Objectives

- More arithmetic in Python
- Format specifiers
- Function calls
- Importing modules

Documenting Your Code

- Use English to describe what your program is doing in **comments**
  Pound sign
  - 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

Good Development Practices

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

Review: Numeric Primitive Types

Data Type	Description	Examples
int	Plain integers (32-bit precision)	-214, -2, 0, 2, 100 Range: -2^{31} to $2^{31}-1$
float	Real numbers	.001, -1.234, 1000.1, 0.00, 2.45
long	Bigger integers (neg or pos, precision limited by computer memory)	2147483648L
complex	Imaginary numbers (have real and imaginary part)	1j * 1J --> (-1+0j)

Review: Assignment statements

- Assignment statements are NOT math equations!

  count = count + 1
- These are commands!

  x = 2

  y = x

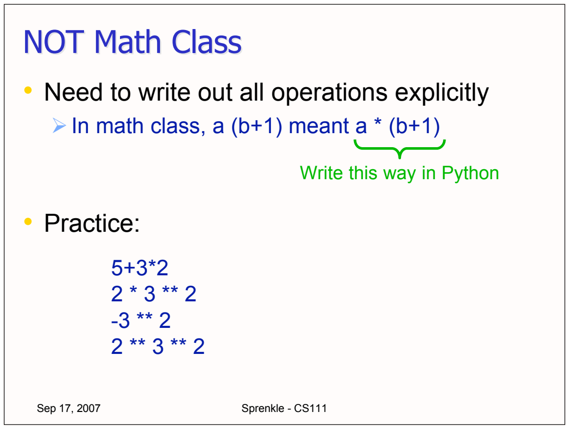
  x = x + 3
  - What's the value of y?

Review: Arithmetic Operations

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

Usual precedence rules (P E - DM% AS)

negation



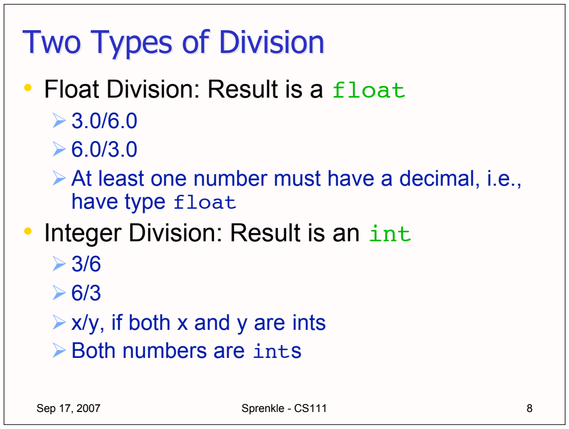
NOT Math Class

- Need to write out all operations explicitly
  - In math class, a (b+1) meant $\underbrace{\text{a * (b+1)}}_{}$

Write this way in Python

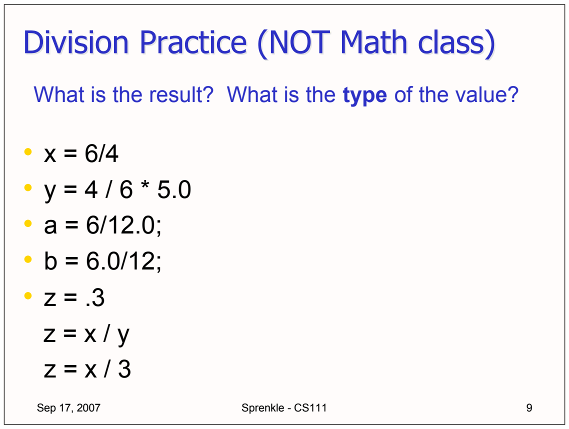
- Practice:

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



Two Types of Division

- Float Division: Result is a `float`
  - 3.0/6.0
  - 6.0/3.0
  - At least one number must have a decimal, i.e., have type `float`
- Integer Division: Result is an `int`
  - 3/6
  - 6/3
  - x/y, if both x and y are ints
  - Both numbers are `ints`



