

OBJECTS AND GRAPHICS

CSSE 120 – Rose-Hulman Institute of Technology

Outline

- Eclipse
- The object of objects
- Graphics
- Creating and using objects
- Interactive graphics
- Coordinate systems
- In-class practice time

input() and raw_input() are related through the `eval` function

- Syntax:
 - ▣ `eval(<string>)`
- Semantics
 - ▣ Input: any string
 - ▣ Output: result of evaluating the string as if it were a Python expression
- On yesterday's quiz, you investigated the relationship between `input` and `raw_input`.
- How does `eval` relate them?

Eclipse configuration

- If you haven't yet shown me your working Eclipse configuration, show me:
 - ▣ The output of either spam.py or greeting.py
 - ▣ spam.py **source code** if you have it
 - **Window > Open perspective > Other > SVN Repository Exploring** otherwise
- While I am checking people's code, please do question 1 on the quiz.

Integrated Development Environments (IDEs)

- What are they?
- Why use one?
- Our IDE – Eclipse
 - ▣ Why we chose it
 - ▣ Basic concepts in Eclipse
 - Workspace, Workbench
 - Files, folders, projects
 - Views, editors, perspectives
 - <http://www.rose-hulman.edu/class/csse/resources/Eclipse/installation.htm>

The next slides address the listed points

If your Eclipse still doesn't work

- Go to course Angel page:
Resources → Course Resources section →
Software Installation links →
Configuring Python on Eclipse
- Scroll down to the section:
Configuring PyDev
- Complete the instructions from there to the end of the document
- Get help as needed

IDEs – What are they?

An IDE is an application that makes it easier to develop software.

They try to make it easy to:

The image shows a screenshot of the PyDev IDE interface. The main window is titled "Pydev - test.py - Eclipse SDK". The interface is divided into several panes:

- Pydev Package Explorer:** Located on the left, it shows a tree view of the project structure with folders like "test" and "src", and a file named "test.py". A callout box points to it with the text "See the outline of the entire project".
- Code Editor:** The central pane shows Python code:

```
import math  
  
print "I am a newer Python module"  
for i in range(10):  
    print math.pow(i,2)
```

 A callout box points to the editor with the text "Type and change code (editors)".
- Outline:** Located on the right, it shows a hierarchical view of the code, currently displaying "math". A callout box points to it with the text "See the outline of a chunk of code".
- Console:** Located at the bottom, it shows the output of the code:

```
<terminated> C:\Documents and Settings\defoe\Python Workspace\test\src\test.py  
16.0  
25.0  
36.0  
49.0  
64.0  
81.0
```

 A callout box points to it with the text "See output".
- Toolbar:** Located at the top, it contains various icons for file operations, running, and debugging. A callout box points to it with the text "Compile, run, debug, document".

IDEs – Why use one?

An IDE is an application that makes it easier to develop software.

They try to make it easy to:

The screenshot shows the Eclipse IDE interface with the Pydev plugin. The main editor displays a Python script named `test.py` with the following code:

```
import math

print "I am a newer Python module"
for i in range(10):
    print math.pow(i,2)
```

The interface includes several panels: the Package Explorer on the left shows the project structure; the Outline on the right shows the code structure; the Console at the bottom shows the output of the program. Callouts point to various features: the top menu bar for compilation and debugging; the Package Explorer for project overview; the Outline for code chunk overview; the main editor for code editing; and the Console for output viewing.

See the outline of the entire project

Compile, run, debug, document

See the outline of a chunk of code

Type and change code (editors)

See output

Eclipse is:

- **Powerful** -- everything here and more
- **Easy** to use
- **Free** and **open-source**
- An IDE for **any language**, not just Python
- **What our upper-class students told us to use!**

