“A compiler is, more generally, a program that converts a program in one programming language into a program in another programming language.”
[source: StackOverflow]

Many modern programming language implementations use some combination of compilation + interpretation. cPython (Python’s reference implementation) compiles Python into bytecode, and then interprets it.
[source: Wikipedia]

PROGRAM →Tokenize → Parse → Evaluate →OUTPUT 

“(- (+ 3 2) 5) ; test comment” →

tokenize → ['(', '-', '(', '+', '3', '2', ')', '5', ')']

parse → ['-', ['+', 3, 2], 5]

evaluate → 0

Online environment diagram tool*

*this was created by my friend for his Master’s thesis, it hasn’t been updated in a while and may be a bit buggy

Question 1

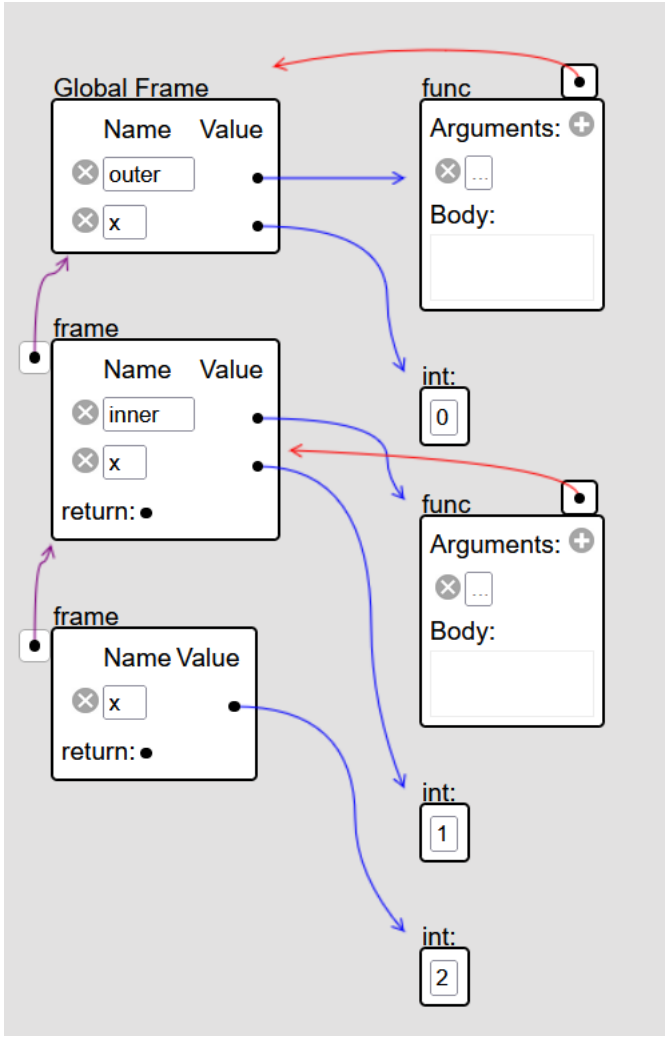
```
1       x = 0
2       def outer():
3           x = 1
4           def inner():
5               x = 2
6               print("inner:", x)
7
8           inner()
9           print("outer:", x)
10
11      outer()
12      print("global:", x)
```

Output:

Inner 2

Outer 1

Global 0



More environment diagram examples from a previous semester:

Question 2a


```
1       x = 0
2       def outer():
3           x = 1
4           def inner():
5               print("inner1:", x)
6               x = 2
7               print("inner2:", x)
8
9           inner()
10          print("outer:", x)
11
12      outer()
13      print("global:", x)
```

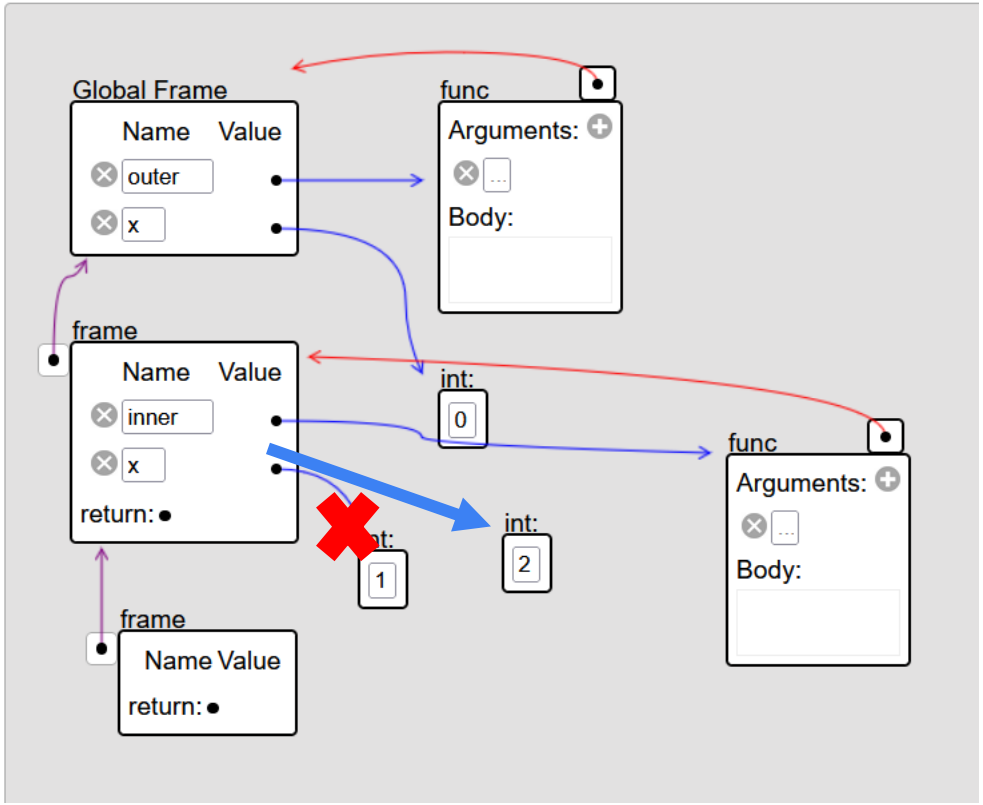
Output:

