+
Warm Up

www.pollev.com/python002

Announcements

- Office hours starts this week!
- First in-class workshop will be on 7/14
- Say your name when you answer questions in class!

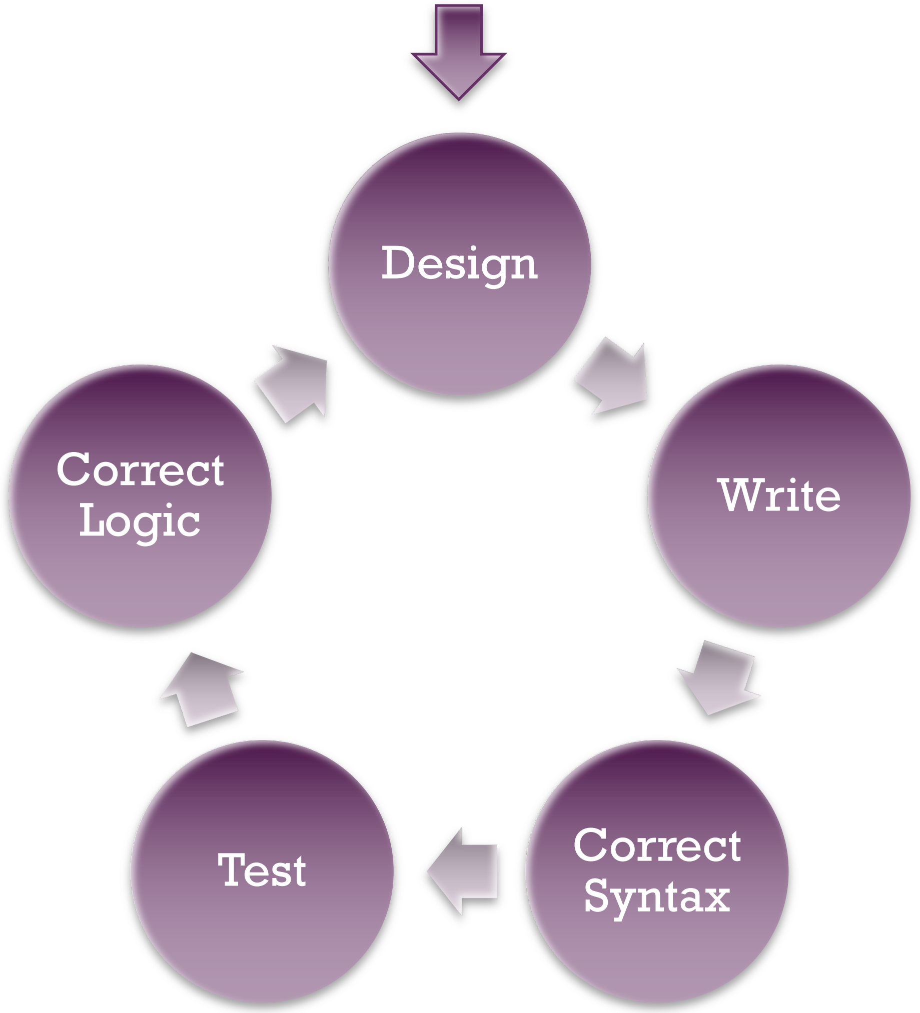
Software Development, Variables, Working with User Input, Math Operators & Data Types

CSCI-UA.002 - 006

+

Software Development

Software Development Loop



Design

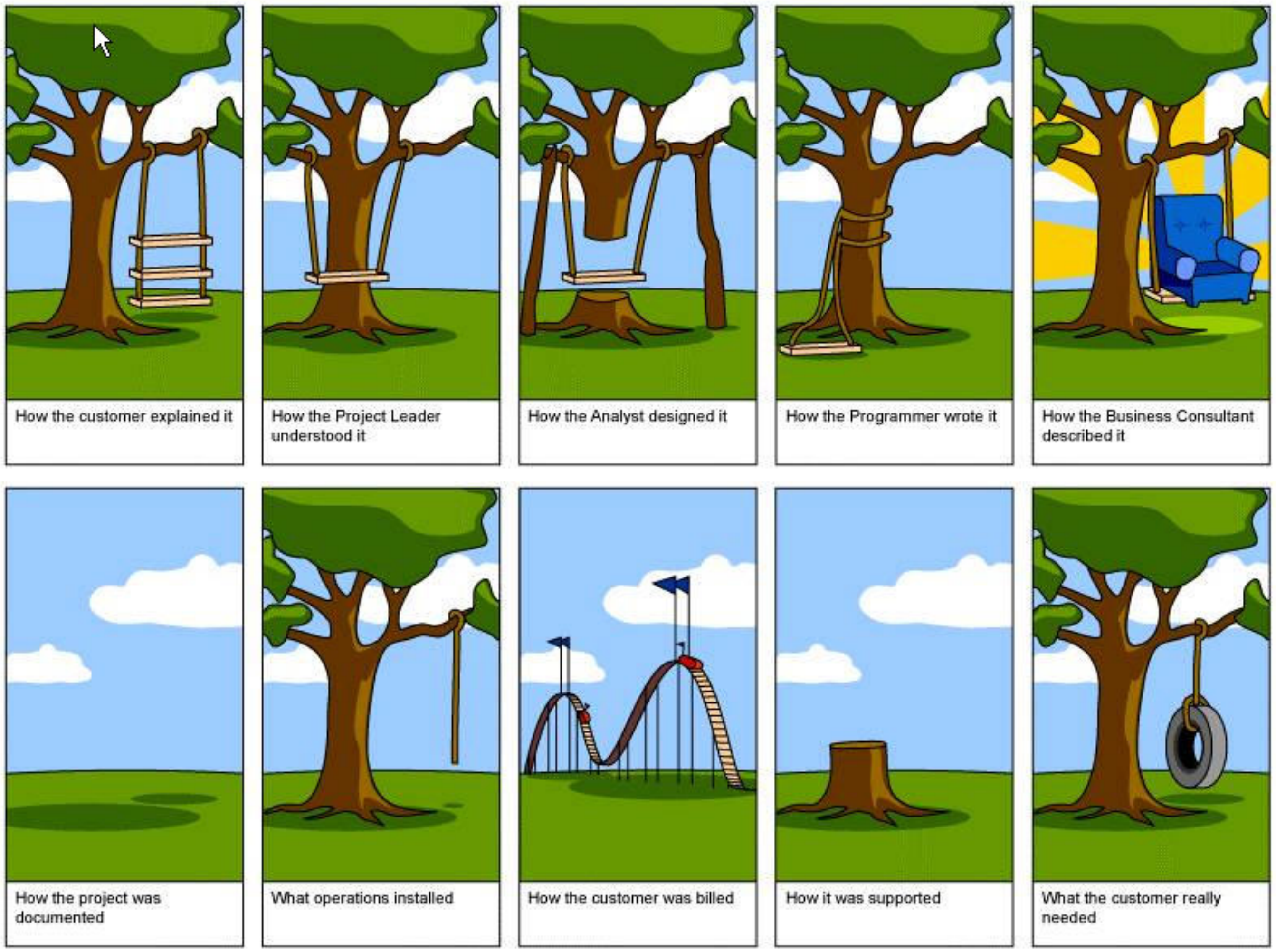
- Programmers need to establish a solid foundation before they begin coding a project
- This involves understanding the task that the program must perform
- Next, we need to determine the steps that need to be taken in order to perform the task

