Objectives

- A little more on arithmetic operators
- Input

- Get handouts from last time
- New handouts in Canvas

Computer Science Student Assistants

Day of the Week	Students
Tuesday	Coletta *Laurie
Wednesday	Abdel Sam
Thursday	Sam (6-8 p.m.) *Laurie (7-9 p.m.)

`https://wlu.box.com/v/CSAssistantZoomLinks`

Laurie had my version of CSCI111

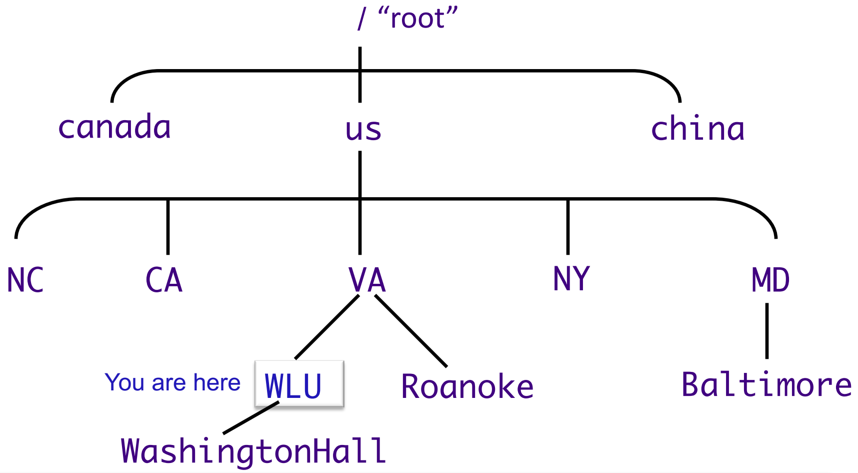
Review

- Linux review:
  - What is the command to …
    - Find out where you are in the file system?
    - Create a directory?
    - Go into a directory?
    - View the contents of a directory?
    - Copy a file?
  - How do you refer to … your home directory? The current directory? The parent directory?
- What is our development process?
  - Programming, in general
  - For lab work
- What are the two division operators?

Lab Retrospective

- Learning how to solve problems
  - Every week: new problems, new techniques to solve problems
- I am explicit in directions/reminders early
  - Then stop reminding because you should know the process later

Relative Paths vs Absolute Paths



- Given that you’re at WLU, how would you get to Washington Hall? To Roanoke? To Baltimore?

