# DEFINING CLASSES IN PYTHON

CSSE 120—Rose Hulman Institute of Technology

# Review: Using Objects in Python

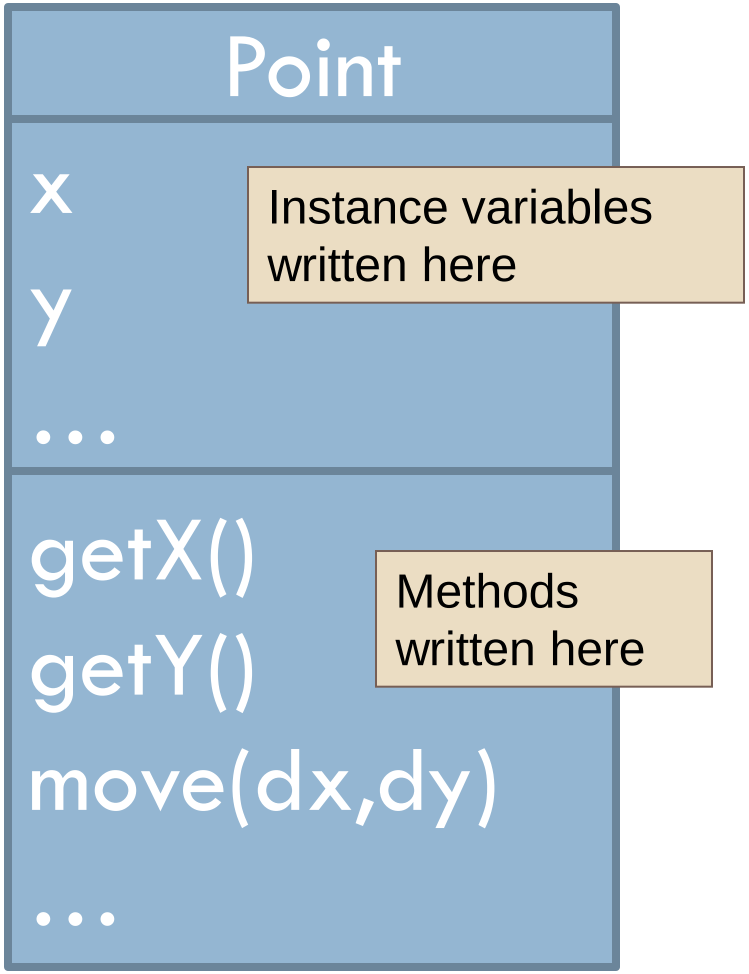
```
WIDTH = 400
HEIGHT = 50
REPEAT_COUNT = 20
PAUSE_LENGTH = 0.25
win = GraphWin(‘Saints Win!', WIDTH, HEIGHT)
p = Point(WIDTH/2, HEIGHT/2)
t = Text(p, ‘Saints—2010 Super Bowl Champs!')
t.setStyle('bold')
t.draw(win)
nextColorIsRed = True
t.setFill('blue')
for i in range(REPEAT_COUNT):
    sleep(PAUSE_LENGTH)
    if nextColorIsRed:
        t.setFill('red')
    else:
        t.setFill('blue')
    nextColorIsRed = not nextColorIsRed
win.close()
```

# 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 a C struct an object?

# 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:



Point

x

y

…

Instance variables written here

getX()

getY()

move(dx,dy)

