THINKING LIKE A COMPUTER SCIENTIST

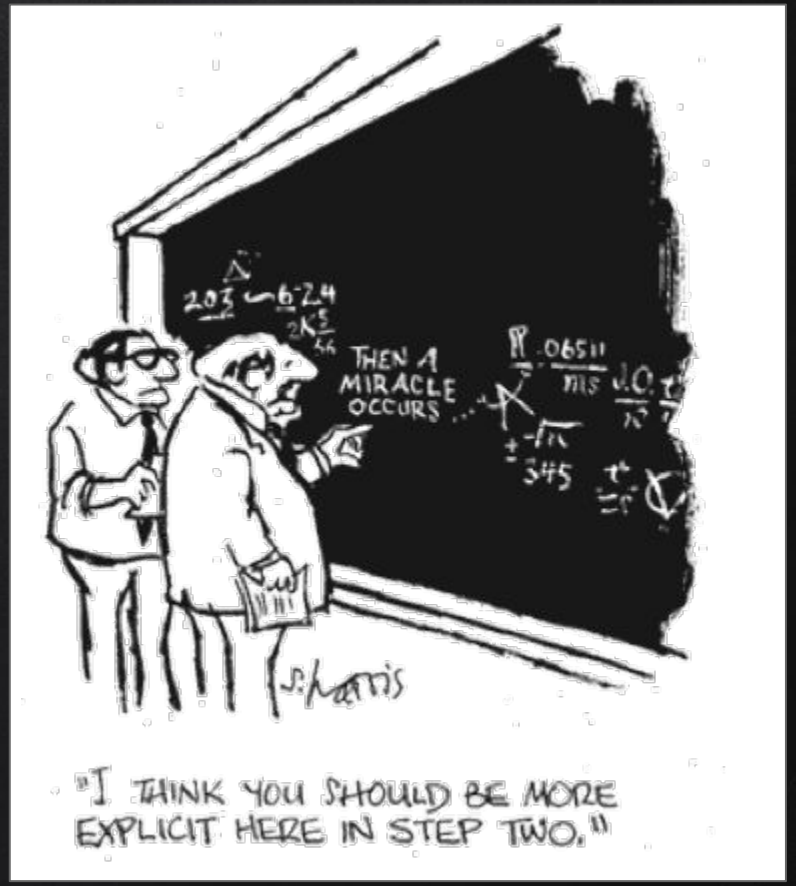
Top down Problem Solving

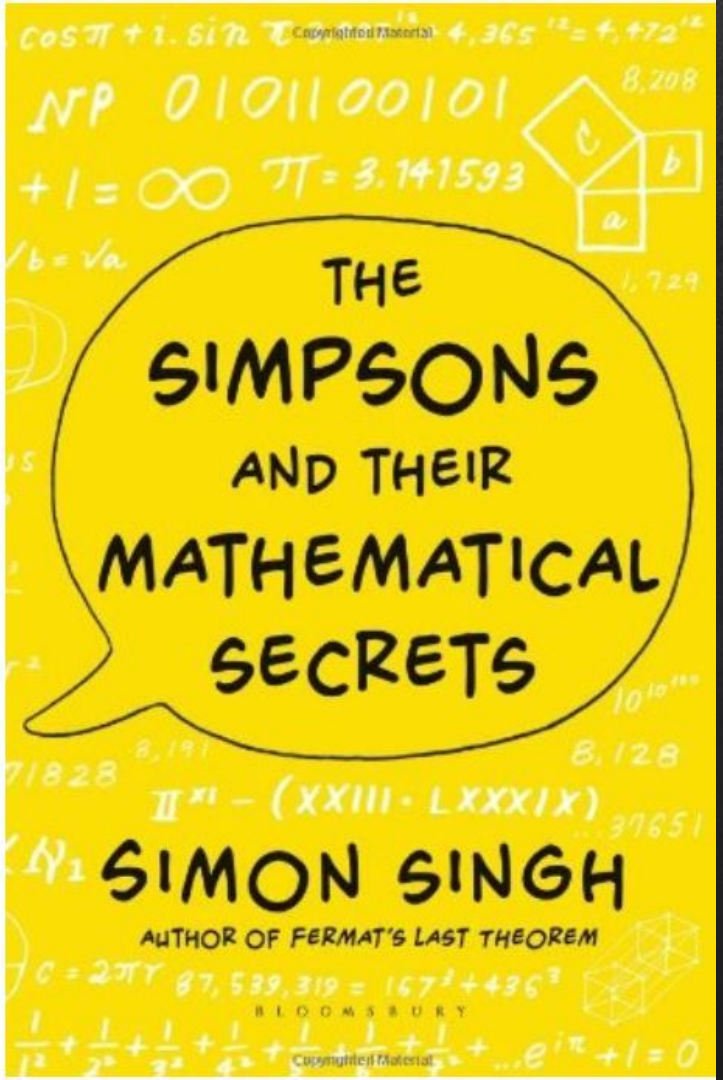
Announcement

- Today's Bei office hour has to be cancelled (5–6 p.m.) If you would like to meet me this week, please send me an email.
- The 2 students who volunteered during the Imitating Imitation game should contact TA Ross at cs1060 AT spam.im to get 2 extra credits. Please do this today.

Techniques for problem solving in general

- CS is about solving problems
- What makes solving problems difficult?
- Limited resources and tools.





The chef in a pancake restaurant making a stack of pancakes is a bit sloppy. The pancakes are all different sizes and stacked on the plate in random order. The waiter doesn't like this so he flips them into a more pleasing arrangement, with the largest on the bottom and the smallest on the top. Flipping is done by inserting a spatula under a pancake and flipping it and any that are above it, the other way up. He wonders, what is the maximum number of flips that are needed for a particular pile to be rearranged nicely, i.e. the worst case scenario. These are called "Pancake Numbers".

How do you solve problems?

For example spatula pancake sorting problem…

A problem in recreational mathematics and computer science…

- What is your strategy (algorithm)?
- How do you know your answer is correct?
  - What was the first relationship we found between # of pancakes and # of flips needed?
  - How did we redefine it?
- BOARD: pictures for 2, 3 pancakes

One strategy

- Flip the biggest pancake on top
- Then it goes to the bottom, ignore
- Repeat the above strategy for the remaining of the pancake
- Bill Gates wrote a paper on Pancake Sorting (his only research paper I believe)...

