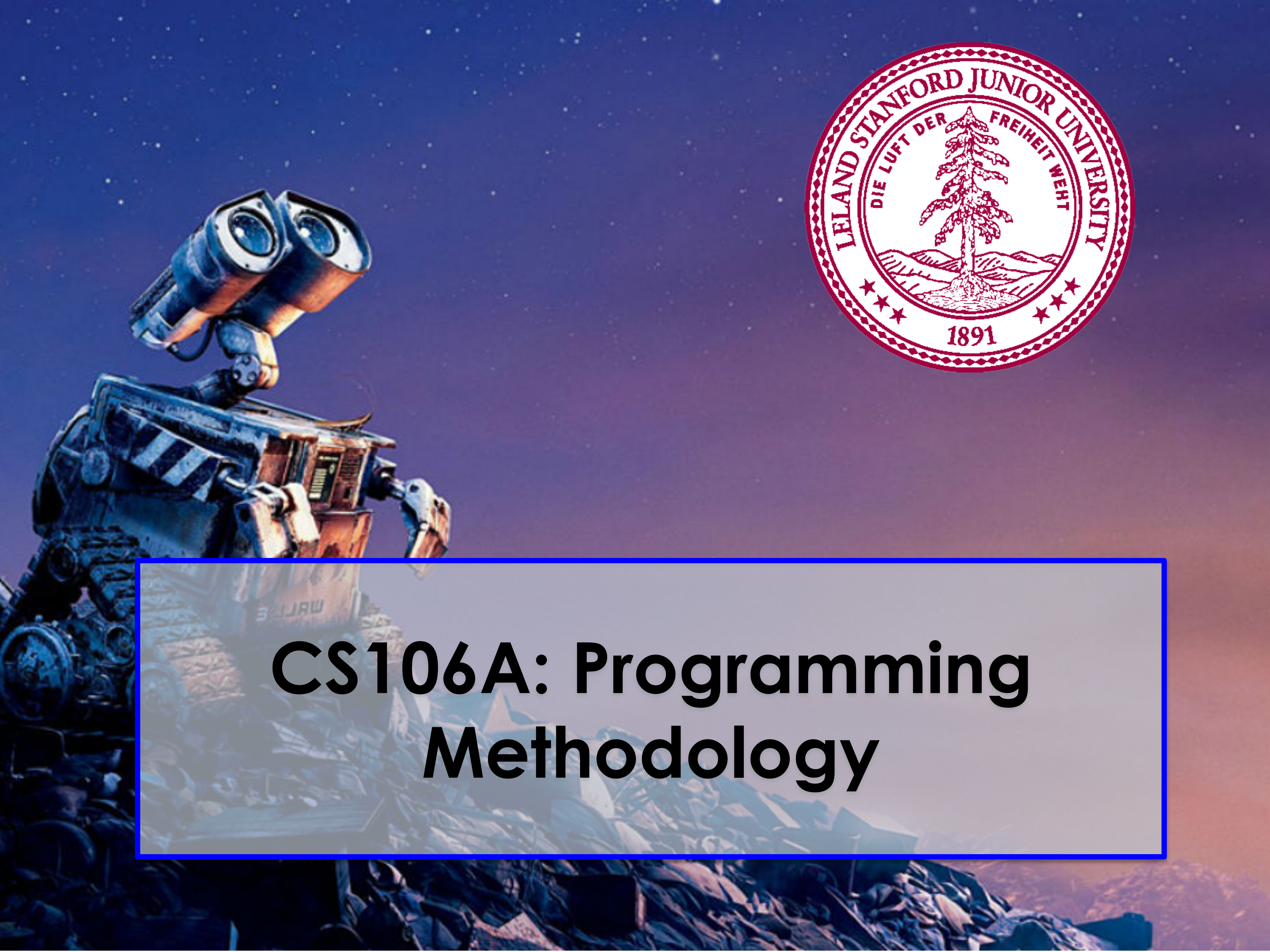


CS106A: Programming Methodology

Chris Gregg

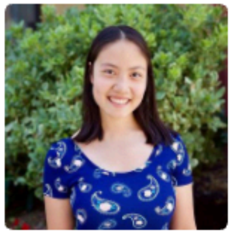
- Electrical Engineering undergraduate degree
- Spent 7 years on active duty in the Navy as a cryptologist, and then 15+ years in the Navy Reserves.
- Masters in Education
  - Taught high school physics for many years in Boston and Santa Cruz, CA
- Ph.D. in Computer Engineering
- Taught at Tufts and have been at Stanford for four years
- I love tech, tinkering, and teaching

Head TA: Wil Kautz

Section Leaders

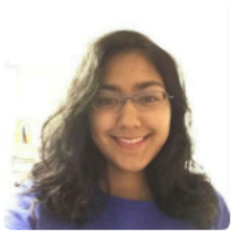
Luciano Gonzalez

Maggie Davis


Marilyn Zhang

Meng Zhang

Nidhi Manoj


Niki Agrawal

Peter Maldonado

Rachel Gardner

Rhea Karuturi

Robbie Jones

Ruiqi Chen

Semir Shafi

Shanon Reckinger

Tessera Chin

Thariq Ridha

Vineet Kosaraju

* Actually some of last year's section leaders

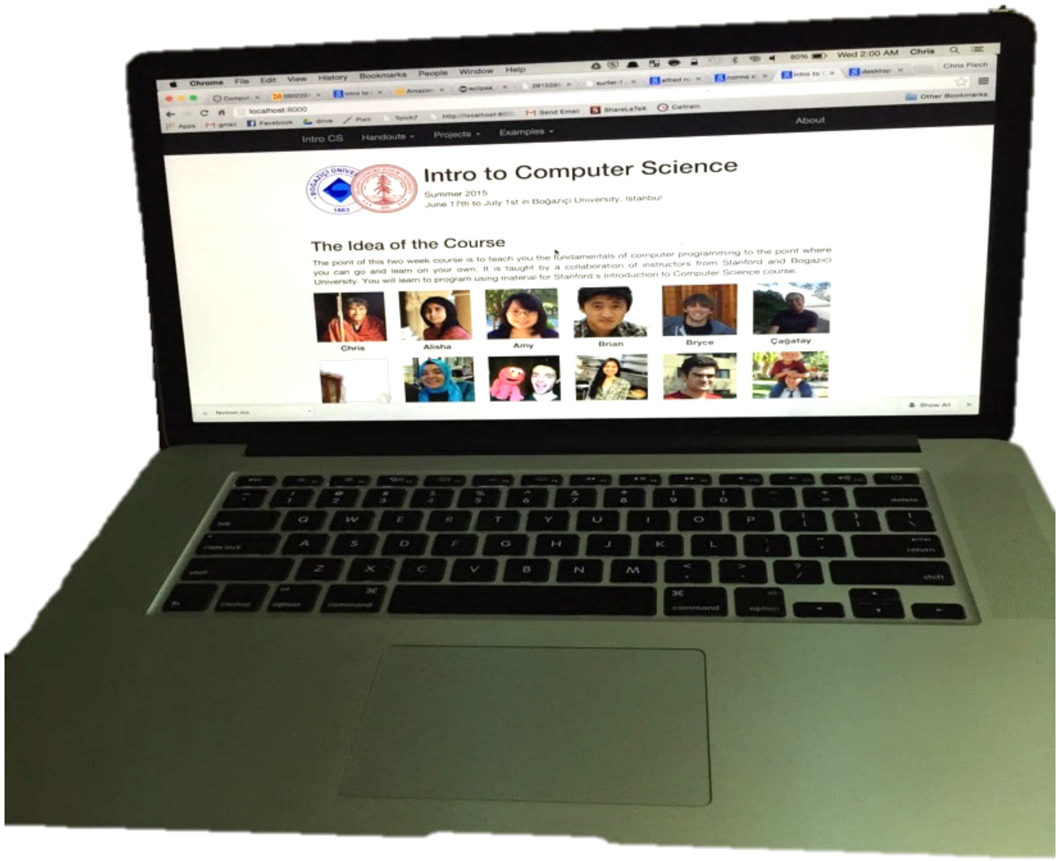
Course mechanics

(This is a brief version.
Please read the handout for full details).

Course Website

http://cs106a.stanford.edu

Prerequisite Test



Intro to Computer Science

Summer 2015

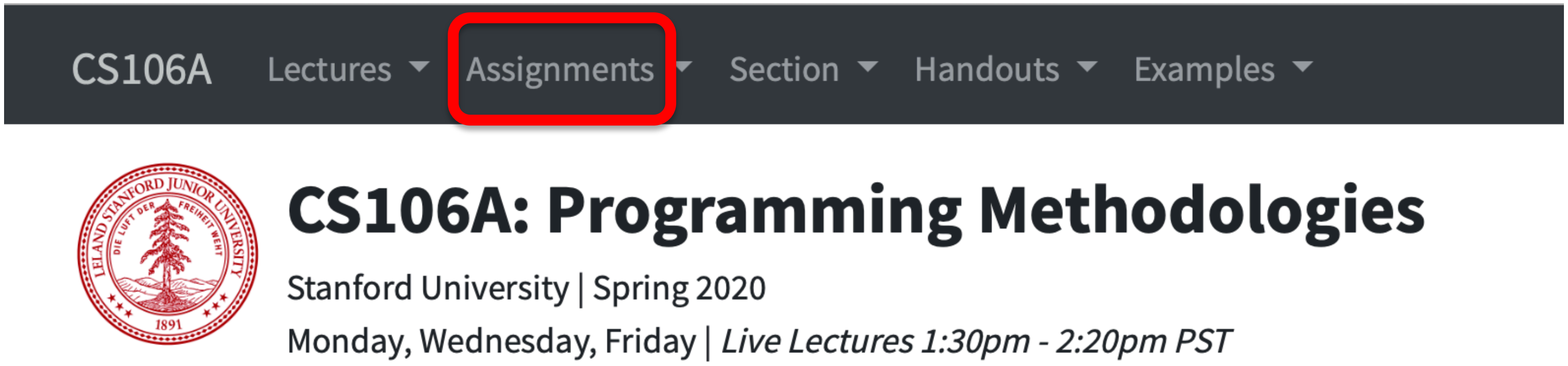
June 17th to July 1st in Boğaziçi University, Istanbul

The Idea of the Course

The point of this two week course is to teach you the fundamentals of computer programming to the point where you can go and learn on your own. It is taught by a collaboration of instructors from Stanford and Bogazici University. You will learn to program using material for Stanford's Introduction to Computer Science course.