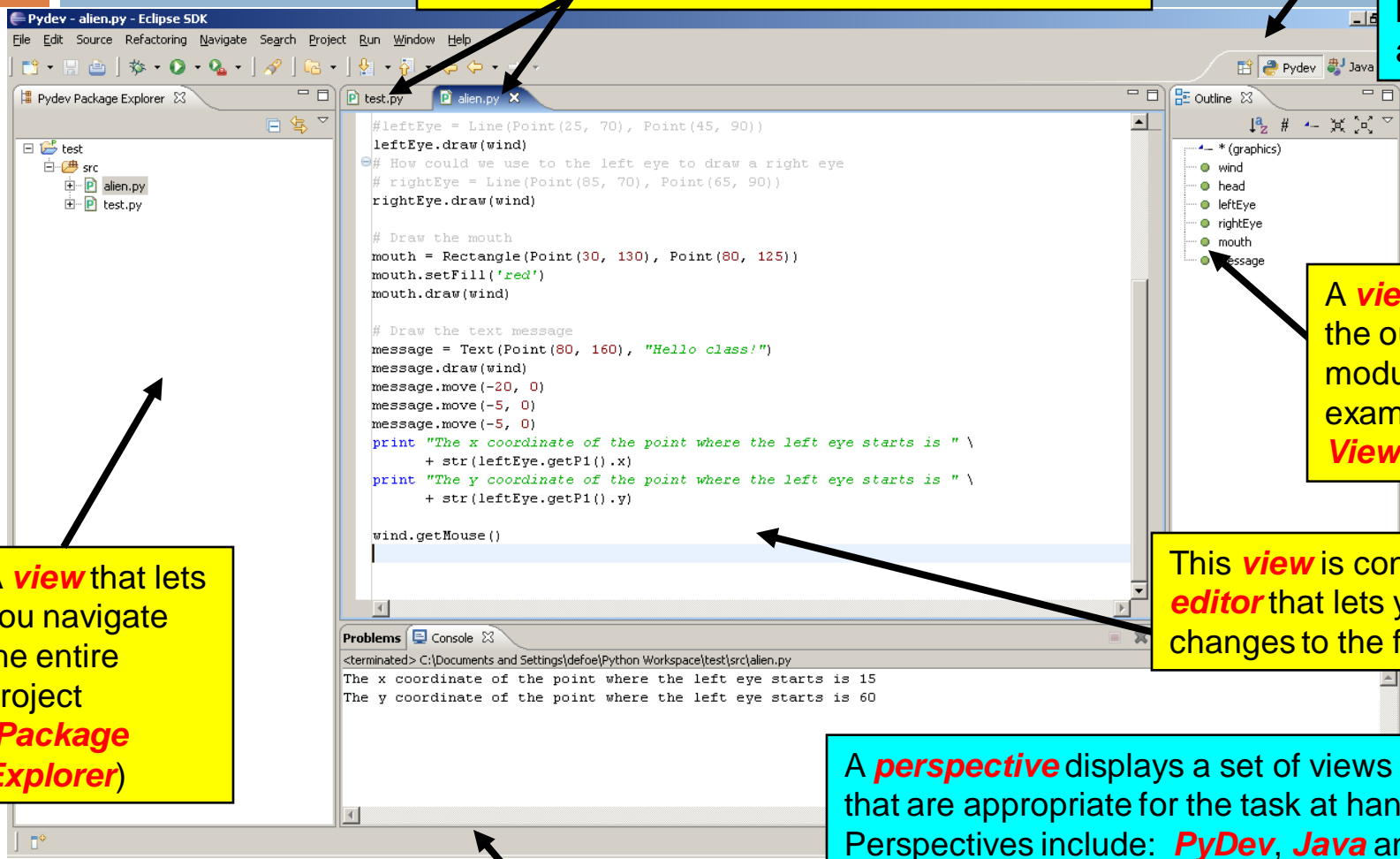
Basic concepts in Eclipse

- **Workspace** – where your *projects* are stored on your computer
- **Project** – a collection of files, organized in folders, that includes:
 - **Source code** (the code that you write)
 - **Compiled code** (what your source code is translated into, for the machine to run)
 - **Design documents**
 - **Documentation**
 - **Tests**
 - And more that you will learn about over time
- **Workbench** – what we saw on the previous slide, that is, the tool in which you do your software development

Views, editors, perspectives

Tabbed **views** of the source code of this project

This is the **PyDev perspective** but just a button click brings us to another



A **view** that lets you navigate the entire project (**Package Explorer**)

A **view** that shows the outline of the module being examined (**Outline View**)

This **view** is controlled by an **editor** that lets you make changes to the file

A **perspective** displays a set of views and editors that are appropriate for the task at hand. Perspectives include: **PyDev, Java** and lots more

Tabbed **views** (**Problems, Console**)

The object of objects

- Data types for strings and numbers are **passive**
 - ▣ Each represents set of values
 - Passive
 - ▣ Each has set of operations
 - Active
- Most modern computer programs built using Object-Oriented (OO) approach
 - ▣ Objects regarded as **active data type**
 - Know stuff
 - Can do stuff