+

Understand the Task

- Most programming projects begin with an interview with the end user
- Programmers must ask lots of questions and get as many details as possible about the task
- Follow ups are usually required
- After an interview a programmer generally constructs a “software requirement” document
- This amounts to an agreement between the end user and the programmer on what the program should actually do

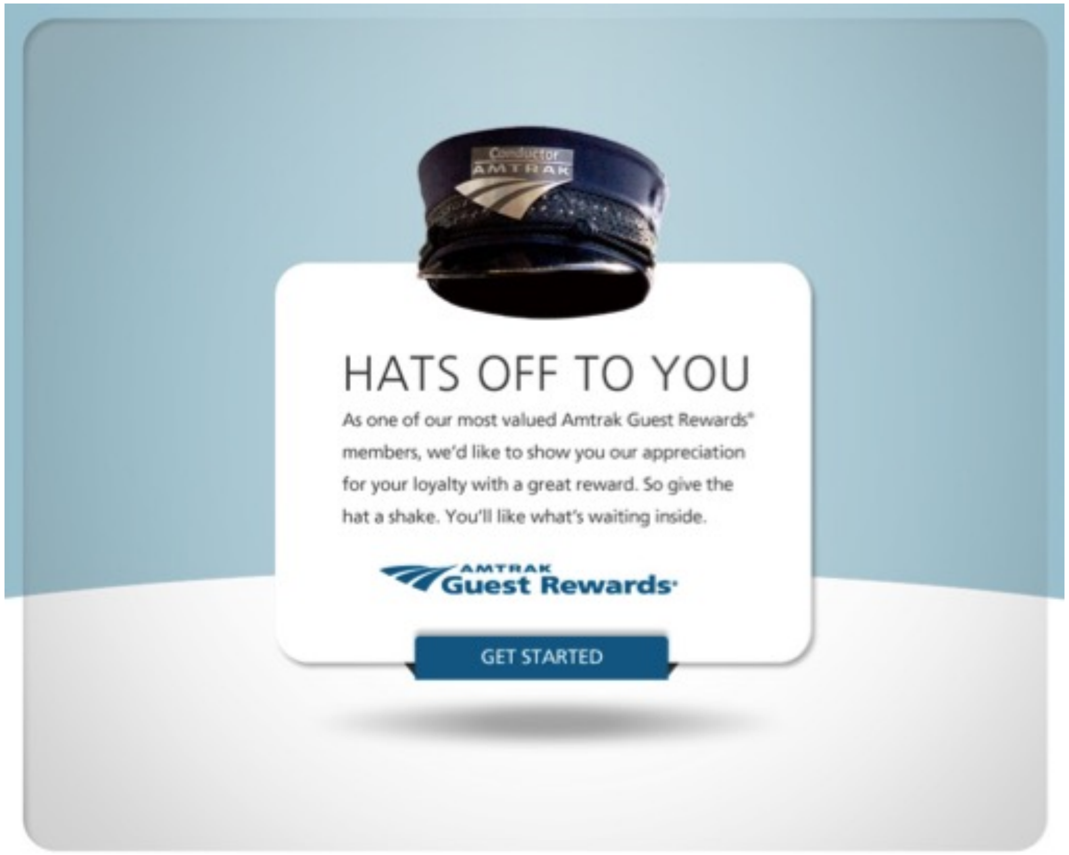
Understand the Task


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

+

Understand the Task

Amtrak Case Study



HATS OFF TO YOU

As one of our most valued Amtrak Guest Rewards® members, we'd like to show you our appreciation for your loyalty with a great reward. So give the hat a shake. You'll like what's waiting inside.

