Objectives

- Review algorithms
- Programming in Python
  - Data types
  - Expressions
  - Variables
  - Arithmetic
- Broader Issue: Algorithms

Review

- What did we learn from the drawing instruction exercise?
- Why do we need programming languages?
  - What are characteristics of programming languages?
- What programming language will we use in this class?
  - What does the *interpreter* do?
    - How can we run the interpreter? What modes does it have?
  - How do we display *output* in that language?
  - What are the *primitive data types*?
- What is an algorithm?
  - What are components/parts of an algorithm?
  - Pick a TV show or movie: What is the algorithm for the TV show/movie?

"Really?" with Professor Sprenkle

- In *TV Guide*, showrunners of *Once Upon a Time* were asked, "Give us an algorithm for your show."

"Really?" with Professor Sprenkle

- In *TV Guide*, showrunners of *Once Upon a Time* were asked, "Give us an algorithm for your show."
  - Example (for first season): 1 part Snow White + 1 part *Lost* + .5 *Alias*
- They said, "We don't understand math. That's why we became writers."

Review: 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

An overview for the semester!

Review: Discussion of PB&J

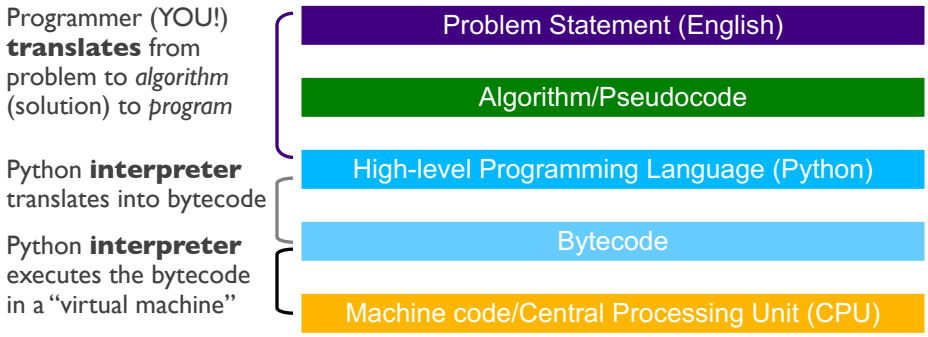
- The computer: a blessing and a curse
  - Recognize and meet the challenge!
- Be unambiguous, descriptive
  - Must be clear for the computer to understand
  - "Do what I **meant**! Not what I said!"
    - Motivates programming languages
- Creating/Implementing an algorithm
  - Break down pieces
  - Try it out
  - Revise

Review: Discussion of PB&J

- Steps need to be done in a particular order
- Be prepared for special cases
- Aren't necessarily spares in real life
  - Need to write correct algorithms!
- Reusing similar techniques
  - Do the same thing with a little twist
- Looping
  - For repeating the same action

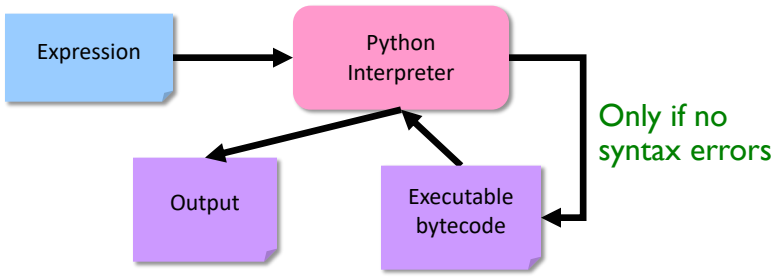
Review: Why Do We Need Programming Languages?

