DEFINING CLASSES IN PYTHON

CSSE 120—Rose Hulman Institute of Technology

Final Exam Facts

- Date: Monday, May 24, 2010
- Time: 6 p.m. to 10 p.m.
- Venue: O167 or O169 (your choice)
- Organization: Paper part and computer part, similar to other exams
  - **The paper part will emphasize both C and Python.**
    - You may bring two double-sided sheets of paper this time.
    - There will be a portion in which we will ask you to **compare and contrast C and Python** language features and properties.
      - Solve same problem in both languages
      - Explain similarities and differences
  - **The computer part will be in C.**
    - The computer part will be worth approximately 65% of the total.
  - Details for both parts to be discussed later today (and placed in **Session 30 Resources** on the course **Schedule** page)

Review: ***Static*** Memory Allocation

- What memory is allocated by the example below?
  - Answer: memory for:
    - An int (called *x*)
    - A char* (called *p*)
    - A double (called *y*)
    - 10 char's (called *string*)

```
void foo(int x, char* p) {
    double y;
    char string[10];
    ...
}
```

- When is that memory allocated?
  - Answer: Every time the function is called
- What is that memory initialized to?
  - Answer: *x* and *p* are <u>*copies*</u> of the actual arguments passed to the function. (Note that *p* is a copy of a pointer, so has the same pointee as its actual argument.) *y* and *string* are uninitialized (i.e. garbage).
- When is that memory returned to the system?
  - Answer: Every time the function returns to whatever called it
- What happens to that memory after it is returned to the system?
  - Answer: Anything! You ***cannot*** count on it remaining unchanged. **Q3-6**
- This is called ***static allocation***. The memory is allocated from the ***stack***.

Review: *Dynamic* Memory allocation

- Suppose we want to reserve space for 10 doubles.
- We would do:

```
double* samples;

samples = (double*) malloc(10 * sizeof(double));
```

- The memory returned to you can store objects of any type (*void* pointer). We give it the desired type by *typecasting*.
  - That's the `(double*)`
- Use the allocated memory using the usual array notation (if more than one place is allocated), e.g.

```
for (k = 0; k < 10; ++k) {
    samples[k] = …
}
```

Review: Using Objects in Python

```
WIDTH = 400
HEIGHT = 50
REPEAT_COUNT = 20
PAUSE_LENGTH = 0.25

win = GraphWin('Saints Win!', WIDTH, HEIGHT)
p = Point(WIDTH/2, HEIGHT/2)
t = Text(p, 'Saints—2010 Super Bowl Champs!')
t.setStyle('bold')
t.draw(win)
t.setFill('blue')

nextColorIsRed = True
for i in range(REPEAT_COUNT):
    sleep(PAUSE_LENGTH)
    if nextColorIsRed:
        t.setFill('red')
    else:
        t.setFill('blue')
    nextColorIsRed = not nextColorIsRed
win.close()
```

Constructing objects: a **GraphWin**, a **Point**, and a **Text**

Doing things with the ***t*** object that is a **Text**

Review: What is an Object?

- An Object:
  - knows things about itself
    - fields
      - a.k.a. instance variables
  - can be asked to (based on what it knows)
    - do things
      - mutator methods
    - provide info about itself and/or other objects that it knows about
      - accessor methods
- Is an instance of a C structure an Object?