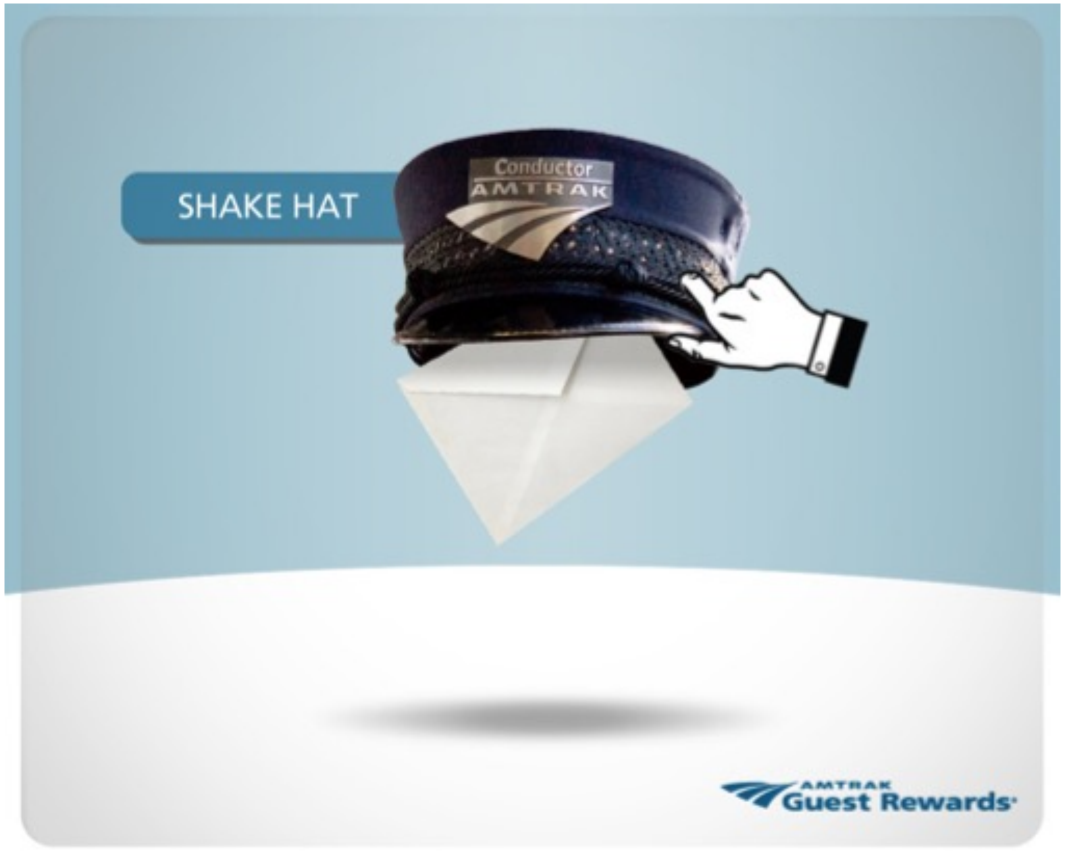
Understand the Task

Amtrak Case Study

GRAB HAT
Conductor
AMTRAK
AMTRAK
Guest Rewards

Understand the Task

Amtrak Case Study



+

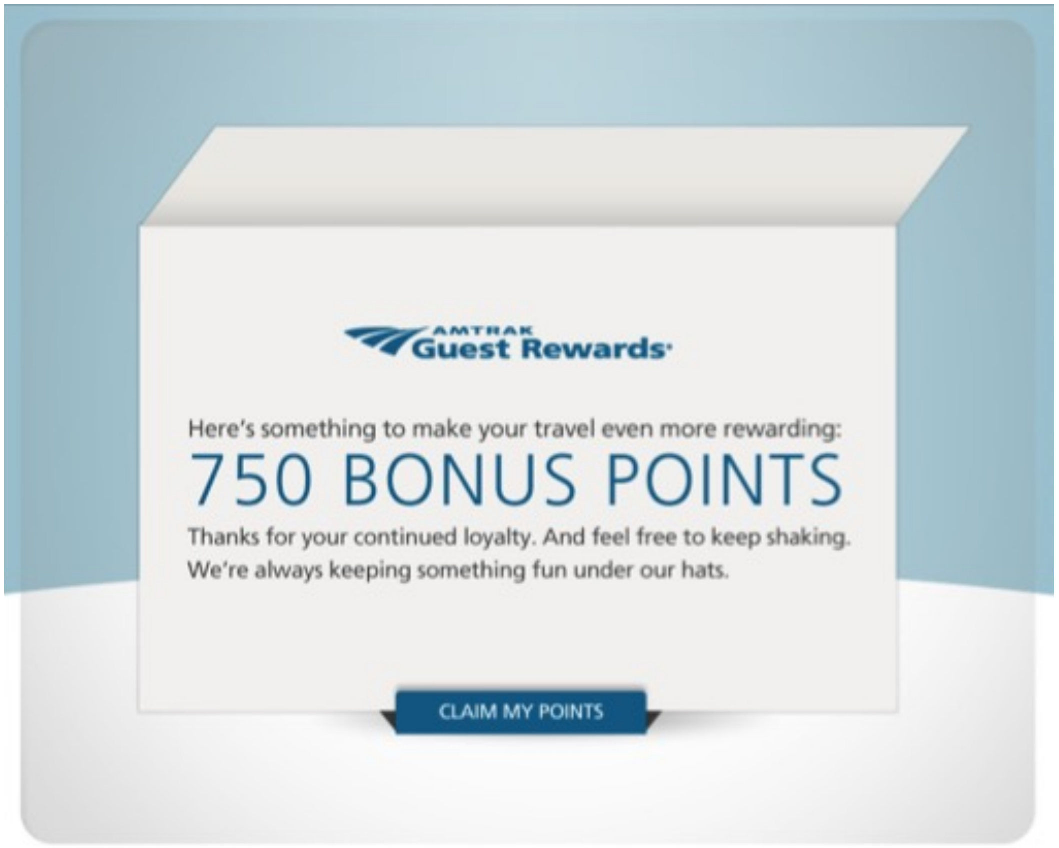
Understand the Task

Amtrak Case Study

+

Understand the Task

Amtrak Case Study



AMTRAK Guest Rewards

Here's something to make your travel even more rewarding:

750 BONUS POINTS

Thanks for your continued loyalty. And feel free to keep shaking. We're always keeping something fun under our hats.

Understand the Task

Amtrak Case Study

- Company will send out an e-mail that has a link to a specially designed e-card.
- This e-card is designed to give only one type of prize, and this prize is set by the company. There can be multiple prizes.
- When the card opens the user will be able to shake a hat.
- When the hat shakes enough a prize will appear.

Determining the Steps Needed to Perform a Task

- Next we break down the task into a series of concrete steps that can be followed (like a recipe)
- Remember that computers need each step to be broken down into minute detail.
- They don't have the ability to infer intermediate steps like we can!

Algorithms

- "A series of well defined, logical steps that must be taken in order to perform a task"
- Algorithms serve as a necessary intermediate step between understanding the task at hand and translating it into computer code

Pseudocode

(aka "fake" code)

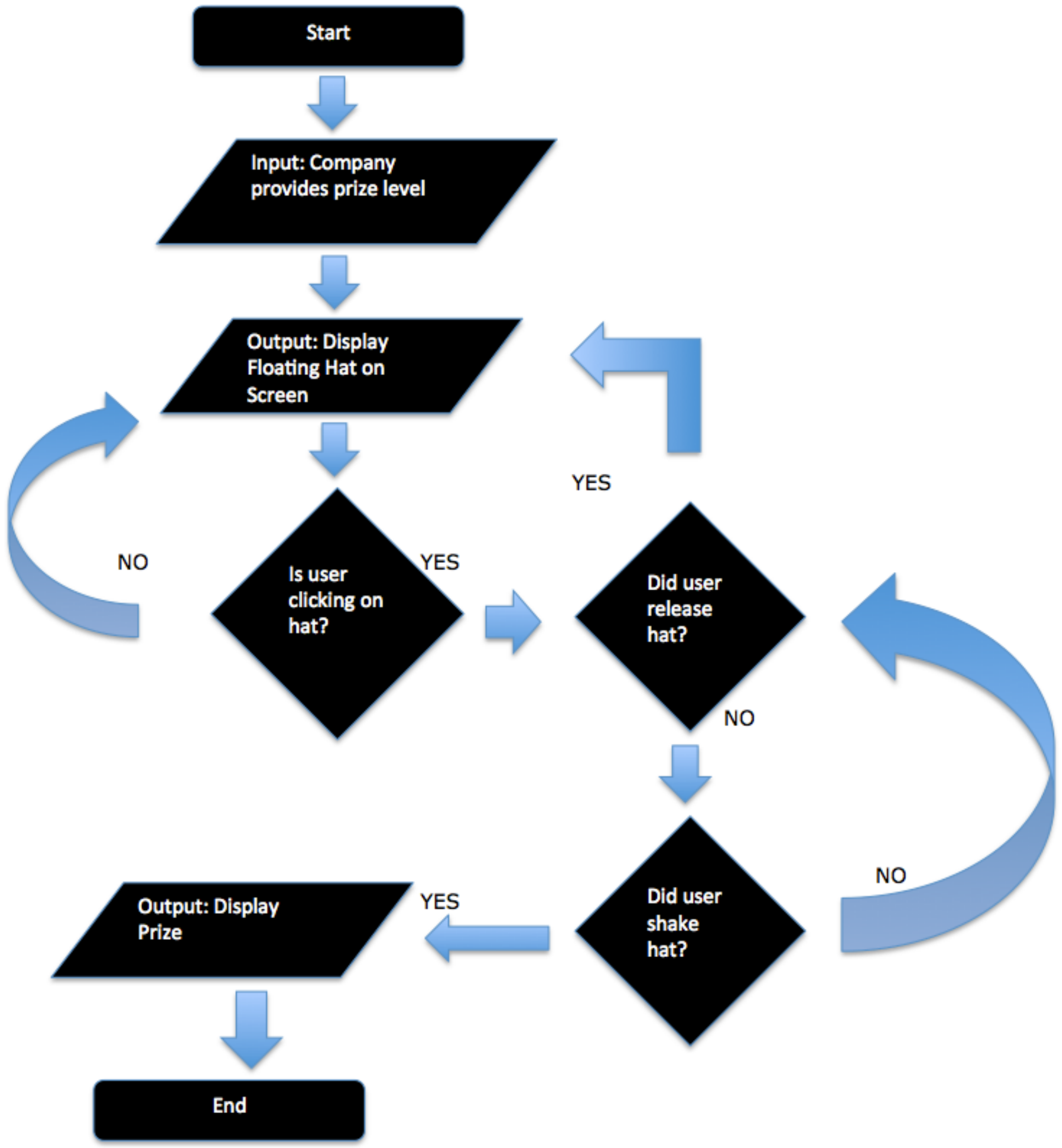
- A useful technique for breaking down an algorithm into meaningful chunks and aligning them with the toolset of a language
- In pseudocode, we don't have to worry about syntax or spelling