The object of objects

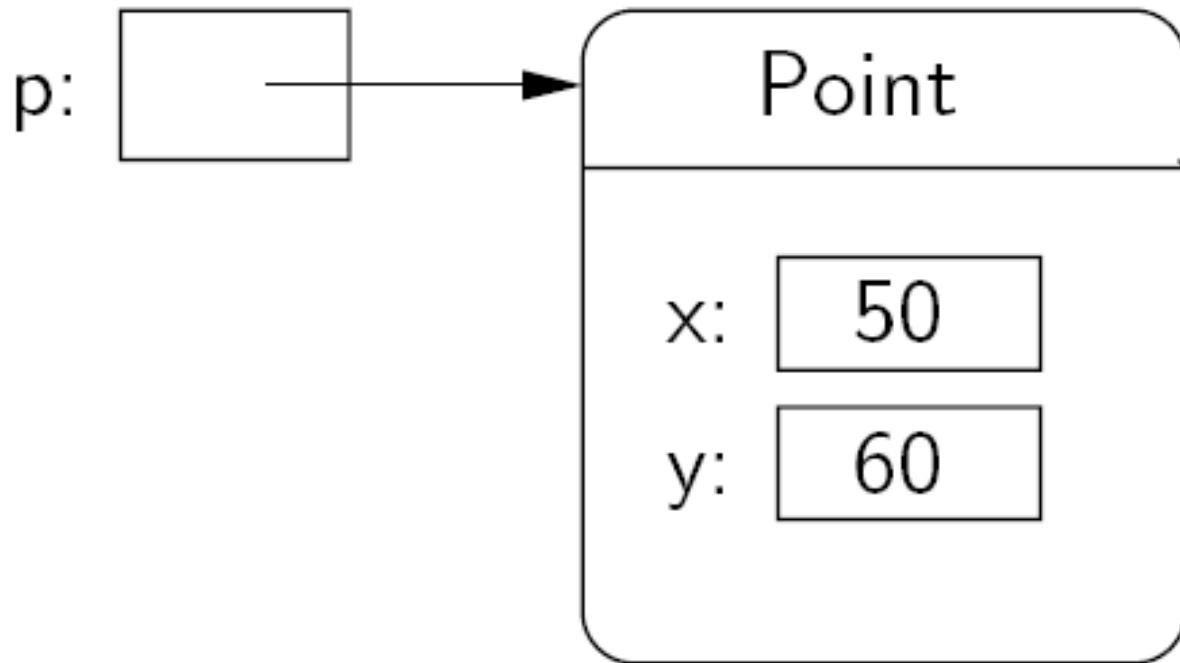
- Basic Idea of OO development
 - ▣ View complex system as interaction of simple objects
 - ▣ Example: the human body is a complex system

How do objects interact?

- Objects interact by sending each other **messages**
 - ▣ Message: request for object to perform one of its operations
 - ▣ Example: the brain can ask the feet to walk
 - ▣ In Python, messages happen *via* **method calls**.
- `>>> win = GraphWin()`
- `>>> p = Point(50, 60)` `# constructor`
- `>>> p.getX()` `# accessor method`
- `>>> p.getY()`
- `>>> p.draw(win)`

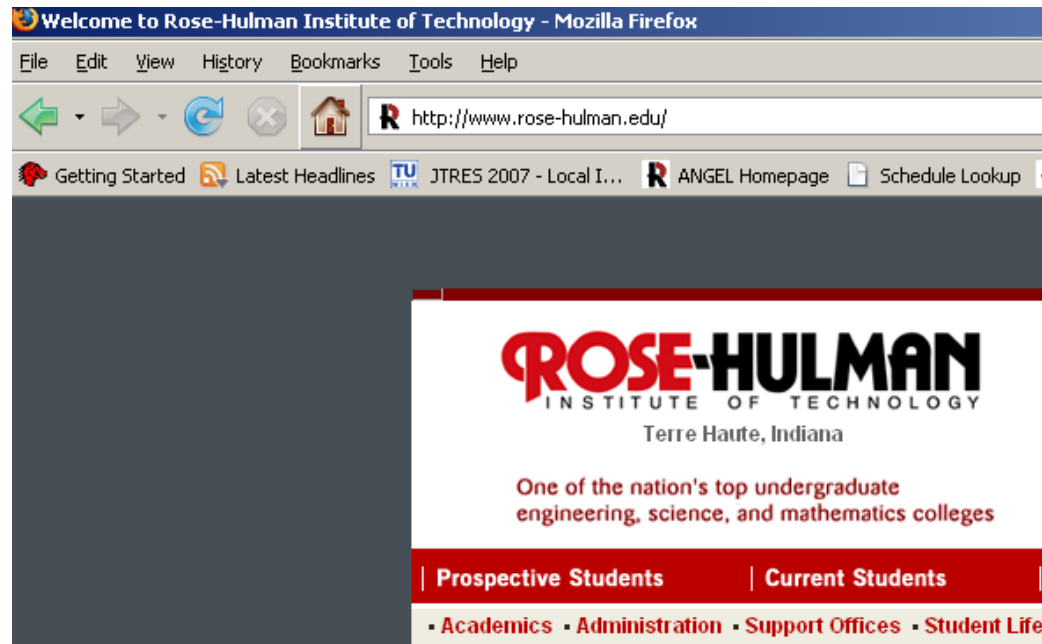
How do objects interact? Point

```
p = Point(50, 60)
```



Simple graphics programming

- Graphics is fun and provides a great vehicle for learning about objects
- Computer graphics: study of graphics programming
- Graphical User Interface (GUI)



You choose how to import

- Must import graphics library before accessing it
 - `>>> import zellegraphics`
 - `>>> win = zellegraphics.GraphWin()`
- Another way to import graphics library
 - `>>> from zellegraphics import *`
 - `win = GraphWin()`

Using graphical objects

- Using different types of objects from the graphics library, draw the following **alien face** and message



Paige clearly isn't working on homework for CSSE1 20

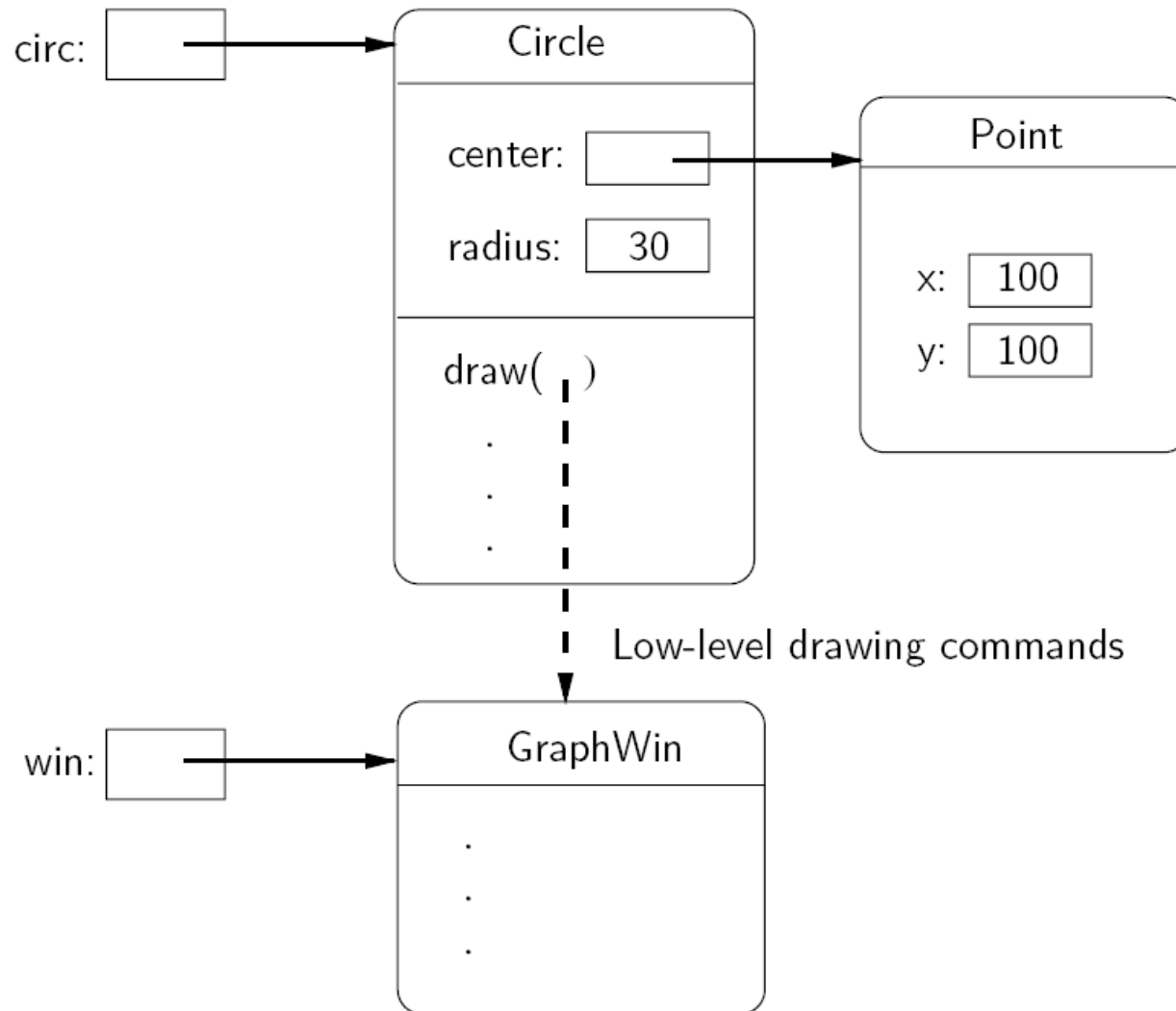


- Preview of tonight's homework:
 1. Read in and draw cool plots from the points in the files you generated in HW5
 2. Create a cool slideshow picture viewer!