- Computers can't understand English
  - Too ambiguous
- Humans can't easily write machine code



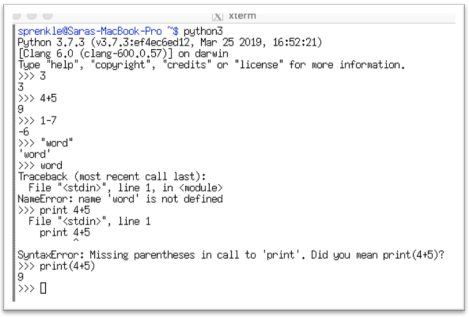
Review: Python Interpreter

1. Validates Python programming language expression(s)
   - Enforces Python **syntax**
   - Reports **syntax** errors
2. Executes expression(s)
   - Runtime errors (e.g., divide by 0)
   - **Semantic** errors (not what you *meant*)



Review: Modes to Execute Python Code

- **Interactive/Shell**
  - Try out Python expressions



- **Batch**: execute *scripts* (i.e., files containing Python code)
  - What we'll write usually

Review: Primitive Types

Python Data Type	Description	Examples
`int`	Plain integers (32-bit precision)	`-214, -2, 0, 2, 100`
`float`	Real numbers	`.001, -1.234, 1000.1, 0.00, 2.45`
`complex`	Imaginary numbers (have real and imaginary part)	`1j * 1J → (-1+0j)`
`str`	Text, enclosed in matching single or double quotes	`"my cat", "123", 'Hello!'`
`bool`	True or false	`True, False`

What is the value's type?

Value	Type
`52`	
`-0.01`	
`4+6j`	
`"3.7"`	
`4047583648`	
`True`	
`'false'`	

What is the value's type?

Value	Type
52	int
-0.01	float
4+6j	complex
"3.7"	str
4047583648	int
True	boolean
'false'	str

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

Introduction to Variables

- Variables save data/information
  - Example: first slice of bread or knife A
  - Type of data the variable holds can be any of primitive data types as well as other data types we'll learn about later
- Variables have names, called ***identifiers***

Variable Names/Identifiers

- A variable name (identifier) can be any one word that:
  - Consists of letters, numbers, or _
  - Does *not* start with a number
  - Is not a Python reserved word
    - Examples: `for while def`
- Python is case-sensitive:
  - change isn't the same as Change

Variable Name Conventions

- **Variables** start with lowercase letter
- Convention: **Constants** (values that won't change) are all capitals
  - (more on this later…)
- Example: Variable for the current year
  - `currentYear`
  - `current_year`
  - `CURRENT_YEAR`
  - ~~`currentyear`~~ Harder to read
  - ~~`current year`~~ No spaces allowed

Naming doesn't matter to computer, matters to humans

Importance of Variable Naming

- Helps you *remember* what the variable represents
- Easier for others to *understand* your program
- Examples:

Info Represented	Good Variable Name
A person's first name	`firstName, first_name`
Radius of a circle	`radius`
If someone is employed or not	`isEmployed`

Review: Computational Problem Solving

- **Computational Problem:**
  A problem that can be solved by logic

- To solve the problem:
  - Create a **model** of the problem
  - Design an **algorithm** for solving the problem using the model
  - Write a **program** that *implements* the algorithm

Modeling Information

- How would you ***model*** this information?
- What data type best represents the info?

Info Represented	Data Type	Variable Name
A person's salary		
Sales tax		
If item is taxable		
Course name		
Graduation Year		

Modeling Information

- How would you ***model*** this information?
- What data type best represents the info?

Info Represented	Data Type	Variable Name
A person's salary	int or float	`salary`
Sales tax	float	`salesTax`
If item is taxable	bool	`isTaxable`
Course name	str	`course_name`
Graduation Year	int	`gradYear`

Variable names are just suggestions,
Many other possible variable names

Assignment Statements

- Variables can be given a value using **=**
  - **Syntax**: `<variable> = <expression>`
  - **Semantics**: `<variable>` is set to value of `<expression>`
- After a variable is set to a value, the variable is said to be ***initialized***
- Examples:

```
month = 1
impt_num = 4.5
monthName = 'January'
```

These are **not** equations!
Read "=" as "is set to"

Variables: The Rules

- Only the variable(s) to **left** of the = in the current statement change
  - We'll usually only have one variable on the left
- **Initialize** a variable **before** using it on the right-hand side (rhs) of a statement

