

Objectives

- More arithmetic operators
- Software development practices
 - Testing
 - Debugging
 - Iteration

Office hours:

- Today, 2:30-2:55, 5-5:50 p.m.
- Thursday: 1 – 5 p.m.

Review

- What are the two ways we can use Python?
- What are the commands we use to be able to use Python in those ways?
- What is our development process?

Review: 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

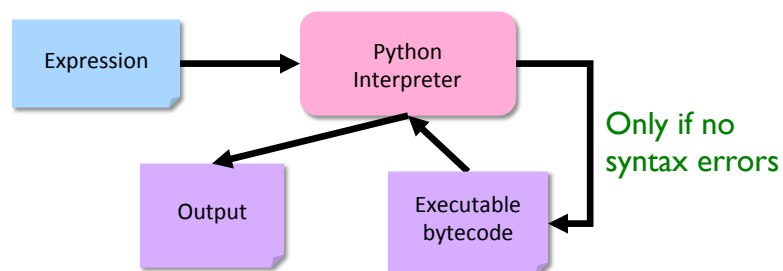
Jan 17, 2018

Sprenkle - CSCI111

3

Review: Python Interpreter

1. Validates Python programming language expression(s)
 - Enforces Python syntax rules
 - Reports syntax errors
2. Executes expression(s) ← Have a lot of these early on!



Jan 17, 2018

Sprenkle - CSCI111

4

Review: Two Modes to Execute Python Code

- **Interactive/Shell:** using the *interpreter*
 - Try out Python expressions
- **Batch:** execute *scripts* (i.e., files containing Python code)
 - What we'll write usually

Jan 17, 2018

Sprenkle - CSCI111

5

Review: Formalizing Process of Developing Computational Solutions


1. Create a sketch of how to solve the problem (the algorithm)
2. Fill in the details in Python
3. Test the Python program with *good* test cases
 - a. If errors found, debug program
 - b. Repeat step 3

Jan 17, 2018

Sprenkle - CSCI111

6

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, 2018

Sprenkle - CSCI111

7

Two Division Operators

/ Float Division

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

// Integer Division

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

Integer division is the default division used in most programming languages

Jan 17, 2018

Sprenkle - CSCI111

8

Division Practice

- $a = 12 // 4$
- $12 // 4 * 5.0$
- $b = 6 / 12$
- $6.0 // 12 * 5.0$
- $z = a / b$

Jan 17, 2018

Sprenkle - CSCI111

9

More on Arithmetic Operations

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

Precedence rules: P E - DM% AS

negation CSCI111

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

Jan 17, 2018

Math Practice

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

How should we verify our answers?

Jan 17, 2018

Sprenkle - CSCI111

11

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

Jan 17, 2018

Sprenkle - CSCI111

12

Modulo Practice

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

Jan 17, 2018

Sprenkle - CSCI111

13

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?

Jan 17, 2018

Sprenkle - CSCI111

14

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`

Jan 17, 2018

Sprenkle - CSCI111

15

Trick: Type Conversion

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


Conversion Function/Constructor	Example	Value Returned
<code>int(<number or string>)</code>	<code>int(3.77)</code>	3
	<code>int("33")</code>	33
<code>float(<number or string>)</code>	<code>float(22)</code>	22.0
<code>str(<any value>)</code>	<code>str(99)</code>	"99"

Jan 17, 2018

Sprenkle - CSCI111

16

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, 2018

Sprenkle - CSCI111

17

Interactive Programs

2.8 in Text Book

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

Jan 17, 2018

Sprenkle - CSCI111

18

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

Jan 17, 2018

Sprenkle - CSCI111

19

Getting Input From User

- Typically used in assignments
- Examples:
 - **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

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

Jan 17, 2018

Sprenkle - CSCI111

20

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 !
```

Jan 17, 2018

Sprenkle - CSCI111

input_demo.py

21

Restricting User's Inputs

```
>>> x = 7
>>> yourVal = input("My val is: ")
My val is: x
>>> print(yourVal)
x
```

Jan 17, 2018

Sprenkle - CSCI111

22

Restricting User's Inputs

```
>>> x = 7
>>> yourVal = input("My val is: ")
My val is: x
>>> print(yourVal)
x
>>> yourVal = eval(input("My val is: "))
My val is: x
>>> print(yourVal)  What happened here?
7
>>> yourVal = int(input("My val is: "))
My val is: x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10:
'x'
```

Jan 17, 2018

Sprenkle - CSCI111

23

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 Ryan Gosling? "))
print("Cool! I like him", rating*1.8, "much!")
```

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

Jan 17, 2018

Sprenkle - CSCI111

input_demo.py

24

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 Ryan Gosling? "))
print("Cool! I like him", rating*1.8, "much!")
                        expression
```

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

Jan 17, 2018

Sprenkle - CSCI111

25

Looking Ahead

- Lab 1 due Friday
- Broader Issue due Friday

Office hours:

- Today, 2:30-2:55, 5-5:50 p.m.
- Thursday: 1 – 5 p.m.

Jan 17, 2018

Sprenkle - CSCI111

26