Welcome to CS61A Lab 6 Sect. 29/47 :D

Dickson Tsai
dickson.tsai+cs61a@berkeley.edu
OH: Tu, Th 4-5pm 411 Soda

Previous: Rooted Trees, Dictionaries, Linked lists **>>>**

Today: OOP >>>

Next stop: Generic functions, linked lists, memoization

Anuncios

- HW5 due on Wednesday 10/15 @ 11:59PM
- Project 3 due 11:59 PM next Thursday 10/23
  - Master OOP on simple examples before attempting
  - Try things that don't work as well. Familiarize yourself with Python's OOP-related error messages beforehand
- Midterm 2 on Monday 10/27 7-9 PM
  - Lots of material! If you feel like you have fallen behind, please set up an appointment at dicksontsai.com/meet, and I will try my best to help you.

OOP Meets Env Diagram

1. Rules when seeing class statement:
   1. Create a new class object.
   2. Run through the code in the body, with all saved names going to the class object instead of the current frame
      1. E.g. the assignment **a = 2** will save in the class object as a class attribute, not in the current frame.
   3. Function parents are still the current frame
2. Rules for calling class constructor:
   1. Create an empty instance object
   2. Run the class object's __init__ with self bound to the empty object and the other parameters bound to the other operands
   3. __init__ does not return, but the constructor call expression evaluates to the instance object

Tip of the Day

- Tired of using print statements? Use a debugger!
  - In Python, there is a standard library module pdb
  - You can also use a debugger to help you understand new code (along with Pythontutor)!
- Demo:
  - Download dicksontsai.com/cs61a/discussion/pdb_example.py
  - Start up the debugging with `python3 –m pdb pdb_example.py`
    - Or you can have `import pdb; pdb.set_trace()` somewhere in your code. Once that code is executed, you will start a debugging session.
  - Common commands are on the next slide

Tip of the Day

- Tired of using print statements? Use a debugger!
  - Some common pdb commands: (you can use the one letter)
  - For more info: https://docs.python.org/3.4/library/pdb.html

1. `h(elp) [command]`
   - You can use this to look at all available commands. If you input a command, will tell you how to use it
2. `b(reak) [(linenumber | function) [,condition]]`
   - Set a breakpoint. Then when the program is running, it will stop execution as soon as it hits the breakpoint
3. `s(tep), n(ext)`
   - Step into a function call or next line. Next just goes to the next line
4. `c(ontinue)`
   - Continue execution of the program (stop when you hit breakpoint or finish execution)

Optional Conceptual Questions (Friday 10/10 lecture)

1. True/False: Classes cannot take attributes
2. True/False: An instance attribute whose value is a function is a method for that object.
3. Suppose `A.foo = lambda x: x * x` for a class A. Suppose `a` is an instance object for `A`. Is the bound method object `a.foo` the same as the function object `A.foo`?
4. What does inheritance provide us? Give a real life example of inheritance. (e.g. what classes inherit from Verb?)
5. When designing for inheritance, why should we always try to look up attributes from the instance?
6. How are is-a relationships represented? Has-a relationships? What do these relationships mean?

Optional Conceptual Questions (Monday 10/13 lecture)

1. What are the two string representations that all objects produce? Why do we make a distinction between them?
2. Which function's result is the same as what comes out of the interactive session? Out of the print function?
3. What's a polymorphic function?
4. What's an interface?
5. What's the purpose of a property method? Compare using @property with using just an instance attribute.
6. Summarize at a high level how we can support a Complex class with addition using rectangular coordinates and multiplication using polar coordinates through interfaces. How was the complex number interface specified?