CS1 Lecture 2 Jan. 20, 2017

- On ICON: first of 12 short surveys related to the research study this class is part of. Due Jan. 28. Overall, the surveys are worth 2% of your grade (you'll get 0-5 points, proportional to how many surveys you complete)
- TA office hours
  - You can go to *any* TA office hours, not just TA of your discussion section.
- Contacting me/TAs by email
  - You may send questions/comments to me/TAs by email.
    - For discussion section issues, sent to TA and me
    - For homework or other issues send to me (your discussion section TA might not be involved in grading your homework)
  - DO NOT sent multiple messages (one to TA, one to me). This often wastes people's time and makes a mess!
  - For homework-related email: *ALWAYS* **attach** file of your current code. Don't just copy/paste code into message body
- First homework will be available Monday, due late Sunday (but practice some Python right away!)

Today

- Course overview
- Chapter 1 of text
  - Programs
  - Python
  - Evaluating simple expressions
  - Types
  - Natural vs formal languages
  - (debugging will be deferred until later)
- Example GUI/web program demo you'll soon be able to create
- Graph problems posed at end of first lecture

Course overview

- 6 weeks basic python programming
  - Expressions, types, variables, conditions, functions, iteration
  - Use of sequence and dictionary data types
- Additional topics (approx. one week each)
  - Classes and object oriented programming
  - Comprehensions and other more powerful Python language tools
  - Program design and debugging
  - Algorithmic efficiency
  - Searching and sorting
  - GUIs
  - Graphing, plotting
  - Accessing web data
  - Machine learning or other advanced topics

Ch 1: Programs

Textbook: "A program is a sequence of instructions that specifies how to perform a computation."

Many people like to distinguish "algorithm" and "program."

dictionary.com: An algorithm is a "finite set of unambiguous instructions performed in a prescribed sequence to achieve a goal."

(non-computing examples?)

I like to say:

program = algorithm + programming language

or:

Programming is the process of expressing an algorithm in a programming language (so you can execute it on a computer)

Ch 1: Programs

Key components of programming, independent of particular programming language.

- Expressions
- Variables and assignment
- if-then-else (decision making/branching)
- Iteration
- Functions

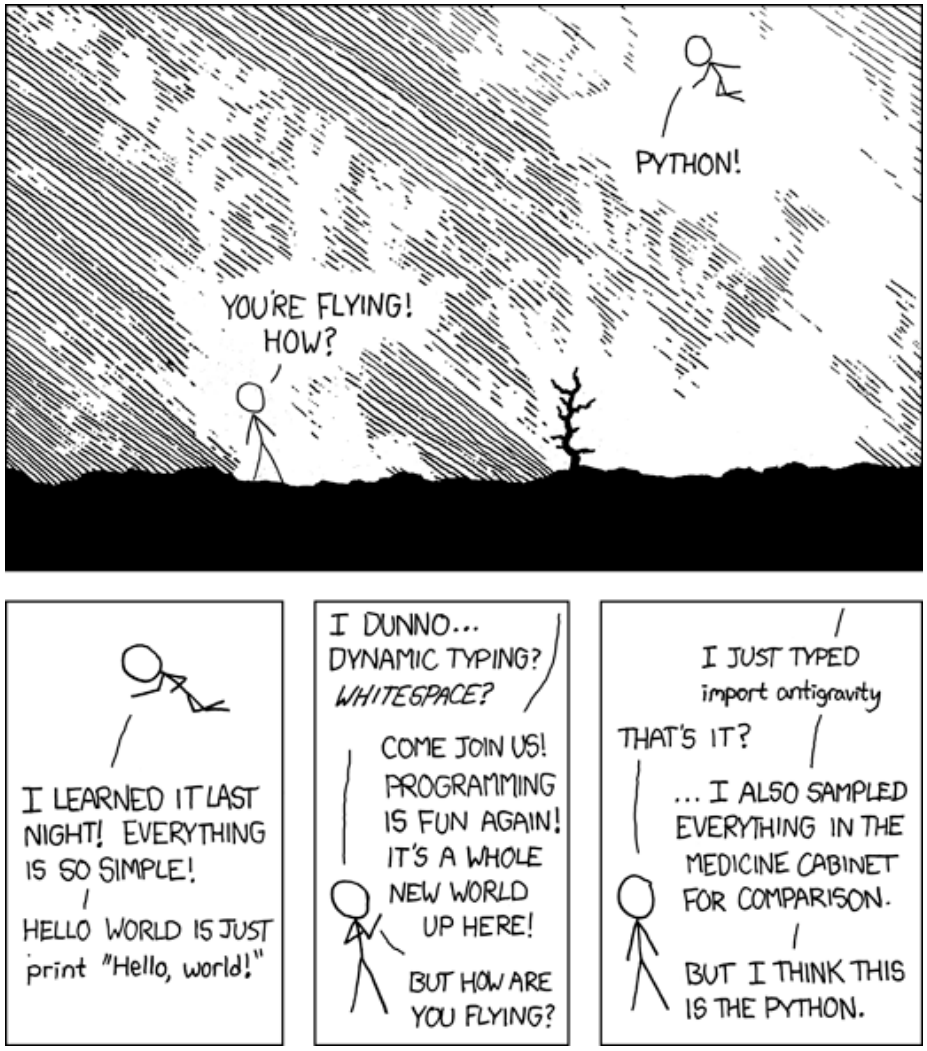
Essentially all programming languages are built around these components. Once you understand how to describe computations using them, you can program. **Learn these basics well!** Changing programming languages is usually just a matter of looking up details of how to "say" a particular standard thing in the different language.