Amtrak in Pseudocode

```
1. Start Program. You can assume you will be told what prize the user has won when the program begins.
2. Display a floating hat on the screen
3. Allow the user to click his or her mouse on the hat
   1. When clicked, drag the hat around with the user
   2. If the user releases the mouse button, snap the hat back into place and go back to step #3
   3. If the user vigorously moves the hat around, proceed to step #4
4. Stop dragging the hat around the screen
5. Display an animation that contains the prize amount specified in step #1 on the screen
```

Flowcharts

- A graphical model that helps programmers conceptualize the task at hand


Start
Input: Company provides prize level
Output: Display Floating Hat on Screen
NO
Is user clicking on hat?
YES
YES
Did user release hat?
NO
NO
Did user shake hat?
YES
Output: Display Prize
End

+

Program Structure

Input, Processing & Output

- Programs typically perform the following 3 steps
  - INPUT is received
  - Some kind of PROCESSING is performed
  - OUTPUT is produced

Input

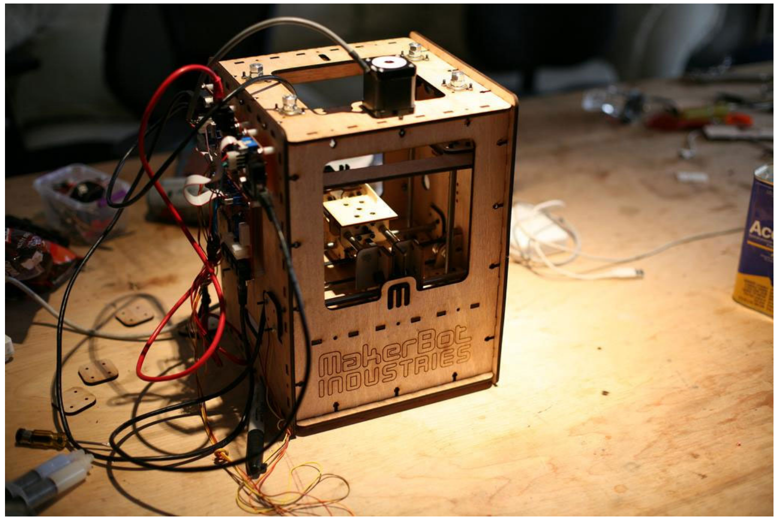
- Can be from a variety of sources
  - User: keyboard, mouse, etc.
  - Hardware: scanner, camera, etc.
  - Data: file, the Internet, etc.

Processing

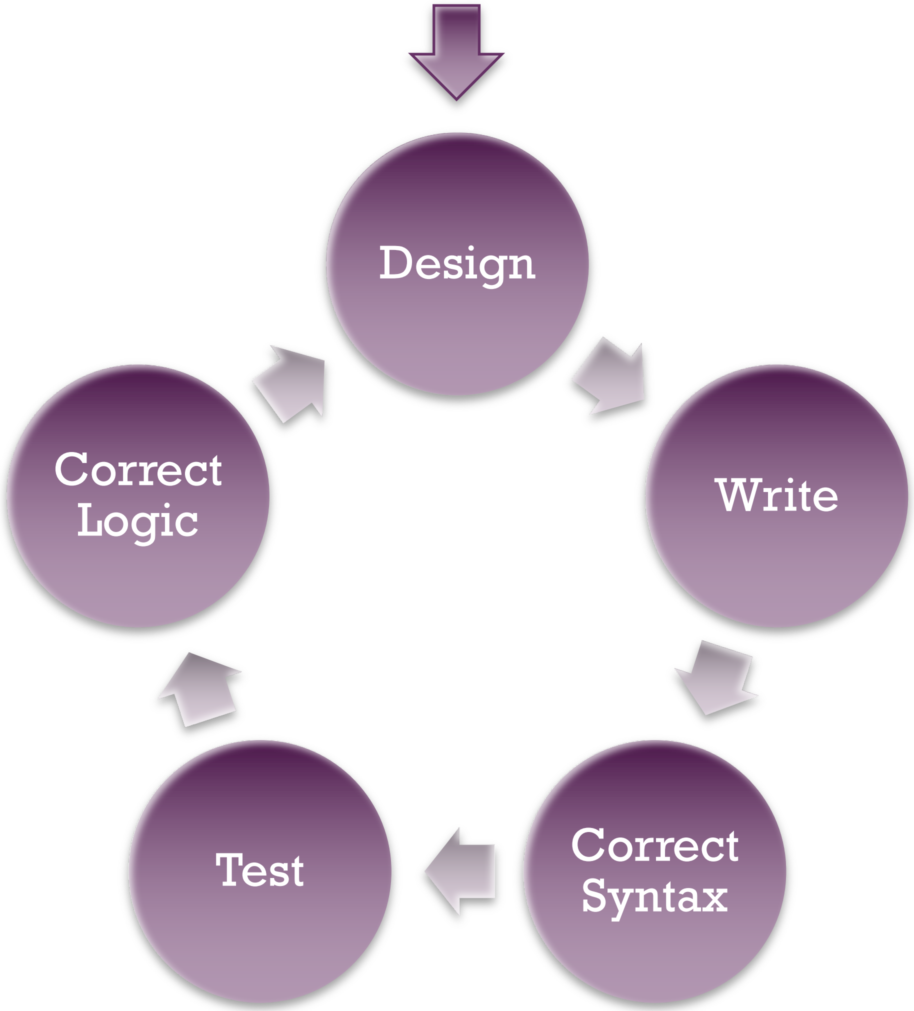
- A series of mathematical or logical processes are applied to the input
  - Compare values
  - Add, multiply, divide or subtract numbers
  - Perform calculations on an item over and over again (i.e. blurring an image)

+
Output

- Some kind of tangible / visible / readable product is constructed
  - Printout
  - Screen display
  - 3D fabrication



Software Development Loop



Any questions?

…. then on to Python!