Review: Two Division Operators

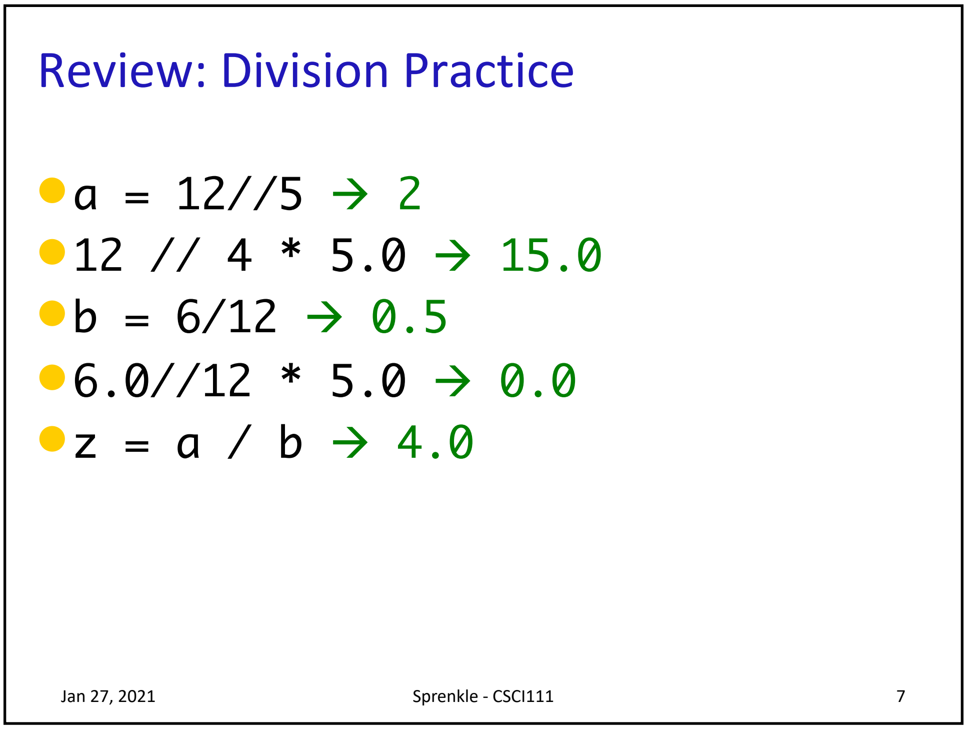
/ Float Division

- Result is a **float**
- Examples:
  - 6/3 → 2.0
  - 10/3 → 3.3333333333333335
  - 3.0/6.0 → 0.5
  - 19/10 → 1.9

// Integer Division

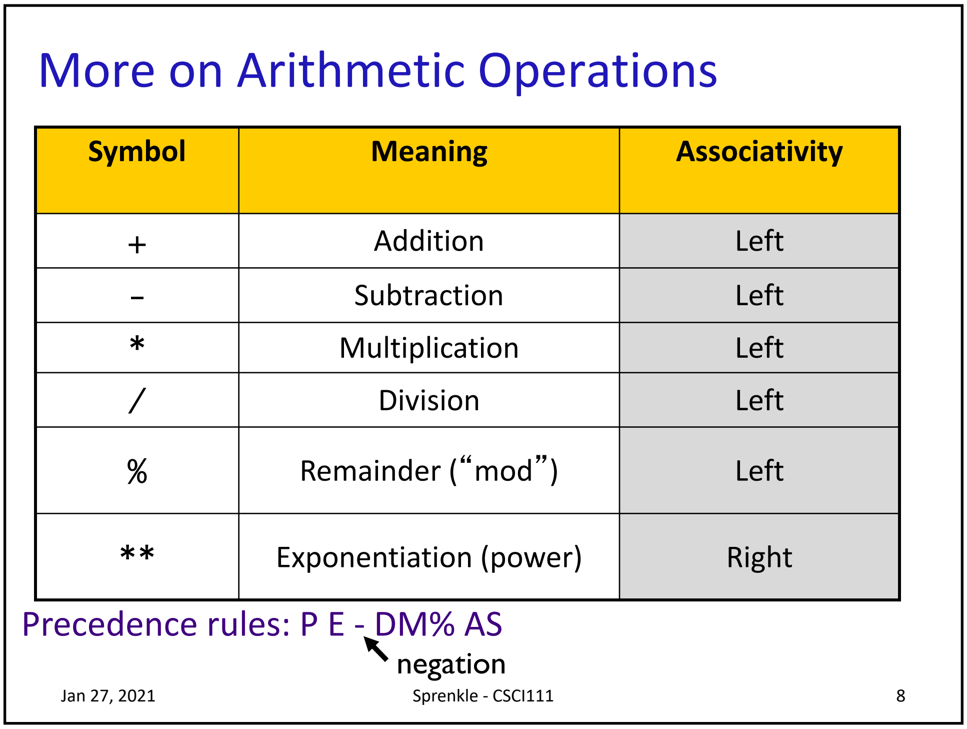
- Result is an **int**
- Examples:
  - 6//3 → 2
  - 10//3 → 3
  - 3.0//6.0 → 0.0
  - 19//10 → 1

Integer division is the default division used in many programming languages



Review: Division Practice

- a = 12//5 → 2
- 12 // 4 * 5.0 → 15.0
- b = 6/12 → 0.5
- 6.0//12 * 5.0 → 0.0
- z = a / b → 4.0



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

More on Arithmetic Operations

Symbol	Meaning	Associativity
+	Addition	Left
–	Subtraction	Left
*	Multiplication	Left
/	Division	Left
%	Remainder ("mod")	Left
**	Exponentiation (powe	