UnboundLocalError: cannot access local variable 'x' where it is not associated with a value

Question 2b

```
1         x = 0
2         def outer():
3             x = 1
4             def inner():
5                 nonlocal x
6                 x = 2
7                 print("inner:", x)
8
9             inner()
10            print("outer:", x)
11
12        outer()
13        print("global:", x)
```

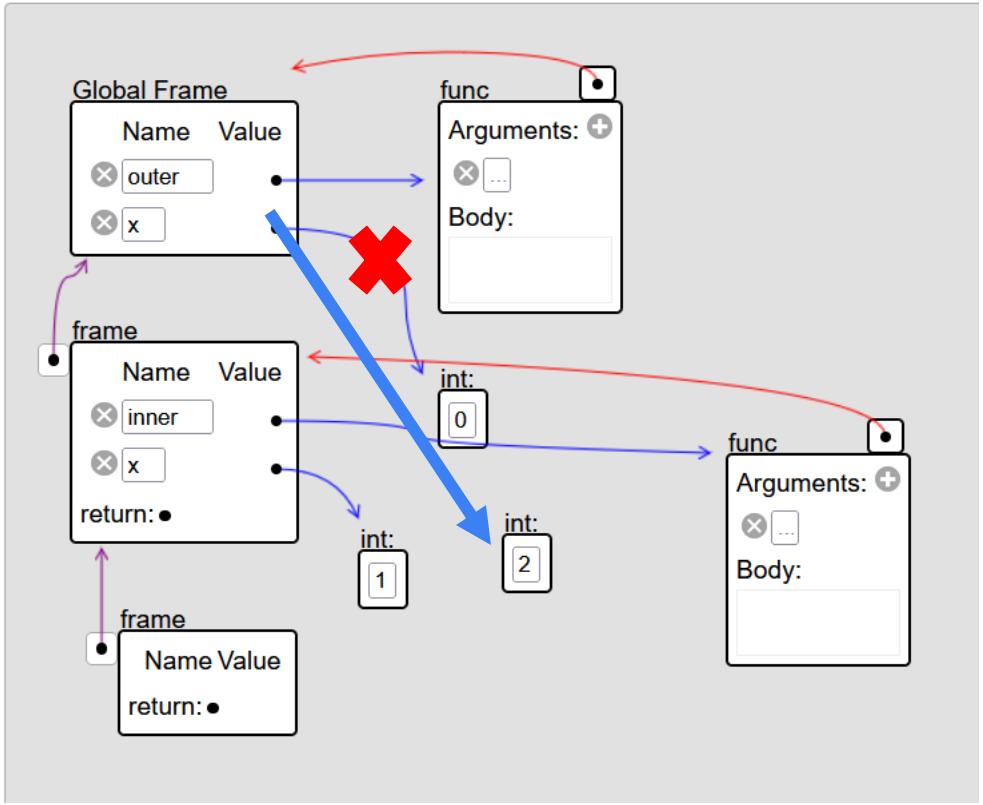
Output:
Inner 2
Outer 2
Global 0



Question 2c

```
1         x = 0
2         def outer():
3             x = 1
4             def inner():
5                 global x
6                 x = 2
7                 print("inner:", x)
8
9             inner()
10            print("outer:", x)
11
12        outer()
13        print("global:", x)
```

Output:
Inner 2
Outer 1
Global 2



Extra Practice with environment diagrams

- What does the program below output? How would you represent it using an environment diagram?

```
class A():
    x = 5

class B():
    other = A
    x = 3

A.other = B
print(B.x, B.other.x, B.other.other.other.other.x)
print(A.x, A.other.x, A.other.other.other.other.x)
```

Extra Practice with environment diagrams

- What does the program below output? How would you represent it using an environment diagram?

Answer:

3, 5, 3

5, 3, 5

For more info on circular class definitions see:
https://stackoverflow.com/questions/23026530/circular-dependency-between-python-classes
Or
https://www.tutorialspoint.com/How-do-we-handle-circular-dependency-between-Python-classes