+

Python: Getting Started

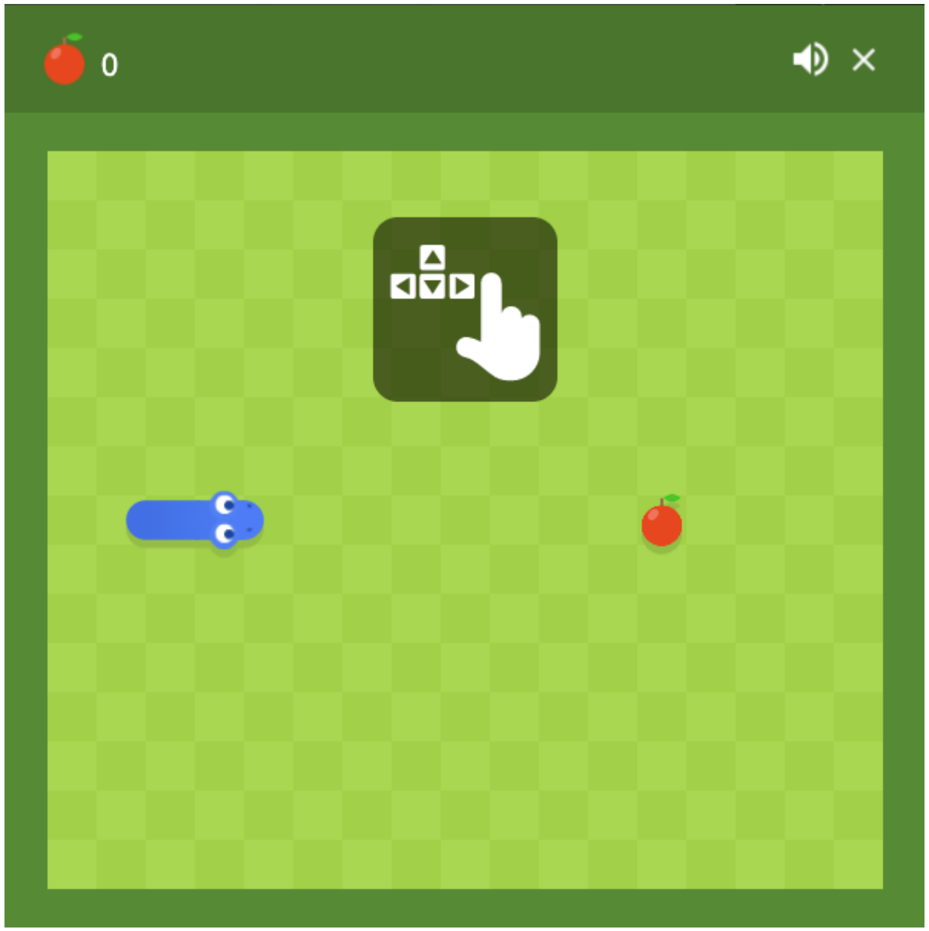
- IDLE: Integrated Development Environment
- Has two modes
  - Interactive – commands are immediately processed as they are received
  - Script – allows to write a program (saved as a "text file" on your computer) and have your commands processed whenever you'd like
- We will mainly be using "script" mode during this semester

+

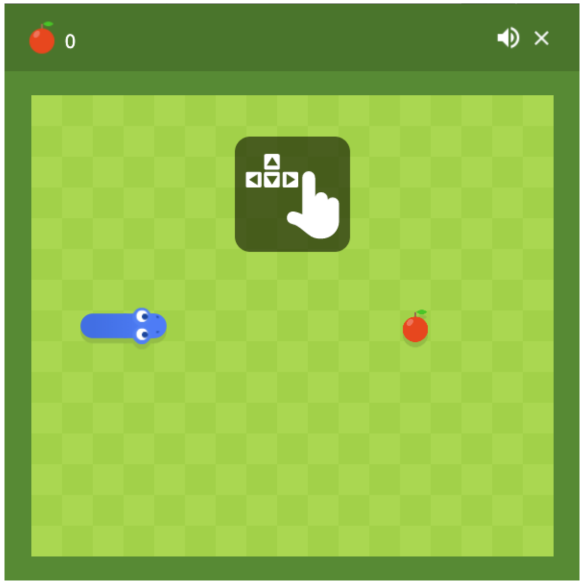
Python: Creating a new program

- Open IDLE
- Click on File -> New Window
- Click on File -> Save
- Save your file somewhere on your computer. You will need to add the '.py' file extension to your file if IDLE does not place it there automatically.
- With your program open, click on Run -> Run Module
- If you need to open your program you can click on File -> Open and browse to find the desired Python source file (.py

+ Functions



Functions



English	Code
Go right 4 units.	right(4)

+

Functions

- A "function" is a pre-written piece of computer code that will perform a specific action or set of actions
- Python comes with a number of built-in functions, and you can also write your own (more on that later in the semester)
- Functions always begin with a keyword followed by a series of parenthesis.
  - Ex: print ()
- You can "pass" one or more "arguments" into a function by placing data inside the parenthesis
  - Ex: print ('Hello, World!')
- Different functions "expect" different arguments. The print function, for example, expects printed text as an argument
- When you ask Python to run a function we say that you have "called" the function.

+ Strings

- Data that is textual in nature (i.e. “Hello, World!”) is called a “String”
- Strings can contain 0 or more printed characters
- “String Literals” are strings that you define inside your program. They are “hard coded” values and must be “delimited” using a special character so that Python knows that the text you’ve typed in should be treated as printed text (and not a function call)
  - Ex: print (‘hello, world!’)
- Python supports three different delimiters
  - The single “tick” (‘)
  - The single quote (“)
  - The triple quote (“””)

+
Printing multiple arguments

- The print() function can accept zero or more more arguments
- If you decide to pass multiple arguments to the print() function you should separate them by a comma. Example:

```
print (“Hello!  My name is”, “Craig”)
```

- Note that when Python executes the function call above it will insert a space between the two arguments for you automatically.
- Also note that the print() function will automatically add a line break after it prints the last argument it was passed. We will discuss how to override this behavior.

+
Line endings

- When using the print() function you probably have noticed that Python automatically places a newline character at the end of each line
- You can override this behavior and have Python use a character of your choice by using the optional ‘end’ argument when using the print() function
- Example:

```
print (‘one’, end=‘’)
print (‘two’, end=‘’)
```

Separating print() function arguments

- By default, Python will place a space between arguments that you use in print() function calls
- You can override this behavior by using the optional ‘sep’ argument
- Example:

```
print (‘one’, ‘two’, sep=‘*’)

# output:  one*two
```

Combing both line endings and separators

- You can use both the ‘sep’ and the ‘end’ arguments at the same time in the same print() function call.
- Example:

```
print (‘a’, ‘b’, ‘c’, sep=‘*’, end=‘’)
```