Precedence rules: P E - DM% AS

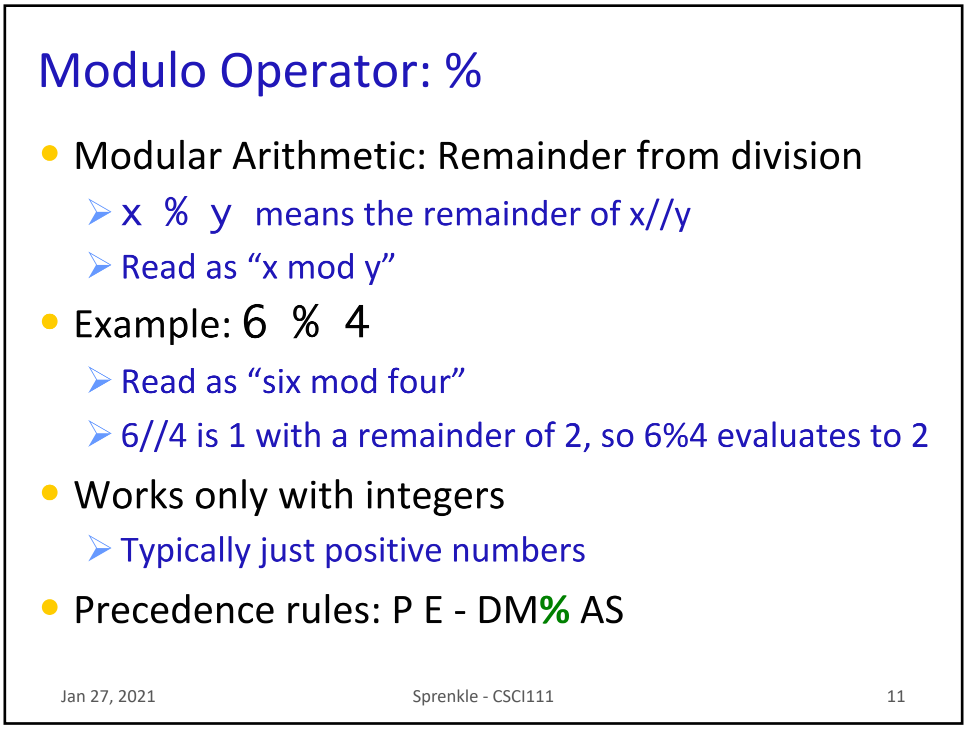
negation

Associativity matters when you have the same operation multiple times. It tells you where you should start computing.

Python Math Practice

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

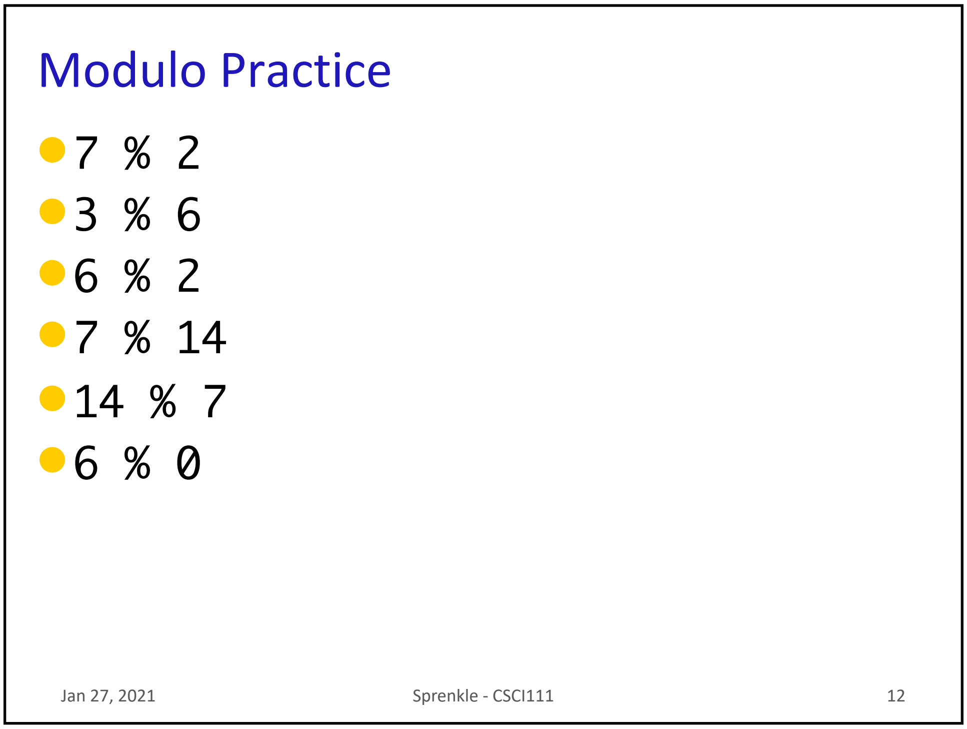
How should we verify our answers?