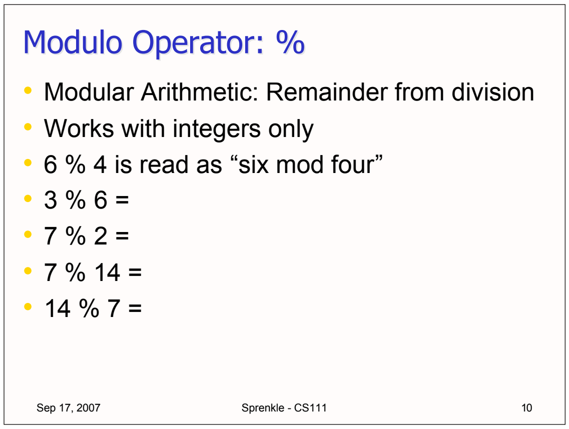
Division Practice (NOT Math class)

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

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

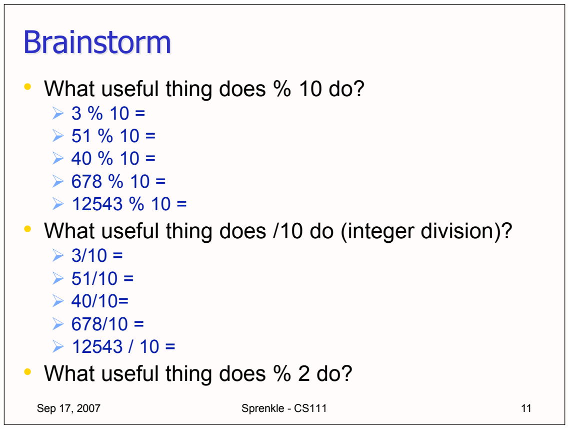
  z = x / y

  z = x / 3



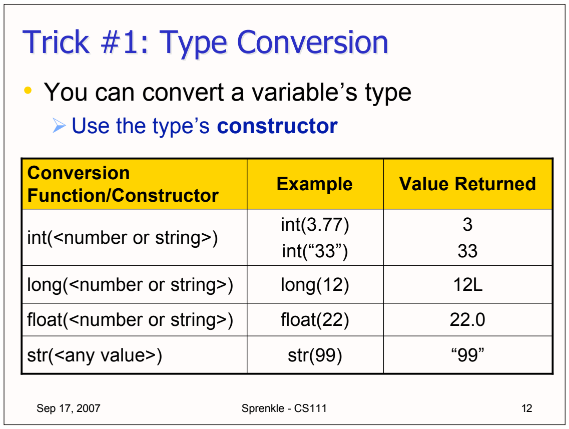
Modulo Operator: %

- Modular Arithmetic: Remainder from division
- Works with integers only
- 6 % 4 is read as “six mod four”
- 3 % 6 =
- 7 % 2 =
- 7 % 14 =
- 14 % 7 =



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?



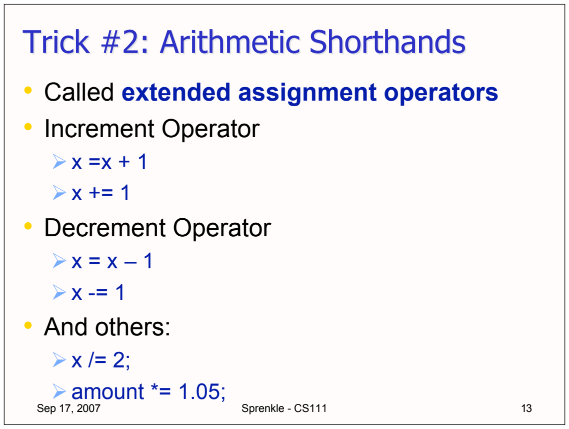
Trick #1: Type Conversion

- You can convert a variable’s type
  - Use the type’s **constructor**

Conversion Function/Constructor	Example	Value Returned

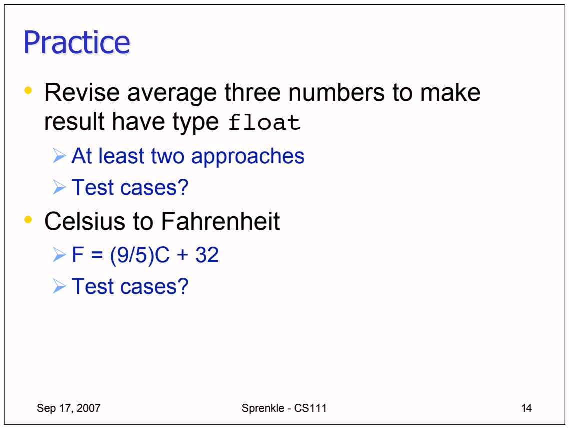
int(<number or string>)	int(3.77)
	int(“33”)
		3
		33

long(<number or string>)	long(12)	12L
float(<number or string>)	float(22)	22.0
str(<any value>)	str(99)	“99”



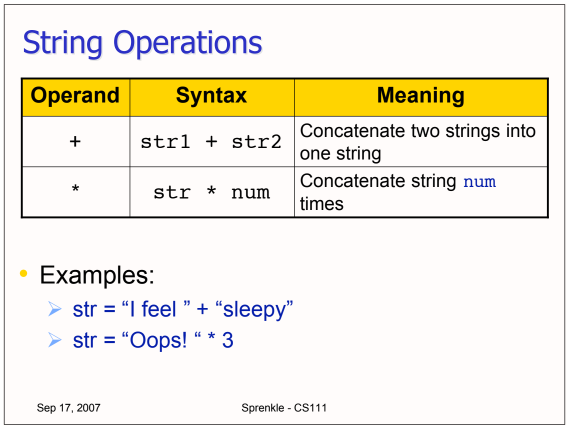
Trick #2: Arithmetic Shorthands

- Called **extended assignment operators**
- Increment Operator
  - x =x + 1
  - x += 1
- Decrement Operator
  - x = x – 1
  - x -= 1
- And others:
  - x /= 2;
  - amount *= 1.05;



Practice

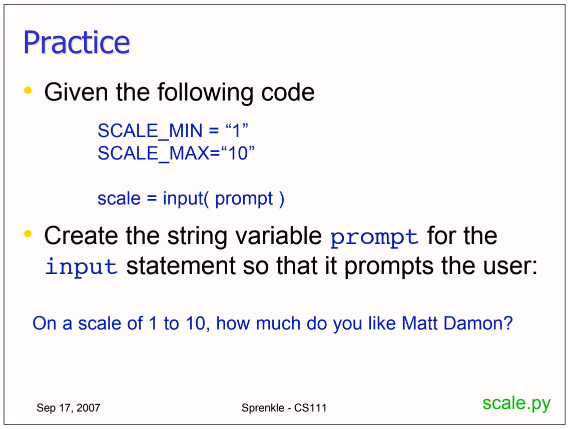
- Revise average three numbers to make result have type `float`
  - At least two approaches
  - Test cases?
- Celsius to Fahrenheit
  - F = (9/5)C + 32
  - Test cases?



String Operations

Operand	Syntax	Meaning
+	`str1 + str2`	Concatenate two strings into one string
*	`str * num`	Concatenate string `num` times

- Examples:
  - str = “I feel ” + “sleepy”
  - str = “Oops! “ * 3



Practice

- Given the following code

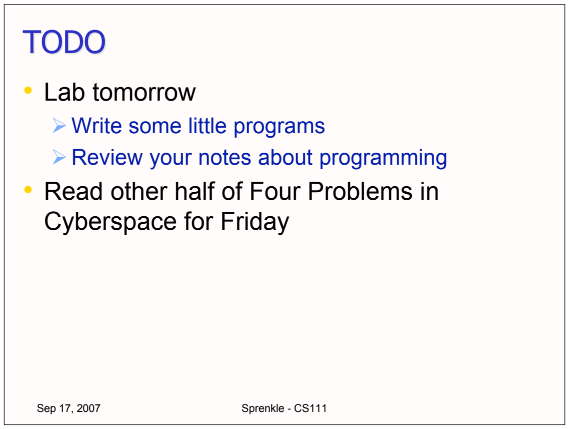
```
SCALE_MIN = “1”
SCALE_MAX=“10”

scale = input( prompt )
```

- Create the string variable `prompt` for the `input` statement so that it prompts the user:

On a scale of 1 to 10, how much do you like Matt Damon?

scale.py



TODO

- Lab tomorrow
  - Write some little programs
  - Review your notes about programming
- Read other half of Four Problems in Cyberspace for Friday