+

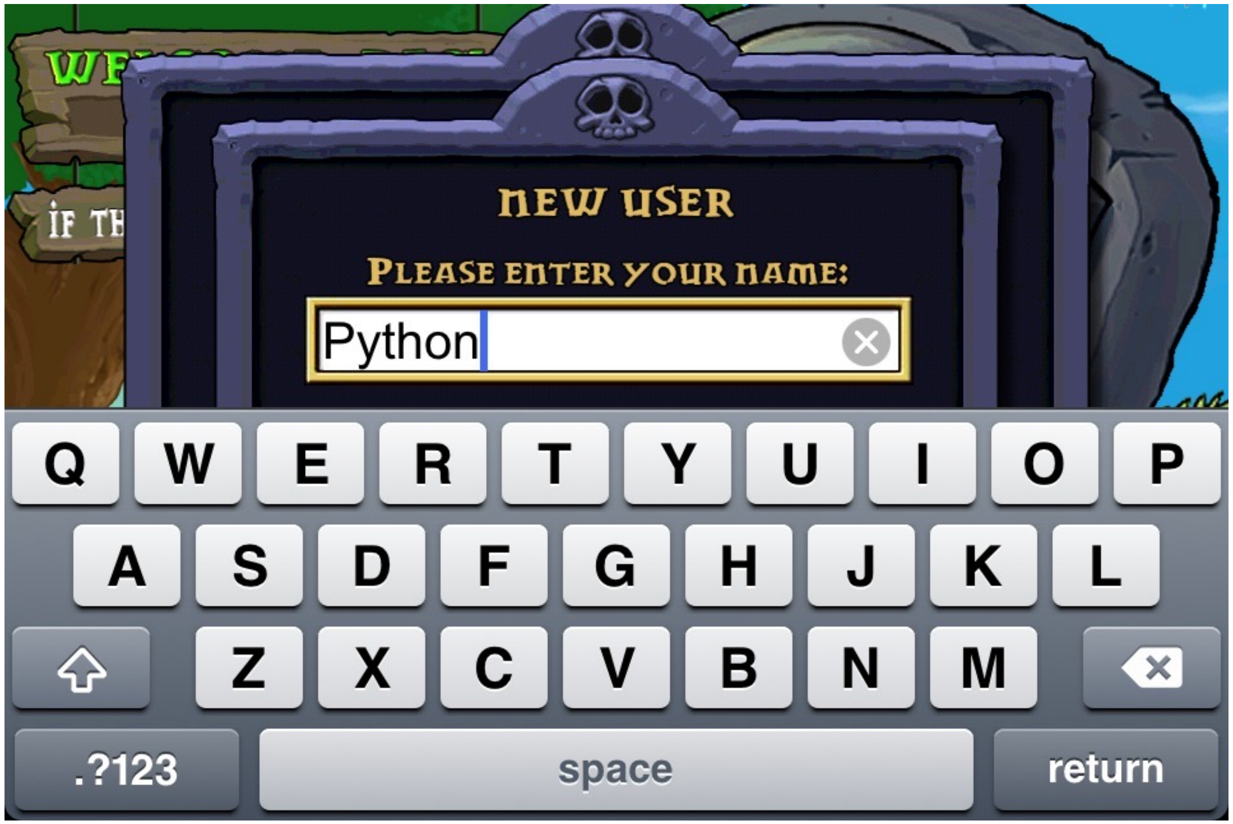
In-class programming exercise

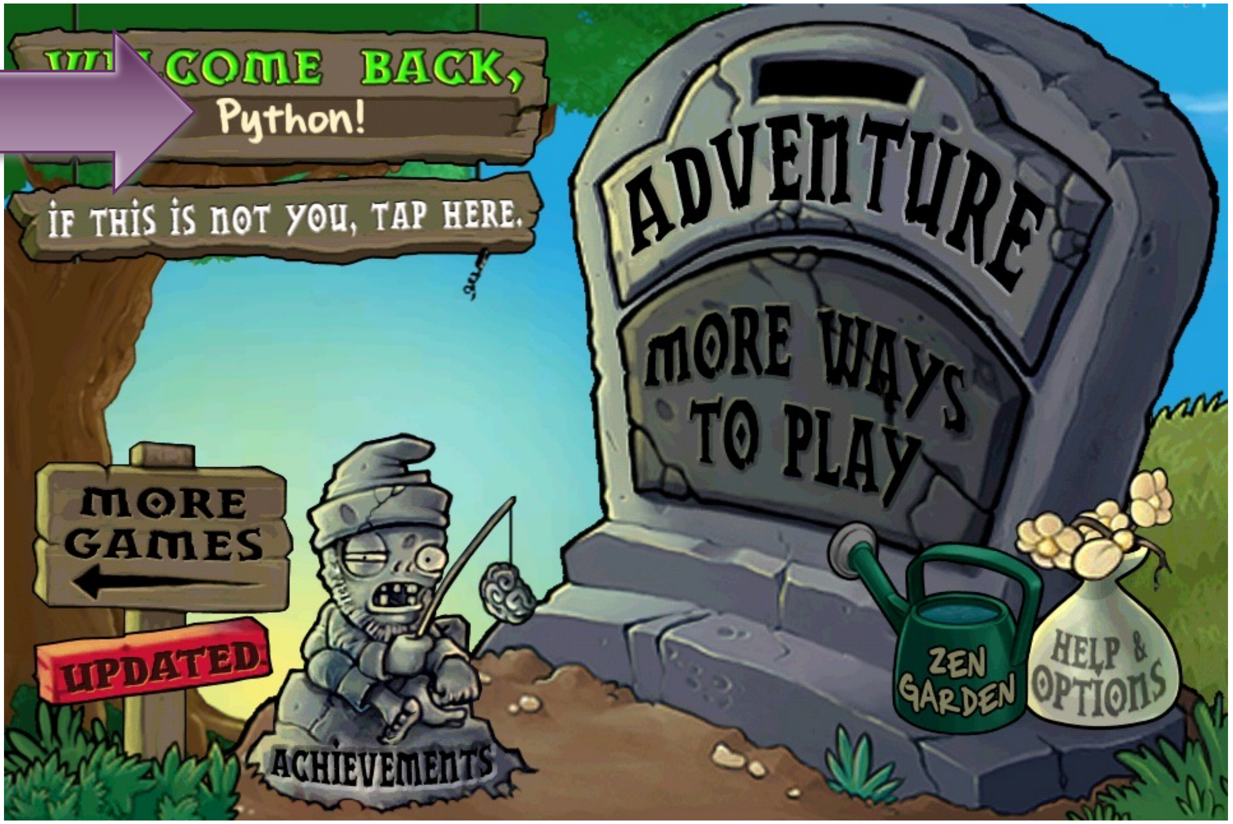
- Write a program that will produce the following output. The 2 sentences should be printed on the same line.

```
Welcome to “Introduction to Computer
Programming!” I’m Julie and my favorite animal
is a sloth.
```

+

Variables

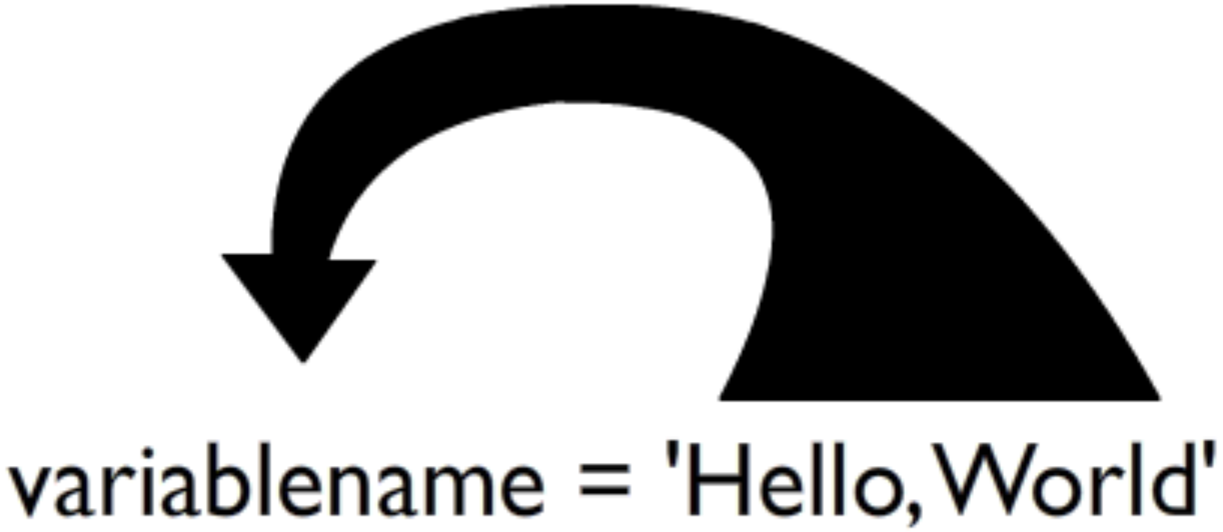

NEW USER
PLEASE ENTER YOUR NAME:
Python
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123
space
return