Getting To Know You

- Assignment #0 on website ("Who are you?")



CS106A: Programming Methodologies

Stanford University | Spring 2020

Monday, Wednesday, Friday | *Live Lectures 1:30pm - 2:20pm PST*

- Please fill this out so we can know who you are and where you are (for planning purposes)
- We also want to know a bit more about you!

Lectures and Sections

- Lectures MTuWTh 10:30am-11:20am
  - Will be recorded (available on Canvas)
- Weekly 50-min section led by awesome section leaders (the backbone of the class!)
  - Section signups are already open -- see the class webpage (not Axess)
  - Signups will close on Tuesday at 5pm. Sections start this week!

Office Hours

LaIR: evenings Sunday through Wednesday
(starting Tuesday)

Grading Scale

Functionality and **style** grades for the assignments use the following scale:

++	A submission so good it "makes you weep"
+	Exceeds requirements (and has great style)
✓+	Satisfies all requirements, with good functionality and style
✓	Meets the requirements, but perhaps with small problems
✓−	Has some somewhat serious problems
−	Is worse than that, but shows real effort and understanding
−−	Better than nothing

You are only competing against yourself.

Meeting with your Section Leader

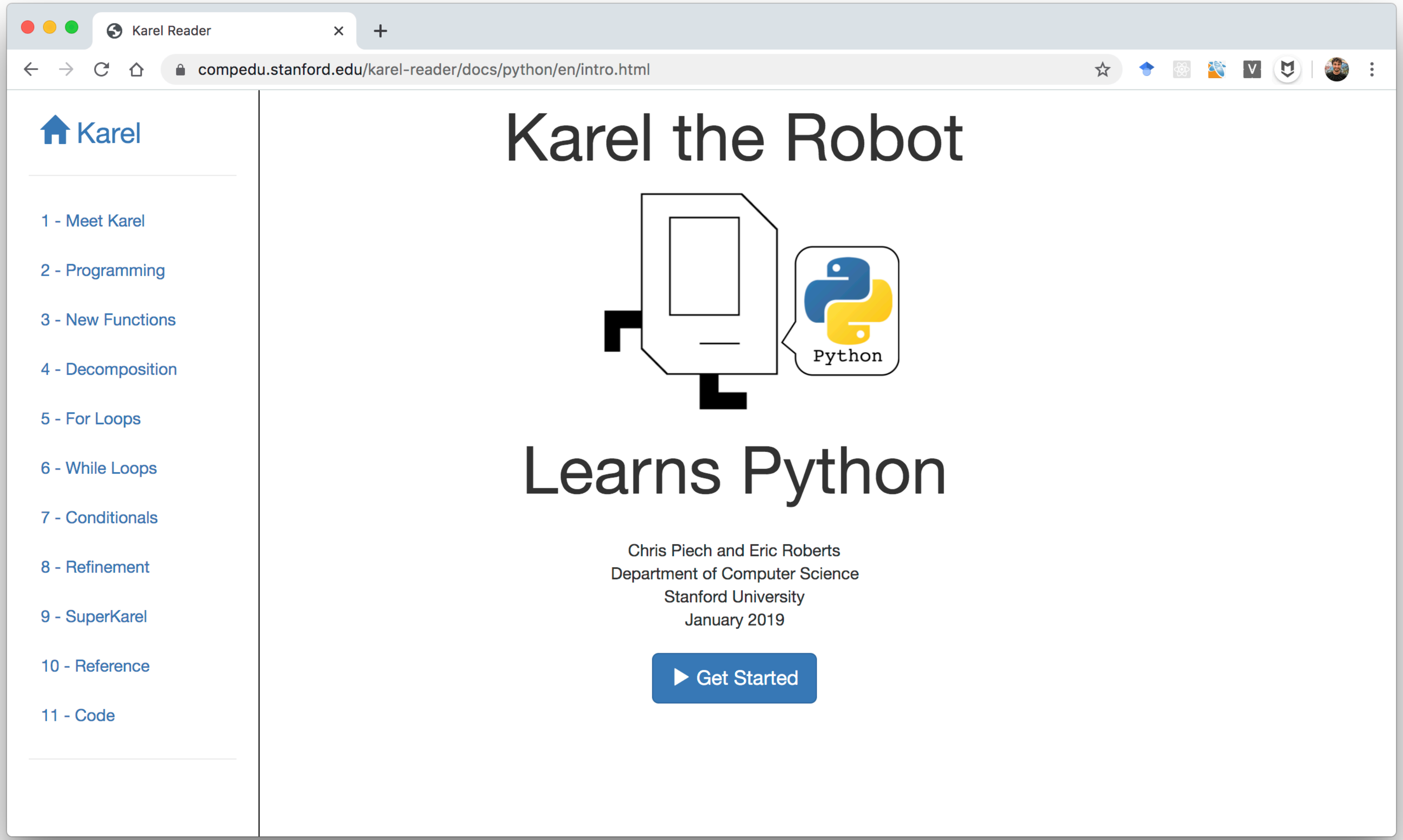
- For at least a couple of assignments, you will be able to meet with you Section Leader one on one to get feedback on your assignments.

- Chance for you to get more feedback than just a grade
- Opportunity to really develop "style" as a programmer
  - We'll talk more about that soon

What we will ask you to do

- 7 programming assignment 65%
  - Get more complicated as quarter progresses
- Week 3 in-class diagnostic assessment (exam) 10%
- Week 9 in-class diagnostic assessment. 15%
- Section participation 10%
- Get 4 free "late days" (on assignments)
  - Each "late day" is a 24-hour period
  - Allows for turning in assignment late without penalty
  - After free late days are used, assignments penalized one grade bucket per day late
  - For extensions beyond free late days, contact Wil (head TA)

Online Text Books



Karel Reader

compedu.stanford.edu/karel-reader/docs/python/en/intro.html

Karel

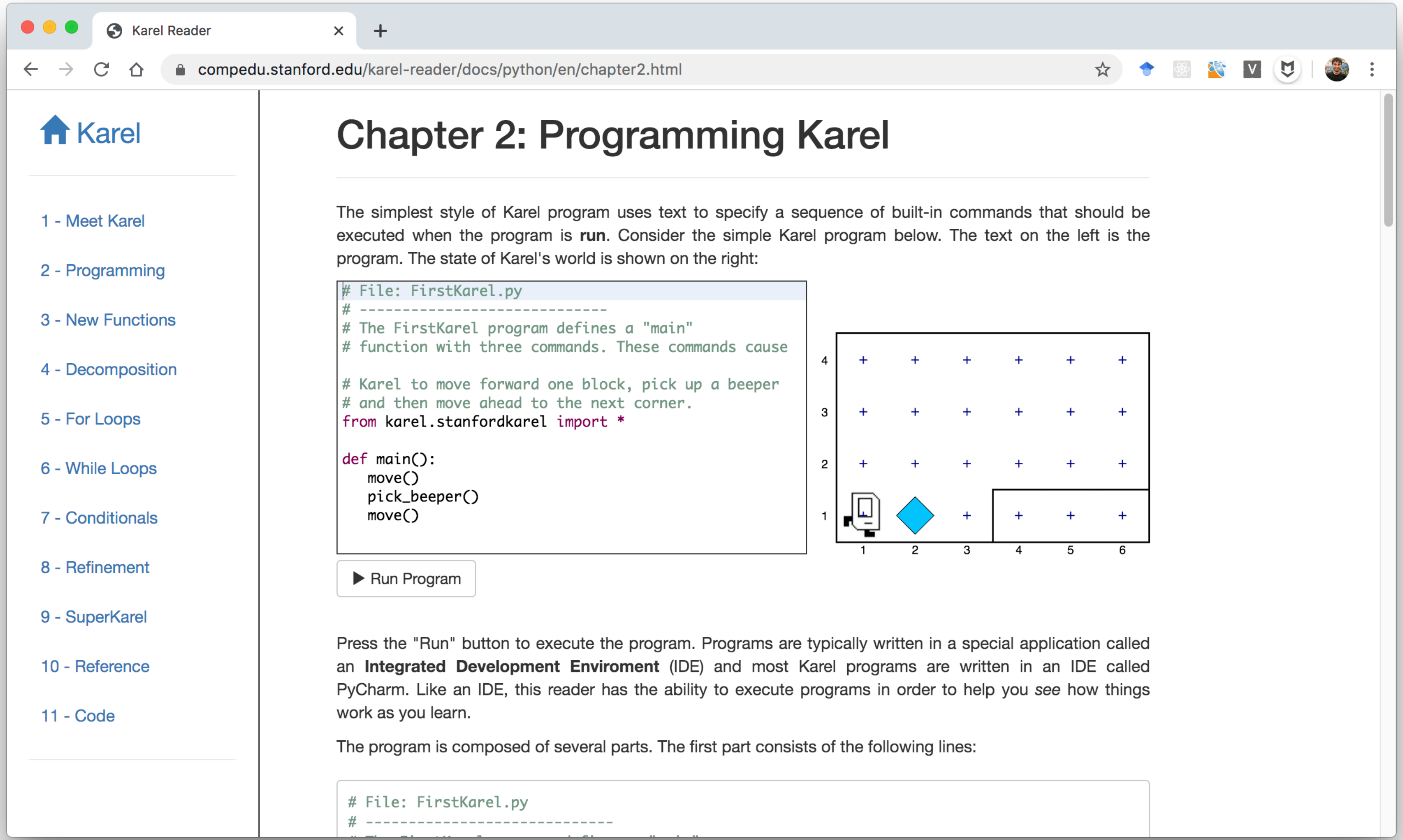
1 - Meet Karel
2 - Programming
3 - New Functions
4 - Decomposition
5 - For Loops
6 - While Loops
7 - Conditionals
8 - Refinement
9 - SuperKarel
10 - Reference
11 - Code

