Objects and a bit of Libraries

Review

- Functions
- Common data structures
  - Lists
  - Tuples
  - Dicts
- Input/Output
- Pygame Events
- Questions?

Overview

- Dict examples (whoops!)
- Classes and Objects
  - Structs
  - Classes
  - Objects
- Libraries
  - Modules
  - Packages

Dict Examples (With Code!)

```
>>> a = {}
>>> a['test'] = 1
>>> a['b'] = 2
>>> a[18] = 'two'
>>> a
>>> a.items()
>>> a.keys()
>>> a.values()
>>> 'test' in a
>>> 16 in a
>>> b = {'1': 2, 'kitten': 'meow', (1, 3): -1}
```

Classes

- Classes are a way to store data and functions that act on that data
- At its most basic a class can act as a bucket filled with different variables
- At its most complex classes can inherit features from other classes in really weird ways (we're gonna skip this)
- We've never heard a good explanation so we'll go by example

Classes (By Example!)

```
class Paddle (object):
  def __init__(self, x, y, dy, color=(255, 0, 255)):
    self.x = x
    self.y = y
    self.dy = dy
    self.color = color
    self.width = 20
    self.height = 100
  def move(self, event):
    if event.key == K_UP:
      self.y -= self.dy
    elif event.key == K_DOWN:
      self.y += self.dy
  def draw(self, screen):
    pygame.draw.rect(screen, self.color, (self.x, self.y,
    self.width, self.height))
```

Classes (By Example!)

```
>>> paddle1 = Paddle(30, height/2-
50, 5)

>>> paddle2 = Paddle(width – 30,
height/2-50, 5, (255, 255, 0))

>>> paddle1.move(e)

>>> paddle2.move(e)
```

Libraries (and why you want them)

- Separating class definitions and functions from your code is good
- To create a module just make a python file with your code in it
- Import that file to have access to all functions and classes defined there
- Unless you use the import * thing your functions will be namespaced (you have to call them with the module name prepended)

  ```
  >>> import my_module
  >>> my_module.test()
  ```

  Or

  ```
  >>> from my_module import *
  >>> test()
  ```
- You've seen this already.

Libraries (continued!)

- Collections of modules can be packaged together in a package!
- You do this by throwing modules in an folder and making an __init__.py file in that folder
- Looks something like this:

- package_name
  - __init__.py
  - Module1.py
  - Module2.py
  - subpackage_name
    - __init__.py
    - Module3.py
    - Etc

- Import packages the same way you import modules
- Namespacing behaves similarly to modules just put the package name(s) prior to the module name

Libraries (One Last Time!)

```
>>> import package_name
>>> package_name.module1.test()
>>> package_name.subpackage_name.module3.test()
```

Or

```
>>> from package_name import *
>>> module1.test()
>>> subpackage_name.module3.test()
```

Or

```
>>> from package_name import subpackage_name
>>> module3.test()
```

Fin

- Let's go over more pong demo code using classes
- Let's go over some sprites using classes
- Try and package your code into a module
- Try and make classes of your interactive animations
- Questions?