…

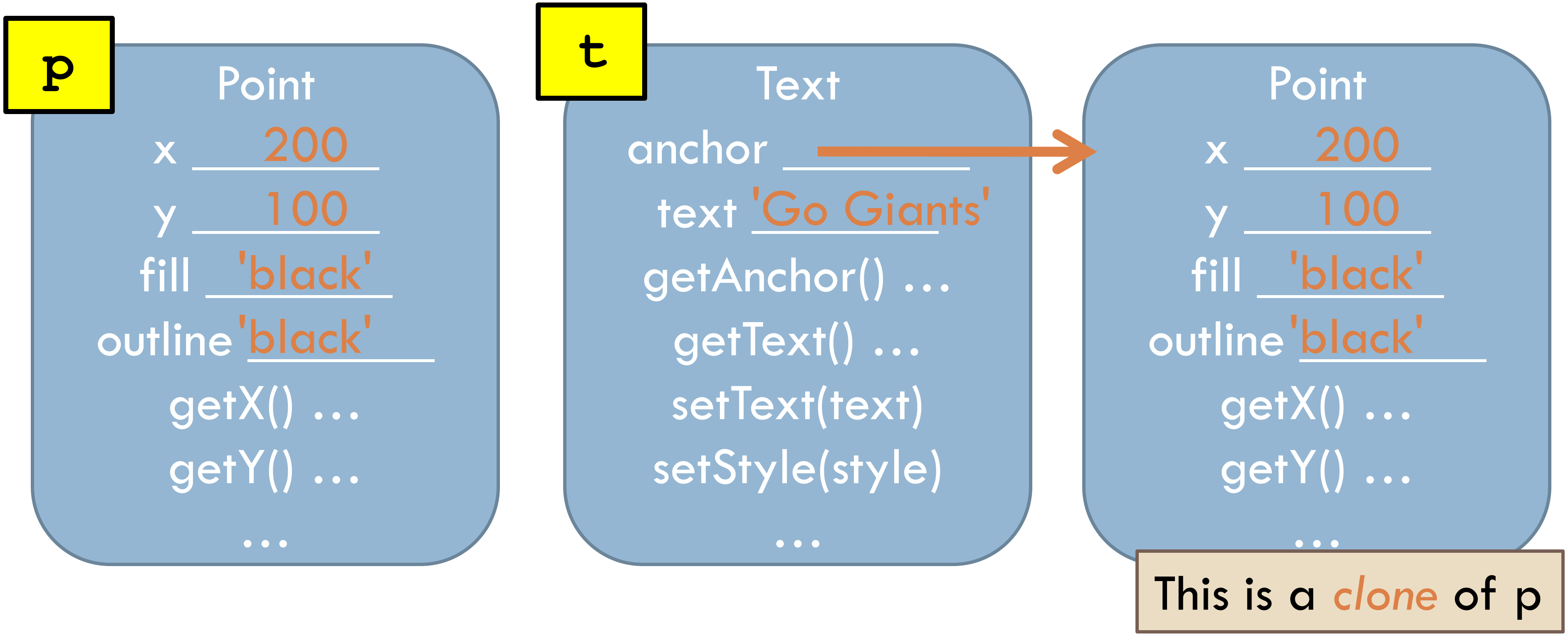
Methods written here

# 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!')`



p

Point

x 200

y 100

fill 'black'

outline 'black'

getX() …

getY() …

…

t

Text

anchor

text 'Go Giants'

getAnchor() …

getText() …

setText(text)

setStyle(style)

…

Point

x 200

y 100

fill 'black'

outline 'black'

getX() …

getY() …

…

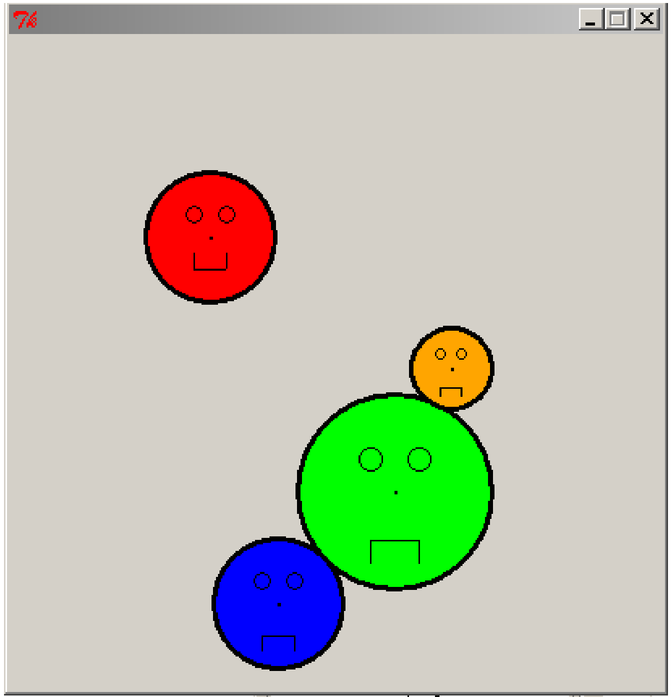
This is a *clone* of p

# 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 **MovingSmileys** project from SVN



# 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 project

- If you have finished the project
  - demonstrate it to your instructor or a TA
  - then you may leave early if you wish
- Come back for Session 30
  - Another example of defining classes
  - Course evaluations
  - Final exam review