Karel the Robot

Learns Python

Chris Piech and Eric Roberts
Department of Computer Science
Stanford University
January 2019

Online Karel Reader



Karel Reader
compedu.stanford.edu/karel-reader/docs/python/en/chapter2.html

Chapter 2: Programming Karel

The simplest style of Karel program uses text to specify a sequence of built-in commands that should be executed when the program is **run**. Consider the simple Karel program below. The text on the left is the program. The state of Karel's world is shown on the right:

```
# File: FirstKarel.py
# -----------------------------
# The FirstKarel program defines a "main"
# function with three commands. These commands cause

# Karel to move forward one block, pick up a beeper
# and then move ahead to the next corner.
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
```

Press the "Run" button to execute the program. Programs are typically written in a special application called an **Integrated Development Enviroment** (IDE) and most Karel programs are written in an IDE called PyCharm. Like an IDE, this reader has the ability to execute programs in order to help you *see* how things work as you learn.

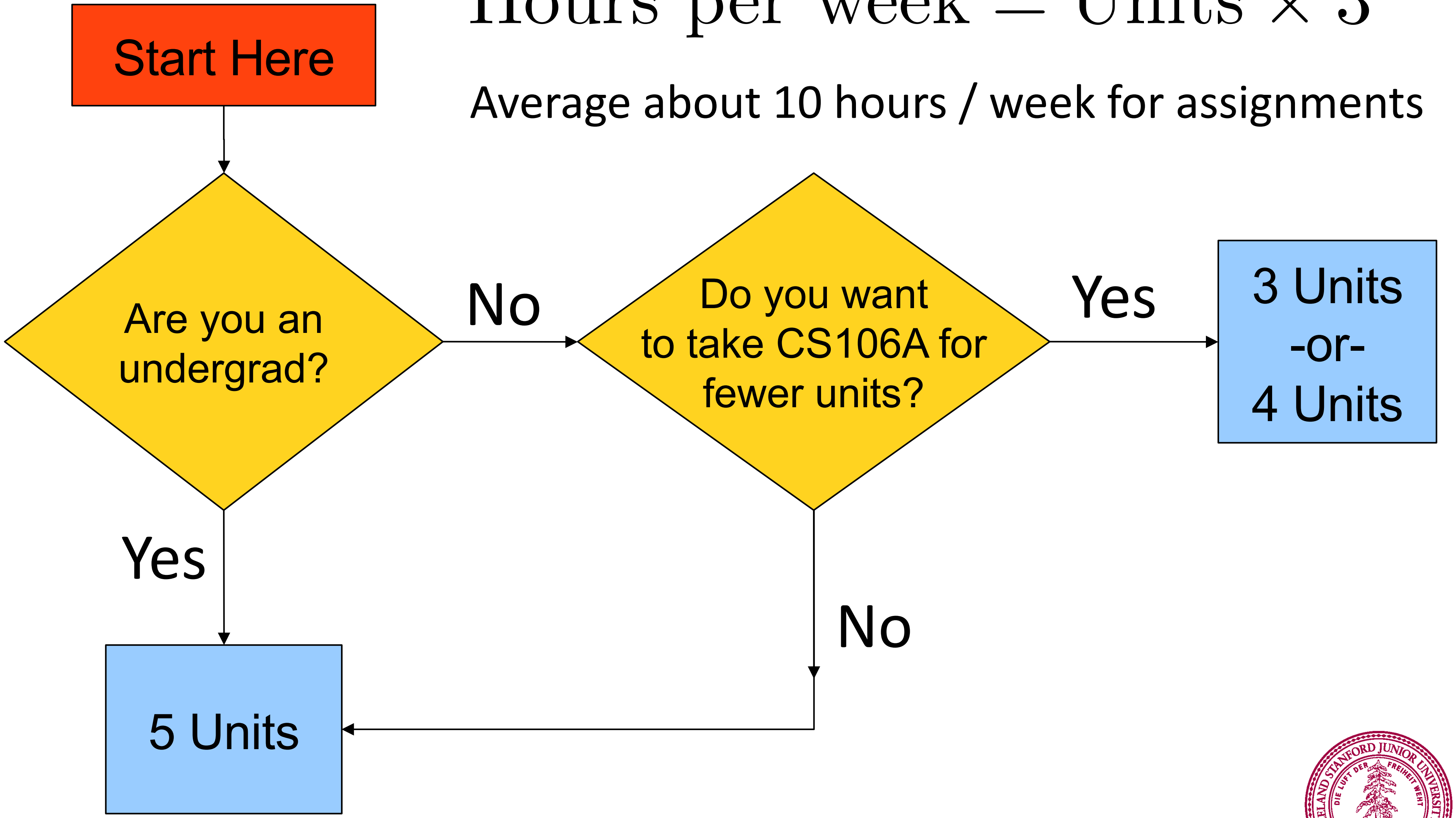
The program is composed of several parts. The first part consists of the following lines:

```
# File: FirstKarel.py
# -----------------------------
```

CS106A Units

$$\text{Hours per week} = \text{Units} \times 3$$

Average about 10 hours / week for assignments



Are you in the right place?

What is CS106A?

Computer Science

“Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools, it is about how we use them and what we find out when we do.”

— Michael Fellows and Ian Parberry

“You must unlearn what you have learned”

— Yoda

Learning Goals

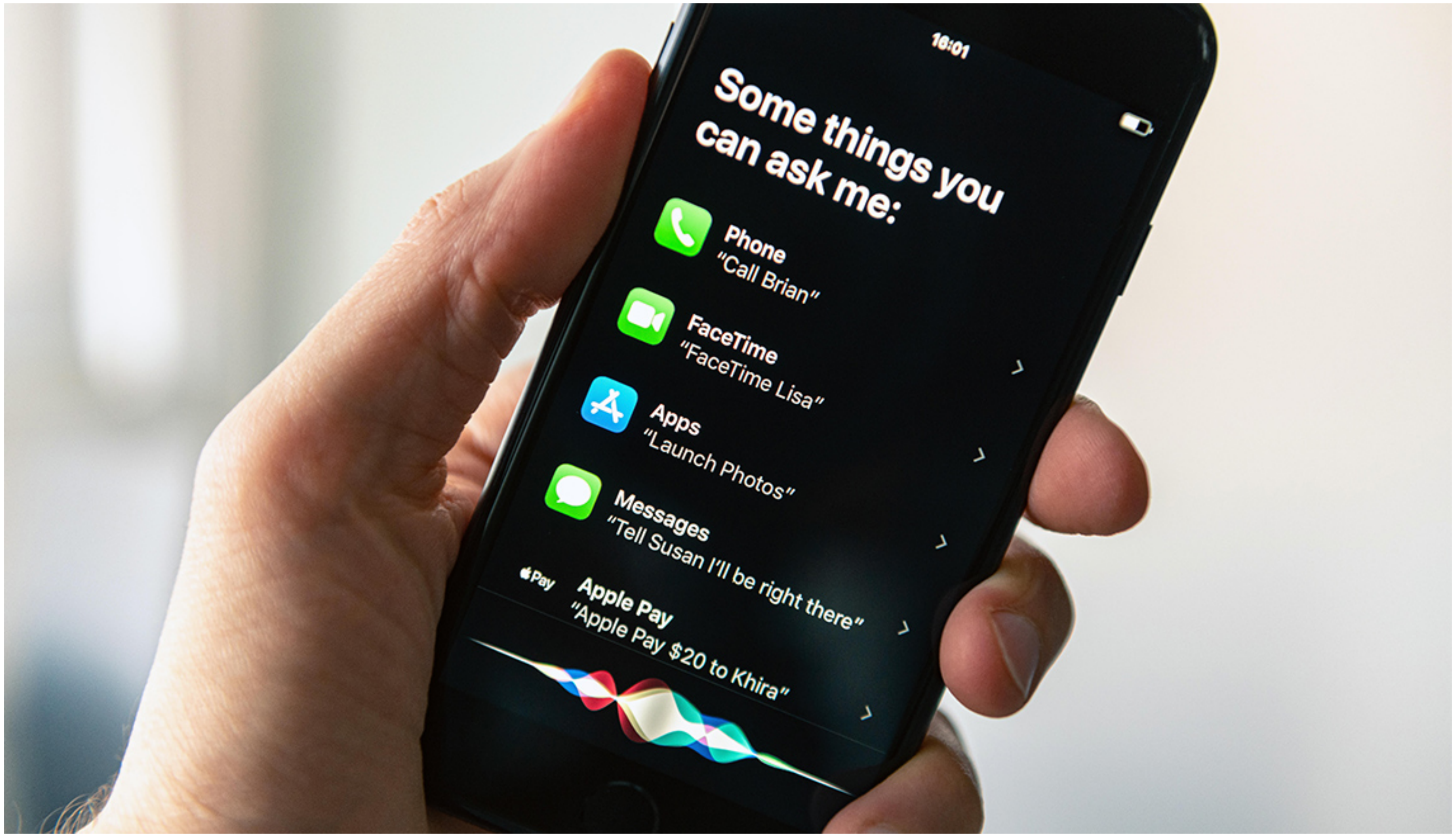
- ***Learn how to harness computing power to solve problems.***
- To that end:
  - Explore fundamental techniques in computer programming.
  - Develop good software engineering style.
  - Gain familiarity with the Python programming language.