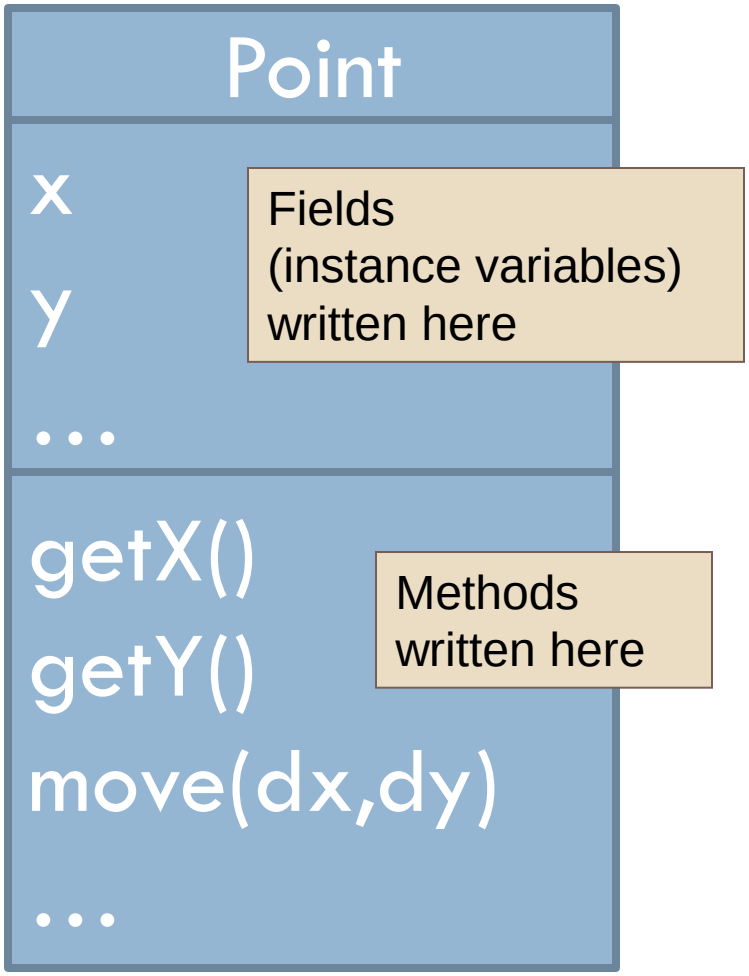
Q11-12

Review: Object Terminology

- Objects are *data types* that might be considered ***active***
  - They **store information** in *fields (aka instance variables)*
  - They **manipulate their data** through *methods*
    - Same concept as *functions*, but OO
- Each object is an *instance* of some *class*
- Objects are created by calling *constructors*

Q13-16

- UML class diagram:

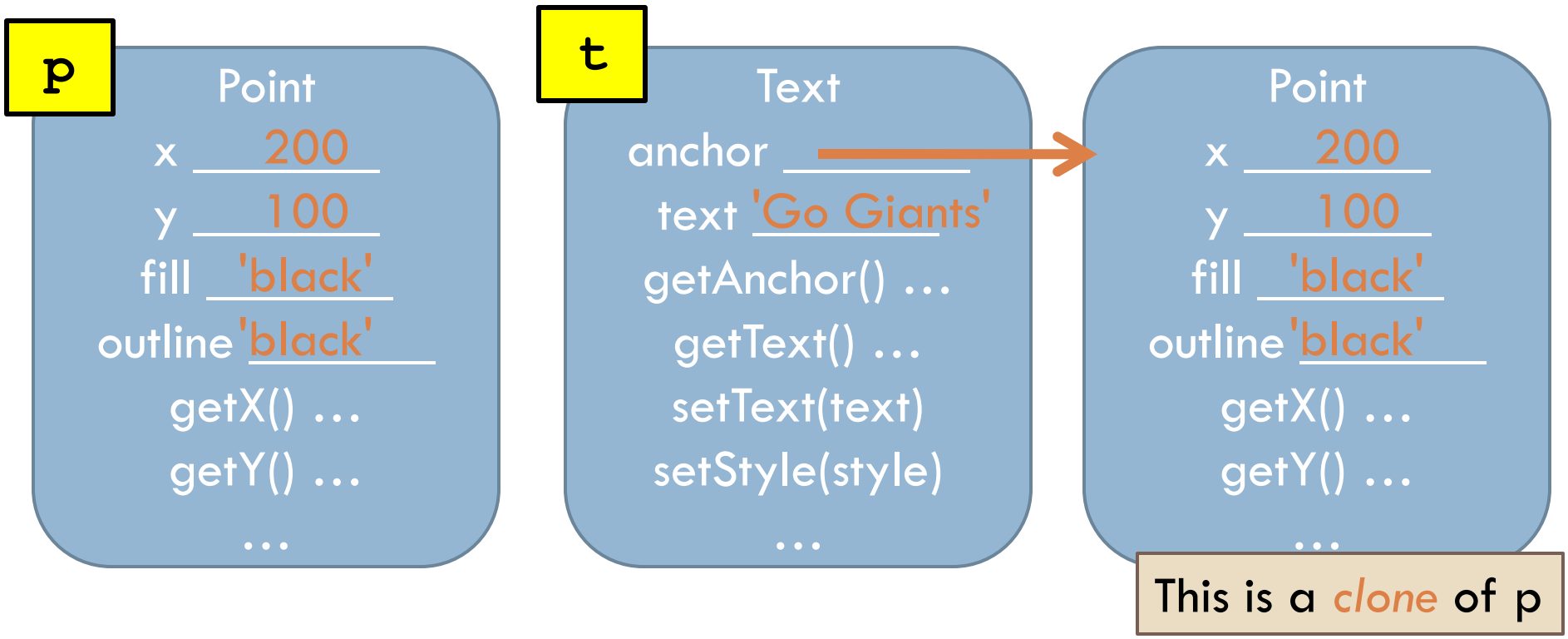


Key Concept!

