Objectives

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

Recap of Programming Fundamentals

- Most important data types (for us, for now): **`int, float, string, boolean`**
  - 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 gets value y”
  - Only LHS of statement changes

Review: Assignment statements

- Assignment statements are NOT math equations!

  count = count + 1
- These are commands!

  x = 2

  y = x

  x = x + 3
  - What’s the value of y?

Review: 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
-- not initialized, can’t use on RHS

Printing Output

- **print** is a special command
  - Displays the result of expression(s) to the terminal
- print “Hello, class” ← **print** automatically adds a ‘\n’ (carriage return) after it’s printed

  string literal
- print “Your answer is”, 4*x
  - Displays same as:
    - print “Your answer is”,
    - print 4*x

Syntax: commas
Semantics: print multiple “things” in one line

Getting Input From User

- **`input`** and **`raw_input`** are functions
  - Function: A command to do something
  - Prompts user for input, gets the user’s input
  - **`input`**: to read in numbers
  - **`raw_input`**: to read in strings
- Syntax:
  - **`input(<string_prompt>)`**
  - **`raw_input(<string_prompt>)`**

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

Getting Input from User

- color = raw_input(“What is your favorite color? ”)

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 !
```

Assigns variable **`color`** that input

input_demo.py

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

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

Identify the Pieces of a Program

```
# Demonstrate numeric and string input
# by Sara Sprenkle for CS111 on 9/12/07
#