There are a lot of cool programs you may one day write

Computer Graphics

Pat Hanrahan, one of the founders of Pixar is a professor here.
He just won the Turing Award – the Nobel Prize of Computer Science

Mobile Computing



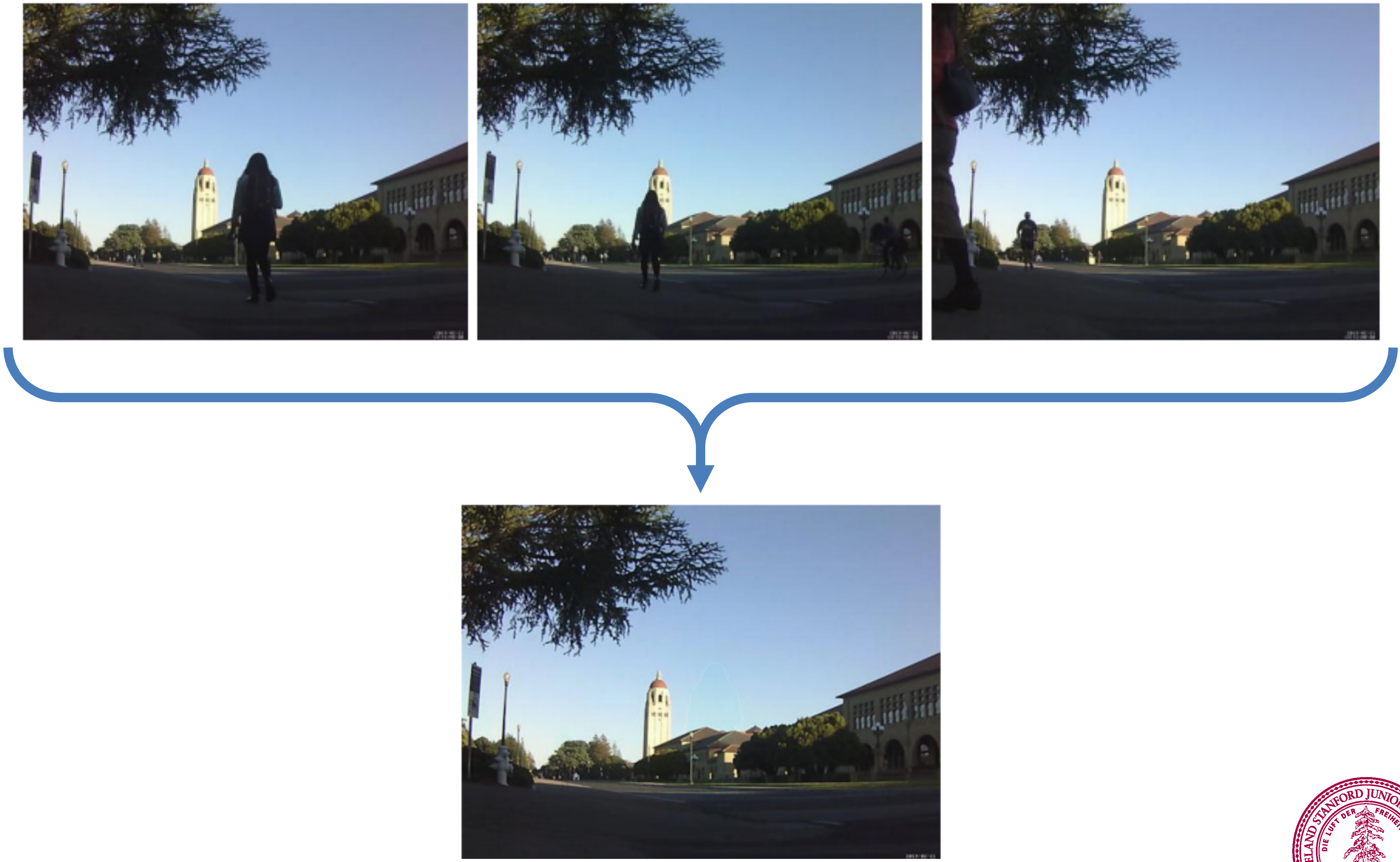
You probably have a supercomputer in your pocket. Learning to code for your phone can be fun!