Ch 1: Python

There are many many programming languages!

Why Python?

- Clean syntax, powerful constructs enable beginners to more quickly get computer to do interesting things
- Interactive mode encourages experimentation
- Extensive standard and third party libraries for web programming, scientific computation, data analysis, etc.



https://xkcd.com/353/

Ch 1: Using Python

- Book encourages starting with browser-based Python. I recommend instead that you install Python with Idle or Anaconda on your own machine (see course website)
- When working with Python, we will usually use a Python **interpreter**, which is a program that reads Python code and executes it.
- Demo: Python interpreter via pythonanywhere.com
- Demo: Python interpreter from Mac terminal
- Demo (the most common way I will do things and recommended approach for students): Python interpreter plus IDLE IDE.

Ch 1: expressions, arithmetic, types

- You can use the Python interpreter like a calculator, typing in many different mathematical (and other) expressions and seeing immediate results
  - print("hello")
  - 3 + 2
  - 2**128
  - 5/2+1
- Expressions yield **values.** Every value has a **type.**
  - 3's type in **int** (for integer)
  - 3.5's type is **float** (for floating point number)
    - (What about 3.0? 3.0's type is float. It is not the exact same thing as 3 in Python but (fortunately) it *is* "equal" to 3. More on this later.)
  - "hello"'s type is **string**
  - You can ask Python for a value's type

    >>> type(3.5)

(Note: don't use commas in big numbers: 1,000,000 is not million in Python!)

Ch 1: natural vs formal languages

One major difference: formal programming languages are unambiguous.

Natural languages are ambiguous, which can be annoying and confusing but also enables playfulness and creativity (jokes, poetry, etc.)

Natural languages have lots of redundancy – many ways to express the same thing. Programming languages also redundant but less so.

Programming languages are dense but they can be directly and unambiguously translated into basic machine-level operations

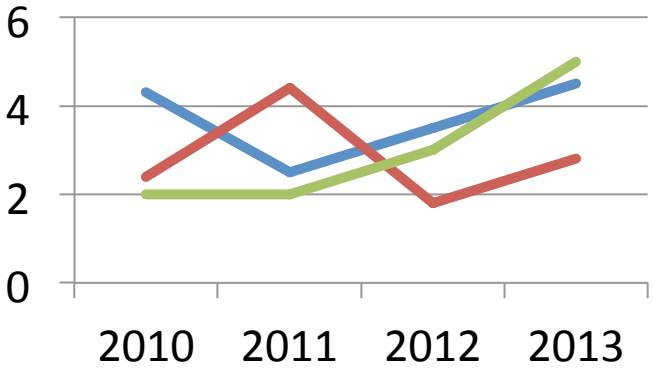
> *Computer systems read them quickly but humans need to read them too. STRONG RECOMMENDATION: get in the habit of reading programs slowly, carefully, and critically. Make sure you understand what they really say rather than what you believe or hope they say)*

- IC Arrest Blotter
- Demo: blotter.py

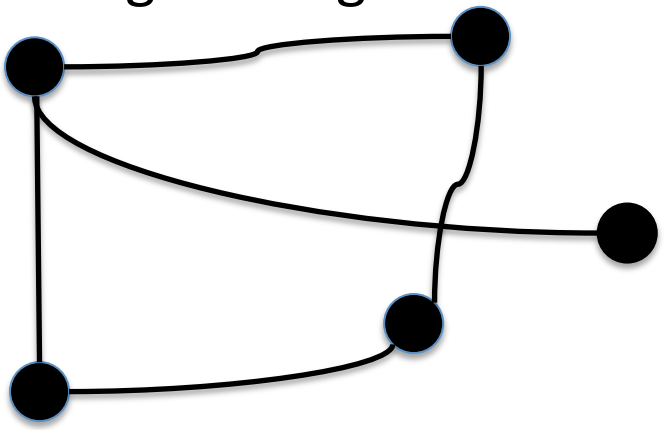
  Later in the semester you should be able to make this program.

Problems from end of last class (*examples of computational complexity)*

- Euler vs Hamiltonian circuits and TSP. *Graphs* in computer science: different notion than the graphical plots taught in high school



vs.



1. Can you find a path that traverses every edge (connection) exactly once?
2. Can you find a path that visits every node (vertex) exactly once?
3. If the edges are assigned "weights" (representing distance/ cost to traverse), can you find path of minimum cost that visits every node exactly once?

Are these similarly computationally hard or easy?

Solve for extra credit?

(Euler paths/circuits – i.e. problem 1 - examples drawn/shown via doc cam)

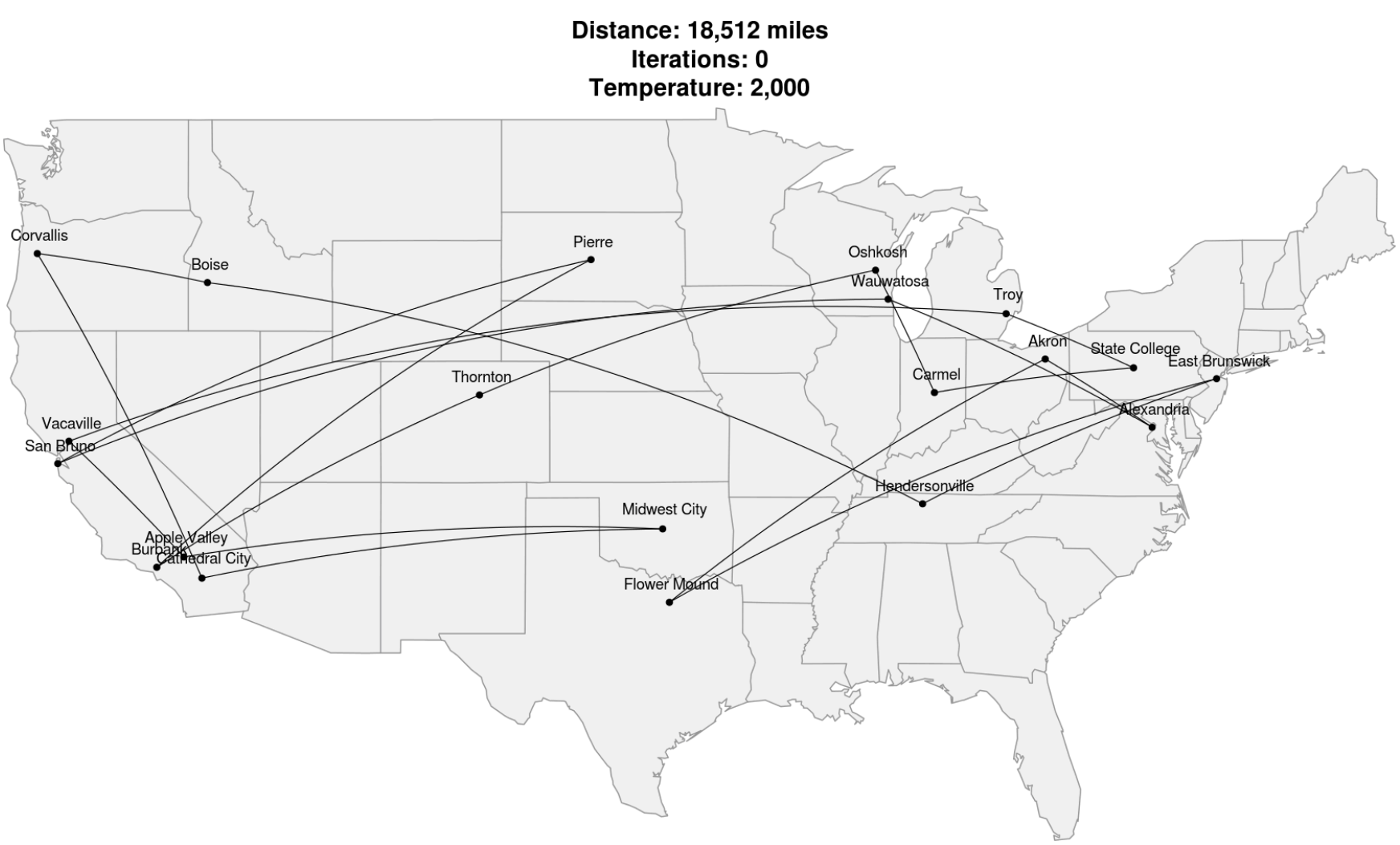
Problem 1 is easy! Super fast, efficient algorithm

Problems 2 and 3 are very hard! No efficient optimal solutions are known (and might not exist)

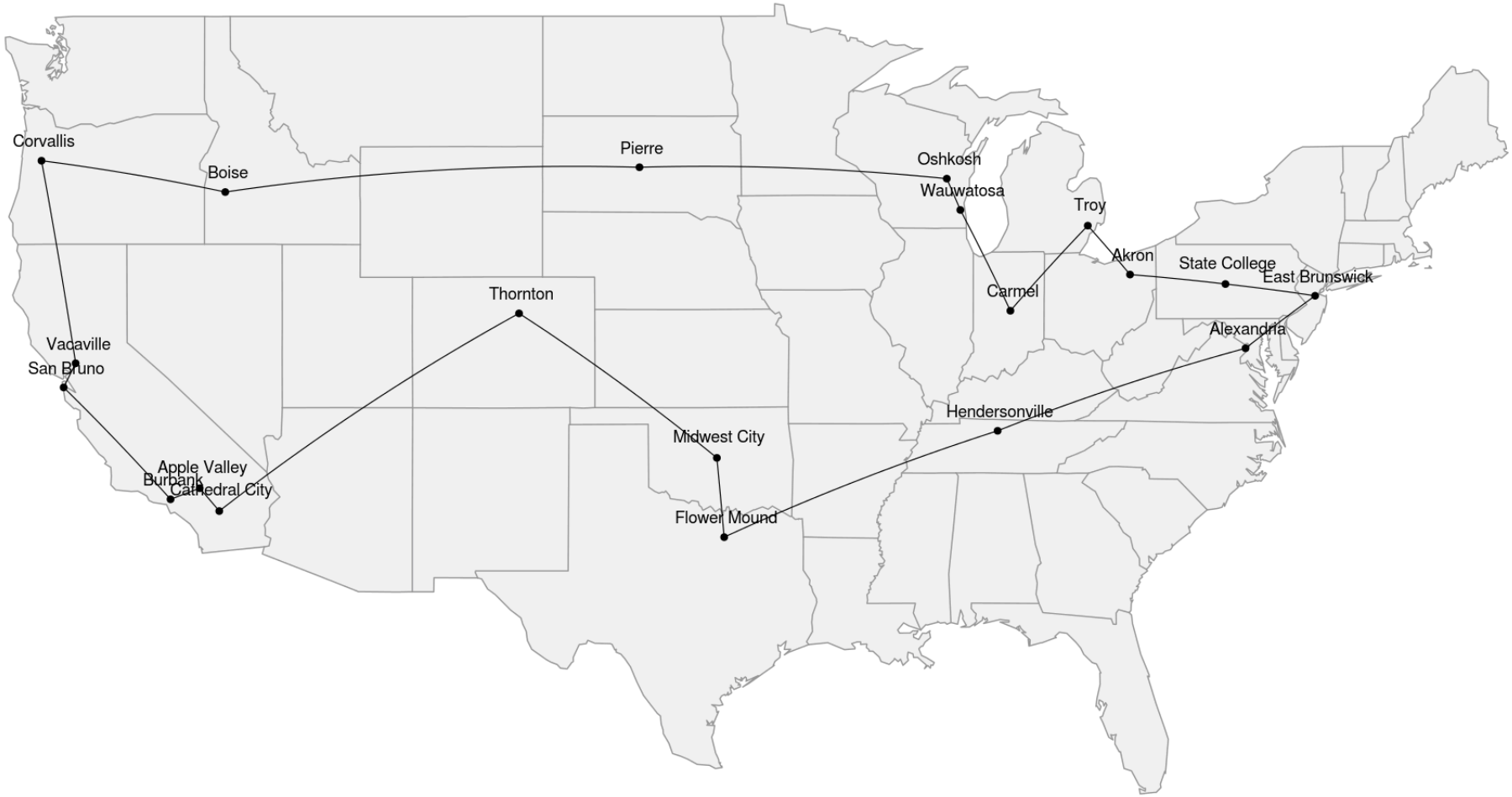
Next time

- More on values, expressions, and types
- Functions
- For next time, read: Ch2, and Ch3 up through "Adding New Functions"

- The next two slides were not shown in Lecture 2 but contain links to web pages/videos giving nice information about the difficulty of the third path-finding problem



Source:
http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/



Source:
http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/

See also nice short video at:
http://www.youtube.com/watch?v=SC5CX8drAtU&list=PLxH6ufuE9gKtM5-bbFMTp_1-avAN-iuiq&index=3