Debugging; Objects and Graphics

Rose-Hulman Institute of Technology

Computer Science and Software Engineering

Check out 05-DebuggingObjectsAndGraphics from SVN. Get help if you're stuck.

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

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

Q1

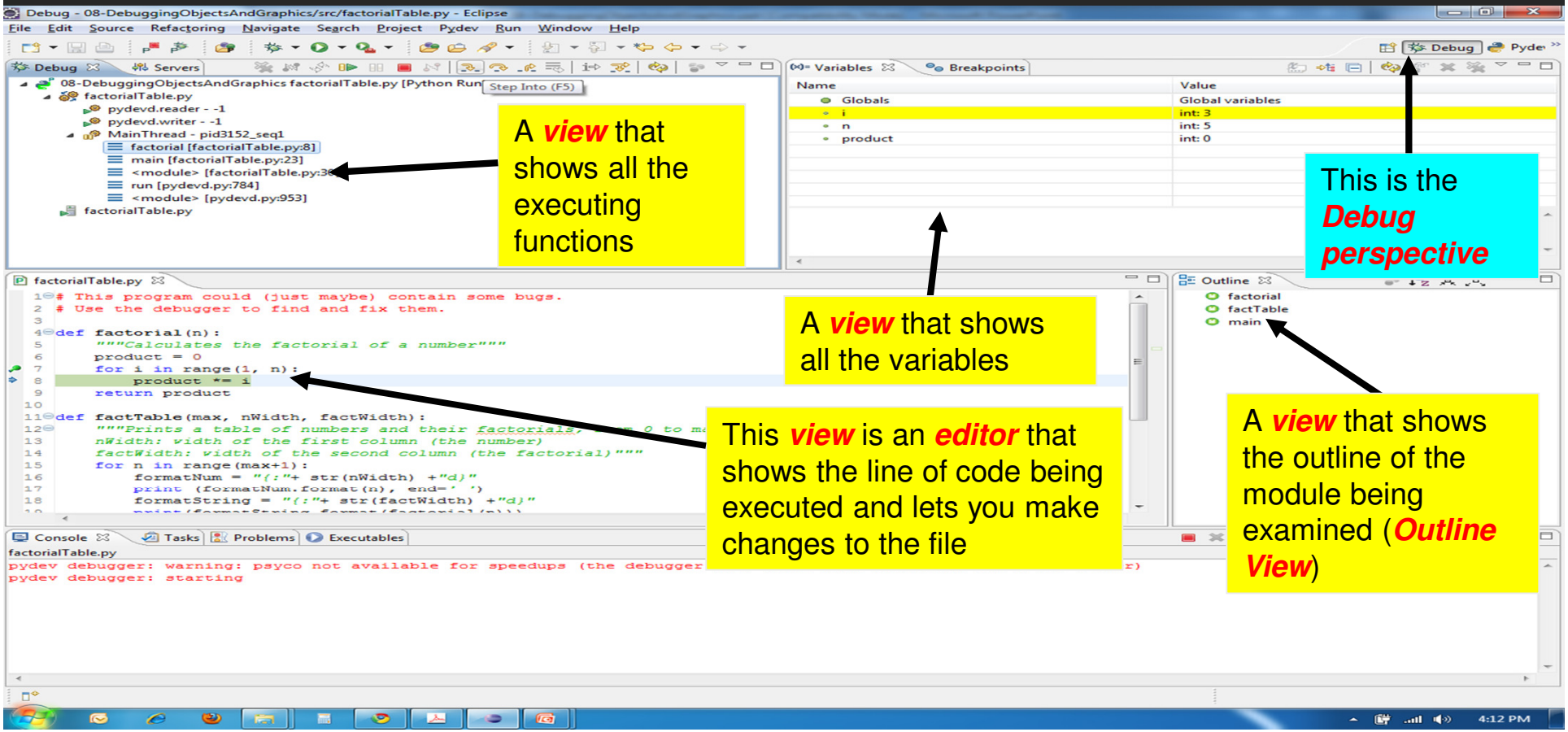
Using a Debugger

- 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 **factorialTable.py** file
  - Start a debugging session:
    - Window → Open Perspective→ other→ Debug → Okay
    - Click debug icon (top-left side of work bench)

Using the debugger in Eclipse

- Set a breakpoint
  - Double click in left margin of editor view
- Step over (when you know a function works)
  - Click step-over icon or use F6 key
- Variable inspection
  - Look at the new value of i, cir after each time though the loop

Sample Debugging Session: Eclipse



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!