With 2 pancakes, how many flips do you need to get them in order in the worst case scenario?

The pancake # for 2 pancakes is 1

The pancake # for 3 pancakes is 3

The pancake # for 4 pancakes is 4

The pancake # for 5 pancakes is 5

The pancake # for 6 pancakes is 7

The pancake # for 7 pancakes is 8

The pancake # for 17 pancakes is 19

The pancake # for 20 pancakes is

Unknown

Why is this a hard problem?

Need to look at all arrangements of 20 pancakes: 20! =

2,432,902,008,176,640,000

roughly 2 billion billion

Polya's How to Solve It

In 1945, George Polya wrote the book that is the classic description of the problem-solving process:

How to Solve it: A New Perspective of Mathematical Method

1. Understand the problem
2. Devise a plan
3. Carry out the plan
4. Look Back

1. Understand the problem

Why is this a separate step, isn't it obvious?

This is the step for asking questions:

- What do I know about the problem? What don't I know?
- What does the solution look like?
- What sort of special cases exist?
- How will I recognize when I have found the solution?

As needed, drew figures and introduction notation.

2. Devise a Plan

Do not reinvent the wheel...

- If a solution already exist, use it
- Look for the familiar. Can you relate this to a similar problem?
- If a solution to a similar problem exists, start from there

Divide and conquer

- Break a large problem into smaller units that you can handle

3 & 4. Carry out and look back

Carry out the plan:

- Check / execute each step of your solution
- Ensure that result / output after each step is correct

Look back:

- Is the final result correct?
- If not, revisit each phase of the problem-solving process to find your mistake
- You may have to go back to the beginning (did you understand the problem)?

