The Intro to Intro

CS 110: Introduction to Computer Science

What is Computer Science?

- Computer science is the study of **algorithms**.
- Computer science is the study of **building digital systems**.
- Computer science is the study of how **people interact with technology**.
- In this class, we will study all of these different aspects of computer science.
- To facilitate our study, we will study the **Java** programming language.

Programming Languages

- In order to get a machine to do something we have to communicate with it.
- You already communicate with machines all the time.
  - *e.g.*, Flipping light switches communicates on/off
- Computers are more complicated machines, so they're require more complicated languages than "flipping switches"

Programming Languages

- Programming languages, unlike human languages, are **more precise**, **have fewer words**, and have **simpler grammar**.
- This makes them easier to learn, but less forgiving with errors.
- Also, the more precise of a problem we are trying to solve, the easier it is.

Little Snippets

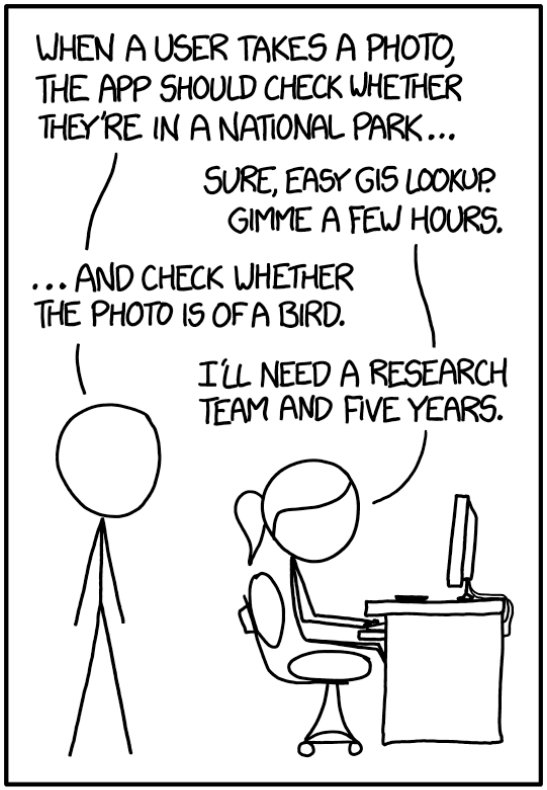
- Solve 334 plus 24 divided by 7.

```
int x = (334 + 24)/7;
```

- Print "big" if 4 is greater than x, else print "small" and "number"

```
if (x > 4) {
    Magic.println("big");
} else {
    Magic.println("small");
    Magic.println("number");
}
```

Limitations



https://xkcd.com/1425/

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

How do you write a program?

Step 1: Come up with the idea

- Programs are written to solve problems.
- So, before you begin, you need to be clear about what you want to solve.
- The problems can be big or small
  - **e.g**. Need to find highest grade on a test
  - **e.g.** Need to enable doctors to store all patient medical records electronically.
- We'll focus on smaller problems in this class

Step 2: Sketch out a solution

- Jot down a few approaches for how you can solve a problem.
- **Try and work through small example problems**.
- If a problem is too big, break it down into smaller problems and solve each of those.
- Your sketch will not be perfect.
- This sketch is called pseudo-code (*i.e.*, fake-code).

https://goo.gl/9Wo2Lz

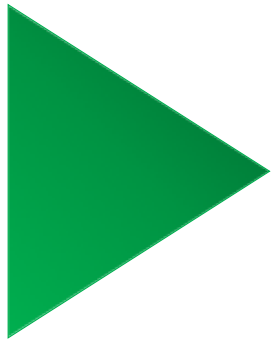
Step 3: Start coding

- Take your sketch and start translating it into actual Java code.
- Build incrementally.
- Don’t be obsessive (yet).
- If you know you need work leave a comment.
- Try and solve one problem at a time

```
Magic.println("Enter a number!");
int x = Magic.nextInt();
if(x < 5) {
    Magic.println("Number is small");
} else {
    Magic.println("Number is big");
}
```

Step 4: Try it out

- Unlike most other disciplines, **you** can test to see if your assignment works.
- So, run your code, run it often.
- When you run it use example values.
- You want to find problems before you turn it in.
- So, use values that you think will break your code.



Step 5: Fix problems

- Sometimes your code won't run.
  - **That's OK!** Java will tell you the line where it breaks
  - *Be careful, the problem could be **above** that line.*
- Sometimes, your code won't produce the correct result.
  - **That's OK!** Nothing magical happens in a program. Look at your code in reverse. Find out where the data went from good to bad.
- Sometimes, your code will crash.
  - **That's OK!** Look the line where it crashes. Did you call the wrong thing? Did you pass bad data? Nothing is magic. Again, trace your code back by hand find where the mistake came.

Step 6: Cry

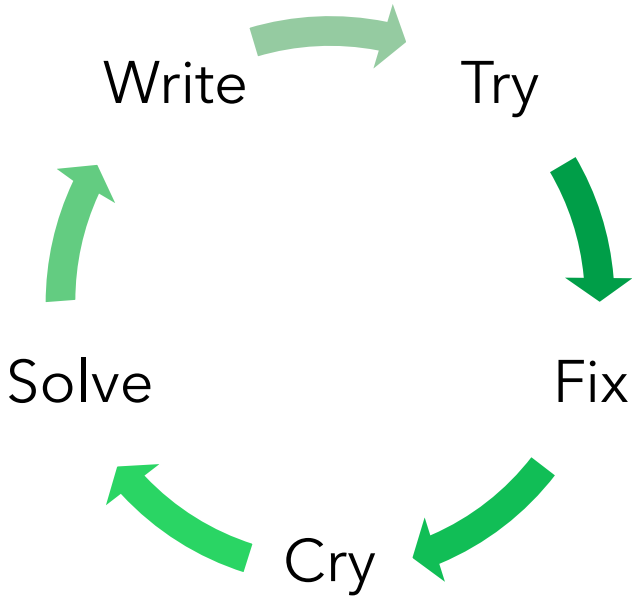
- You won't get everything right away.
- It will seem like it's an impossible problem.
- You'll think to yourself….
  - *I AM NEVER GOING TO GET THIS FREAKING PROBLEM. HOW AM I SUPOS…..*

Step 7:Solve your problem

- *Oh, that was the solution!*
- *Why didn't I see that an hour ago?*
- *Who cares!*
- *I'm going to tell everyone!*
- ***SO COOOL!***

Step 8: Repeat steps 3-7

- Now that you've gotten past that one problem, do it again, and again, and again, and again until you're done.
- What you will make will be amazing!



Syllabus