- A class is an "object factory"
  - Calling the constructor tells the classes to make a new object
  - Parameters to constructor are like "factory options", used to set instance variables
- Or think of class like a "rubber stamp"
  - Calling the constructor stamps out a new object shaped like the class
  - Parameters to constructor "fill in the blanks". That is, they are used to initialize instance variables.

Example

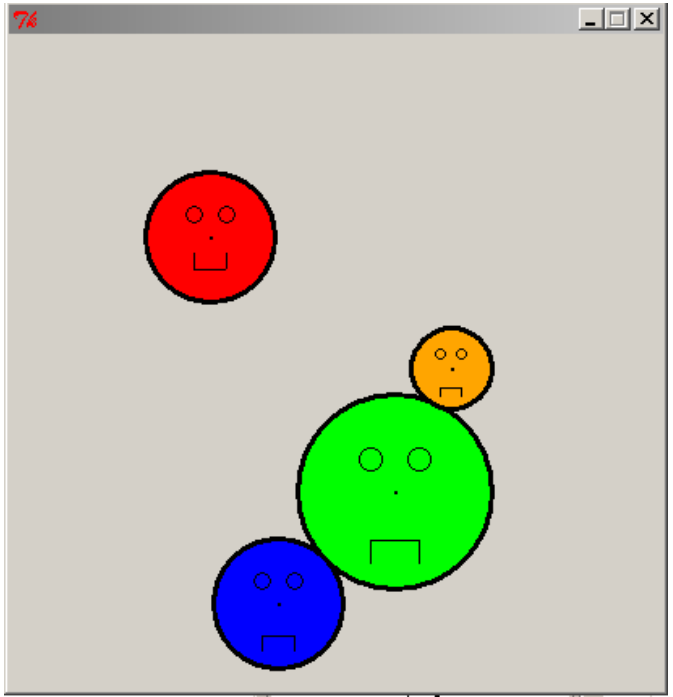
- `p = Point(200, 100)`
- `t = Text(p, 'Go Giants!')`



Creating Custom Objects: Defining Your Own Classes

- Custom objects:
  - Hide complexity
  - Provide another way to break problems into pieces
  - Make it easier to pass information around
- Example:

  Moving "Smiley" class.
  - Switch workspace to your Python workspace
  - Checkout the `30-ClassesSmileys` project from SVN



Summary: Defining classes

```
class Smiley
    def __init__ (self, initX, initY, dx, dy, size=40...):
        self.dx = dx
        self.dy = dy
          ...
        self.moving = True
          ...
        self.head = Circle(Point(initX, initY), size)
          ...
        self.parts = [self.head, self.leftEye, self.rightEye,
                      self.smileBase, self.smileLeft,
                      self.smileRight, self.centerPoint]

    def draw(self, win):
        for part in self.parts:
            part.draw(win);

      ...
```

Review of Key Ideas

- ***Constructor***:
  - Defined with special name `__init__`
  - Called like `ClassName()`
- ***Fields*** *(aka instance variables)*:
  - Created when we assign to them, using
    `self.blah = ...`
  - Live as long as the object lives
  - Can be referenced anywhere in the class definition
- `self` formal parameter:
  - Implicitly get the value *before the dot* in the call
  - Allows an object to "talk about itself" in a method

- Rest of class:
  - Do survey on Angel, our course, under **Lessons**: ***End of course survey for CSSE 120, Robotics Section***
  - Do **anonymous course evaluation** on **Banner Web**
  - From the Schedule page, **Session 30**, download **Final Exam Topics and Sample Problems**
    - Look it over and ask questions
    - Work problems from it
- I will be in my office or CSSE lab F-217:
  - Friday: 9:30 a.m. to 5 p.m.
  - Saturday: 10 a.m. to noon and 4 p.m. to 6 p.m.
  - Sunday: 12:30 to 4:30 p.m.
  - Monday: 9:30 a.m. to 5 p.m.

- **Best way to prepare for the final exam:**
  1. Prepare a good cheat sheet for the written problems, based on the **Final Exam Topics and Sample Problems.**
  2. Do sample problems from that document that you are unsure about.
     - Do them in the CSSE lab F-217, where you can get help from me or a student.
  3. Review/do the C homeworks.
     - Have your examples ready for the exam