An extensive list of approaches

- Abstraction: solving the problem in a model before applying it to the real system
- Analogy: using a solution that solved an analogous problem
- Brainstorming: (especially among groups of people) suggesting a large number of solutions or ideas and combining and developing them until an optimum is found
- Divide and conquer: breaking down a large, complex problem into smaller, solvable problems
- Hypothesis testing: assuming a possible explanation to the problem and trying to prove (or, in some contexts, disprove) the assumption
- Lateral thinking: approaching solutions indirectly and creatively
- Means-ends analysis: choosing an action at each step to move closer to the goal
- Reduction: transforming the problem into another problem for which solutions exist
- Research: employing existing ideas or adapting existing solutions to similar problems
- Root cause analysis: eliminating the cause of the problem
- Trial-and-error: testing possible solutions until the right one is found

Real world Examples

Making a peanut butter and jelly sandwich

- Understand the problem (ask questions)
- Devise a plan (look for the familiar, divide and conquer)
- Carry out the plan (check the result of each step)
- Look back

PageRank and Google: Sergey Brin and Lawrence Page, 1998

The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as Yahoo! or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10^{100} and fits well with our goal of building very large-scale search engines.

- **The Anatomy of a Large-Scale Hypertextual Web Search Engine**

Divide and Conquer

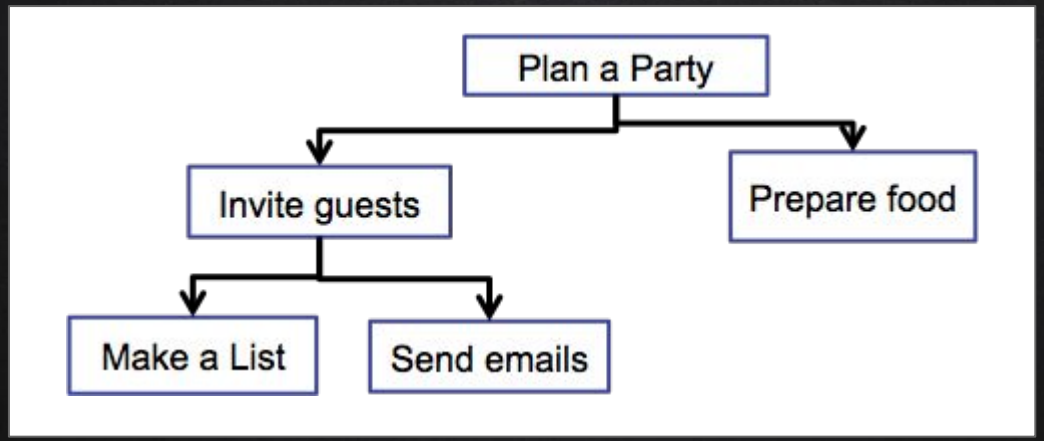
Divide and conquer

People are only good at holding a few things in their heads at once...

- Hide the details until needed
- Use abstraction
- Top-down design is a divide and conquer strategy

Top-down design

- Breakdown the problem into a set of subproblems, and more sub-subproblems, until no further decomposition is necessary
- Combine subproblem solutions to form the overall solution
- An example:

