

# DEBUGGING AND INDEFINITE LOOPS

CSSE 120—Rose Hulman Institute of Technology

# Exam 1

- Questions?
- Solutions are posted in the Exam 1 folder
- Later in class, we'll discuss solutions to computer problem #2.

# 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 the debugger will pause the program
  - ▣ Single step—execute one line at a time
  - ▣ Inspect a variable—look at its changing value over time
- Debugging Example
  - ▣ Checkout the Session10 project from your repository and open `factorialTable.py`

# Sample Debugging Session: Eclipse

**Debug - printFactorial.py - Eclipse SDK**

File Edit Source Refactoring Navigate Search Project Run Window Help

Debug Pydev Java

**Debug**

- test printFactorial.py [Python Run]
- printFactorial.py
  - MainThread
    - printFactorial [printFactorial.py:4]
    - factTable [printFactorial.py:22]
    - <module> [printFactorial.py:24]
    - run [pydevd.py:634]
    - <module> [pydevd.py:779]

**Variables**

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

int: 21

**printFactorial.py**

```
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**

- printFactorial
- factTable

**Console**

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

Writable Insert 4 : 1

**Annotations:**

- A **view** that shows all the executing functions
- This is the **Debug perspective**
- A **view** that shows all the variables
- This **view** is an **editor** that shows the line being executed and lets you make changes to the file
- A **view** that shows the outline of the module being examined (**Outline View**)

# 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

# Exam Question 2

---

- Let's solve it together, using the debugger as needed.

# Review: Definite Loops

- Review: For loop
  - ▣ Definite loop: knows *a priori* the number of iterations of loop body
  - ▣ Counted loop: sequence can be generated by `range()`
  - ▣ Example for loop in `slideshow.py`
- Syntax:
  - ▣ `for <var> in <sequence>:`  
    <body>

# Is This Loop a Definite Loop?

```
#Open the file
inputFile = open(inputFileName, 'r')

# process each line of file
for line in inputFile:
    image = Image(imageCenter, line.rstrip())
    image.draw(win)
    time.sleep(delay)

win.getMouse()
inputFile.close()
win.close()
```

# Indefinite Loops

- Number of iterations is not known when loop starts
- Is a conditional loop
  - ▣ Keeps iterating as long as a certain condition remains true
  - ▣ Conditions are Boolean expressions
- Typically implemented using while statement
- Syntax:

```
while<condition> :  
    <body>
```

# While Loop

- A *pre-test loop*
  - ▣ Condition is tested at the top of the loop
- Example use of while loops

Nadia deposits \$100 in a savings account each month. Each month the account earns 0.25% interest on the previous balance. How many months will it take her to accumulate \$10,000?

# Exercise: While Loops

- Open `guessMyNumber.py` in the Session10 project.
- Follow the instructions there and demo your program to your instructor or an assistant when you finish.
- Commit your work



- When you are done, please start HW10.