

Objectives

- Review algorithms
- Programming in Python
 - Data types
 - Expressions
 - Variables
 - Arithmetic

Review

- What is an algorithm?
- What did we learn from the PB&J demonstration?

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!

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

Discussion of PB&J

- Steps need to be done in a particular order
- Be prepared for special cases
 - Any other special cases we didn't discuss?
- 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

Other Lessons To Remember

- A cowboy's wisdom: Good judgment comes from experience
 - How can you get experience?
 - Bad judgment works every time
- Program errors can have **bad** effects
 - Prevent the bad effects--especially before you turn in your assignment!

Computational Problem Solving 101

- **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

Jan 11, 2019

Sprenkle - CSCI111

7

Why Do We Need Programming Languages?

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

Live Jazz!

Problem Statement (English)



Machine code/Central Processing Unit (CPU)

000000 00001 00010 00110 00000 100000

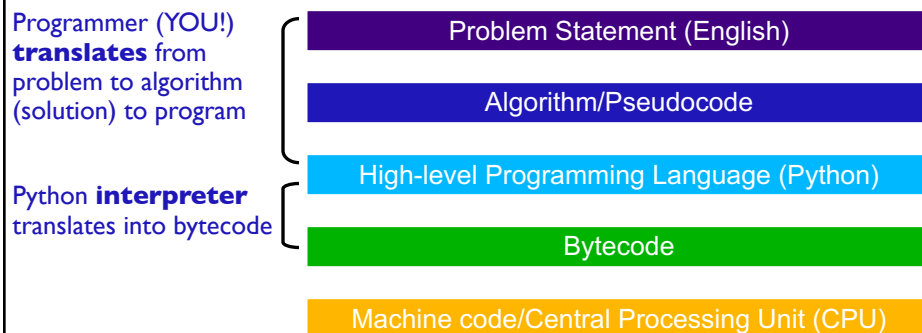
Jan 11, 2019

Sprenkle - CSCI111

8

Why Do We Need Programming Languages?

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



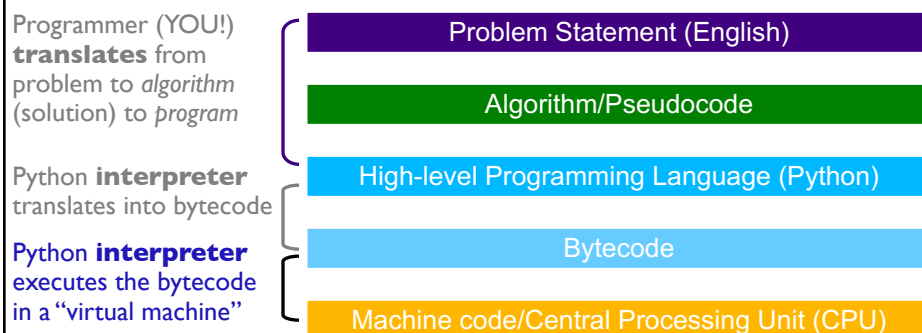
Jan 11, 2019

Sprenkle - CSCI111

9

Why Do We Need Programming Languages?

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



Jan 11, 2019

Sprenkle - CSCI111

10

Programming Languages

- Programming language:
 - Specific rules for what is and isn't allowed
 - Must be exact
 - Computer carries out commands as they are given
- **Syntax:** the symbols given
- **Semantics:** what it means
- Example:
 - `III * IV` means 3×4 which evaluates to 12
 - `cp src dest` means copy the file named src to dest
- Programming languages are **unambiguous**

Another Syntax and Semantics Example

What does this syntax mean?



Python Is ...

- A **programming language**
 - 3rd most popular programming language, according to Tiobe survey

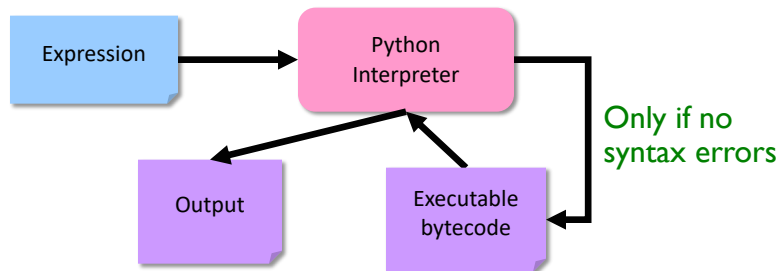
<http://www.tiobe.com/tiobe-index/>
- An **interpreter** (which is a program) that understands and executes Python code

Python Programming Language

- A common **interpreted** programming language
 - Runs on many operating systems
- First released by Guido van Rossum in 1991
- Named after *Monty Python's Flying Circus*
- Minimalist syntax, emphasizes **readability**
- Flexible, fast, useful language
- Used by scientists, engineers, systems programmers

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*)



Jan 11, 2019

Sprenkle - CSCI111

15

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

Jan 11, 2019

Sprenkle - CSCI111

19

Printing Output

- **print** is a special command or a *function*
 - Displays the result of expression(s) to the terminal
 - Automatically adds a '\n' (carriage return) after it's printed
 - Relevant when have multiple print statements

● `print("Hello, class")`
 string literal

Syntax: a set of double quotes
Semantics: represents text

Printing Output

- **print** is a special command
 - Displays the result of expression(s) to the terminal


