

# OBJECTS AND GRAPHICS

CSSE 120 – Rose-Hulman Institute of Technology

# Outline

---

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

# Integrated Development Environments (IDEs)

Slide 3

- 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

# 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 Eclipse SDK IDE. The interface is divided into several panes:

- Pydev Package Explorer:** Located on the left, it shows a 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".
- Editor:** The central pane shows a Python code file named 'test.py'. The code is:

```
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 in the editor, listing 'math'. A callout box points to it with the text "See the outline of a chunk of code".
- Problems/Console:** Located at the bottom, it shows the output of the code execution. The output is:

```
<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 the console with the text "See output".
- Menu Bar:** At the top, it contains menus like File, Edit, Source, Refactoring, Navigate, Search, Project, Run, Window, and Help. A callout box points to the Run menu 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 showing the project structure, the Outline on the right showing the code structure, and the Console at the bottom showing the output of the program. The console output is:

```
<terminated> C:\Documents and Settings\user\workspace\test.py
16.0
25.0
36.0
49.0
64.0
81.0
```

Annotations with arrows point to various parts of the IDE:

- Compile, run, debug, document**: Points to the top toolbar.
- See the outline of the entire project**: Points to the Package Explorer.
- See the outline of a chunk of code**: Points to the Outline panel.
- Type and change code (editors)**: Points to the main code editor.
- See out...**: Points to the Console panel.