It's the scientific method!
•hypothesize,
•experiment,
•fix bug,
•repeat experiment

Q2

The object of objects

- Data types for numbers are passive
- Most modern computer programs are built using an *Object-Oriented* (OO) approach
  - An *object* is an active data type
    - It *knows* stuff
    - It *does* stuff

The object of objects

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

Q4,5

How do objects interact?

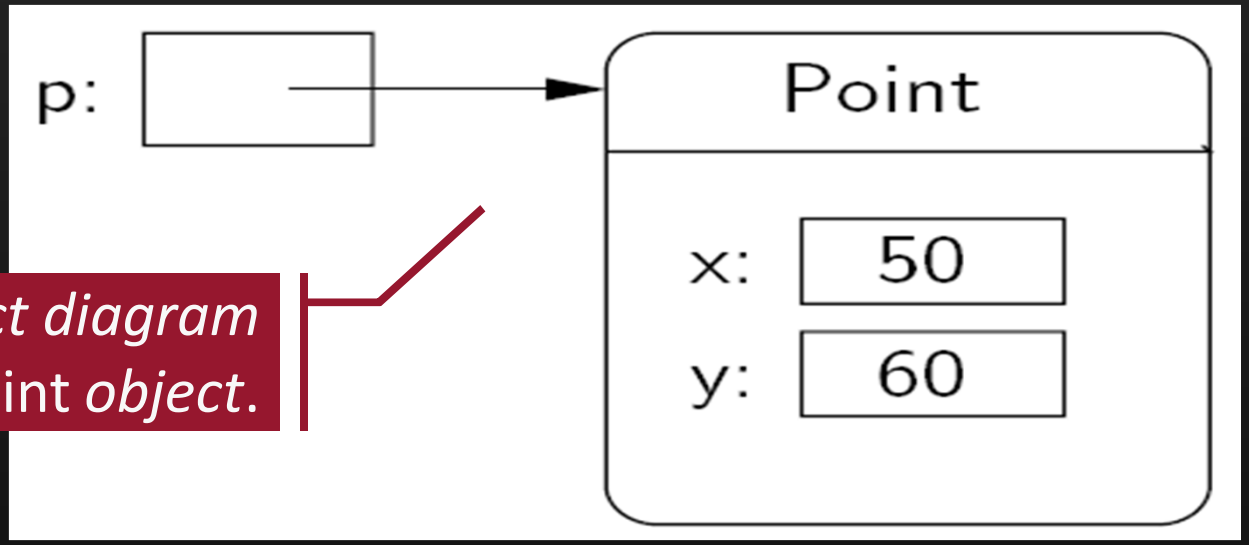
- Objects interact by sending each other *messages*
  - Message: request for object to perform one of its operations
  - Example: the brain can ask the feet to walk
  - In Python, messages happen via *method calls*.
- win = GraphWin("Window", 10, 20) **# constructor**
- >>> p = Point(50, 60) **# constructor**
- >>> p.getX() **# *accessor* method**
- >>> p.getY() **# *accessor* method**
- >>> p.draw(win) **# method**

Q6,7

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

How do objects interact? Point

p = Point(50, 60)



UML *object diagram* for a point *object*.