Review: Class and object terminology

- Different types of objects
 - ▣ Point, Line, Rectangle, Oval, Text
 - ▣ These are examples of *classes*
- Different objects
 - ▣ head, leftEye, rightEye, mouth, message
 - ▣ Each is an *instance* of a class
 - ▣ Created using a *constructor*
 - ▣ Objects have *instance variables*
 - ▣ Objects use *methods* to operate on instance variables

Object interaction to draw a circle



Interactive graphics

- *GUI*—Graphical User Interface
 - Accepts input
 - Keyboard, mouse clicks, menu, text box
 - Displays output
 - In graphical format
 - On-the-fly
- Developed using *Event-Driven Programming*
 - Program draws interface elements (*widgets*) and waits
 - Program responds when user does something

getMouse

- `win.getMouse()`
 - ▣ Causes the program to halt, waiting for the user to click the mouse somewhere in the window
 - ▣ To find out where it was clicked, assign it to a variable:
 - `p = win.getMouse()`

Mouse Event Exercise

Together, let's solve the following problem:

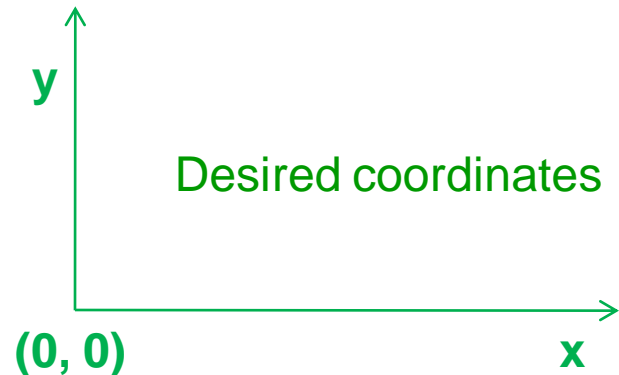
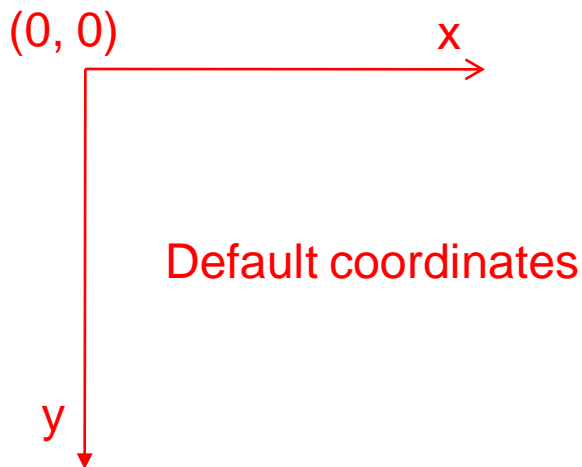
Create a program, `clickme.py`, with a window labeled "Click Me!" that displays the message *You clicked (x, y)* the first 5 times the user clicks in the window.

The program also draws a red-filled circle, with blue outline, for each of these first 5 clicks.

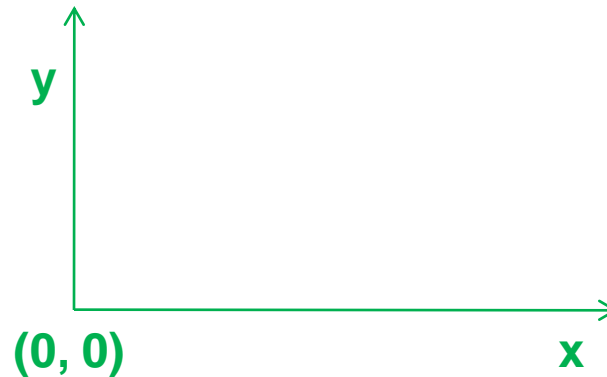
The program closes the window on the 6th click

Coordinate systems

- An important use of graphics is to represent **data** visually
 - ▣ Example: a bar chart
- We really want $(0,0)$ to be in the lower-left corner



Desired coordinate system



- `setCoords(x1, y1, x2, y2)` method from `GraphWin` class
 - ▣ Sets the coordinates to run from (x_1, y_1) in the lower-left corner to (x_2, y_2) in the upper-right corner.