

Accelerating Information Technology

http://aiti.mit.edu

Ghana Summer 2012 Lecture 06 – Classes and Objects





The History of Objects

- Objects weren't always supported by programming languages
- Idea first originated at MIT in the 1960s and was officially incorporated in a few languages in the same decade
- OOP (Object Oriented Programming)
 has now become a core feature of
 nearly all languages

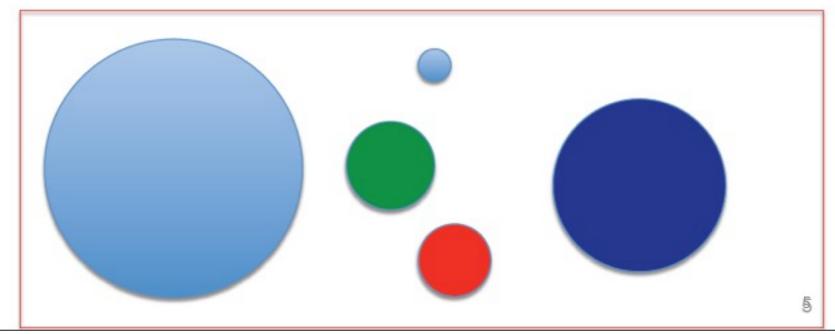
Agenda

- Overview of Objects
 - attributes
 - methods
- Special methods

Introduction to Classes and Objects

Objects and Classes

- An object is an instance of a class!
- Classes have:
 - attributes: data that can be accessed
 - methods: functions associated with object instances



Objects: Shapes Example

Class name

```
class Triangle:
```

```
def __init__(self, base_val, height_val):
    self.base = base
    self.height = height
```

Constructor takes two parameters and sets the attributes self.base and self.height

```
def area(self, f):
    area_val = self.base*self.height/f
    return area_val
```

```
blaTriangle = Triangle(5, 3)
blaTriangle.area(2)
OR
Triangle.area(blaTriangle)
```

area function calculates the area of the triangle from the attributes self.base and self.height and returns the result

Objects: Shapes Example

```
class Triangle:
    def __init__(self, base_val, height_val):
        self.base = base
        self.height = height
        self.areaa = self.area()
    def area(self):
        area_val = self.base*self.height/2
        return area_val
How could we write similar class for rectangles?
```

```
class Rectangle:
    def __init__(self, ...): #fill in parameters
        #your code here
    def area(self):
        # your code here
```

Objects: Shapes Example

```
class Triangle:
   def __init__(self, base_val, height_val):
     self.base = base
     self.height = height
   def area(self):
     area val = self.base*self.height/2
     return area val
Very similar structure: the area function is slightly different
```

```
class Rectangle:
  def init (self, base val,
  height val ):
     self.base = base val
     self.height = height val
  def area(self):
     area val = self.base*self.height
     return area val
```

Objects: Practice

- Should area be an attribute as well as a function?
- Without adding attributes to Rectangle, how would you write a perimeter function for the rectangle?
- Is it possible to calculate the perimeter of a triangle using only self.base and self.height?
 - If yes, explain how to do this.
- If no, explain what attribute you might add to the
 Triangle class to make this possible.

Objects: Practice Solutions

- Should area be an attribute as well as a function?
- Probably not; if it is, be careful with how mutator functions are designed. Consider what happens if we change the value of self.base but do not recalculate the area and update self.area

Objects: Practice Solutions

 How would you write a perimeter function for the rectangle?

```
#calculates the perimeter of a Rectangle
def perimeter(self):
    return 2*self.base + 2*self.height
```

Objects: Practice Solutions

Is it possible to calculate the perimeter of a triangle using only self.base and self.height?

No, the perimeter of a triangle would require one to know each side length. If we specify an angle value between the base and one of the other sides, we could use trigonometry to calculate all the side lengths

Objects vs. Classes

Class is like a factory or a blueprint for producing objects

Objects vs. Classes

This is a Class

```
class Rectangle:
    def __init__(self, base_val, height_val):
        self.base = base_val
        self.height = height_val
    def area(self):
        area_val = self.base*self.height
        return area_val
```

We use classes to create/instantiate objects like this:

myRectangle = Rectangle(5, 3)

myRectangle is an object of class Rectangle

print myRectangle.area()

>>15

Static Variables and Methods

 Static (class) variables can be used to describe attributes that should have one value across the entire class

```
class Car():
    wheels = 4
    def __init__(self, color):
        self.color = color

All Car instances will have four wheels
```

• What static variables can you think of for Triangle and Rectangle?

Special Methods

- "Special methods" help your object interface with built-in operators and data structures
- Example: __str__(self) defines the string representation of an object

 How could we make a string representation of the Car object?

Special Methods

Originally...

```
class Car():
    wheels = 4
    def __init__(self, color, horsepower):
        self.color = color
        self.horsepower = horsepower

>>>Louis_car = Car('red', 400)
>>>print Louis_car
```

< main .Car instance at 0xcea5a8>

Special Methods

Adding a __str__ method...

red car with 400 horsepower

```
class Car():
    wheels = 4
    def init (self, color, horsepower):
        self.color = color
        self.horsepower = horsepower
    def str (self):
        description = self.color, 'car with',
str(self.horsepower), 'horsepower'
        return description
  >>>Louis car = Car('red', 400)
  >>>print Louis car
```

Group Work: Creating the FootballTeam class

- Work in groups (three to four students per group) to design a FootballTeam class
- Want to describe a football Team in one data structure:
 - Name
 - Home City
 - List of Players
 - Points
 - League
- Associated functions: what does a Team do?
 - Teams can play games
 - Teams can acquire/release players

Group Work

 You will continue to practice these concepts in lab!

- aitighana2012.herokuapp.com
- /labs/python-objects