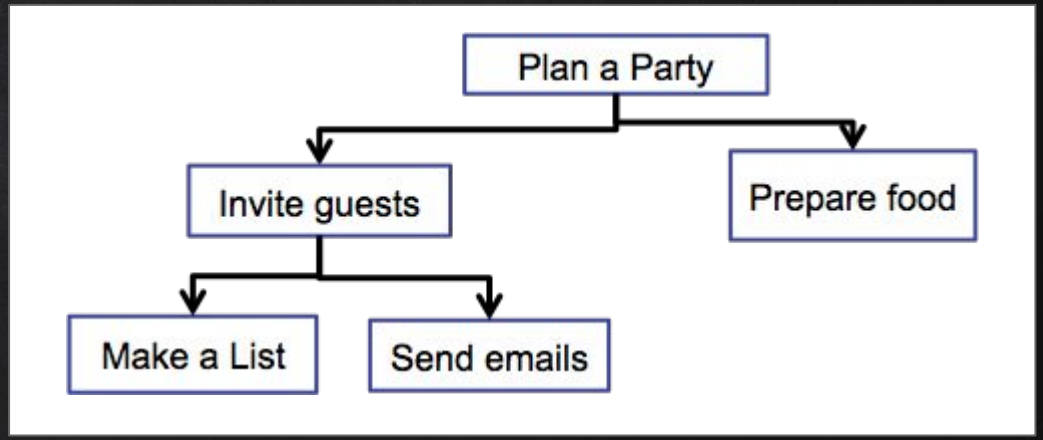


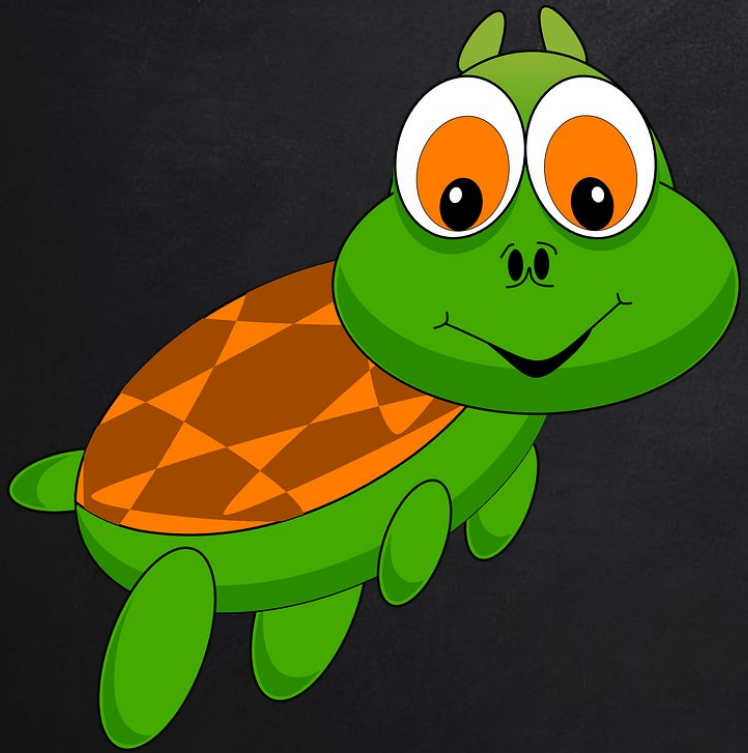
abstract

particular

Writing the solution as an indented list

- Plan a party
  - Invite guests
    - Make a list
    - Send emails
  - Prepare food





credit: https://pixabay.com/static/uploads/photo/2013/07/12/16/53/turtle-151431_960_720.png

Turtle Example

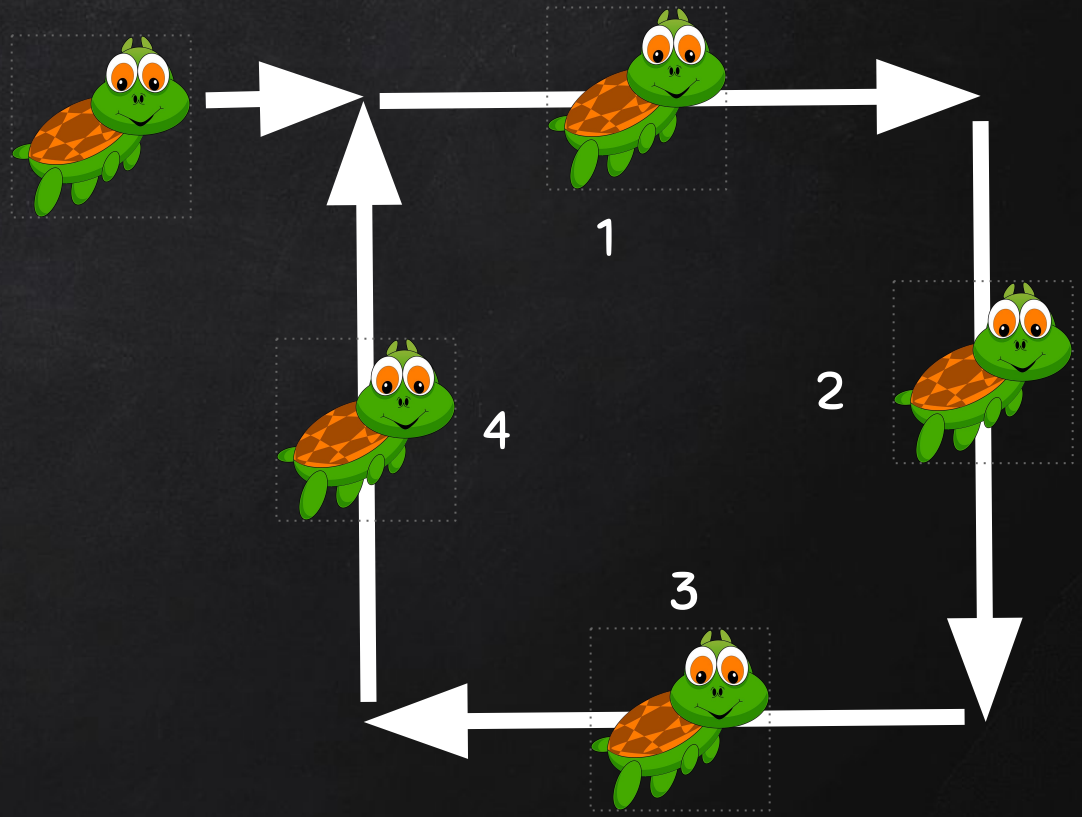
I own a pet turtle named Johnny...