**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!**

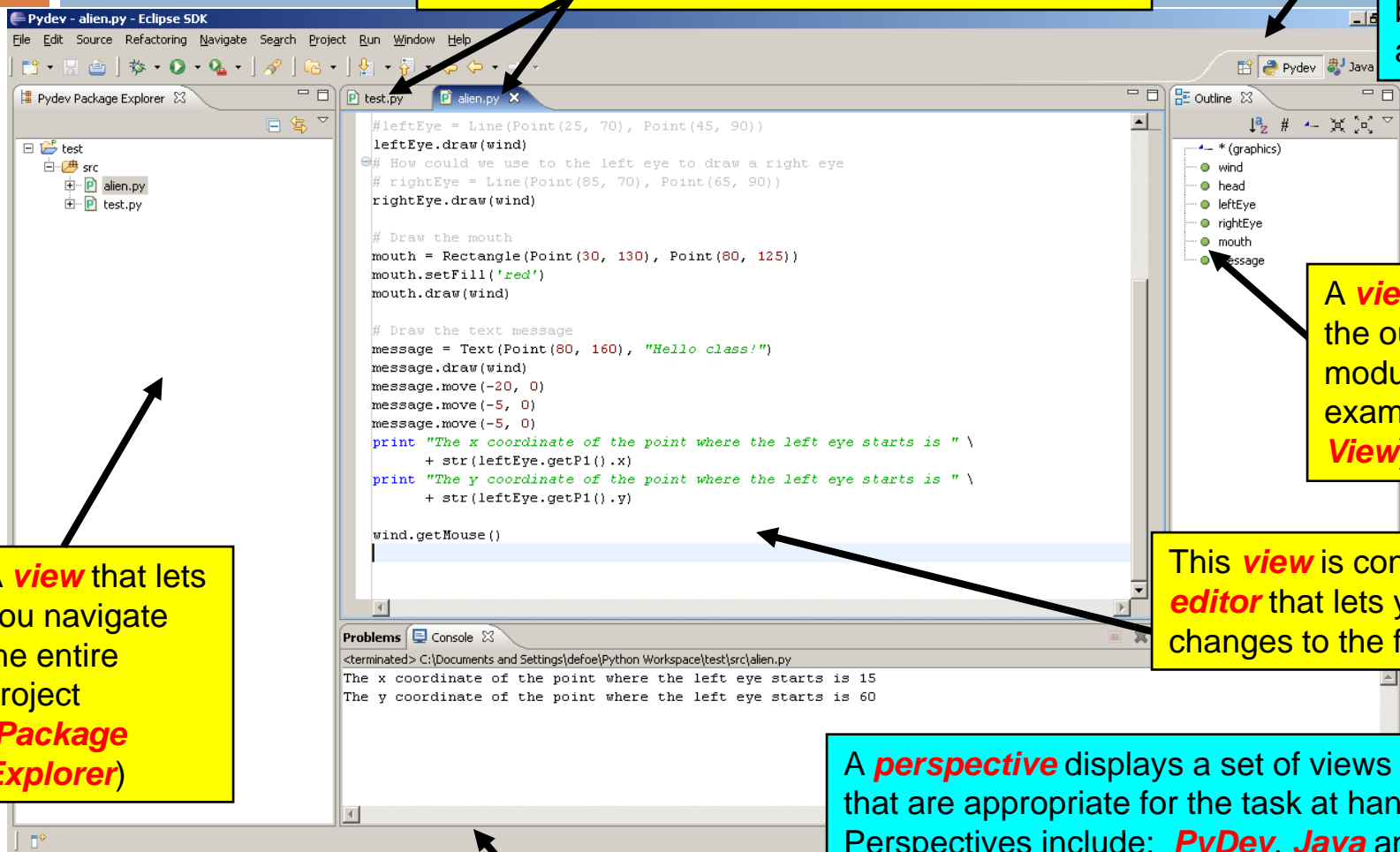
# Basic concepts in Eclipse

- **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**)

# Eclipse in a Nutshell

- **Workspace** – where your **projects** are stored on your computer
- **Project** – a collection of files, organized in folders, that includes:
  - **Source code** and **Compiled code** and more
- **Workbench** – the tool in which to work
  - It has **perspectives** which organize the **views** and **editors** that you use

# Verify that Eclipse Works for You

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

# 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

---

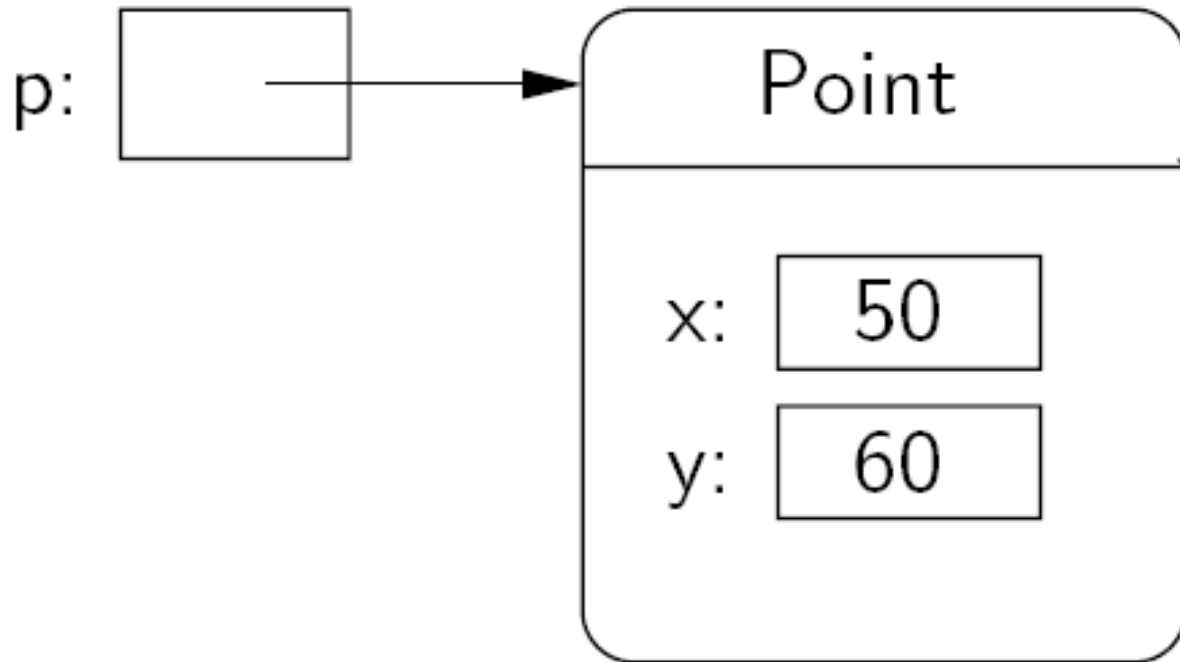
- Basic Idea of OO development
  - ▣ View complex system as interaction of simple objects
  - ▣ Example: the human body is a complex system

# 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
- `>>> win = graphics.GraphWin()`
- `>>> p = Point(50, 60)`
- `>>> p.getX()` # accessor method
- `>>> p.getY()`
- `>>> p.draw(win)`

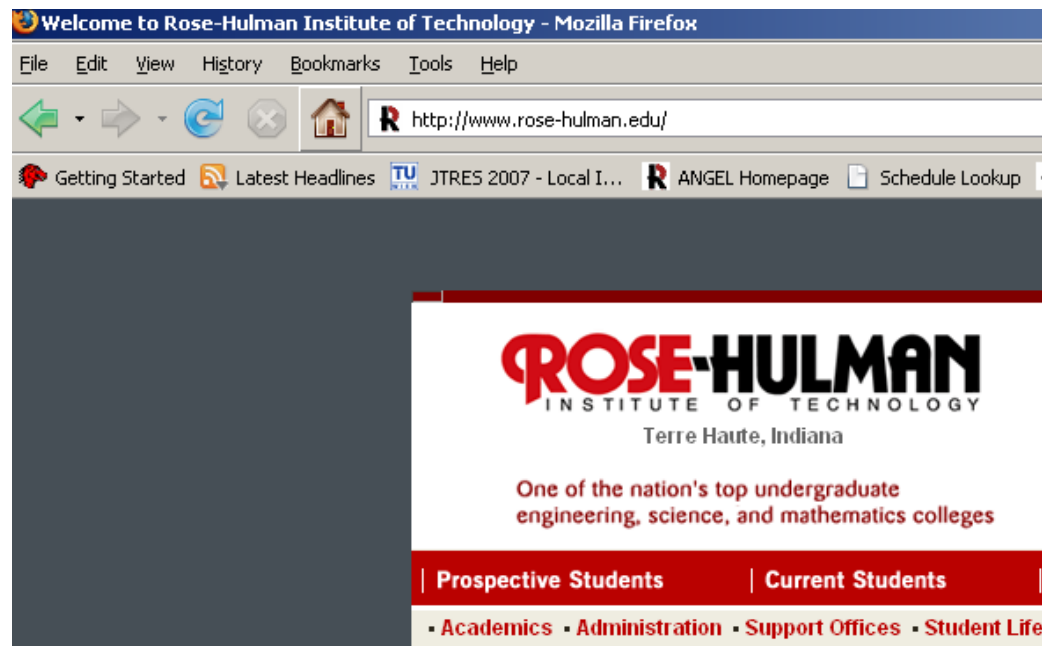
# How do objects interact -- Point

```
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)



# Simple graphics programming

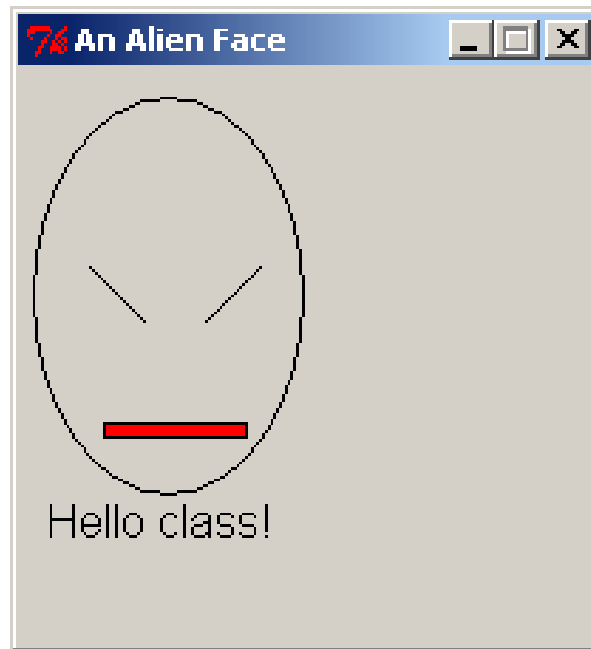
- Must import graphics library before accessing it
  - ▣ `>>> import graphics`
  - ▣ `>>> win = graphics.GraphWin()`
- Another way to import graphics library
  - ▣ `>>> from graphics import *`
  - ▣ `win = GraphWin()`
- What is the difference between these two approaches?

# Graphics window

- Collection of tiny points called pixel
  - Pixel: picture element
  - Has a title, length, and width
  - E.g. height = 200 pixels, width = 200 pixels
    - How many pixels?
- Computer monitor
  - # pixels wide
  - # pixels tall

# Using graphical objects

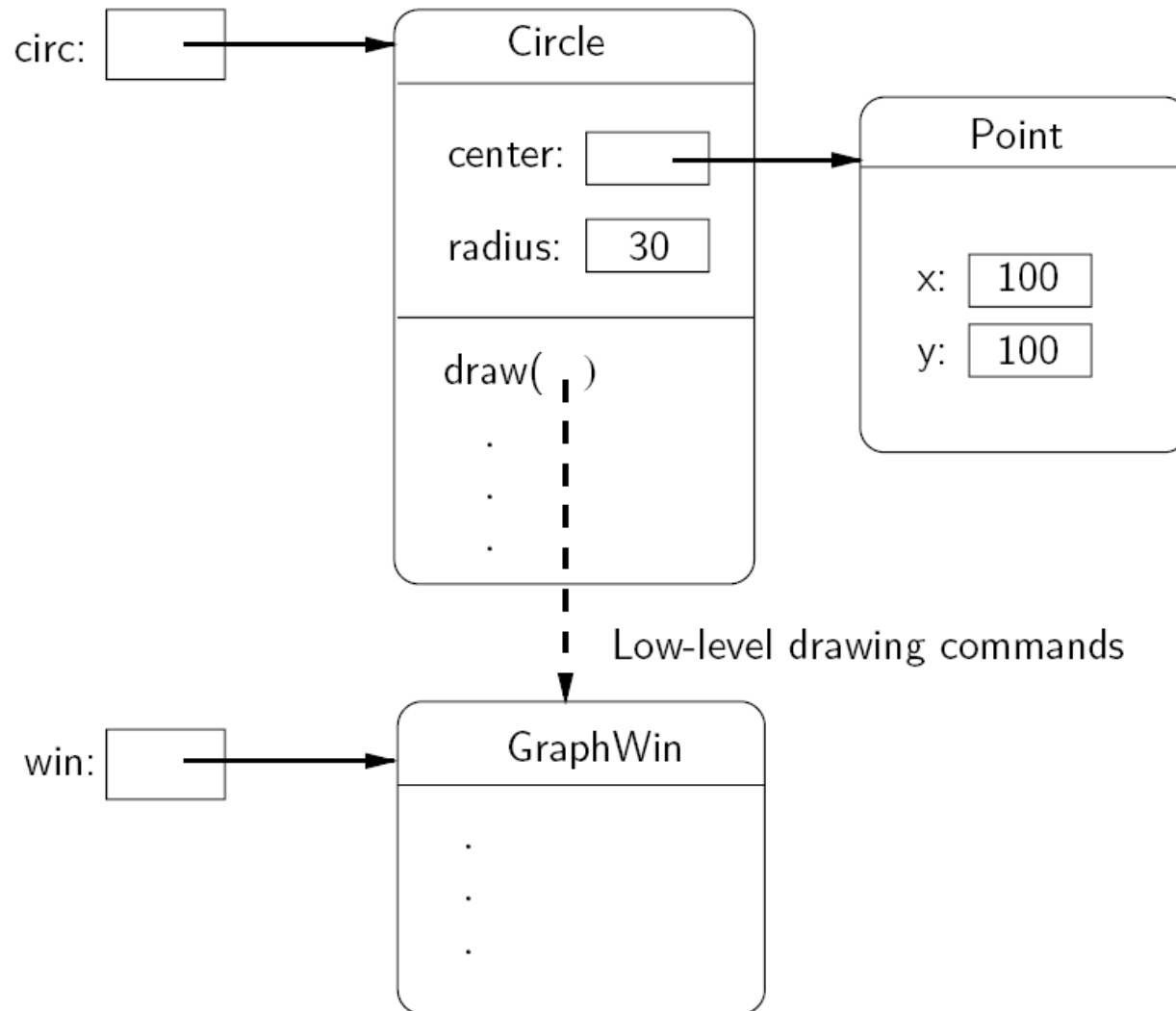
- Using different types of objects from the graphics library draw the following **alien face** and message



# Classes and objects

- 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

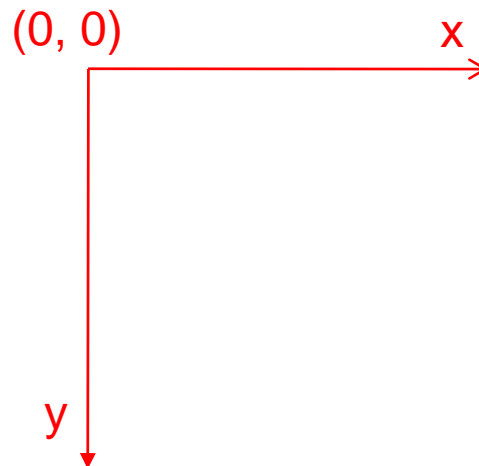


# Producing a copy of an object

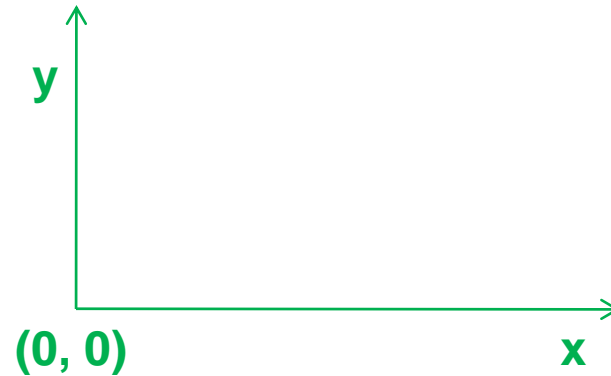
- What if the left eye was a circle and we wanted to use it to generate the right eye?
- What if we redefine leftEye and rightEye as follows?
  - ▣ leftEye = Circle(Point(25, 70), 10)  
leftEye.setFill('yellow')  
leftEye.setOutline('red')
  - ▣ rightEye = leftEye      *# aliasing*  
rightEye.move(55, 0)

# Coordinate systems

- Very important use of graphics is to visually represent data
- What if we want to plot a bar chart?
  - Graphwin



# Desired coordinate system



- Use `setCoords(x1, y1, x2, y2)` method from `GraphWin` class.

# 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

# Mouse Event Exercise

Together with class 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 10 times the user clicks in the window.

The program draws a red-filled circle, with blue outline for each of the first 10 clicks.

The program closes the window on the 11<sup>th</sup> click

# Bar chart problem – in class

- Suppose the number of CSSE majors at Rose increases by 5 % every year starting from 180 majors in the current year.
- Using an appropriate coordinate system, write a program called `majors.py` that draws a bar graph to represent the number of CSSE majors for each year over the next 8 years
- Label the axes appropriately

# Turn-in for Majors Exercise

- In the In-class Exercise Drop Boxes folder on ANGEL, there is a drop box called **Majors Drop Box**.
- Place your Python code file **majors.py** in this drop boxes before the beginning of your section's Session 7 class meeting (before next class).