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

Identify the Pieces of a Program

```
# Demonstrate numeric and string input
# by Sara Sprenkle for CS111 on 9/12/07
#

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

Putting it all together

- Find the area of a rectangle (which has a width and height)
  - What is the algorithm for this problem?

Putting it all together

- 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

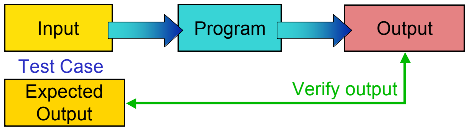
area.py

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
    - “The anser is”
  - Exceptions/Runtime errors
    - answer = 2/0
    - Undefined variable name
- Expose errors when **Testing**

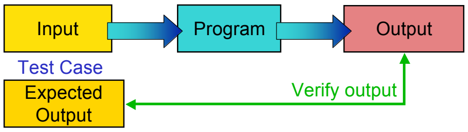
Or, perhaps the answer should be 2*3

Testing Process



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

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!

Practice: Test Cases

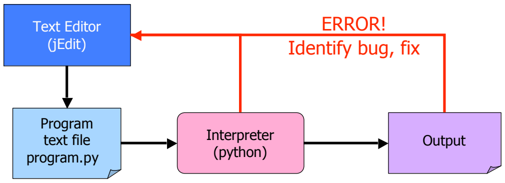
- Test cases for finding the area of a rectangle

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

Debugging

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



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

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

NOT Math Class

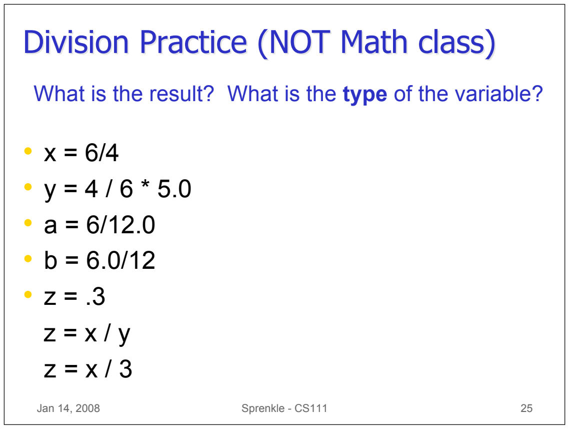
- Need to write out all operations explicitly
  - In math class, a (b+1) meant $\underbrace{a * (b+1)}_{\text{Write this way in Python}}$
- Practice:

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

Two Types of Division

- Float Division: Result is a **`float`**
  - 3.0/6.0 --> 0.5
  - 6.0/3.0 --> 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 --> 0
  - 6/3 --> 2
  - x/y, if both x and y are **`int`s**
  - If both numerator and denominator are **`int`s**, result is **`int`**

Not always obvious



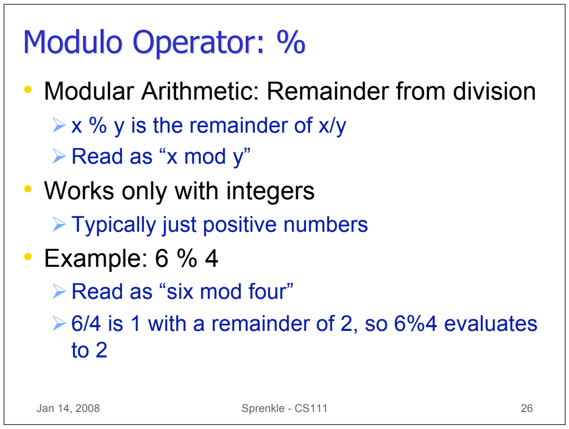
Division Practice (NOT Math class)

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

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

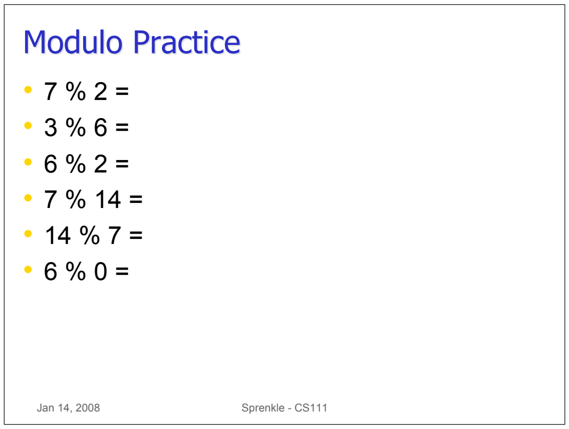
  z = x / y

  z = x / 3



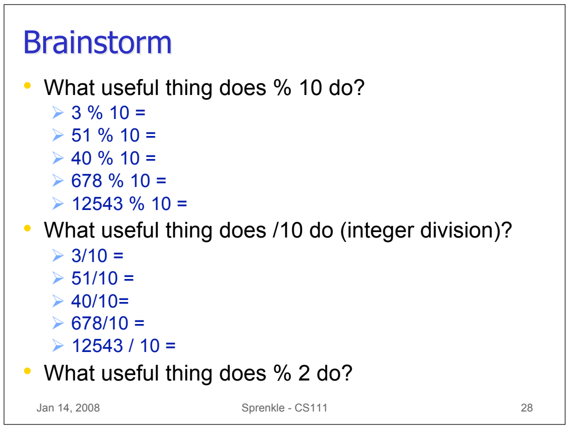
Modulo Operator: %

- Modular Arithmetic: Remainder from division
  - x % y is the remainder of x/y
  - Read as “x mod y”
- Works only with integers
  - Typically just positive numbers
- Example: 6 % 4
  - Read as “six mod four”
  - 6/4 is 1 with a remainder of 2, so 6%4 evaluates to 2



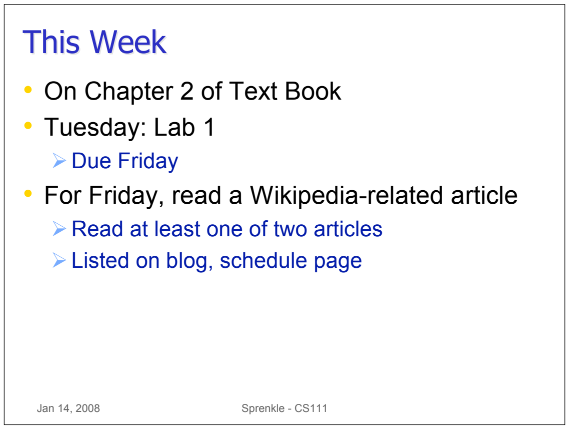
Modulo Practice

- 7 % 2 =
- 3 % 6 =
- 6 % 2 =
- 7 % 14 =
- 14 % 7 =
- 6 % 0 =



Brainstorm

- What useful thing does % 10 do?
  - 3 % 10 =
  - 51 % 10 =
  - 40 % 10 =
  - 678 % 10 =
  - 12543 % 10 =
- What useful thing does /10 do (integer division)?
  - 3/10 =
  - 51/10 =
  - 40/10=
  - 678/10 =
  - 12543 / 10 =
- What useful thing does % 2 do?



This Week

- On Chapter 2 of Text Book
- Tuesday: Lab 1
  - Due Friday
- For Friday, read a Wikipedia-related article
  - Read at least one of two articles
  - Listed on blog, schedule page