As you arrive:

1. Start up your computer and plug it in
2. ***Log into Angel*** and go to CSSE 120
3. Do the ***Attendance Widget*** – the PIN is on the board
4. Go to the course ***Schedule Page***
5. Open the ***Slides*** for today if you wish
6. Check out today's project: `Session17_MovingSmileys`

Plus lots of in-class time to work on team project.

Defining classes part 1

- Review objects & object terminology
- Defining your own classes
- Instantiating and using objects
- Object interaction

Project work:

Work in your team to complete next milestone

Checkout today's project: `Session17_MovingSmileys`

Troubles getting today's project? If so: →

Are you in the Pydev perspective? If not:

- `Window ~ Open Perspective ~ Other`
  then `Pydev`

Messed up views? If so:

- `Window ~ Reset Perspective`

No SVN repositories view (tab)? If it is not there:

- `Window ~ Show View ~ Other`
  then `SVN ~ SVN Repositories`

In your SVN repositories view (tab), expand your repository (*the top-level item) if not already expanded.*

- If no repository, perhaps you are in the wrong Workspace. Get help as needed.

Right-click on today's project, *then select* ***`Checkout`****. Press* ***OK*** *as needed.*

The project shows up in the
`Pydev Package Explorer`
to the right. Expand and browse the modules under
`src` as desired.

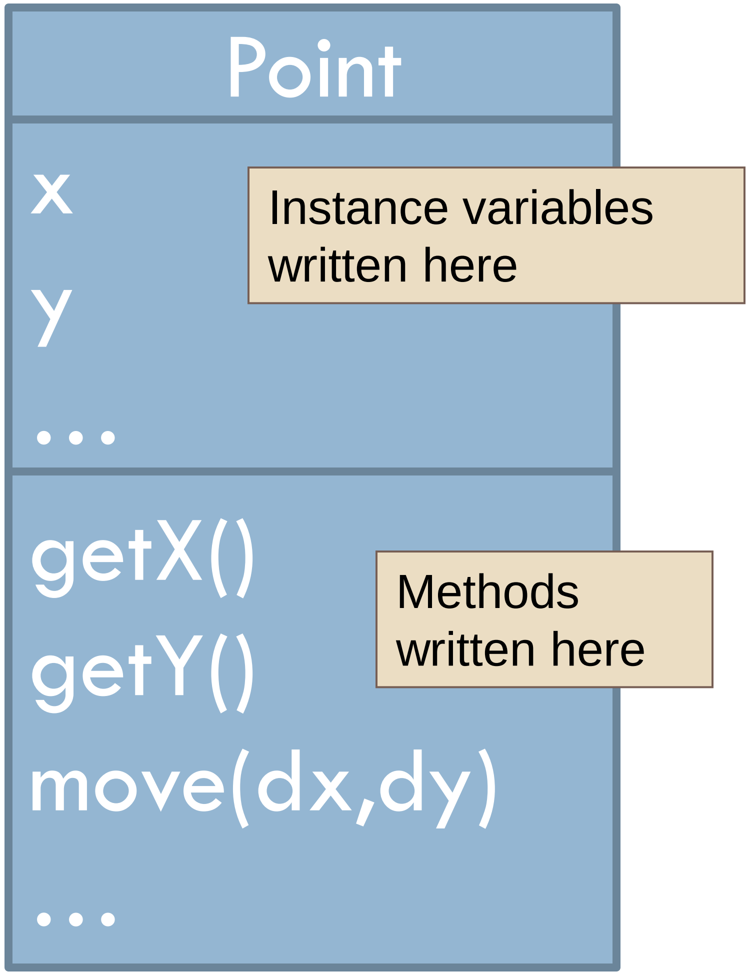
Review: What is an Object?

- An Object is an active data-type:
  - knows things about itself
    - fields
      - a.k.a. instance variables (or fields)
  - 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

Review: Object Terminology

- Objects are *data types* that might be considered ***active***
  - They **store information** in *instance variables*
  - They **manipulate their data** through *methods*
- Objects are *instances* of some *class*
- Objects are created by calling *constructors*

- **UML class diagram:**



Key Concept!

