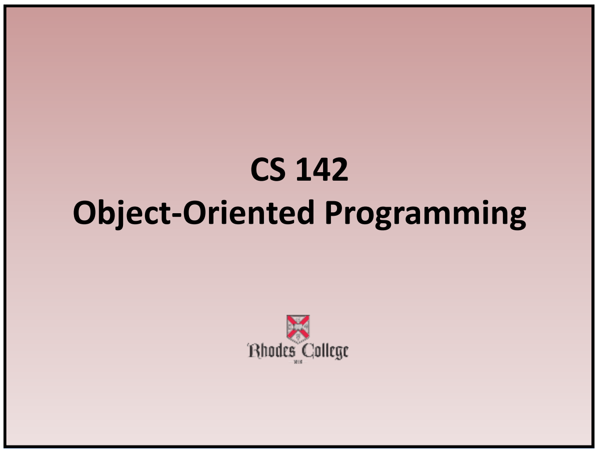
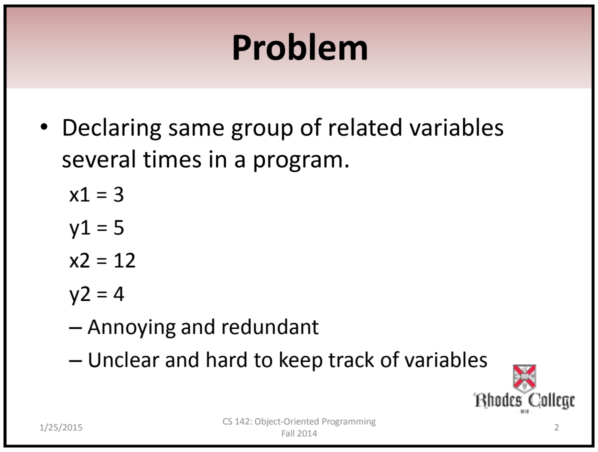


CS 142 Object-Oriented Programming



Problem

- Declaring same group of related variables several times in a program.

  x1 = 3

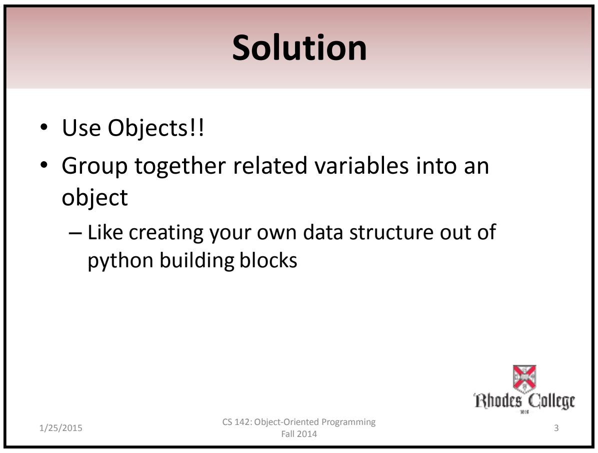
  y1 = 5

  x2 = 12

  y2 = 4

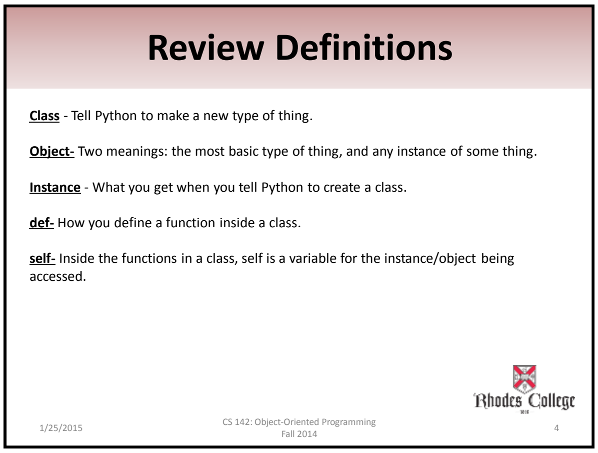
  – Annoying and redundant

  – Unclear and hard to keep track of variables



Solution

- Use Objects!!
- Group together related variables into an object
  - Like creating your own data structure out of python building blocks



Review Definitions

Class - Tell Python to make a new type of thing.

Object- Two meanings: the most basic type of thing, and any instance of some thing.

Instance - What you get when you tell Python to create a class.

def- How you define a function inside a class.

self- Inside the functions in a class, self is a variable for the instance/object being accessed.

New Definition

<u>Operator overloading</u> - A class can implement certain operations that are invoked by special syntax by defining methods with special names. This allows classes to define their own behavior with respect to language operators. (__str__, __add__, __mul__,….)

Why overload an operator?
- Special behavior appropriate for the class you are writing

Operator methods that can be overloaded in Python

Method	Returns
__add__(self, other)	self + other
__sub__(self, other)	self – other
__mul__(self, other)	self * other
__div__(self, other)	self / other
__neg__(self)	-self
__lt__(self, other)	self < other
__le__(self, other)	self <= other
__gt__(self, other)	self > other
__ge__(self, other)	self >= other
__eq__(self, other)	self == other
__ne__(self, other)	self != other

* Page 62 of your textbook

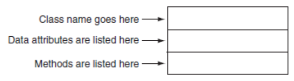
Point ADT

- Instance variables
  - x, y (should be private)
- Operators
  - Create new Points
  - Print out point nicely
  - Add 2 points together
  - See if 2 points are equal
  - Distance between 2 points
  - Distance from origin
  - In first quadrant?

Techniques for Designing Classes

- <u>UML diagram</u>: standard diagrams for graphically depicting object-oriented systems
  - Stands for Unified Modeling Language
- General layout: box divided into three sections:
  - Top section: name of the class
  - Middle section: list of data attributes
  - Bottom section: list of class methods

Figure 11-10 General layout of a UML diagram for a class



Finding the Classes in a Problem

- **When developing object oriented program, first goal is to identify classes**
  - Typically involves identifying the real-world objects that are in the problem
  - Technique for identifying classes:
    1. Get written description of the problem domain
    2. Identify all nouns in the description, each of which is a potential class
    3. Refine the list to include only classes that are relevant to the problem

Identifying a Class's Responsibilities

- A classes responsibilities are:
  - The things the class is responsible for knowing
    - Identifying these helps identify the class's data attributes
  - The actions the class is responsible for doing
    - Identifying these helps identify the class's methods
- To find out a class's responsibilities look at the problem domain
  - Deduce required information and actions

Your Turn

Rational class handout