Autonomous Surgery

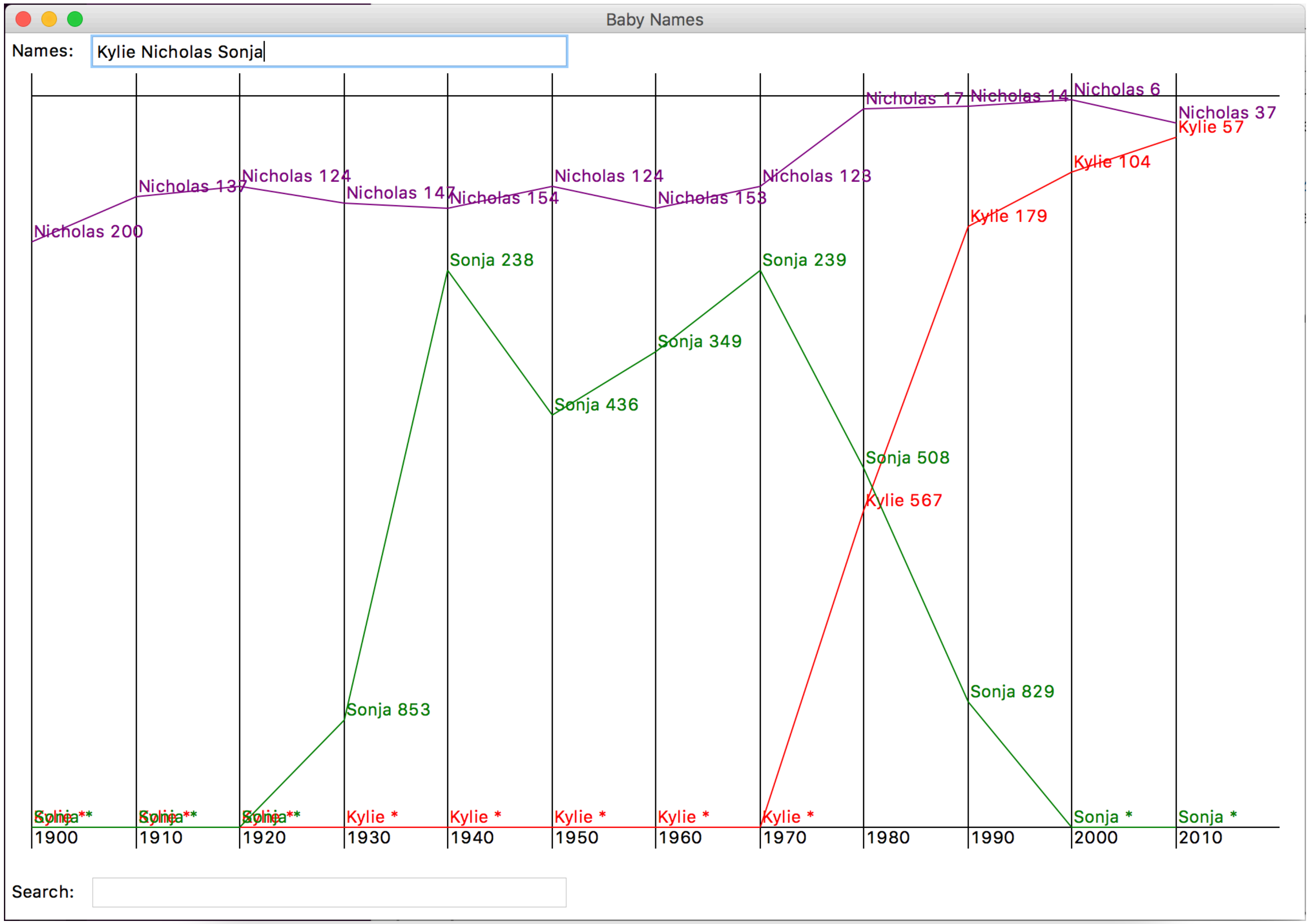
Self-Driving Car

If only we could program self-driving cars…

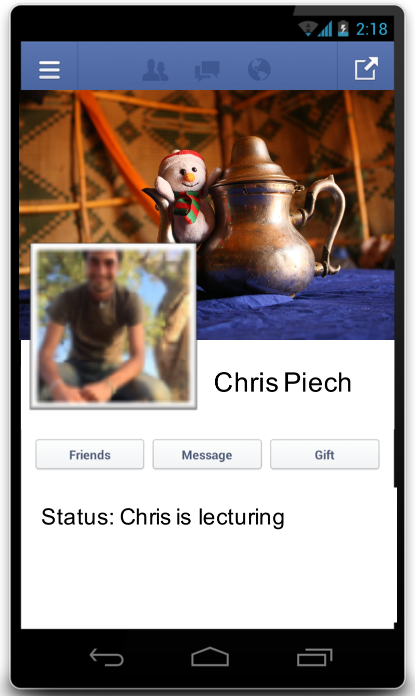
Image Transformation

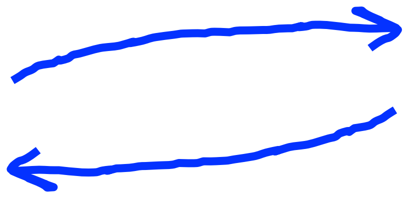


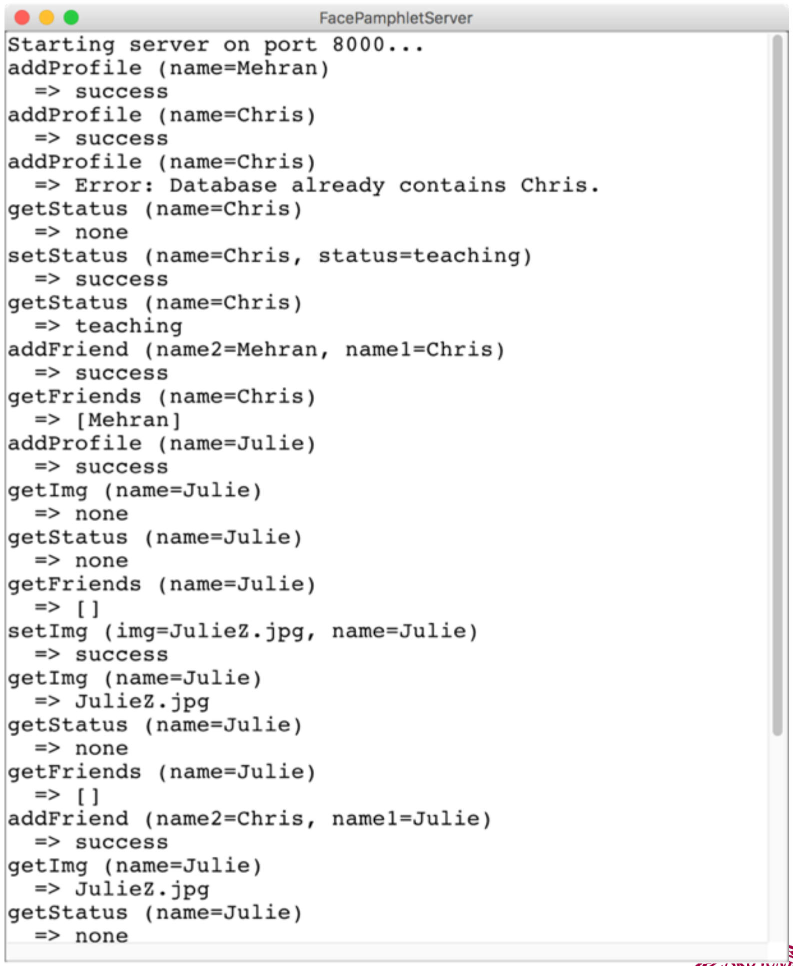
Data Visualization



Internet Applications







```
Starting server on port 8000...
addProfile (name=Mehran)
  => success
addProfile (name=Chris)
  => success
addProfile (name=Chris)
  => Error: Database already contains Chris.
getStatus (name=Chris)
  => none
setStatus (name=Chris, status=teaching)
  => success
getStatus (name=Chris)
  => teaching
addFriend (name2=Mehran, name1=Chris)
  => success
getFriends (name=Chris)
  => [Mehran]
addProfile (name=Julie)
  => success
getImg (name=Julie)
  => none
getStatus (name=Julie)
  => none
getFriends (name=Julie)
  => []
setImg (img=JulieZ.jpg, name=Julie)
  => success
getImg (name=Julie)
  => JulieZ.jpg
getStatus (name=Julie)
  => none
getFriends (name=Julie)
  => []
addFriend (name2=Chris, name1=Julie)
  => success
getImg (name=Julie)
  => JulieZ.jpg
getStatus (name=Julie)
  => none
```

Art of Computer *Science*

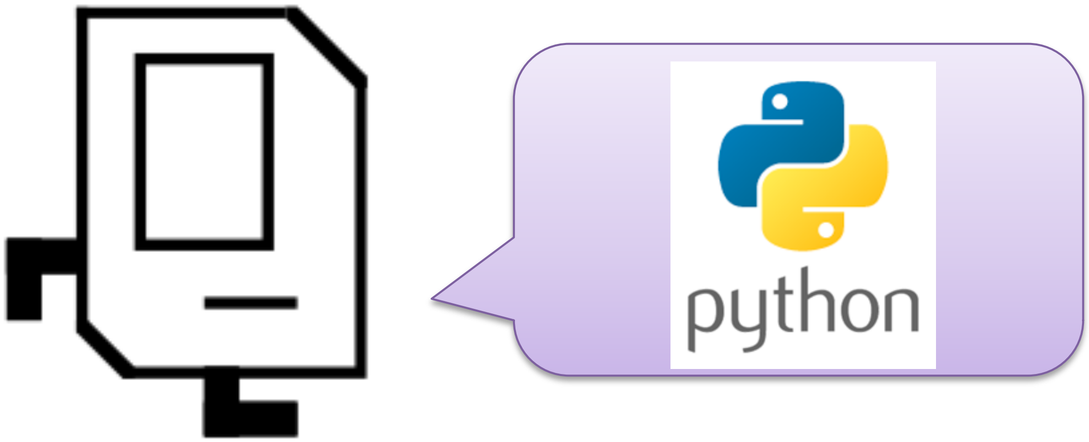
Strive for Everyone to Succeed

Lets Get Started

Meet Karel the Robot

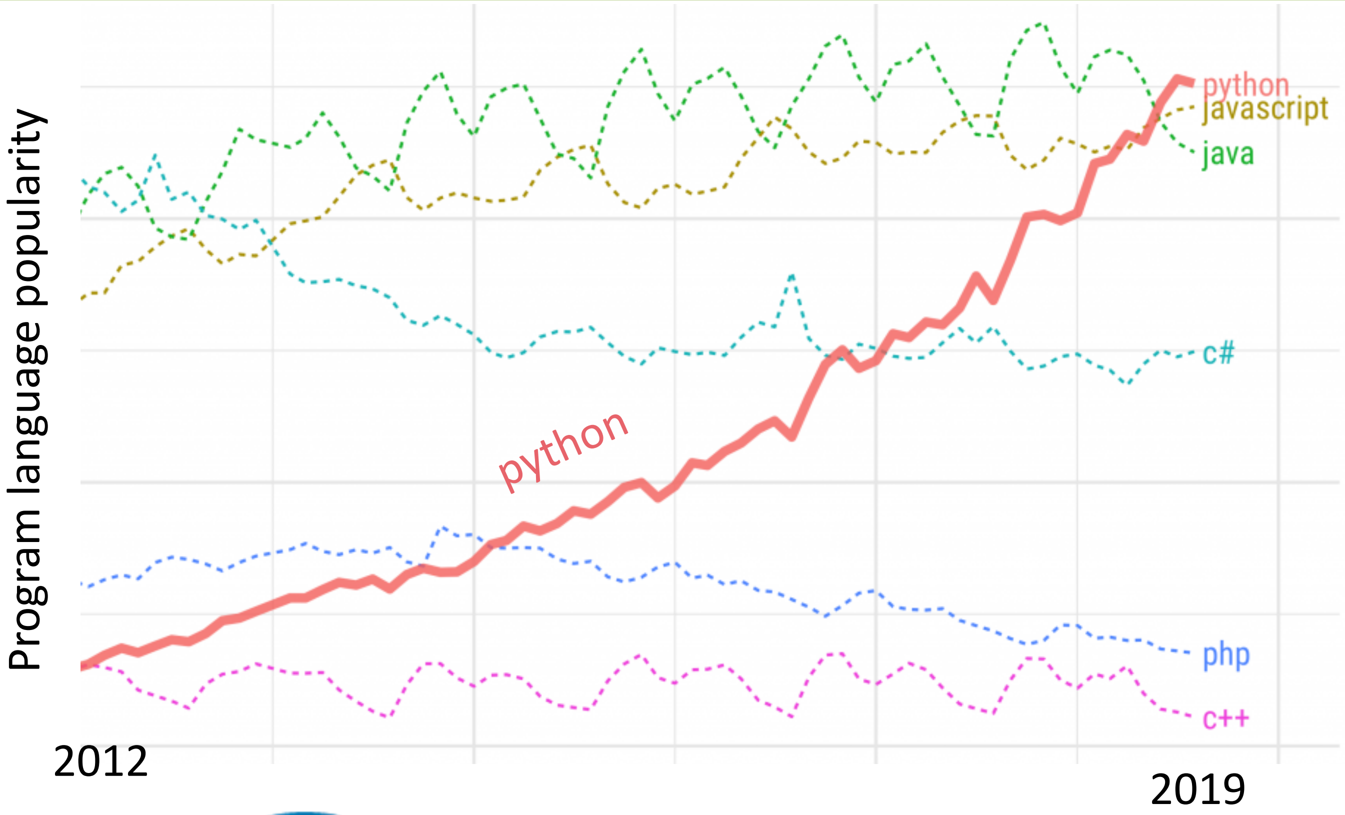
Good morning

Karel Speaks Python



Why Python?