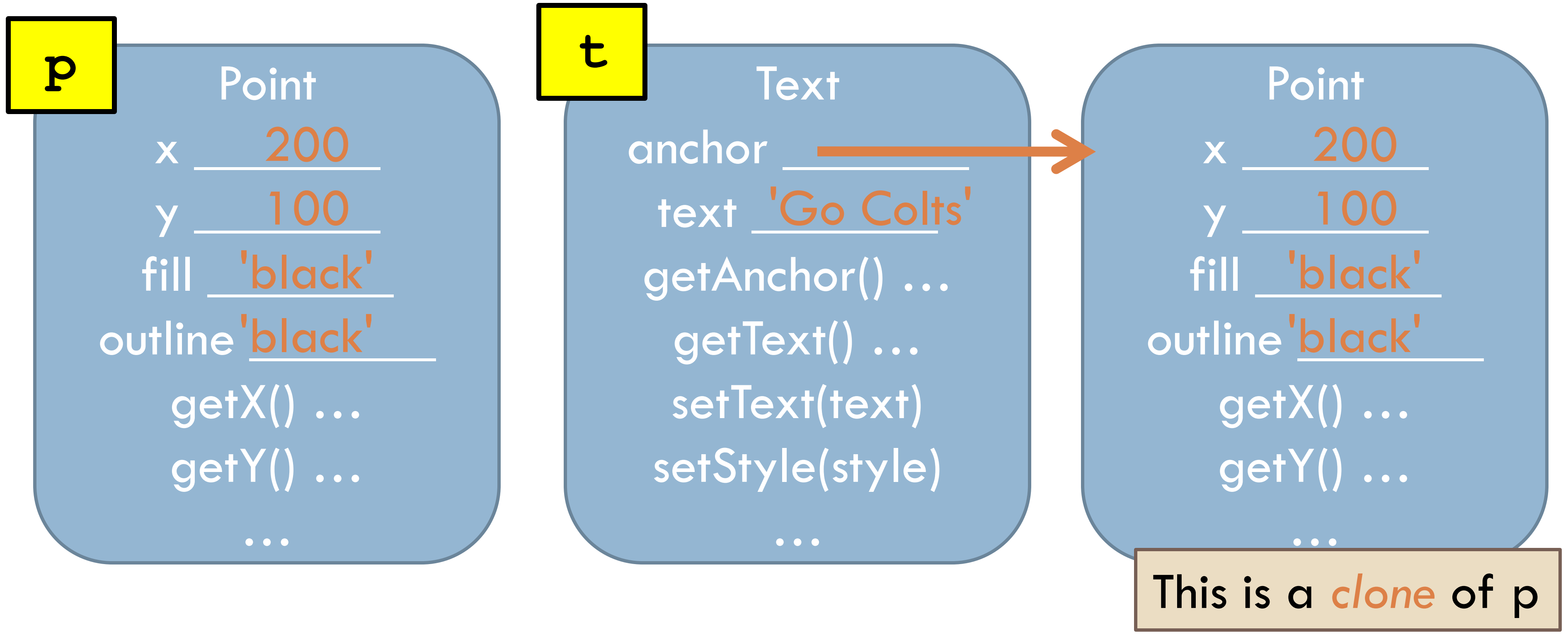
- A class is an "object factory"
  - Calling the constructor tells the classes to make a new object
  - Parameters to constructor are like "factory options", used to set instance variables
- Or think of class like a "rubber stamp"
  - Calling the constructor stamps out a new object shaped like the class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

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

Example

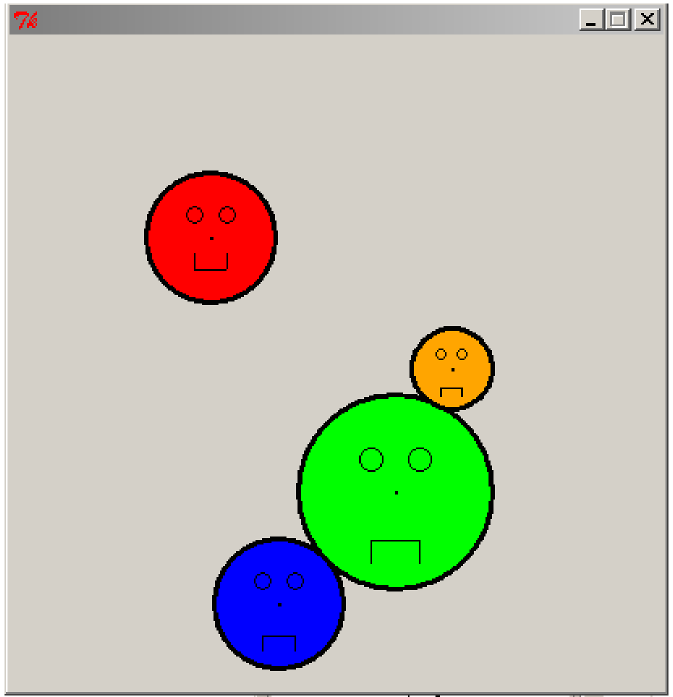
```
p = Point(200, 100)
t = Text(p, 'Go Colts!')
```



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.
- Let's create our own custom class and use it to instantiate objects.
- Use modules in project you checked out earlier



Coding MovingSmileys

- Create constructor noting default parameters
  - Defaults are size, color, and isSmiling
  - Study the code for creating parts
  - Explore how parts list is created
- Create draw() method and run scene1
- Add move() method, and run scene1
- Add smile and frown methods, which need to know about size
- Run scene 2, point out that 3 other methods needed for collisions to work

Review of Key Ideas

- *Constructor*:
  - Defined with special name `__init__`
  - Called like `ClassName()`
- *Instance variables*:
  - Created when we assign to them
  - Live as long as the object lives
- `self` formal parameter:
  - Implicitly get the value *before the dot* in the call
  - Allows an object to "talk about itself" in a method

Work on your team project

- Meet with your project team
  - Finish up what is due for session 17 milestone
  - Continue working on next milestone
  - Decide on time/venue for next meeting
- Next session
  - Another example of defining classes
  - More project work