Assignment Statements

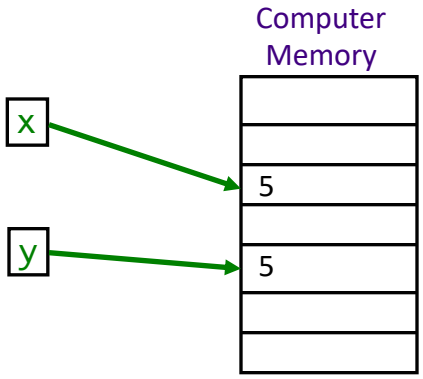
```
x = 5
y = x
```

- Statements execute in order, from top to bottom
- Value of **x** does not change because of second assignment statement

Assignment Statements

```
x = 5
y = x
```

Does a "lookup" in memory to find value of x



- Statements execute in order, from top to bottom
- Value of **x** does not change because of second assignment statement

Literals

- Pieces of data that are not variables are called ***literals***
  - We've been using these a lot
- Examples:
  - `4`
  - `3.2`
  - `'q'`
  - `"books"`

Numeric Arithmetic Operations

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

Arithmetic & Assignment

- You can use the assignment operator (=) and arithmetic operators to do calculations
  1. Calculate right hand side
  2. Assign value to variable
- Remember your order of operations! (PEMDAS)
- Examples:

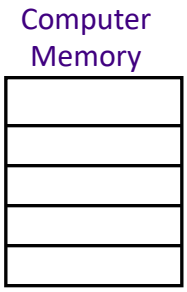
```
x = 4+3*10
y = 3/2.0
z = x+y
```

The right-hand sides are ***expressions***, just like in math.

Arithmetic & Assignment

- Examples:

```
x = 4+3*10
y = 3/2.0
z = x+y
```

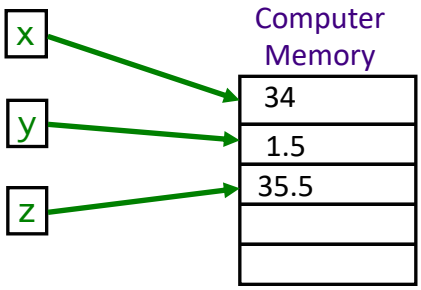


- For last statement
  - need to "lookup" values of x and y
  - computer remembers the result of the expression, not the expression itself

Arithmetic & Assignment

- Examples:

```
x = 4+3*10
y = 3/2.0
z = x+y
```



- For last statement
  - need to "lookup" values of x and y
  - computer remembers the result of the expression, not the expression itself

Broader Issue: Typical Process

1. Break into assigned groups
2. Introduce yourselves
3. Answer questions in groups
4. Discuss in class

Broader CS Issues

- Good summaries!
  - Good English, complete sentences
  - Followed the specifications
- Good, thoughtful questions
  - A lot are teasers to what I hope we'll talk about later this term
- Interest scale is 0 to 9
  - Recall: Lab 0
  - Why we start at 0 will be clearer soon…

Algorithms Everywhere

- Comment on these from articles:
  - "Because it's less familiar, *algorithm* tends to emphasize our uncertainty."
  - "An algorithm is, essentially, a brainless way of doing clever things."
- What are examples of algorithms that you do every day?
- What is machine learning useful for?
- What aren't algorithms useful for?
- What would be some useful algorithms, specific to W&L students?
  - What are problems that are difficult—but useful—to solve?

My Corrections to Articles

- "In his book *The Master Algorithm*, Pedro Domingos offers a masterfully simple definition: 'An algorithm is,' Domingos writes, 'a sequence of instructions telling ~~a computer~~ what to do.'"
- "An algorithm is, essentially, a ~~brainless~~ way of doing clever things."
  -

Looking Ahead

- Extra Credit Opportunity:
  - Read an article that relates to CS
  - Summarize it on the discussions under "Extra Credit"
    - 5 pts extra credit added to lab grade
- Textbook Pre Lab 1 assignment due before lab on Tuesday
  - We won't have covered everything in assignment until after Monday's class
  - Book has changed since I last used
    - Let me know if there are problems