Intro to Programming

Summer 2022

About this Course

- We will learn to design and implement computer programs in order to solve problems.
- Assumption: you have *never* written a line of code before.

- output
- variables
- errors
- reserved words
- operators + computations
- constants
- built-in math functions
- conditional execution

- iteration
- programmer-defined functions
- strings
- lists (1D and 2D)
- Files
- Exceptions
- Dictionaries
- Objects and Classes

Programming and CS

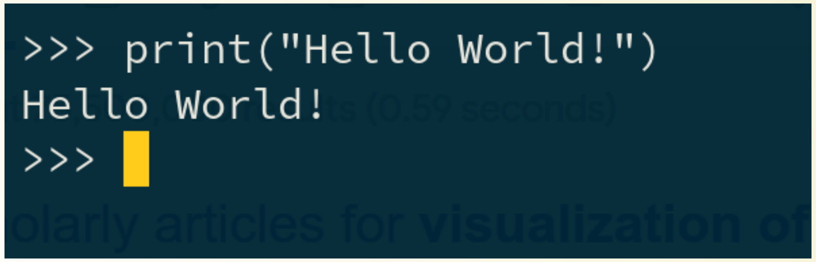
Computer science explores what it's possible to do with computers. Programming is getting the computer to actually do it.

It might be possible to create a self-driving car, but without the programming and software to control the vehicle, it's not going to work!

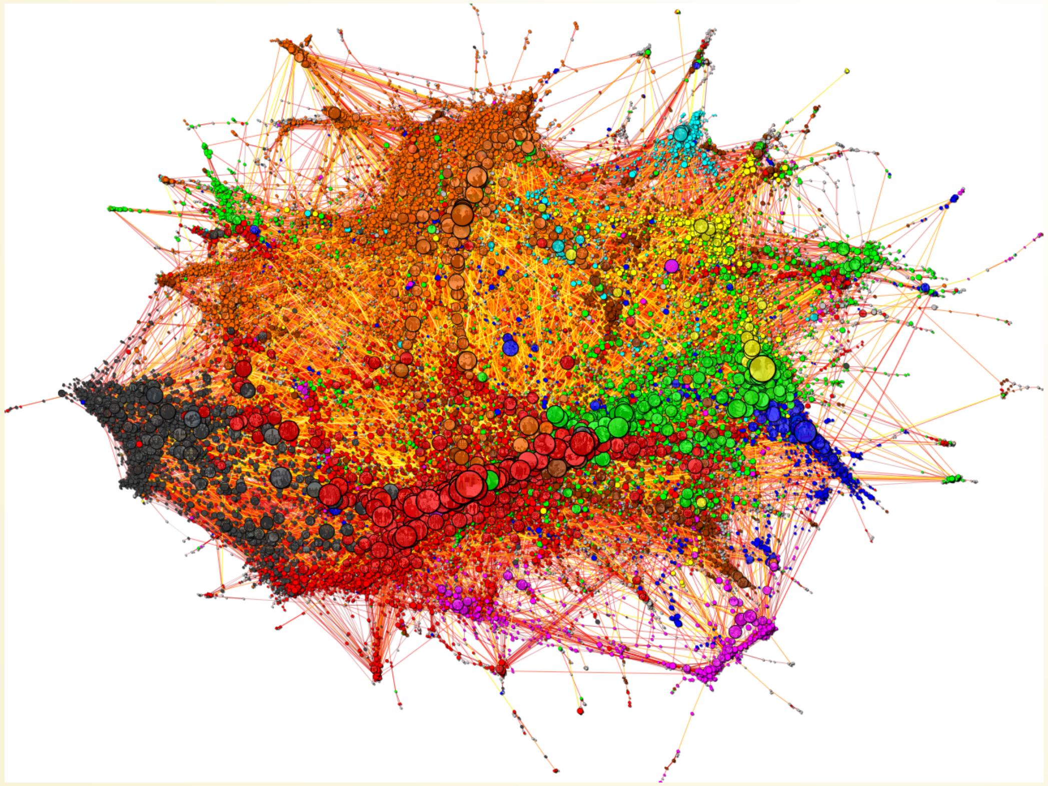
Programming

Starts off simple....

...but gets really hard, *really* fast.



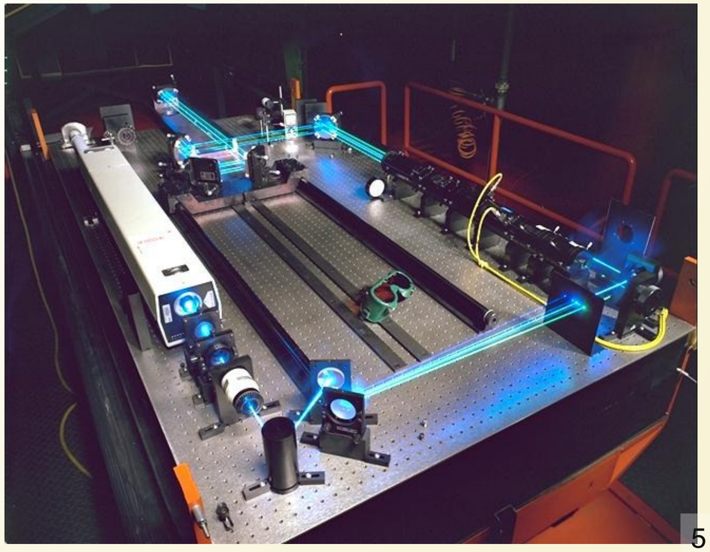
```
>>> print("Hello World!")
Hello World!
>>> 
```



Given the structure to the left, where each line has a cost associated with it, find the cheapest way to travel between any two circles.

About Me

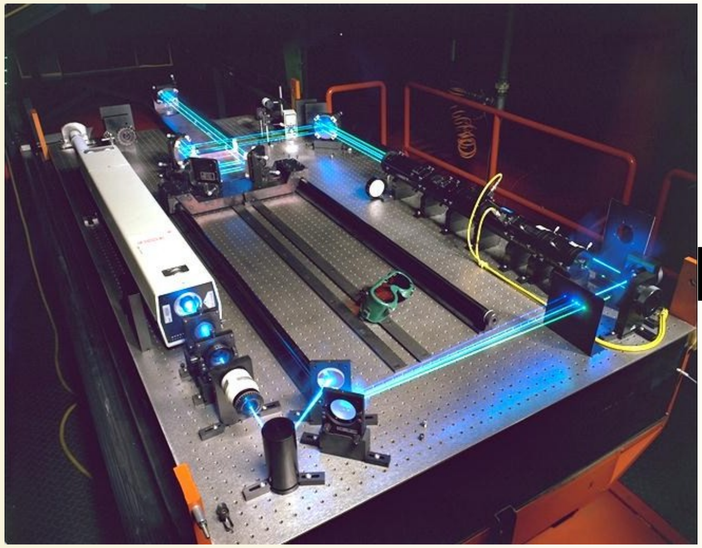
- Graduate Student in CS Department
- Undergraduate: UC Merced
  - Chemistry (spectroscopy + nanoparticle chemistry)
  - Applied Mathematics
- Helped teach (TA) several courses in the CS department

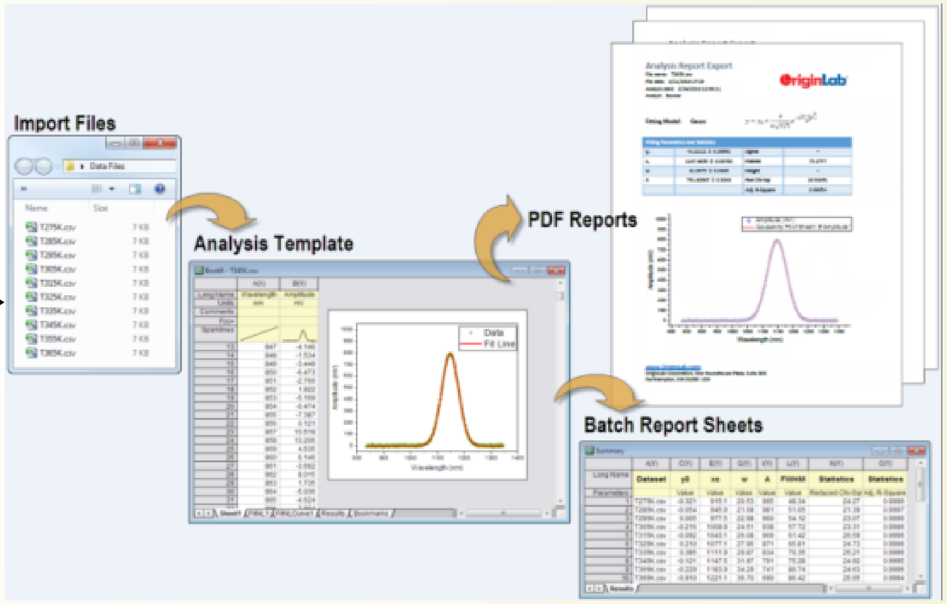


How I Got into Computer Science

Very late entry into the field

Initially focused around supporting experiments





Soon became interesting in constructing systems as an activity.

Past Computational Projects

- Can we identify what medication a particular pill corresponds to from a photo? (UC Merced)
- Does allowing cross-disciplinary study in academics create more innovation? (UC Merced)
- How can we accelerate classical machine learning algorithms to work on huge image systems? (UIUC)
- How should we design large-scale systems in the presence of hardware errors? (ORNL)

Hamming's Maxim

“The purpose of computation is ***insight****, not* ***numbers***

Course Structure

Materials

There is no required book for this class. The syllabus has online resources if you would like them.

We will write some programs together in class--these will be uploaded to Canvas.

Grading

Homework (45%)

There are two homework assignments a week.

- Assignment "A" released Monday after class, due end of day on Friday
- Assignment "B" released Wednesday after class, due end of day the next Monday

Grades will be based off of the 1/3/5 system:

- 1: Made an effort, but didn't really complete it
- 3: Had a general idea of what was going on
- 5: Everything is correct

Note: I will leave feedback for you on your programs. **YOU NEED TO READ THIS**.

Exam (15%)

There will be a midterm at the start of July and a final exam at the end. Each will be worth 15% of your grade.

A mix of multiple choice/short answer. Will generally have three components:

- Answer questions about the Python language
- Answer questions about code in the Python language
- Write Python code

Final Project (20%)

A project chosen by you! Can work alone or in pairs.

You should start thinking about project ideas shortly after the midterm (I will remind you about this when the time comes)

The idea needs to be approved by me, to make sure you're not trying to do something *way* too hard or too easy.

Last day of class will (partially) be used to present and demo final projects. This presentation will be part of your grade.

Participation (5%)

Based on how much you participate in class.

May be extra credit in this section depending on participation levels.

Assignments

- Start off simple, but get challenging *fast*
- **Individual assignments**
- Turn in the right thing: fill out the header, and make the format correct, or you will lose points
- Assignments come out after class, and are due at the end of the day on their due date.
  - Designed so that you will have two class periods to ask for help for every assignment.
  - This is going to be a little weird around the midterm + 4th of July.
- Graded on a combination of *correctness* and *program hygiene* (coding style, best practices, etc.)

Personally, I would suggest getting A done on Tuesday and B done on Thursday so you can have the weekend off!

Insight Questions

Starting with Homework 1b, the assignments will start asking you to do a little work beyond simply writing a correct program.

We will try to gain *insight* on the nature of the problem. This may take the form of seeing how the problem works, how long it takes to run, how much space it takes, or other such questions.

Academic Honesty

In professional engineering contexts, signing your name on something means you *personally* attest to its correctness

“ Gillum and Associates failed to review the initial design thoroughly...

“ Reports...cited a feedback loop of architects' unverified assumptions, each having believed that someone else had performed calculations and checked reinforcements

Result: Gilium and engineers at the company were **criminally charged**, and lost their licenses to practice engineering.

Honesty in Programming

It becomes much easier to lie about the big things when you've already lied about smaller things.

All code that is not your own **must** be tagged with the following information:

- How much of the code is not yours
- Where it came from
- Why you are allowed to use it

Do not write anyone else's code for them. When helping, make sure that neither of you are actively writing code.

ASK

Office Hours

Getting Help

Post to Piazza

- You can make yourself anonymous to other students
- You can post an instructor-only question
- Do not post more than 2 lines of code on a public post
- Any non-personal course questions sent by email will be redirected to Piazza

Succeeding

Programming is not a spectator sport.

You cannot learn to program by attending class or watching YouTube any more than you can learn to ride a bike by doing these things.

In order to learn to program, you need to actually *write programs.*

Succeeding

Learning something new is *scary!*

- Have to try something without know if you can do it or not
- If you can't, feels *really* bad

FIRST PENGUIN AWARD

Digital Garage

But our goal isn't to feel good all the time....our goal is to learn programming!

It is **okay** to make mistakes.

Generally, people who have made more mistakes know more things, because they've just *done more stuff*.

Succeeding

This entire course is cumulative. The material builds on itself. If you're lost, it's not going to get better unless you try to make it better.

- Come to class and participate!
- Start on homework early
- Get help from me or on Piazza when you get stuck.
- Ask questions when you need help
- Do more practice problems!
  - https://codingbat.com/python
  - Selection on Canvas

Succeeding

- Cannot succeed by brute-force memorization.
- Programming is a skill.
- Learn by doing.
- If you are new to programming, I strongly recommend doing lots of practice problems.

Suggested Startup

- Make sure you can access Canvas and Piazza.
- Download and install Python (see Assignment 0)
- Review Bill Young's How to Succeed in CS 303E. It's not the exact same class as ours, but the advice there applies to this class.
- Check out Julia Evans's suggestions on how to ask good questions

A Quick Demo!