

Objectives

- Review
- Lab 1
 - Linux practice
 - Programming practice
 - Numeric operations
 - Getting input from the user

Jan 19, 2010

Sprenkle - CSCI111

1

Review: Testing Process



- Test case: **input** used to test the program, **expected output** given that input
- Verify if **output** is what you expected
- Need good test cases
 - Good that you know the “problematic” test cases, even if we don’t know how to address them yet

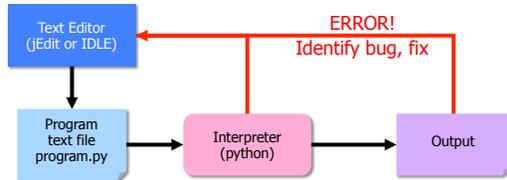
Jan 19, 2010

Sprenkle - CSCI111

2

Debugging

- Edit the program, re-execute/test until everything works
- The error is often called a “bug”
- Diagnosing and fixing it is called **debugging**



Jan 18, 2010

Sprenkle - CSCI111

debug_practice.py 3

Good Development Practices

- Design the algorithm
 - Break into pieces
- **Implement and Test** each piece *separately*
 - Identify the best pieces to make progress
 - Iterate over each step to improve it
- Write comments FIRST for each step
 - Elaborate on what you’re doing in comments when necessary

Jan 18, 2010

Sprenkle - CSCI111

4

Review: Arithmetic Operations

Symbol	Meaning	Associativity
+	Addition	Left
-	Subtraction	Left
*	Multiplication	Left
/	Division	Left
%	Remainder (“mod”)	Left
**	Exponentiation (power)	Right

Precedence rules: P E - DM% AS
 ↑
 negation

Associativity matters when you have the same operation multiple times

Jan 19, 2010

Sprenkle - CSCI111

5

Review: Two Types of Division

- Float Division: Result is a **float**
 - $3.0/6.0 \rightarrow 0.5$
 - $6.0/3.0 \rightarrow 2.0$
 - **At least** one of numerator and denominator must have a decimal, i.e., have type **float**
 - Integer Division: Result is an **int**
 - $3/6 \rightarrow 0$
 - $6/3 \rightarrow 2$
 - x/y , if both x and y are **ints**
 - If both numerator and denominator are **ints**, result is **int**
- Not always obvious

Jan 19, 2010

Sprenkle - CSCI111

6

General Announcements

- CS Issues Grading/Expectations
 - 6 pts for blog entry
 - Common issue – missing answers to one of questions
 - 4 pts for participation in class
- Idea: online grading (in Sakai) in future?
 - No paper copy (less waste)
 - But you'd have to login to Sakai and look at the feedback

Jan 19, 2010

Sprengle - CSCI111

7

Lab 0 Feedback

- Overall, did well
 - Often lost points because missed some directions
 - E.g., broken Web page links, documentation in programs, output of programs
 - Generally, lab grades should be high
- Interesting article links
 - Consider reviewing for extra credit
 - Missed my Sakai extra credit

Jan 19, 2010

Sprengle - CSCI111

8

Linux Command Conventions

- `<arg>` means fill in the appropriate thing
- `[arg]` means optional argument
- Example: Move or Rename a file
 - `mv <sourcefile> <destination>`
 - If `<destination>` is a *directory*, keeps the original source file's name

```
mv ~/labs/file.py labs/lab1/
```

directory

 - File "file.py" will be in labs/lab1 directory

Jan 19, 2010

Sprengle - CSCI111

9

Lab 0 Feedback

- `ls -l` option
 - Demonstrate how different from `-1` option
- Need electronic as well as printed submission
 - I can execute your program, help find mistakes
 - Copy your lab directory into your turnin directory
 - How do you copy a directory?

Jan 19, 2010

Sprengle - CSCI111

10

Lab 1: Linux Practice

- Setting up directories
- Renaming/moving files
- Note: terminal tells you which directory you're in

```
sprengle@hopper:~/home/courses/cs111/handouts/lab1
┌───┴───┐
File Edit View Terminal Help
dnur-xr-x 4 sprengle cs111 4.0K 2009-01-08 11:58 ./
dnur-xr-x 2 sprengle cs111 4.0K 2009-01-09 15:43 ./
-fer-r-- 1 sprengle cs111 387 2009-01-12 15:01 area.py
[sprengle@hopper lab1]$ more area.py
# Calculate the area of a rectangle
# by CS111, 01/07/2009

print "This program calculates the area of a rectangle."
# First, just assign width and height values.
# Then, get from user
width = 3.5
height = 2.2

width = input("Enter the width of the rectangle: ")
height = input("Enter the height of the rectangle: ")

area = width * height

print "The area of the rectangle is", area
[sprengle@hopper lab1]$ nano
[sprengle@hopper lab1]$ xv 6
[!] 4073
[sprengle@hopper lab1]$ []
```

Jan 19, 2010

Lab 1: Programming Practice

- Name them **lab1.n.py**, where *n* is the problem you're working on
- After completed, demonstrate that your program works
 - Close IDLE/Python shell, rerun program
 - Get rid of the output from when you were developing/debugging ("scratch work")
 - Execute using **good** test cases
 - More than one test case if dealing with user input
 - Don't need to exhaustively test
 - Save output for each program in file named **lab1.n.out** where *n* is the problem you're working on

Jan 19, 2010

Sprengle - CSCI111

12

Lab 1 Notes

- Expect comments in programs
 - High-level comments, authorship
 - Notes for your algorithms, implementation
- Expect testing on programs
 - What are good test cases for your programs?
 - Show the output from those test cases
 - But don't go overboard, testing every possible number!
- Honor System
 - Pledge the Honor Code on printed sheets