1

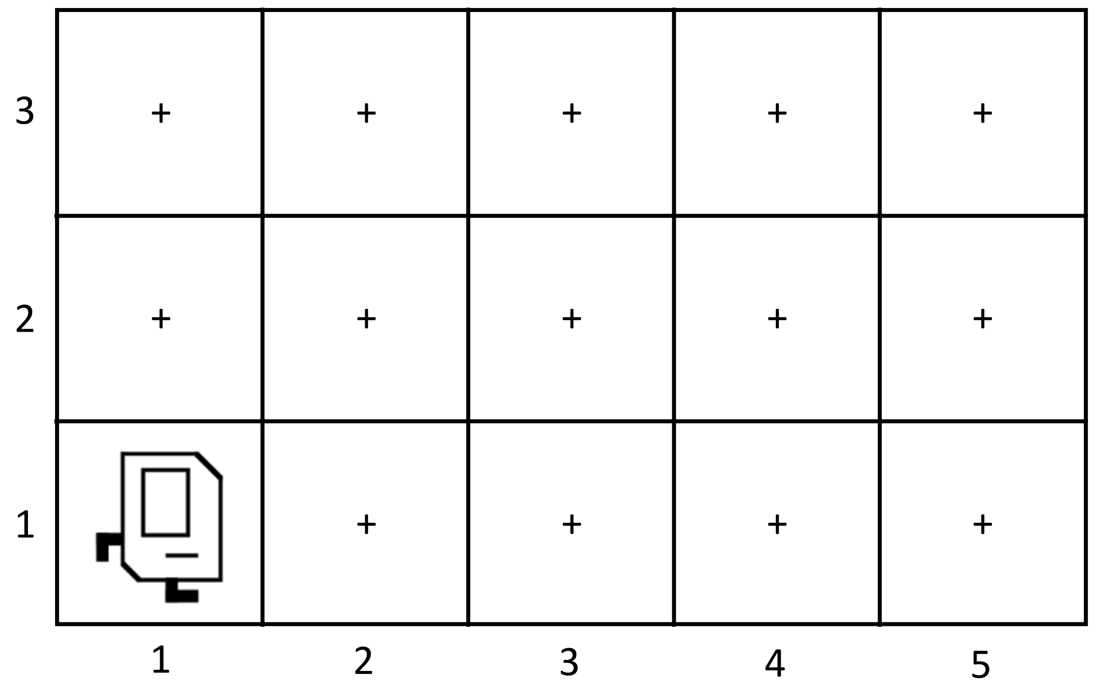


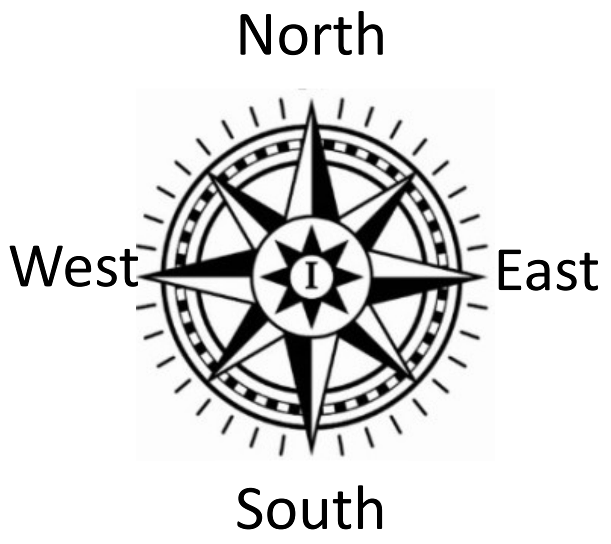
2

https://stackoverflow.blog/2017/09/06/incredible-growth-python/

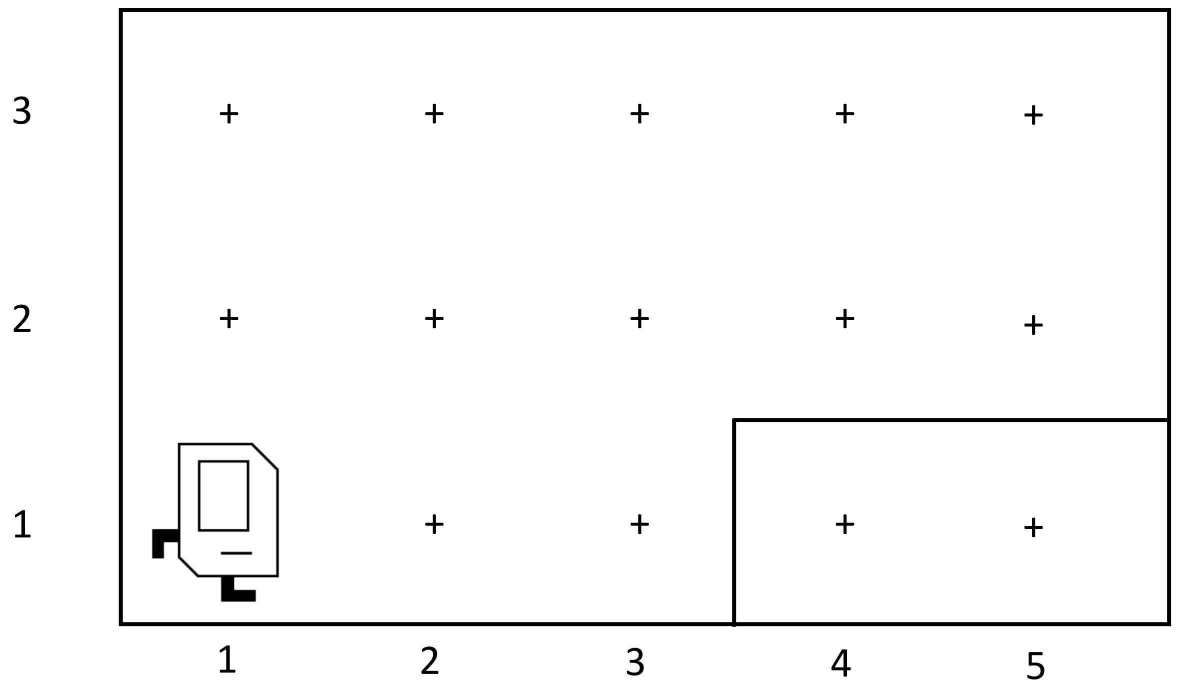
Guido van Rossum

Karel's World

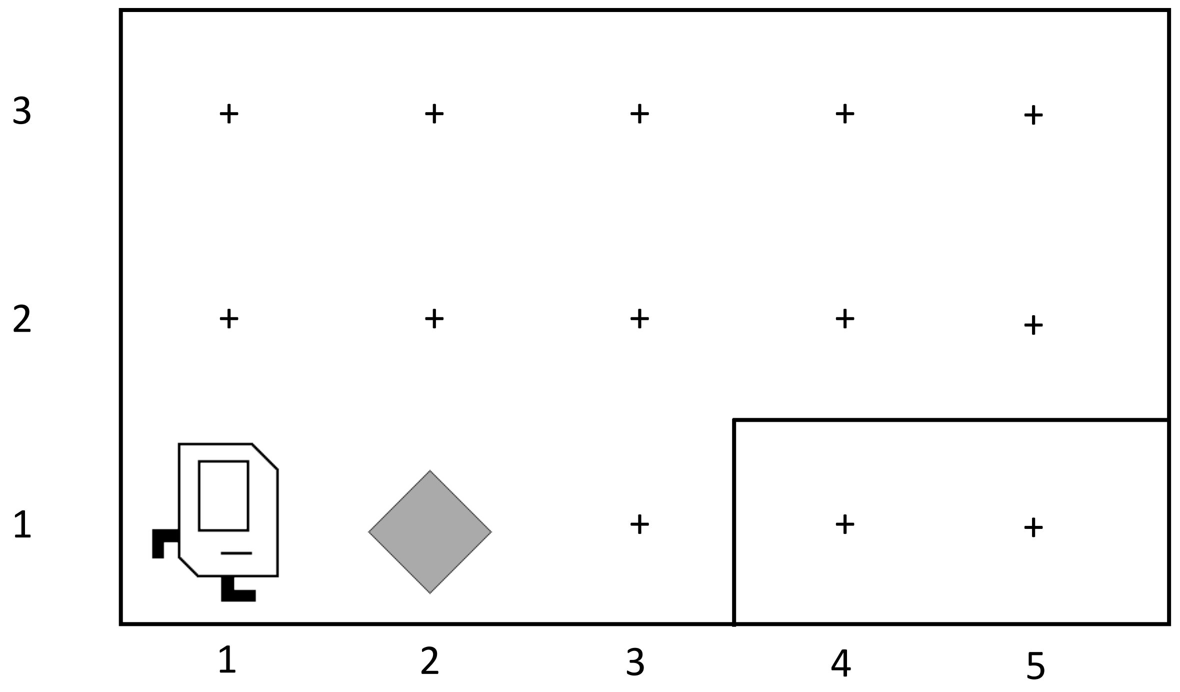




