Objects and Classes
Imperative vs. Object-Oriented Programming

Imperative Programming (50s - 70s)
• uses statements to change a program's state
• a variable contains a value, an assignment statement changes it
• a print statement sends a value to output
• a "go to" transfers control to another statement
• "verbs" are the most important thing
• Functions / subprograms used for reuse and modularization
• Examples: FORTRAN, COBOL, Algol, Basic

Object-Oriented Programming (80s - present)
• Imperative in style, structured, but features added to support "objects"
• Ex: Smalltalk, Simula, C++, Python, Visual Basic, Java, Ruby
Imperative vs. Object-Oriented Programming (cont.)

In Imperative Programming, you:

- break down a task into variables, data structures, and subprograms
- you use subprograms to operate on data structures

In Object-Oriented Programming, you:

- define *objects* that expose behavior (*methods*) and data (*attributes* or *properties*) using well-defined interfaces
- bundle everything together, so that an object only operates on its own attributes using methods
Defining a Class

class Rectangle():
    # create a new Rectangle object
    def __init__(self,length,width):
        self.length=length
        self.width=width

    def getPerimeter(self):
        return 2*self.length + 2*self.width

    def getArea(self):
        return self.length*self.width

    def setLength(self,length):
        self.length = length

    def setWidth(self,width):
        self.width = width

Start with the keyword “class”, followed by the name you want to give the class, and a colon.
Defining a Class (cont.)

class Rectangle:

    # create a new Rectangle object
    def __init__(self,length,width):
        self.length=length
        self.width=width

    def getPerimeter(self):
        return 2*self.length + 2*self.width

    def getArea(self):
        return self.length*self.width

    def setLength(self,length):
        self.length = length

    def setWidth(self,width):
        self.width = width

"initializer": describes how to create a new instance of this class.

The initializer is always named __init__, with two underscores in front and in back.
class Rectangle:

    # create a new Rectangle object
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def getPerimeter(self):
        return 2*self.length + 2*self.width

    def getArea(self):
        return self.length * self.width

    def setLength(self, length):
        self.length = length

    def setWidth(self, width):
        self.width = width

Other methods:
avtions that you can perform using an instance of this class