Coming Up

- Wednesday: Intro to classes
- Thursday @ 10: A2 Due
- Friday: Writing classes
- Sunday @ 10: weekly exercise due
- Monday: Special class methods
- Wednesday: Inheritance

Using objects

- You have used many kinds of objects that are built in to Python: int, str, list, dict, etc.

```
total = 0            # Create a new int object.

name = “Priya”     # Create a new str object.
```

- And others that are part of modules you had to import, such as author_functions.

```
import author_functions

cleaned = author_functions.clean_up(s)
```

Defining New Types

- Consider a bank. One of the basic pieces of data it must track is accounts.
- What should we keep track of for a single account?
- We could define individual variables for each of these, but we'd need to do that for every account. Messy.
- Or we could define a new type of object to bundle these together.
- We have int, str, dict, list, etc.
  Now we can have Account!

Important Terms

- class
- object (or instance)
- attribute
- method

- self
- constructor

Encapsulation

- How is a bank account implemented? How is a withdrawal done?
    - Some things should remain secret.
    - Sometimes, it helps to not know details
- In order to use a bank account, we don't need to know these internal details. We just need to know how to call the methods and functions that work with accounts.
- How do we find out what is available? dir and help.

- So the programmer who uses a bank account can be blissfully ignorant of the details.
- This is good! Why?
- It goes the other way too: The programmer who writes/updates/debugs/modifies/extends the bank account code doesn’t need to know how others are using it.
- This is good, too! Why?
- Encapsulation: keeping data, and the code that uses it, in one place and hiding the details.
- A simple but powerful idea. It helps us manage complicated code.

Python Conventions

- Some languages provide mechanisms for enforcing encapsulation.
    - You might be able to make an attribute "private", for example.

- Python does not. Instead, conventions are used.
    - An underscore denotes a private attribute or method.
    - Two underscores denotes a system attribute or method.

Review Questions

Underscore methods

- What do the underscores signify?

Constructors

- What is the name of every constructor?
- What does a constructor do?
- When does a constructor "happen"?
- What if you don't define your own?

More Review Questions

self

- What is it and what does it refer to?

__str__

- When is it called?
- What if you don’t define your own?

__repr__

- When is it called?
- What if you don’t define your own?
- How does it differ from __str__?