WELCOME BACK,
Python!
IF THIS IS NOT YOU, TAP HERE.
ADVENTURE
MORE WAYS TO PLAY
MORE GAMES
UPDATED
ACHIEVEMENTS
ZEN GARDEN
HELP & OPTIONS

+
Variables

- Variables are like little “buckets” that can store information in your computer’s memory
- You will be using variables constantly when writing your programs in order to keep track of various pieces of information
- You can create a variable by using the following syntax:
  - variablename = somedata
- Examples:
  - speed = 5
  - myname = ‘Craig’
- The ‘=‘ symbol is called the ‘assignment operator’ and will cause Python to store the data on the right side of the statement into the variable name printed on the left side

Variables



Variable Naming Rules

- You can't use one of Python's "reserved words" (see the next slide for a list)
- Variables can't contain spaces (though you can use the underscore character ("_") in place of a space)
- The first character of a variable name must be a letter or the underscore character. Any character after the first can be any valid alphanumeric character (or the underscore character)
- Python is case sensitive

+

Python Reserved Words

(these words can’t be used when declaring a variable in your program)

'False’,'None', 'True', 'and', 'as', 'assert', 'break', 'class',

'continue','def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from',

'global','if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',

'pass','raise', 'return', 'try', 'while', 'with', 'yield'

+
Legal or Illegal variable name?

www.pollev.com/python002

Common Variable Naming Conventions

```
rockettopspeed = 1000          # can be hard to read

rocket_top_speed = 1000        # underscored

rocketTopSpeed = 1000          # “camel case”
```

Displaying Variables with the print function

- You can print the data that is currently being held by a variable by passing the variable name to the print() function as an argument. Example:

```
print (myvar)
```

- As with string literals, you can tell the print() function to display more than one item in the same function call by separating your arguments with commas. Example:

```
name_var = “Harry”
print (“Hello, my name is”, name_var)

>> Hello, my name is Harry
```

+
Reassigning Variables

- Variables are called variables because they can "vary" the data that they store.

- You an re-assign the value that a variable stores by simply using a second assignment statement. Example:

```
dollars = 10.99
print ('I have', dollars, 'in my account')
dollars = 29.99
print ('Now I have', dollars, 'in my account)
```

Multiple Assignments

- It’s possible to set the value of multiple variables on the same line. For example:

```
x, y, z = ‘a’, ‘b’, ‘c’
```

- In this case the variables x, y and z will assume the values ‘a’, ‘b’, ‘c’ in that order

- You can also assign the same value to multiple variables at the same time by doing the following:

```
# a, b and c will all contain the integer 100
a = b = c = 100
```

+
Programming Challenge

You're working on a simple inventory management system for a small store. You'd like to store the name and price of two different products and print them out in the following format.

```
Item: Bread, Price: $ 2.99
Item: Eggs, Price: $ 1.99
```

+

Performing Calculations

+ Performing Calculations

- Algorithms generally require some kind of calculation to be performed
- All programming languages have tools for manipulating numeric data – we generally call these “math operations”

+

Python Math Operations

Operation	Operator
Addition	+
Subtraction	-
Multiplication	*
Division (floating point)	/
Division (integer)	//

Expressions

- We use operators along with numeric data to create “math expressions” that Python can “evaluate” on our behalf

+ Expressions

operator

$$\underbrace{5}_{\text{operand}} + \underbrace{2}_{\text{operand}}$$

```
7 — 6

2 * 3

6 / 2

6 + 2 - 12
```

Outputting math expressions

```
print (5+2)

print (100 * 5 – 12)
```

Storing the results of an expression

```
answer = 5 + 2

print (‘the answer to 5 + 2 is’, answer)
```

+

Using variables in math expressions

- Math expressions don’t necessarily need to involve only numeric literals
- Example:

```
price = 100.00
sales_tax = 0.07

total = price + sales_tax*price
```

In-class programming exercise

- Given the following variables:

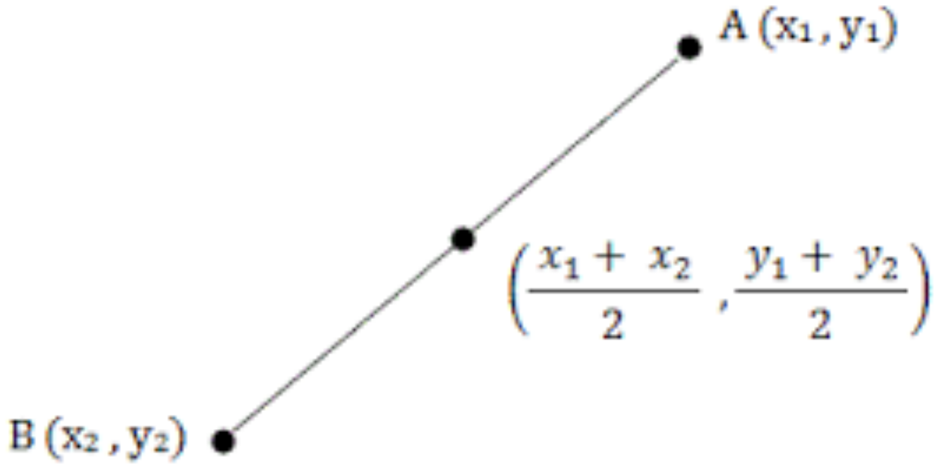
```
x1 = 100

y1 = 65

x2 = 35

y2 = -45
```

- Calculate the midpoint of these 2 points using the formula on the right.



Data Input

+
Reading Input from the Keyboard

- So far we have learned how to:
  - OUTPUT information (via the print function)
  - STORE information (via variables)
- However, all of our programs have been “hard coded,” meaning that we have predefined the operating environment using information that we ourselves have defined. Example:

```
myname = ‘Harry Potter’

print (‘Welcome to my program’, myname)
```

+
Reading Input from the Keyboard

- You can make your programs more interactive by involving the user in the process
- One of the simplest ways to do this is to request information from the keyboard using the **input()** function.
- Example:

```
# ask the user their name
username = input (‘What is your name?’)

# welcome them!
print (‘Welcome,‘, username)
```

The Input function

- Input is a built-in function
- It accepts one argument – a String
- It then prompts the user with that string and waits for them to type in a series of characters. Your program will resume when the user hits the ENTER key
- Whatever the user typed in during that time is sent back to your program as a string. You can store that string in a variable. Example:

```
user_age = input('How old are you?')
```

The Input function

- The input() function always “returns” a String.
- This means that the output it generates is a string, and when you store that output in a variable it will be treated as though it is a string.