● `print("Hello, class")`
 string literal

print automatically adds a '\n' (carriage return) after it's printed

● `print("Your answer is", 4*4)`

Syntax: comma
Semantics: print multiple "things" in one line

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

Primitive Data Types

- Primitive data types represent **data**
 - In PB&J example, our data had **types** slice of bread, PB jar, jelly jar, etc.
- Python provides some basic or **primitive data types**
- Broadly, the categories of primitive types are
 - Numeric
 - Boolean
 - Strings

Numeric Primitive Types

Python Data Type	Description	Examples
<code>int</code>	Plain integers (32-bit precision)	-214, -2, 0, 2, 100
<code>float</code>	Real numbers	.001, -1.234, 1000.1, 0.00, 2.45
<code>complex</code>	Imaginary numbers (have real and imaginary part)	$1j * 1j \rightarrow (-1+0j)$

How big (or small or precise) can we get?

- Computer cannot represent all values
- Problem: Computer has a **finite** capacity
 - The computer only has so much memory that it can devote to one value.
 - Eventually, reach a cutoff
 - Limits size of value
 - Limits precision of value

PI has more decimals,
but we're out of space!

0	0	0	0	0	3	.	1	4	1	5	9	2	6	5
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Example: in Python interpreter, `.1 + .1 + .1` yields `0.30000000000000004`.
* In reality, computers represent data in binary.

Strings: **str**

- Indicated by double quotes " " or single quotes ' '
- Treat what is in the " " or ' ' literally
 - Known as **string literals**
- Examples:
 - "Hello, world!"
 - 'c'
 - "That is Buddy's dog."

Single quote must be
inside double quotes*
* Exception later

Booleans: **bool**

- 2 values
 - True
 - False
- More on these later...

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**~~
 - ~~➤ **current year**~~

Naming doesn't matter to computer,
matters to humans

Harder to read

No spaces allowed

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	boolean	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 any 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”

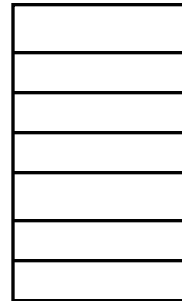
Variables: The Rules

- 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
```

Computer
Memory



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

Jan 11, 2019

Sprenkle - CSCI111

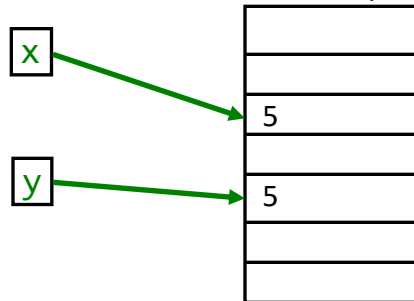
40

Assignment Statements

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

Does a “lookup”
in memory to find
value of x

Computer
Memory



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

Jan 11, 2019

Sprenkle - CSCI111

41

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

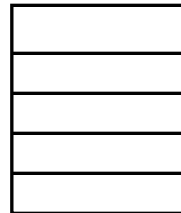
- Examples:

$x = 4 + 3 * 10$

$y = 3 / 2.0$

$z = x + y$

Computer
Memory



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

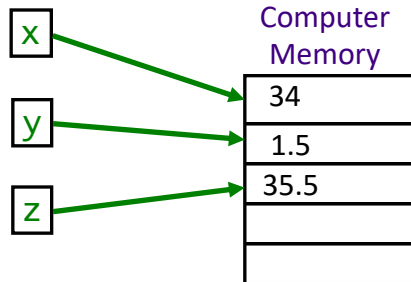
Arithmetic & Assignment

- 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

What are the values?

- After executing the following statements, what are the values of each variable?

➤ $r = 5$

➤ $s = -1 + r$

➤ $t = r + s$

➤ $s = 2$

➤ $r = -7$

How can we verify our answers?

Programming Building Blocks

- Each type of statement is a building block
 - Initialization/Assignment
 - So far: Arithmetic
 - Print
- We can combine them to create more complex programs
 - Solutions to problems

Assign.

print

Assign.
print
Assign.
Assign.
print

Broader Issue Groups

Introduce yourselves!

Charlotte
Callie
Danny
Karel

James
Jake
Natalie
Nate

Cat
Giovanni
Kassi
Matt
Mike

Alice
Daniel
Hayden
Jenna
Melissa

Andrew
August
Bobby
Danielle
Ellis
Laurie

Broader CS Issues

- Good summaries!
 - Good English, complete sentences
- Good, thoughtful questions
- Mechanics details
 - Follow instructions on BI Forum about what summary should contain
 - Should be able to edit your own posts
 - Characters from Word
 - Click button “Paste from Word”
 - Don’t attach Word documents

“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 1st season): 1 part *Snow White* + 1 part *Lost* + .5 *Alias*
- They said, “We don’t understand math. That’s why we became writers.”

AI Everywhere

- “An algorithm is, essentially, a brainless way of doing clever things... Brainlessness, in other words, is no impediment to intelligence. ”
- What are examples of algorithms that you do every day?
- What is AI (which is based on algorithms) 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?

Extra Credit Opportunities

- Read an article that relates to CS
- Summarize it on the forum under “Extra Credit”
 - 5 pts extra credit on lab grade

Looking Ahead

- Textbook Pre Lab 1 assignment due before lab on Tuesday