Walls



Beepers

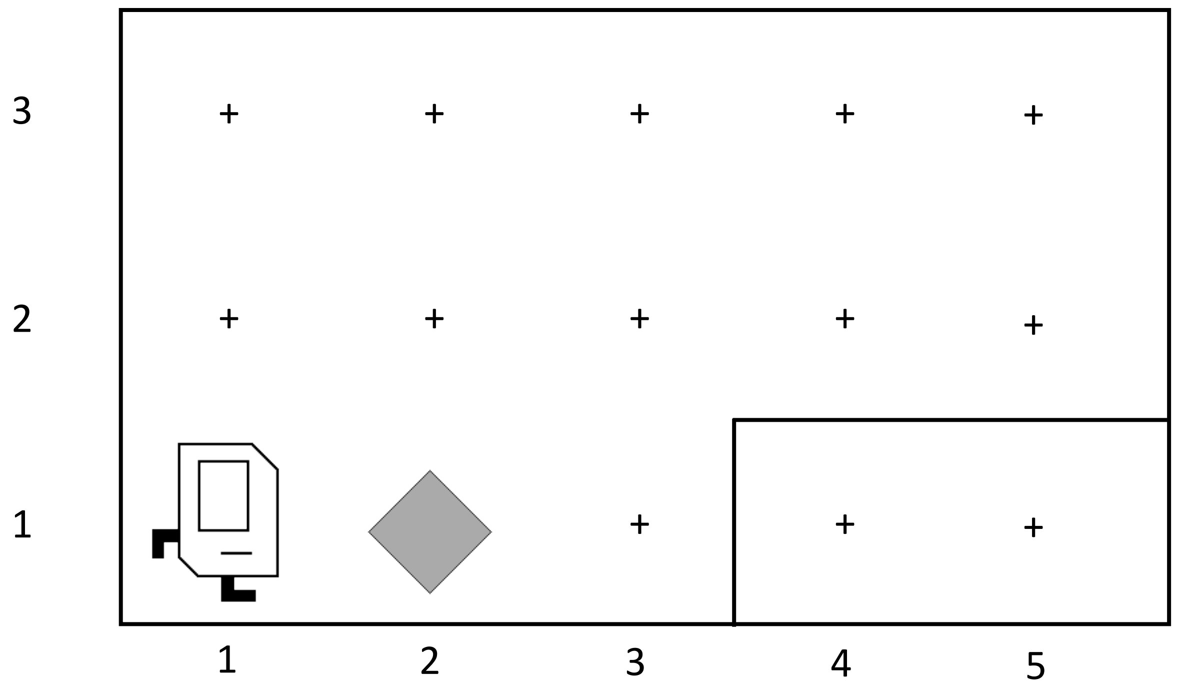


Knows Four Commands

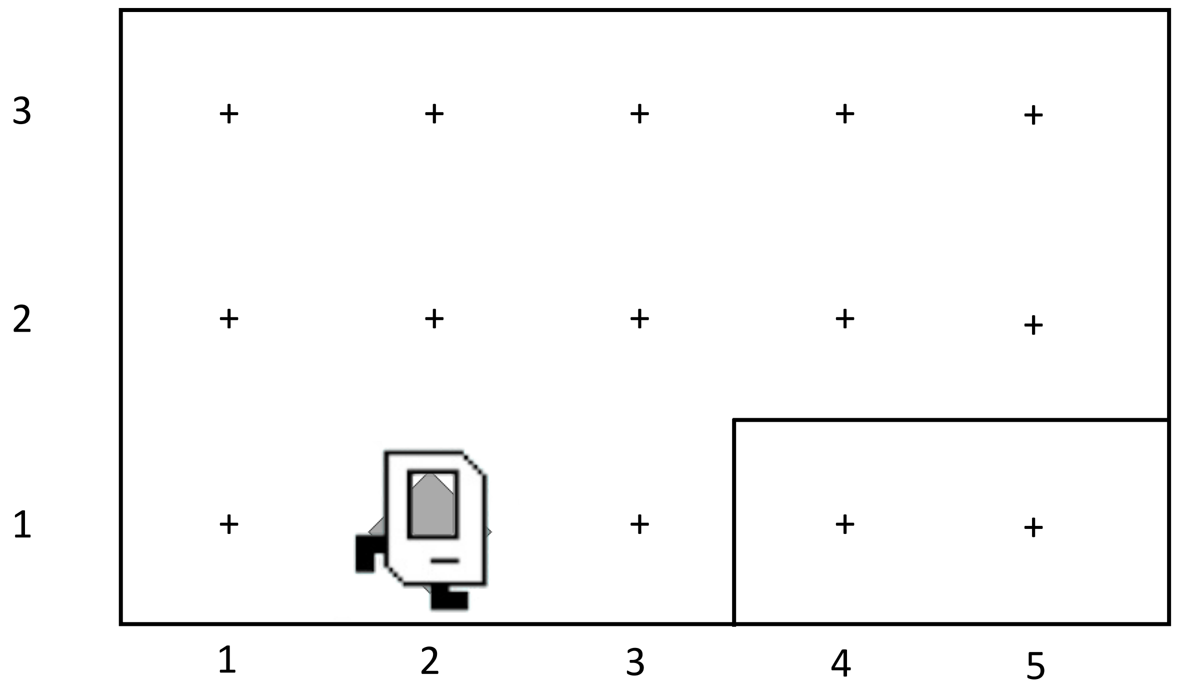
```
move()

turn_left()

put_beeper()

pick_beeper()
```

`move()`

`move()`

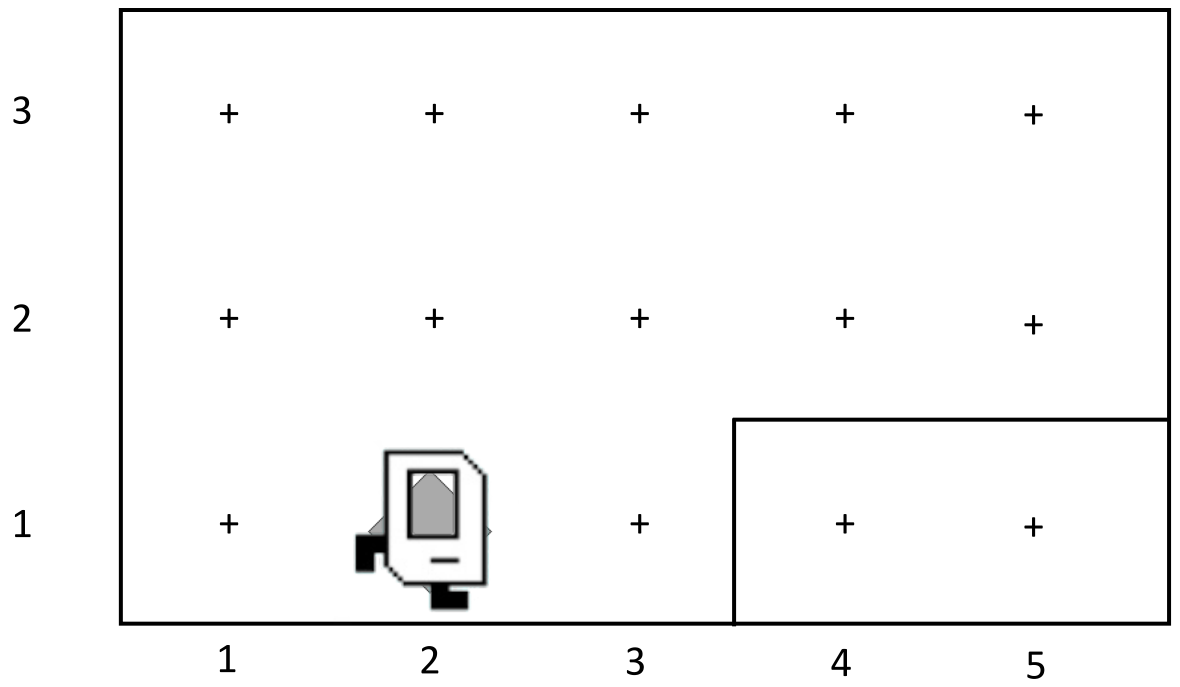


move()

