

## Objectives


- Continuing fundamentals of programming in Python
- Numeric Operations
- Software development practices
  - Testing
  - Debugging
  - Iteration

Jan 17, 2011

Sprenkle - CSCI111

1

## Parts of an Algorithm

- Input, Output
- Primitive operations 
  - What data you have, what you can do to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques

Jan 17, 2011

Sprenkle - CSCI111

2

## Review

- What are Python's primitive data types and what do they represent?
- Note on Linux machines: "Switch User" is not the same as "Logout"

Jan 17, 2011

Sprenkle - CSCI111

3

## Recap of Programming Fundamentals

- Most important data types (for us, for now): **int, float, str, bool**
  - Use these types to represent various information
- Variables have identifiers, (implicit) types
  - Should have "good" names
  - Names: start with lowercase letter; can have numbers, underscores
- Assignments
  - $x = y$  means "x set to value y" or "x is assigned value of y"
  - Only variable on LHS of statement changes

Jan 17, 2011

Sprenkle - CSCI111

4

## Review: Assignment statements

- Assignment statements are NOT math equations!

```
count = count + 1
```

- These are commands!

```
x = 2
y = x
x = x + 3
```

What is the value of y?

Jan 17, 2011

Sprenkle - CSCI111

5

## What are the values?

- After executing the following statements, what are the values of each variable?

```
> a = 5
> y = a + -1 * a
> z = a + y / 2
> a = a + 3
> y = (7+x)*z
> x = z*2
```

### Runtime error:

- x doesn't have a value yet!
- We say "x was not initialized"
- Can't use a variable on RHS until seen on LHS!\*

Jan 17, 2011

Sprenkle - CSCI111

6

## More on 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 17, 2011

Sprenkle - CSCI111

7

## NOT Math Class

- Need to write out all operations explicitly

> In math class,  $a(b+1)$  meant  $a * (b+1)$

Write this way in Python

Jan 17, 2011

Sprenkle - CSCI111

8

## Math Practice

```
5+3*2
2 * 3 ** 2
-3 ** 2
2 ** 3 ** 3
```

How should we verify our answers?

Jan 17, 2011

Sprenkle - CSCI111

9

## Two Types of Division

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

Jan 17, 2011

Sprenkle - CSCI111

10

## Division Practice (NOT Math class)

- What is the result?
- What is the **type** of the LHS variable?

- $x = 6/4$
- $y = 4 / 6 * 5.0$
- $a = 6/12.0$
- $b = 6.0/12$
- $z = .3$
- $z = x / y$
- $z = x / 3$

Jan 17, 2011

Sprenkle - CSCI111

11

## Parts of an Algorithm

- Input, Output ←
- Primitive operations
  - > What data you have, what you can do to the data
- Naming
  - > Identify things we're using
- Sequence of operations
- Conditionals
  - > Handle special cases
- Repetition/Loops
- Subroutines
  - > Call, reuse similar techniques

Jan 17, 2011

Sprenkle - CSCI111

12

## Printing Output

- **print** is a special command
  - Displays the result of expression(s) to the terminal
- `print "Hello, class"`
  - string literal
  - print** automatically adds a '\n' (carriage return) after it's printed
- `print "Your answer is", 4*4`
  - Displays same as:
    - `print "Your answer is",`
    - `print 4*4`
  - Syntax:** comma
  - Semantics:** print multiple "things" in one line

Jan 17, 2011

Sprenkle - CSCI111

## Interactive Programs

- Often, meaningful programs need input from users
- Demo: `input_demo.py`

Jan 17, 2011

Sprenkle - CSCI111

14

## Getting Input From User

- **input** and **raw\_input** are *functions*
  - **Function:** A command to do something
    - A "subroutine"
  - Prompts user for input, gets the user's input
  - **input:** to read in *numbers*
  - **raw\_input:** to read in *strings/text*
- Syntax:
  - `input(<string_prompt>)`
  - `raw_input(<string_prompt>)`

Jan 17, 2011

Sprenkle - CSCI111

15

## Getting Input From User

- Typically used in assignments
- Examples:
  - Prompt displayed to user
  - `width = input("Enter the width: ")`
    - **width** is assigned the number the user enters
    - Use **input** because expect a number from user
  - `name=raw_input("What is your name?")`
    - **name** is assigned the string the user enters
    - Use **raw\_input** because expect a string from user

What do you think the code looks like for `input_demo.py`?

