The Parts of a Program
Today’s plan

• Review Wednesday’s class

• Four things we see in a typical program:
  - Getting input
  - Computation
  - Displaying output
  - Repeating the process

• Basic input and output with Python
But first

- Don’t be afraid to call me out!

```python
>>> len("swarthmore")
10
```
Review

- Data types: int, float, string
- Operators: +, -, *, /, %
  - Behave differently for different types
  - Promotion
  - Integer division rounds down
Review

- Conversion functions: int(), float(), str()

- Building up more complex expressions
  - Combine operators with functions and parentheses
  - Python reduces or evaluates expressions until they’re devoid of variables, functions, operators, parentheses: 17, -3.4, “hello”
Review: Assignment

- Variable name goes to the left of =
  - Variables are case-sensitive, can include letters, numbers, and the underscore, _
  - Variables can’t start with a number
  - Strings go in quotes, variables do not

- An expression goes to the right of =
  - Python evaluates it, associates it with the variable
Review: Assignment

• We can later re-assign the variable to a new value

• We can’t use a variable in an expression before it has been assigned a value
Program pieces

- **Input**
  - Keyboard, mouse, touch screen, voice

- **Computation**
  - Retrieving data, doing math, algorithms

- **Output**
  - Text, graphics, sounds, other forms of feedback

- **Repeat**
Example: Google search

- **Input**
  - User types search query

- **Computation**
  - Algorithm: correct typos, find relevant websites, rank them

- **Output**
  - Show top results on the screen

- **Repeat**
  - Do the same thing again on next query
Example: Instagram app

• Input
  - User scrolls down / taps ‘like’ button

• Computation
  - Algorithm: retrieve next photos / update number of likes

• Output
  - Show new photos / updated number of ‘likes’

• Repeat
  - Continue responding to scrolls and taps
Our programs

• Input
  • `raw_input()` function

• Computation
  • `+, -, *, /, %, len, int, str, float`

• Output
  • `print()` function

• Repeat
  • For now, just run the program again
Let’s write programs

• Waitlist students: codeskulptor.org
Python shell vs. Python program

- The Python shell lets you try lines of code one at a time and see their effect.

```bash
$ python
>>> message = "Hello, world"
>>> print(message)
Hello, world
>>> quit()
$
```
Python shell vs. Python program

• With a Python program you can put multiple instructions together

```
$ vim mycalculation.py
[Edit file in vim]
$ cat mycalculation.py
# This is a comment
product1 = 234 * 345
product2 = 87 * 33
total = product1 + product2
print("The result of my calculation is: " + str(total))
$ python mycalculation.py
The result of my calculation is: 83601
$`

Functions

• Expanded notion of a function:
  • Takes zero or more **arguments**
  • Possibly has **side effects**: printing, getting user input, etc.
  • Possibly produces a **return** value
len function

- One argument: any string (or expression that evaluates to a string)
- Side effects: none
- Returns: number of individual characters in that string

```
length_of_c = len(c)
```
raw_input function

- One argument: a string containing the prompt
- Side effect: stops the program to wait for user input
- Returns: a string containing what the user typed

```python
name = raw_input("Enter your name: ")
```
print function

• One argument: the string you want to print
• Side effect: displays the string (without quotes)
• Returns: nothing

```python
print("Hello, " + name)
```
Have a nice weekend!