CIS 122

Coding with Class
Building Things Up

• How do we build up sequences?

• Lists are mutable
  ○ So we can append things to them
    >>> L = [1, 2, 3]
    >>> L.append(4)

• Strings are immutable
  ○ So we can't modify them
  ○ We can only reassign them
    >>> s = "abc"
    >>> s = s + "d"  # alternatively, s += "d"
Personalized Objects

● We've seen a lot of types of objects...
  ○ Integers
  ○ Floats
  ○ Strings
  ○ Booleans
  ○ Lists
  ○ Dictionaries

● Different objects are good for different purposes
  ○ Integers - performing calculations
  ○ Booleans - conditional code
  ○ Lists - grouping things together
Personalized Objects

- Python objects are general purpose

- But what if we're performing some specific task?
  - It might be nice to have more specialized objects

- If we're working with coordinate systems...
  - It might be nice to have a Point object

- If we're writing music...
  - It might be nice to have a Note object

- If we're studying genetics...
  - It might be nice to have a Chromosome object
Personalized Objects

- Python can't include all these objects
  - There are far too many

- Fortunately, it lets you define your own objects
  - Classes
  - Custom objects for specific tasks

- Classes are collections of attributes and methods
  - Attributes - What does my object store?
  - Methods - What can my object do?
The turtle module defines a Turtle class
○ Allows you to make individual Turtle objects

```python
import turtle

t1 = turtle.Turtle()

t2 = turtle.Turtle()

t1.forward(10)

t2.backward(10)
```
Turtle Aside

- Turtle attributes
  - x coordinate
  - y coordinate
  - heading

- Turtle methods
  - forward
  - backward
  - left
  - right
  - ...
Making a Point

- Suppose we wanted a Point class
- What attributes would we want to store?
- What would we like to be able to do with points?
Making a Point

● Suppose we wanted a Point class

● What attributes would we want to store?
  ○ x coordinate
  ○ y coordinate

● What would we like to be able to do with points?
  ○ find distance to origin
  ○ find distance between points
  ○ add points
Making a Point

- Where do we start?
- Need to define our Point class

```python
class Point:
    # <Point code goes here>
```
Making a Point

- Now what?

- Need a method for constructing new Points
  - A "constructor"

- __init__ method
  - __init__
  - (special methods are surrounded by underscores)

- The first argument to __init__ is special
  - It refers to the object being created
  - Customary to call it self
Making a Point

class Point:

    def __init__(self):
        """Point constructor""
        self.xcor = 0  # Set point's x coordinate to 0
        self.ycor = 0  # Set point's y coordinate to 0
Making a Point

● We can now construct new Points
  ○ \( p = \text{Point()} \)

● Our constructor doesn't take any arguments right now
  ○ self doesn't count

● So right now, all Points default to \((0, 0)\)

● What if we wanted to be able to construct a point with specific coordinates?
  ○ Add some more arguments to our constructor
  ○ Any arguments after the first act normally
Making a Point

class Point:

def __init__(self):
    """Point constructor"""
    self.xcor = 0  # Set point's x coordinate to 0
    self.ycor = 0  # Set point's y coordinate to 0
class Point:

    def __init__(self, x, y):
        """Point constructor"""
        self.xcor = x  # Set point's x coordinate
        self.ycor = y  # Set point's y coordinate
You've Made Your Point

● We can now construct Points with arguments
  ○ p = Point(1,2)

● We can see those arguments if we ask for them
  ○ p.xcor
  ○ p.ycor

● But what if we try to print p itself?
  ○ Python doesn't tell us anything useful right now
  ○ But we can fix that
The \_\_repr\_\_ method tells Python how to print an object
  ○ Short for representation

The first argument to repr refers to the object being printed
  ○ Same for all class methods

The \_\_repr\_\_ method doesn't print anything
  ○ It returns a string

When python wants to print an object
  ○ It calls the object's \_\_repr\_\_ method
  ○ And prints the string it returns
class Point:

    def __init__(self, x, y):
        """Point constructor""
        self.xcor = x  # Set point's x coordinate
        self.ycor = y  # Set point's y coordinate

    def __repr__(self):
        """Return string representation of Point"""
Making a Point

class Point:

    def __init__(self, x, y):
        """Point constructor""
        self.xcor = x  # Set point's x coordinate
        self.ycor = y  # Set point's y coordinate

    def __repr__(self):
        """Return string representation of Point""
        return "(" + str(self.xcor) + ", " + str(self.ycor) + ")"
Special Class Methods

- `__init__`
  - Constructor
  - Produces new objects

- `__repr__`
  - Print method
  - Returns a string for displaying object

- `__cmp__`
  - Comparison method
  - Defines comparisons between objects

- Many others...