Modulo Operator: %

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

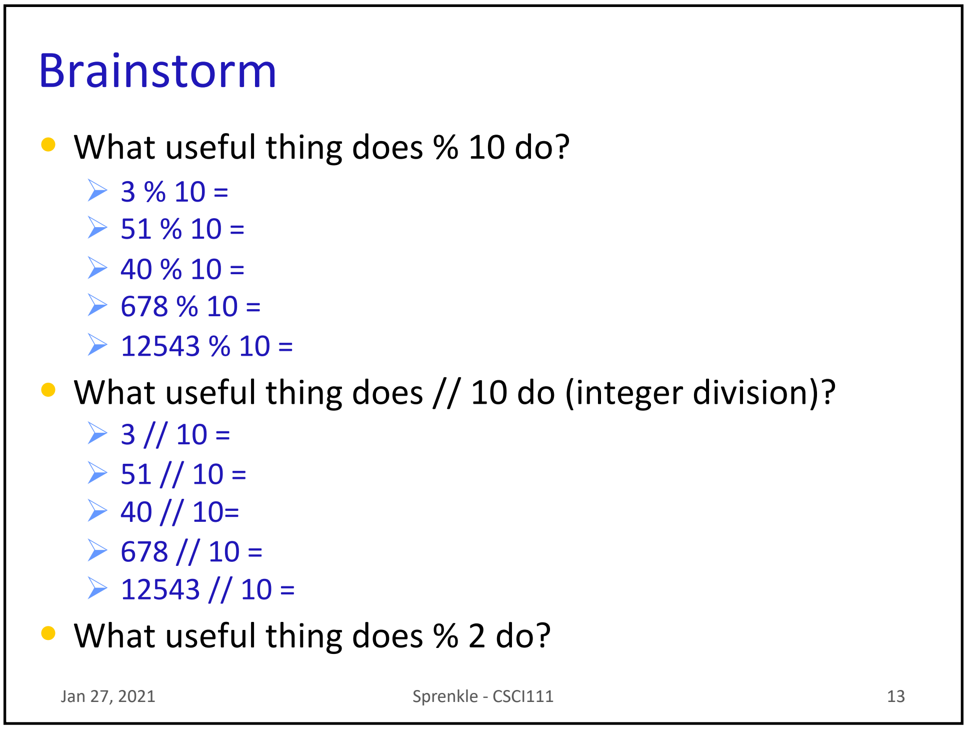
11



Modulo Practice

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

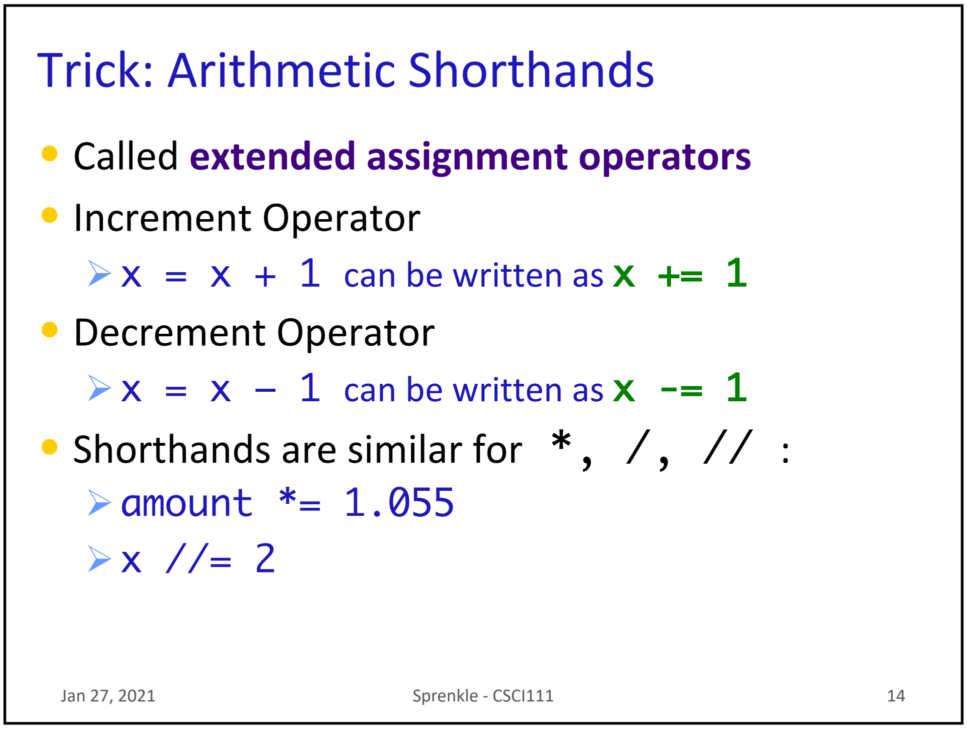
12



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?