Jan 17, 2011

Sprenkle - CSCI111

16

## Getting Input from User

```
color = raw_input("What is your favorite color? ")
```

Semantics: Assigns variable **color** the user's input

### Terminal:

Grabs every character up to the user presses "enter"

```
> python input_demo.py
What is your favorite color? blue
Cool! My favorite color is _light_ blue !
```

Jan 17, 2011

Sprenkle - CSCI111

input\_demo.py 17

## Documenting Your Code

- Use English to describe what your program is doing in *comments*
  - Everything after a **#** is a comment
    - Color-coded in IDLE, jEdit
  - Python does not execute comments
- Does not affect the correctness of your program
- Improves program's *readability*
  - Easier for someone else to read and update your code

Jan 17, 2011

Sprenkle - CSCI111

18

## When to Use Comments

- Document the author, high-level description of the program at the top of the program
- Provide an outline of an algorithm
  - Separates the steps of the algorithm
- Describe difficult-to-understand code

Jan 17, 2011

Sprenkle - CSCI111

19

## Identify the Parts of a Program

```
# Demonstrate numeric and string input
# by Sara Sprenkle for CSCI111
#

color = raw_input("What is your favorite color? ")
print "Cool! My favorite color is _light_", color, "!"

scale = input("On a scale of 1 to 10, how much do you
like Matt Damon? ")
print "Cool! I like him", scale*1.8, "much!"
```

Identify the comments, variables, functions,  
expressions, assignments, literals

Jan 17, 2011

Sprenkle - CSCI111

input\_demo.py 20

## Identify the Parts of a Program

```
# Demonstrate numeric and string input
# by Sara Sprenkle for CSCI111
#

color = raw_input("What is your favorite color? ")
print "Cool! My favorite color is _light_", color, "!"

scale = input("On a scale of 1 to 10, how much do you
like Matt Damon? ")
print "Cool! I like him", scale*1.8, "much!"
                        expression
```

Identify the comments, variables, functions,  
expressions, assignments, literals

Jan 17, 2011

Sprenkle - CSCI111

21

## Our First Computational Algorithm

- Find the area of a rectangle (which has a width and height)

Jan 17, 2011

Sprenkle - CSCI111

22

## Our First Computational Algorithm

- Find the area of a rectangle (which has a width and height)
- Algorithm:
  - Optional: get the width and height from user
    - Alternative: "hard-code" width and height
  - Calculate area
  - Print area

Jan 17, 2011

Sprenkle - CSCI111

area.py

23

## Errors/Bugs

- Sometimes the program doesn't work
- Types of programming errors:
  - Syntax error
    - Interpreter shows where the problem is
  - Logic/semantic error
    - answer = 2+3
    - No, answer should be 2\*3
  - Exceptions/Runtime errors
    - answer = 2/0
    - Undefined variable name

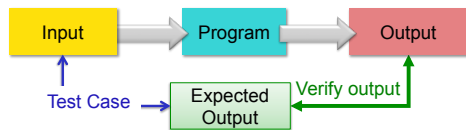
Expose errors when **Testing**

Jan 17, 2011

Sprenkle - CSCI111

24

## Testing Process



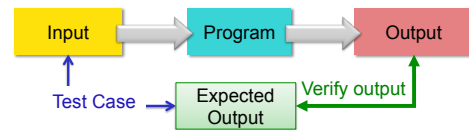
- **Test case:** **input** used to test the program, **expected output** given that input
- Verify if **output** is what you expected

Jan 17, 2011

Sprenkle - CSCI111

25

## Testing Process



- Need **good test cases** to help determine if program is correct
  - Tester plays devil's advocate
  - Want to expose **all** bugs!
  - Find before customer/professor!

Jan 17, 2011

Sprenkle - CSCI111

26

## Practice: Test Cases

- Test cases for finding the area of a rectangle

Input	Expected Output

Jan 17, 2011

Sprenkle - CSCI111

27

## Practice: Test Cases

- Test cases for finding the area of a rectangle
  - Test both integers
  - Test with at least one float for width, height
  - Test numbers less than or equal to 0
    - Shouldn't compute area for those

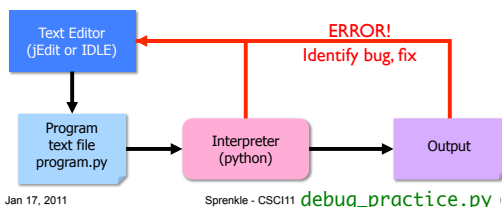
Jan 17, 2011

Sprenkle - CSCI111

28

## Debugging

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



Jan 17, 2011

Sprenkle - CSCI111 debug\_practice.py

## 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 17, 2011

Sprenkle - CSCI111

30

## This Week

- Tuesday: Lab 1
  - Bring your lecture notes and handouts!
  - Due Friday
- Wednesday: 2:05 p.m. - 2:50 p.m.
  - For ODK
  - Pushes my office hours back accordingly
- For Friday, read up to (but not including) "Themes" of Four Puzzles from Cyberspace
  - Post summary on Sakai

Jan 17, 2011

Sprenkle - CSCI111

31