The Input function

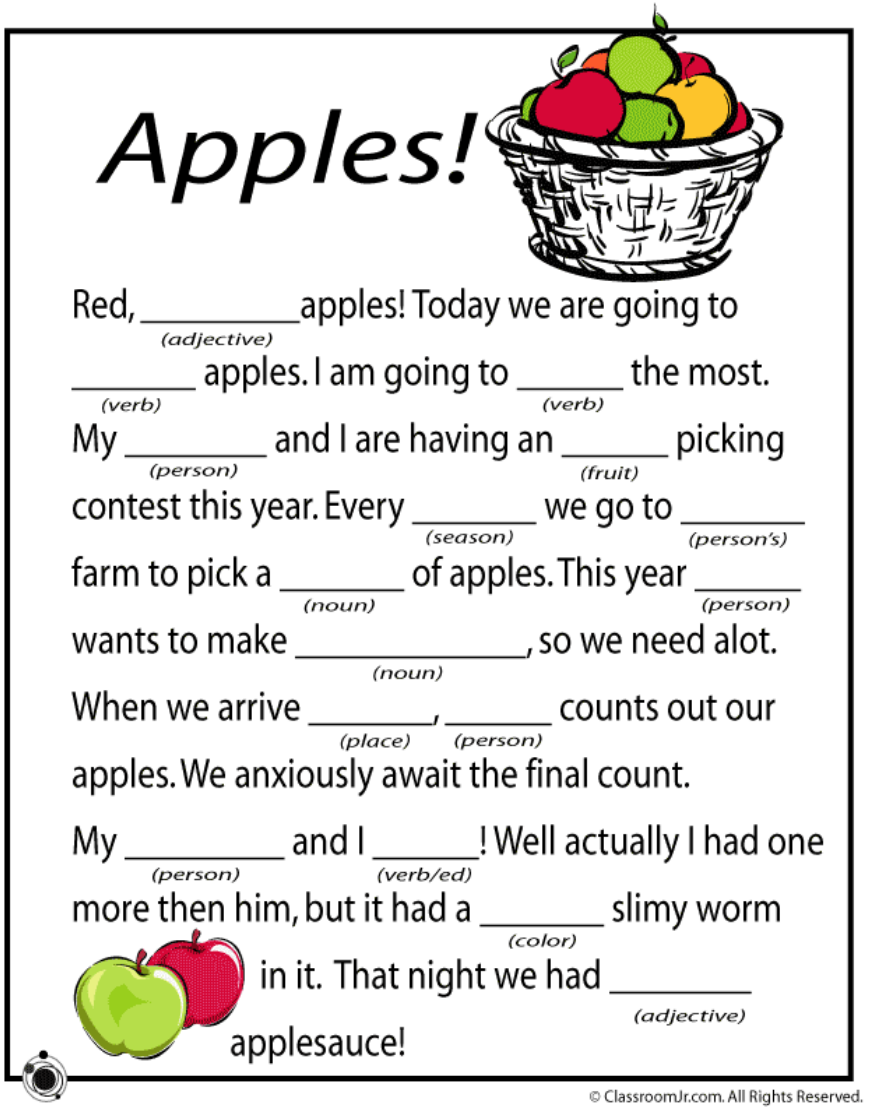
- Note that these two line work using the same mechanics:

```
var1 = “Blue”
var2 = input(“What is your favorite color?”)
```

- In the first line of code we are assigning the String Literal into the variable ‘var1’
- In the second line of code we are assigning the RESULT of the input function into the variable ‘var2’
- Both ‘var1’ and ‘var2’ will be filled with data after these lines execute

+

Input



Apples!

Red, ________ apples! Today we are going to
(adjective)

________ apples. I am going to ______ the most.
(verb) (verb)

My ________ and I are having an ______ picking
(person) (fruit)

contest this year. Every ______ we go to ______
(season) (person's)

farm to pick a ______ of apples. This year ______
(noun) (person)

wants to make ____________, so we need alot.
(noun)

When we arrive ______, ______ counts out our
(place) (person)

apples. We anxiously await the final count.

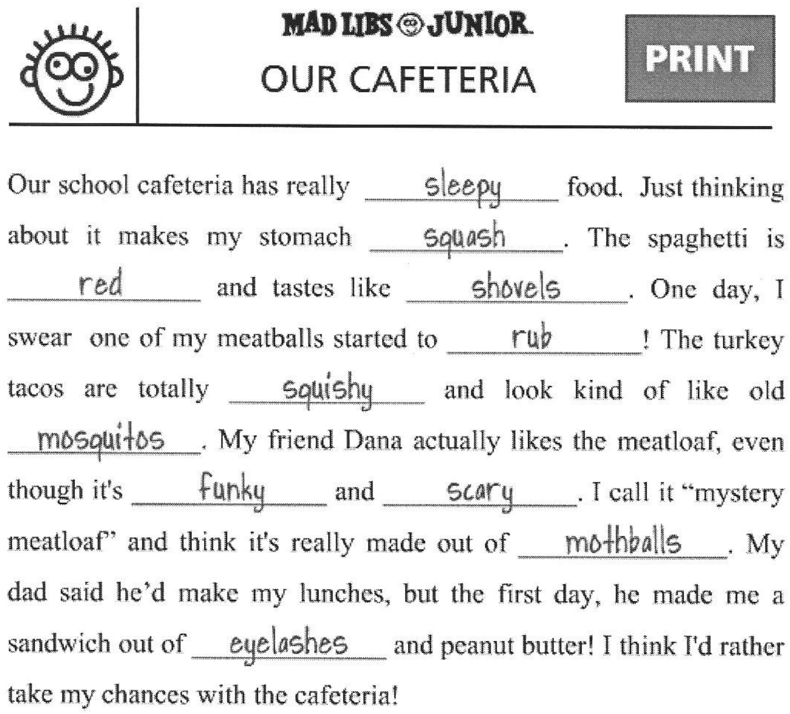
My ________ and I ______! Well actually I had one
(person) (verb/ed)

more then him, but it had a ______ slimy worm
(color)

in it. That night we had ________
(adjective)

applesauce!

Output



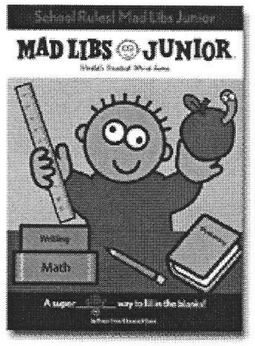
MAD LIBS JUNIOR

OUR CAFETERIA

PRINT

Our school cafeteria has really sleepy food. Just thinking about it makes my stomach squash. The spaghetti is red and tastes like shovels. One day, I swear one of my meatballs started to rub! The turkey tacos are totally squishy and look kind of like old mosquitos. My friend Dana actually likes the meatloaf, even though it's funky and scary. I call it "mystery meatloaf" and think it's really made out of mothballs. My dad said he'd make my lunches, but the first day, he made me a sandwich out of eyelashes and peanut butter! I think I'd rather take my chances with the cafeteria!

This Mad Lib was taken from School Rules! Mad Libs Junior, available at Amazon.com and in bookstores!



+
In-class programming exercise

- Write a program that asks the user to type in 4 different words using the following prompts:

```
enter a noun
enter a verb
enter an adjective
enter an adverb
```

- Use the input to output a "Mad Libs" paragraph using the following text:

```
The [adjective] [noun] was very hungry, so it
decided to [adverb] [verb] to the nearest
restaurant.
```

Input in other languages …

- HTML & PHP: http://cookingwithpixels.com/python2011/input/form.html

Work on Assignment #1: "Hello, World!"