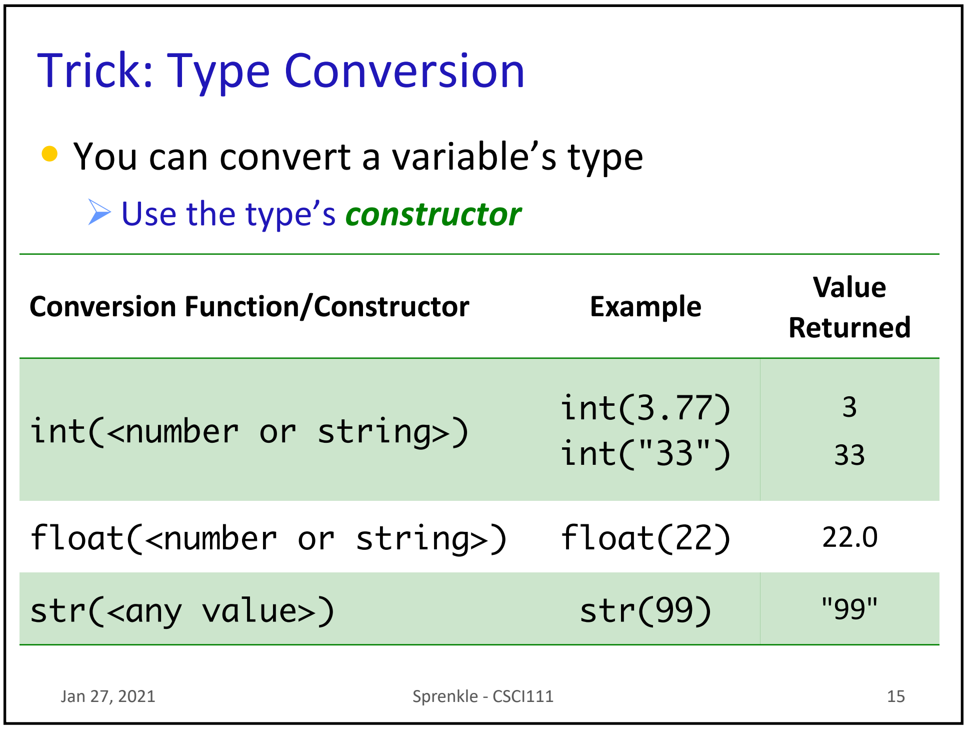
13



Trick: Arithmetic Shorthands

- Called **extended assignment operators**
- Increment Operator
  - `x = x + 1` can be written as **`x += 1`**
- Decrement Operator
  - `x = x – 1` can be written as **`x -= 1`**
- Shorthands are similar for `*, /, //` :
  - `amount *= 1.055`
  - `x //= 2`

14



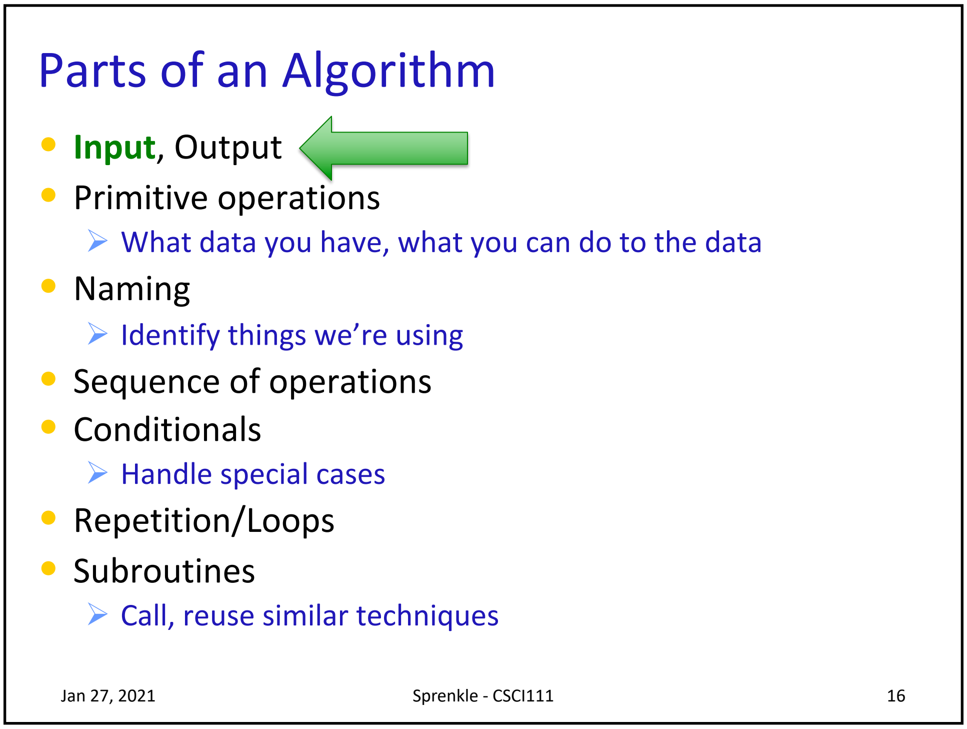
Trick: Type Conversion

- 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

`float(<number or string>)`	`float(22)`	22.0
`str(<any value>)`	`str(99)`	"99"



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

Interactive Programs

- Meaningful programs often need input from users
- Demo: `input_demo.py`

Getting Input From User

- `input` is a ***function***
  - **Function**: A command to do something
    - A “subroutine”
- Syntax:
  - `input(<string_prompt>)`
- Semantics:
  - Display the prompt `<string_prompt>` in the terminal
  - Read in the user’s input and *return* it as a string/text

Getting Input From User

- Typically used in assignments
- Examples:

Prompt displayed to user

```
name=input("What is your name? ")
```

- `name` is assigned the string the user enters

```
width=eval(input("Enter the width:"))
```

- **What the user enters is evaluated (as a number) and assigned to `width`**
- Use `eval` function because expect a number from user

Getting Input from User

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

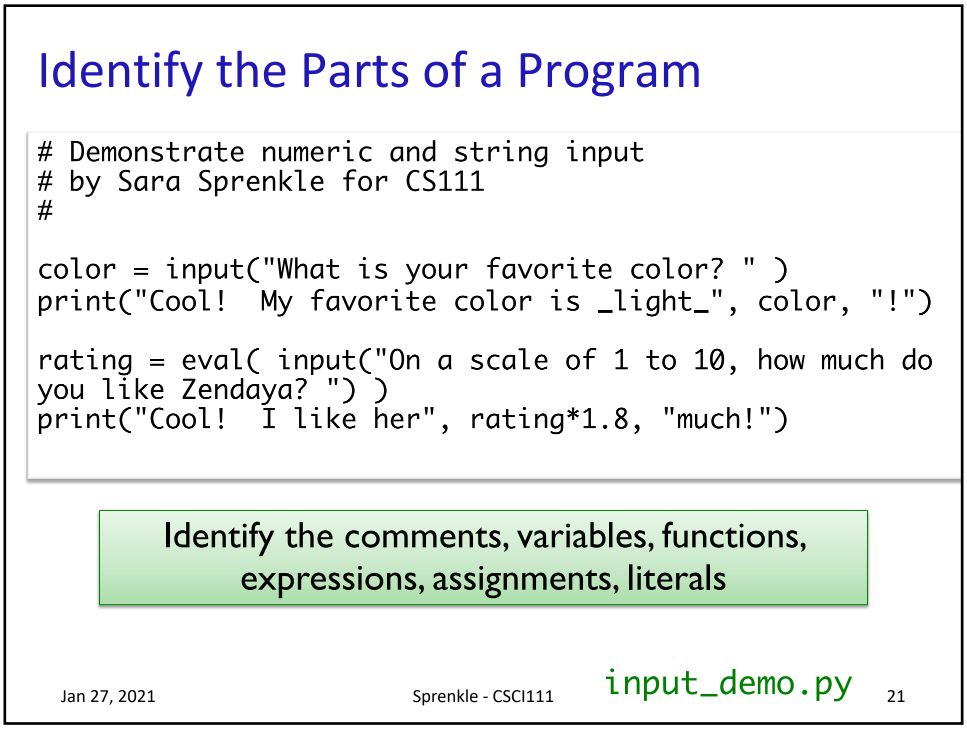
Semantics: Sets the variable `color` to the user's input

Terminal:

Grabs every character up to the user presses "enter"

```
> python3 input_demo.py
What is your favorite color? blue
Cool!  My favorite color is _light_ blue !
```

input_demo.py



Identify the Parts of a Program

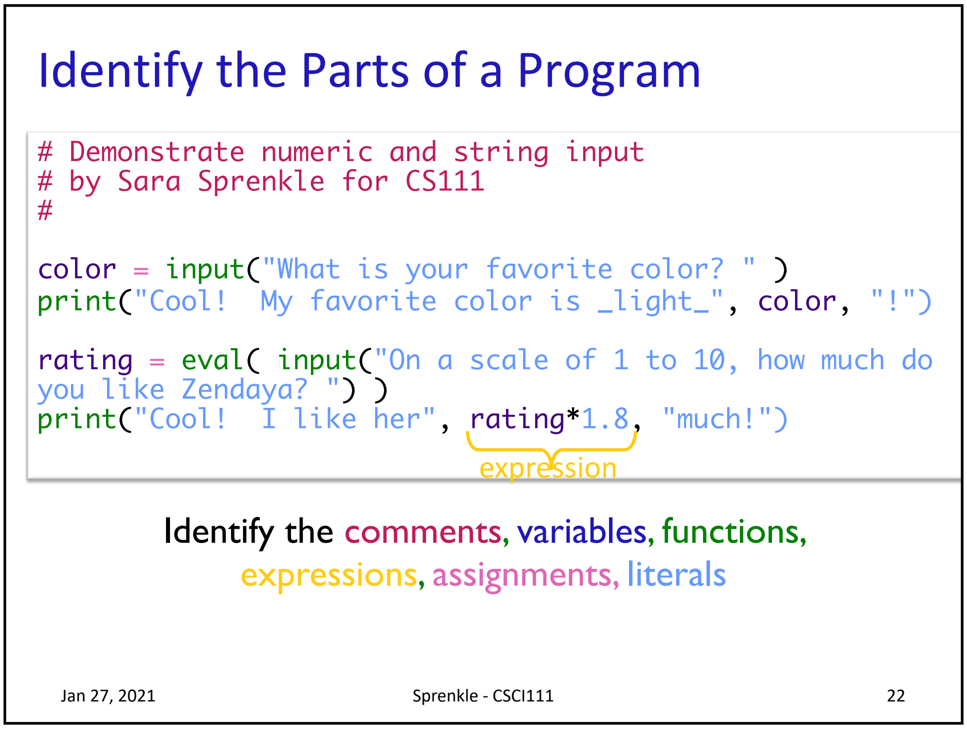
```
# Demonstrate numeric and string input
# by Sara Sprenkle for CS111
#

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

rating = eval( input("On a scale of 1 to 10, how much do
you like Zendaya? ") )
print("Cool!  I like her", rating*1.8, "much!")
```

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

input_demo.py



Identify the Parts of a Program

```
# Demonstrate numeric and string input
# by Sara Sprenkle for CS111
#

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

rating = eval( input("On a scale of 1 to 10, how much do
you like Zendaya? ") )
print("Cool!  I like her", rating*1.8, "much!")
```

expression

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

Improving `average2.py`

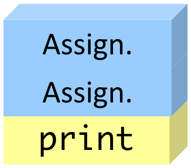
- With what we just learned, how could we improve `average2.py`?
- Example of suggested approach to development
  - Input is going to become fairly routine.
  - Wait on input until you have figured out the rest of the program/problem.

Design Patterns

- General, repeatable solution to a commonly occurring problem in software design
  - Template for solution

Design Patterns

- General, repeatable solution to a commonly occurring problem in software design
  - Template for solution
- Example (Standard Algorithm)
  - Get input from user
  - Do some computation
  - Display output



```
x = input("…")
ans = …
print(ans)
```

Looking Ahead

- Lab 1 due Friday
- Broader Issue write up due Friday
  - TED talk