Q8,9

UML ➔ Unified Modeling Language

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Simple graphics programming

- Great way to learn about objects
- *Computer Graphics*: study of graphics programming
  - Important for gaming and movie industries
  - Military applications
- Graphical User Interface (GUI)

Review: 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

- Look at the **alienFace** module in today's SVN project

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

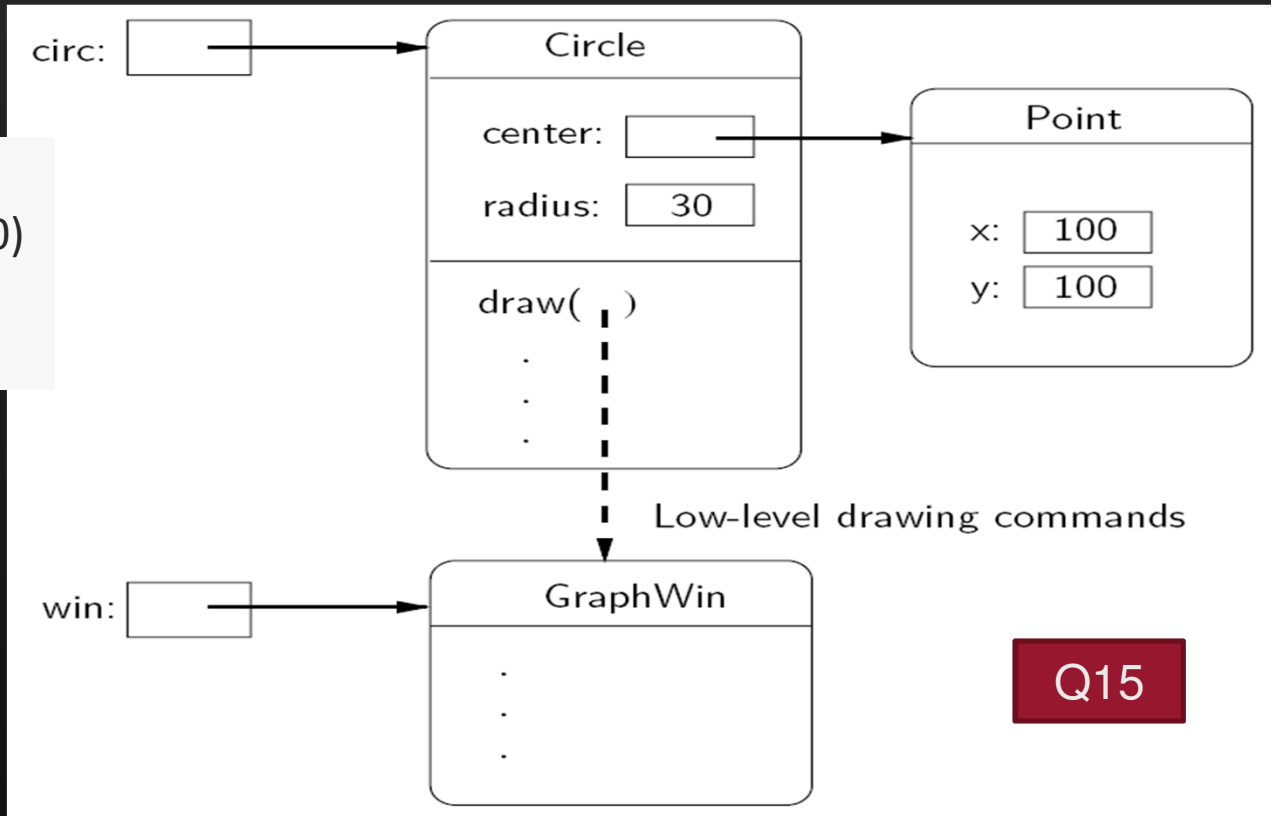
Recap: Class and object terminology

- 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

Q12,13,14

Object interaction to draw a circle

```
from zellegraphics import *
circ = Circle(Point(100, 100), 30)
win = GraphWin()
circ.draw(win)
```



Q15

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

Example: getMouse

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

Mouse Event Exercise

- Create a program in module, **clickMe**, with a window labeled “Click Me!” that displays the message **You clicked (x, y)** to the console the first 5 times the user clicks in the window.
- The program also draws a red-filled circle, with blue outline, in the location of each of these first 5 clicks.
- The program closes the window on the 6th click

Would you like to see more examples?

Practice Problem Sundays?

Q17,18

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY