Admin

• **Pre-term survey**: reminder to complete pre-term survey by 11:59pm tonight for bonus to course grade
  https://survey.ubc.ca/s/cpsc301_presurvey_2016w2/
  (link also pinned on Piazza)

• **TA office hours**: Starting Wednesday
  – Schedule will be posted to Piazza

• **Waitlist**: Everyone who signed in Thursday has been added
  – come talk to JD after class if you’re still waiting to join

• **Lab 2**: Due Sunday @ 9am
  – This is the usual due date / time

• **Lab 3**: will be up by Thursday evening
Objectives

At the end of this lecture you should be able to:

• trace Python code that involves numerical and string operations
• trace Python code that uses numerical and string variables
• draw the state of memory after a series of variable assignment operations
• identify errors in code related to numerical and string operations
• write simple code involving these operations
• write code that reads input from the user, perform simple calculations and displays results to the user
Scratch vs Python

• Python is a general programming language, different than Scratch in many ways

• Scratch:
  – Multiple sprites, each with multiple scripts
  – Build scripts by plugging together blocks from a visual library
  – Blocks cause sprites to do things: move, change costume, play sounds, draw lines, etc.
  – Inputs from mouse, keyboard, microphone
  – Outputs to screen, speaker

• Python
  – One program doing one thing at a time
  – Build program by writing commands
  – Commands read input, change variables, write output
  – Inputs from keyboard, files
  – Outputs to screen, files
Getting Started in Python

• Technically, “Python” refers to the basic language
  – However, many of the most interesting features require the use of separate packages
  – A “Python distribution” includes not just the basic language but also a collection of common packages and often a simplified installer for the whole thing

• To practice Python at home, download and install the Anaconda distribution:
  
  http://continuum.io/downloads
  
  – Choose the Graphical Installer for Python 3.5 (NOT 2.7) for your operating system.

• We will not be using exactly the same Python as the book
  – We will be using the Spyder Integrated Development Environment (IDE) instead of IDLE
Spyder Integrated Development Environment (IDE)

- One of many ways of working with & running Python code

Area for editing your Python programs: the “text editor” or “editor”

Area for running Python code or watching the output of a running program: the “console” (we suggest you make it bigger)

Area for looking at variable values or documentation
Python Misbehaving?

• What should you do if you are typing commands into Python at the >>> prompt and the responses are not what you expect:

  1) Check what you typed. A single mistyped character can cause an entirely different response.

  2) Restart Python. You can do this by selecting “open a Python Console” from either the Consoles menu (at the top of the Spyder window) or from the small white dropdown button on the top right edge of your console window (if you have not closed the console window). See the next slide.

  3) Check what you typed again.

  4) Restart Spyder.
Starting a new Python Console

Choose “open a Python console” from either of these menus

The new window will normally appear here as a tab called “Python #” (where “#” is an integer)
We will not use IPython

- Spyder offers the options of opening an “IPython console”
  - IPython is another way of interacting with Python which offers some powerful features
  - We will stick with regular Python consoles because that is much closer to the textbook (e.g. the IDLE platform)
  - If you use Python interactively (e.g. not just running other people’s scripts) in the future, you should strongly consider using IPython

Pay attention to the names on these tabs
How to Get Rid of IPython Consoles

• An IPython console may create up to two windows:
  – “Kernel ####” in the Console tab, where the regular Python consoles are open. (only appears in some versions of Spyder, including on lab computers)
  – “Console ####/A” in the IPython tab

• To get rid of these windows (may vary depending on Spyder version):
  – close the IPython console window window (or type “exit” in it) and then when asked “Do you want to kill the associated kernel and all of its clients” click “yes”
  – From the spyder menu bar select ‘View’ > ‘Panes’ and then deselect ‘IPython Console’

Do not close this window (it will close by itself)

Close this window
Online Python Tutor

- [http://pythontutor.com/visualize.html](http://pythontutor.com/visualize.html)
- Useful for visualizing Python execution
  - You can step back and forth through the code, watching how data changes in memory
Two types of numbers:
  – integers (called *int*): 0, 5, -20
  – decimal numbers (*float*): 2.5, 0.25555, 5.0, 0.0

The following arithmetic operations are defined for both types:
  – +, -, *: addition, subtraction and multiplication
  – /: division; always returns a float
    • i.e. 6 / 3 is 2.0
  – //: integer division; always returns an int (but not if you use a float)
    • i.e. 14 // 5 is 2 (14.0 // 5 is 2.0)
  – %: remainder (or modulo) operator
    • i.e. 14 % 5 is 4
  – **: exponentiation
    • i.e. 2 ** 6 is 64

The result of an operation is of the same type as the operands, if they are of same type. Otherwise the int operand is converted to float. I.e.
  – 5 + 2 is 7, but 5.0 + 2 is 7.0
  – 5 //2 is 2, but 5.0 // 2 is 2.0
Numeric Values and Operations (cont)

• Floats have finite precision:
  – Equality on floats is tricky
  – 2 / 3 = 0.6666666666666666
  – 5 / 3 = 1.6666666666666667
  – 1 + 2 / 3 is not the same as 5 / 3

• Operators are performed from left to right with this precedence:
  – **
  – - (negation)
  – *, /, //, %
  – +, -
  – For instance: 10 - 2 * 5 + 6 / 2 is ____

• Can use () to force different order.
  – I.e. 10 - 2 * (5 + 6) / 2 is ____
Variables

- Variables are names that have values we like to keep and use later (like Scratch variables)
- Variable names may only have letters, digits and underscore(_) and the first character must be a letter or _
- A variable is created (declared) using an assignment statement which assigns a value to a variable using the assignment operator "="
  
  ```
  speed = 4
  time = 2.5
  ```

- A variable’s value is stored in a memory location, and the variable refers to the appropriate memory location by its address
- When we trace Python code, to keep track of the variables and their values we use the following notation (memory model):
  
  ```
  speed → 4
  time → 2.5
  ```

- After created, when a variable is used Python uses its value
  
  ```
  distance = speed * time
  time = time + 2
  distance = speed * time
  ```

Assignment statements are executed as following:
1. evaluate the expression on the right of =
2. assign the value to the variable on the left of =
Combined Operators (Augmented Assignment)

- Notice the difference between the following operations:
  
  ```
  time = 20
  time = time + 20
  ```

- Operations like the last one can be combined as following
  - `time += 20` means `time = time + 20`
  - `time -= 20` means `time = time - 20`
  - `time *= 20` means `time = time * 20`
  - `time /= 20` means `time = time / 20`

- Evaluation of these expressions:
  - first evaluate the expression on the right of `=`
  - apply the operation (attached to `=`) to the value of the variable on the left of `=`
  - assign the result to variable on the left of `=`

- Suppose distance, time and speed are as shown on the box to the right. Show their new values after executing
  ```
  distance += speed * time
  ```

```plaintext
speed ➔ 8
time ➔ 10
distance ➔ 20
```
Change log from pre-class slides

• Added additional learning goal regarding being able to draw state of memory after a series of variable assignments