

DEBUGGING AND INDEFINITE LOOPS

Exam 1

- Questions?
- Solutions are posted in the Exam 1 folder on ANGEL

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
 - ▣ 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

Package Explorer: 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] → printFactorial.py

Variables View:

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

int: 21

Code Editor:

```
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
```

Console:

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

Annotations:

- A *view* that shows all the executing functions** (points to Package Explorer)
- This is the *Debug perspective*** (points to the top toolbar)
- A *view* that shows all the variables** (points to Variables View)
- This *view* is an *editor* that shows the line of code being executed and lets you make changes to the file** (points to Code Editor)
- A *view* that shows the outline of the module being examined (*Outline View*)** (points to Outline View)

Writable Insert 4 : 1

Q1

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

Review: Definite Loops

□ Review: For loop

- ▣ Definite loop: *knows before the loop starts to execute the number of iterations of the 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?

Infinite loops on purpose

- With **for** loops, we could make the program run for a really long time, but not forever.
- Create a very simple **while** loop that runs forever.

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.