- My turtle can follow very simple instructions, such as walk in straight line, and turn with some angle.
- I put a paint box on the back of my turtle
- The problem: I would like my turtle to draw a square with edge length 100 steps on the ground by following my instructions
- Devise a plan: what should my step by step instructions be for my turtle?

What are my instructions for the turtle?

1. Walk forward 100 steps
   Turn right (90 degree)
2. Walk forward 100 steps
   Turn right
3. Walk forward 100 steps
   Turn right
4. Walk forward 100 steps
   Turn right

Done!



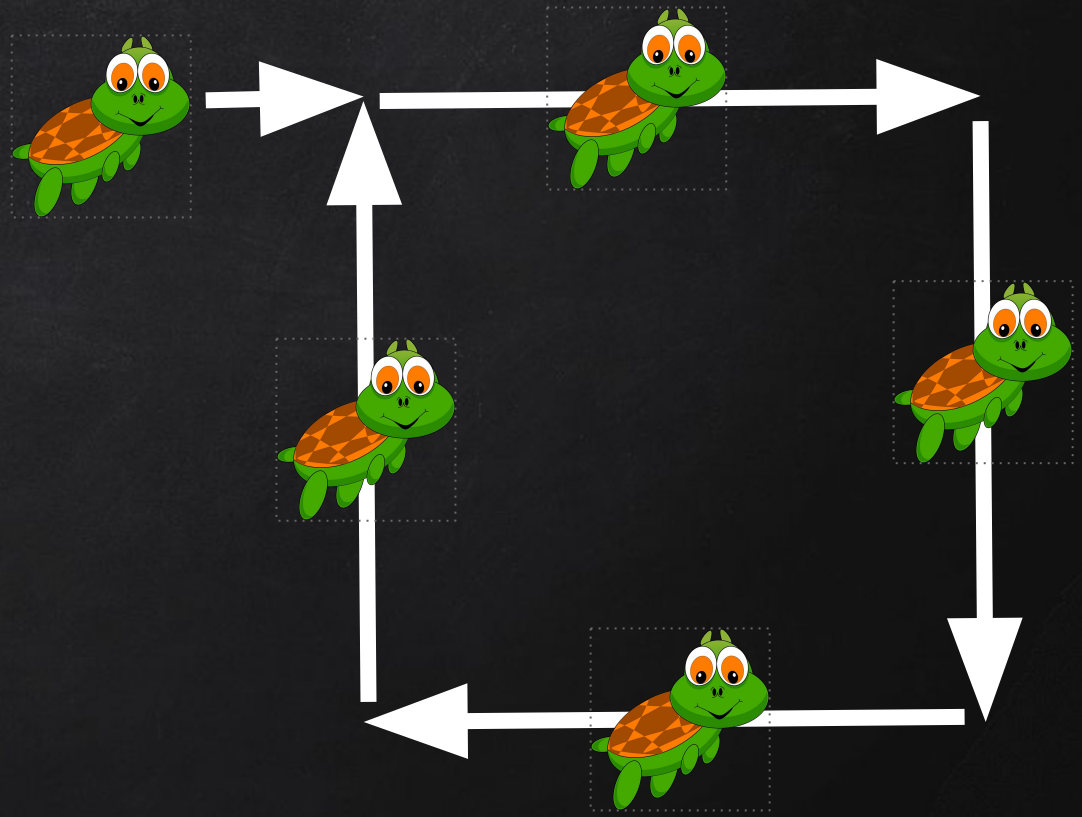
Alternatively

Turtle, please repeat my instructions below 4 times:

Walk forward 100 steps

Turn right (90 degree)

Done!



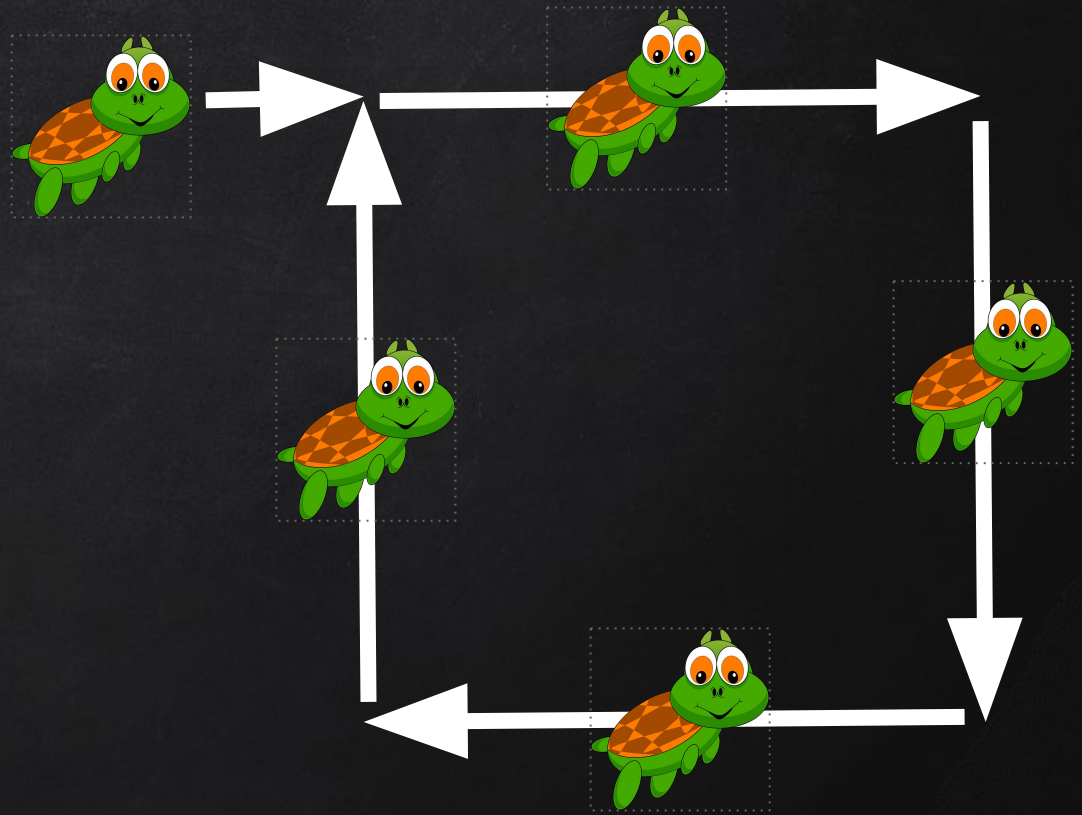
Python Turtle

Let's change our language a little bit

```
import turtle
johnny = turtle.Turtle()
for i in range(0,4):
    johnny.forward(100)
    johnny.right(90)
```

http://www.skulpt.org/

http://interactivepython.org/courselib/static/thinkcspy/index.html



Guess what?
You just wrote your first computer program in Python!

We are going to learn some light coding in Python just like this...

Let's change things a bit

```
import turtle
johnny = turtle.Turtle()
for i in range(0,6):
    johnny.forward(100)
    johnny.right(60)
```

http://www.skulpt.org/

http://interactivepython.org/courselib/static/thinkcspy/index.html

Let's look at this Python program more carefully...

```
import turtle                    ──────►  a library of tools/personalities
johnny = turtle.Turtle()         ──────►  johnny is now a turtle object
for i in range(0,6):             ──────►  Loop / repeat
    johnny.forward(100)          ──────►  johnny performs some
    johnny.right(60)             ──────►  pre-defined tasks (functions)
```

Coming up next:

Do u speak binary?

Coding Basics

THANKS!

Any questions?

You can find me at
beiwang@sci.utah.edu

http://www.sci.utah.edu/~beiwang/teaching/cs1060.html

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash