

PRACTICE, ECLIPSE. DEBUGGER

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

- **In class today:**

- Look on with someone else during debugger demo
- Use other person's computer when pair programming

- **Later:**

- Follow the instructions in HW 4
- See the lab assistants or in-class assistants if you need help

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 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 and callouts:

- Pydev Package Explorer:** Shows the project structure with folders like `test` and `src`, and files like `test.py`. Callout: "See the outline of the entire project".
- Main Editor:** The central area where code is written and edited. Callout: "Type and change code (editors)".
- Outline:** A panel on the right showing the structure of the code, such as the `math` module. Callout: "See the outline of a chunk of code".
- Console:** A panel at the bottom showing the output of the code execution. Callout: "See output".
- Toolbar:** A row of icons at the top for various actions. Callout: "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 window 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 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 program output.

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!**

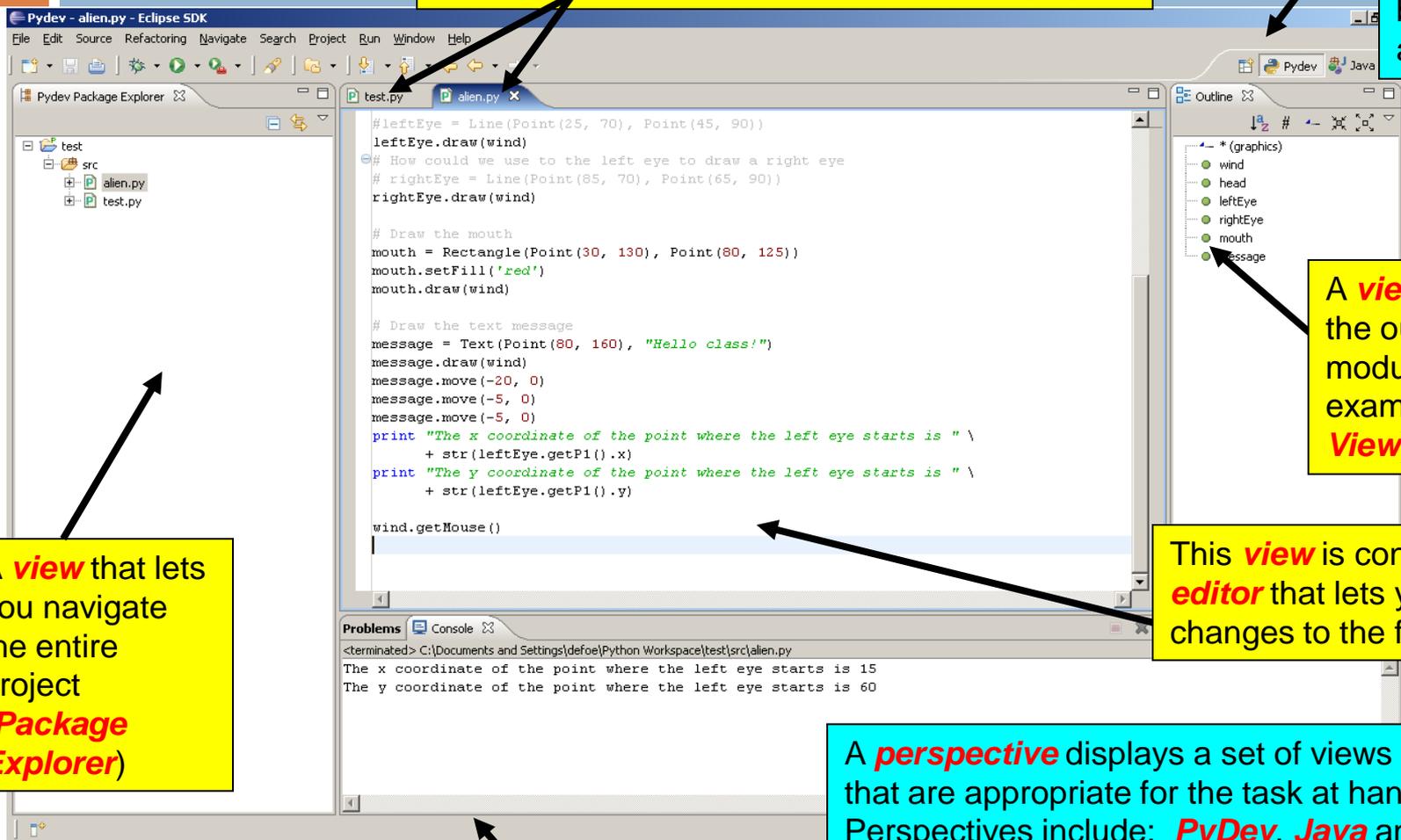
Basic concepts in Eclipse

- **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**)

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
- **View** – a "window within the window"
 - displays code, output, project contents, debugging info, etc.

Debugging

- Debugging includes:
 - ▣ Discovering errors
 - ▣ Coming up with a hypothesis about the cause
 - ▣ Testing your hypothesis
 - ▣ Fixing the error
- Ways to debug
 - ▣ Insert print statements to show program flow and data
 - ▣ Use a debugger:
 - A program that executes another program and displays its runtime behavior, step by step
 - Part of every modern IDE

Using a Debugger

- Typical debugger commands:
 - ▣ Set a breakpoint—place where you want the debugger to pause the program
 - ▣ Single step—execute one line at a time
 - ▣ Inspect a variable—look at its changing value over time
- Debugging Example
 - ▣ In the `MoveCircle.py` file, your instructor will show you how to double-click to set a breakpoint at the line that contains the call to the `sleep` function.

Sample Debugging Session: Eclipse

Debug - printFactorial.py - Eclipse SDK

File Edit Source Refactoring Navigate Search Project Run Window Help

Debug Pydev Java

Debug View: A view that shows all the executing functions

Name	Value
Globals	Global variables
formatString	str: %21d
n	int: 0
product	int: 1
width	int: 21

Variables View: This is the *Debug perspective*

Editor: A view that shows all the variables

```
1 def printFactorial(n, width):
2     formatString = "%"+str(width)+ "d"
3     product = 1
4     for i in range(1, n+1):
5         product = product * i
6
7     print formatString % (product)
8
9 #printFactorial(5, 6)
10 #printFactorial(15, 20)
11
12 print "Factorial Table"
13
14
```

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

- printFactorial
- factTable

Console:

```
printFactorial.py
pydev debugger
Factorial Table
0
```

Writable Insert 4 : 1

Tips to Debug Effectively

- Reproduce the error
- Simplify the error
- Divide and conquer
- Know what your program should do
- Look at the details
- Understand each bug before you fix it
- Practice!

Use the scientific method:

- hypothesize,
- experiment,
- fix bug,
- repeat experiment

Practice with Loops, Lists, Strings

- Work with another student, pair programming
- Several small programs/exercises
- If you do not finish them, do so for homework
- Use Eclipse, so you can get practice with it
- Make a new PyDev project called Session6
- Make a new Pydev module called Session6
- Put all of your code/answers for the in-class exercises in this file(Session6.py)
- Details are in the HW 6 document
 - ▣ Accessible frm the Schedule page.