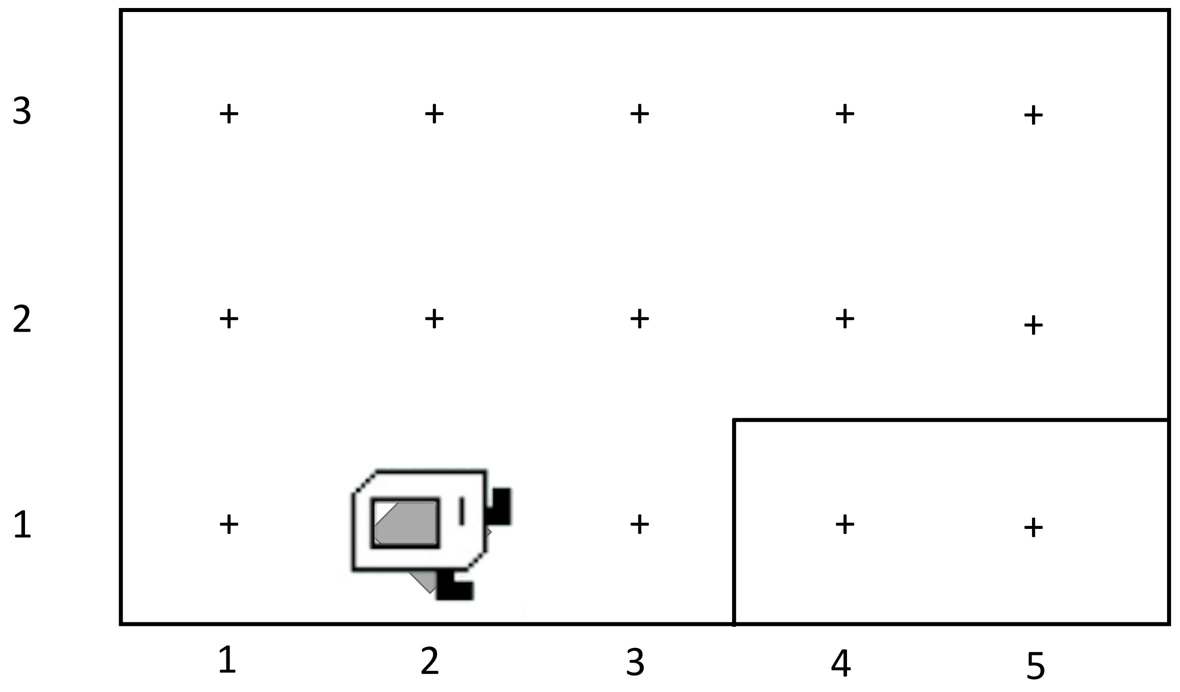


```
turn_left()
```

`turn_left()`

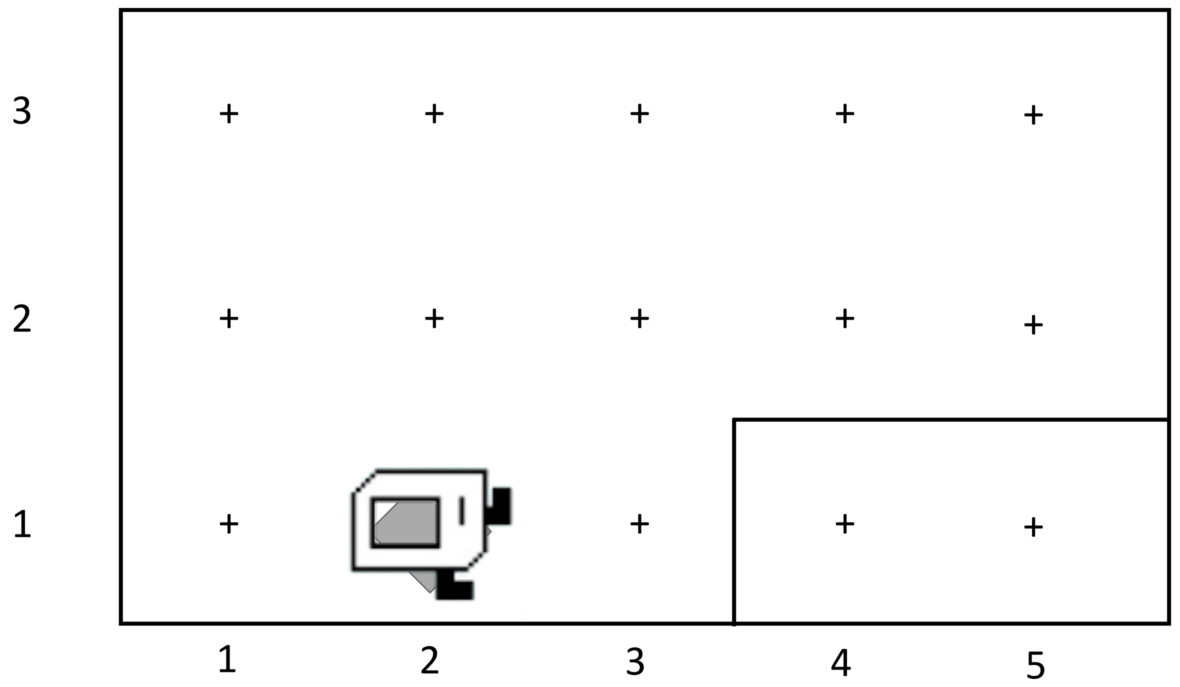


`turn_left()`

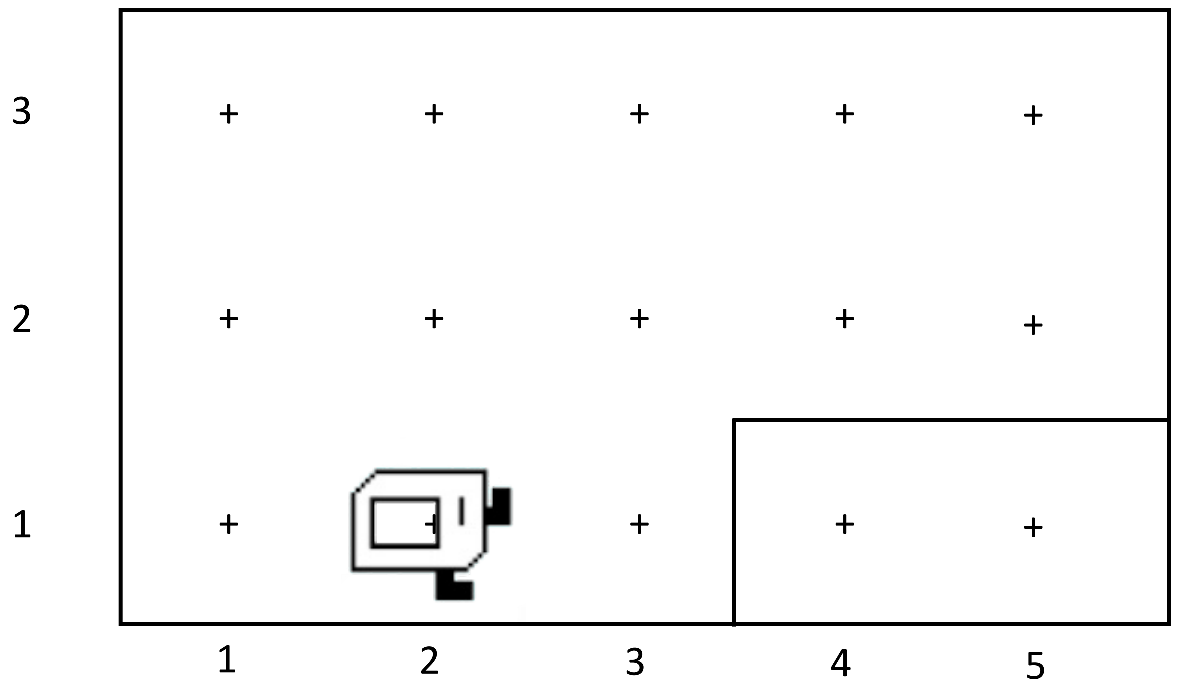


```
pick_beeper()
```

`turn_left()`

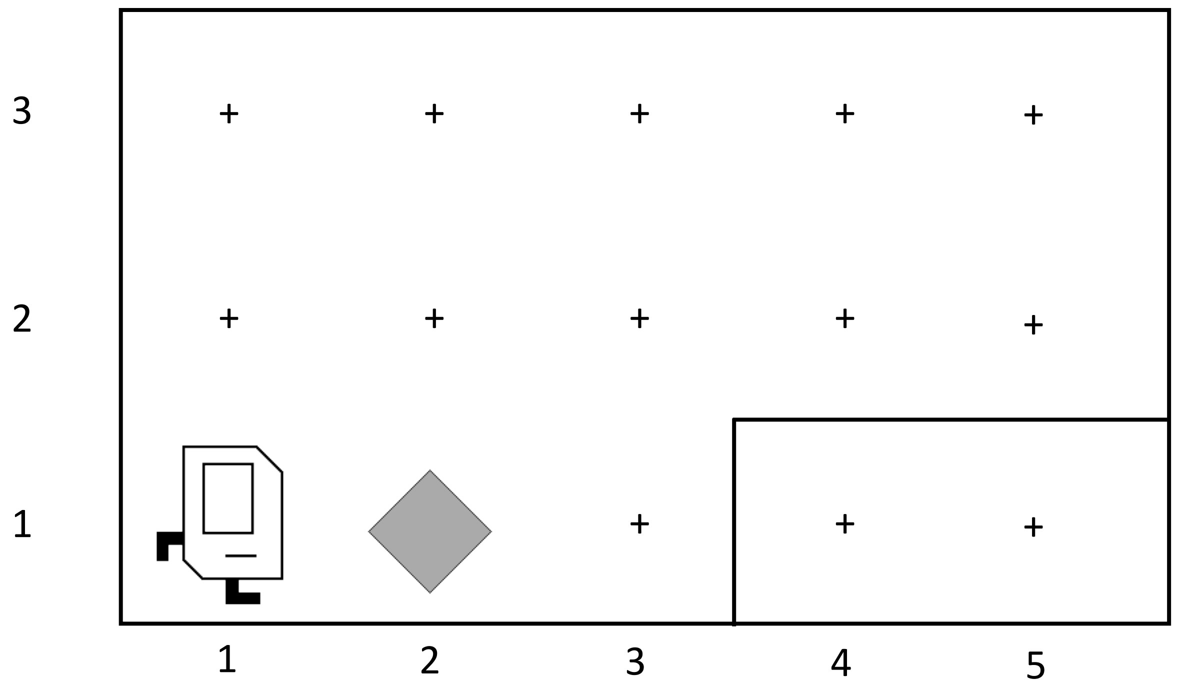


`turn_left()`

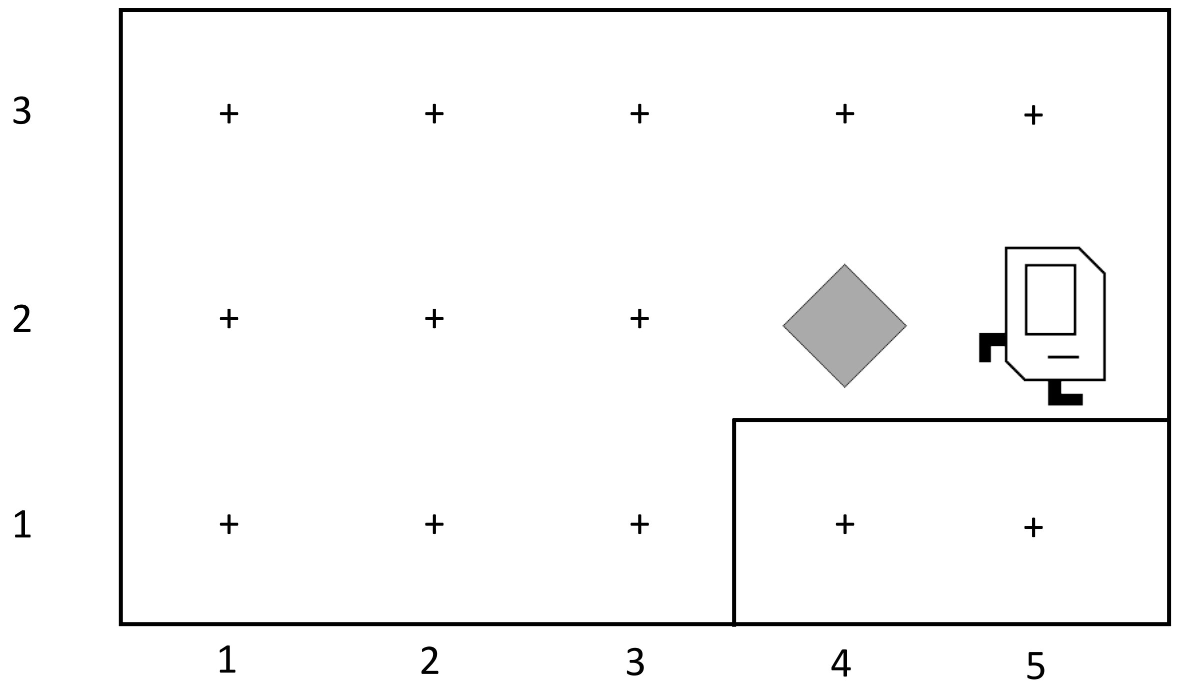


Make Sense?

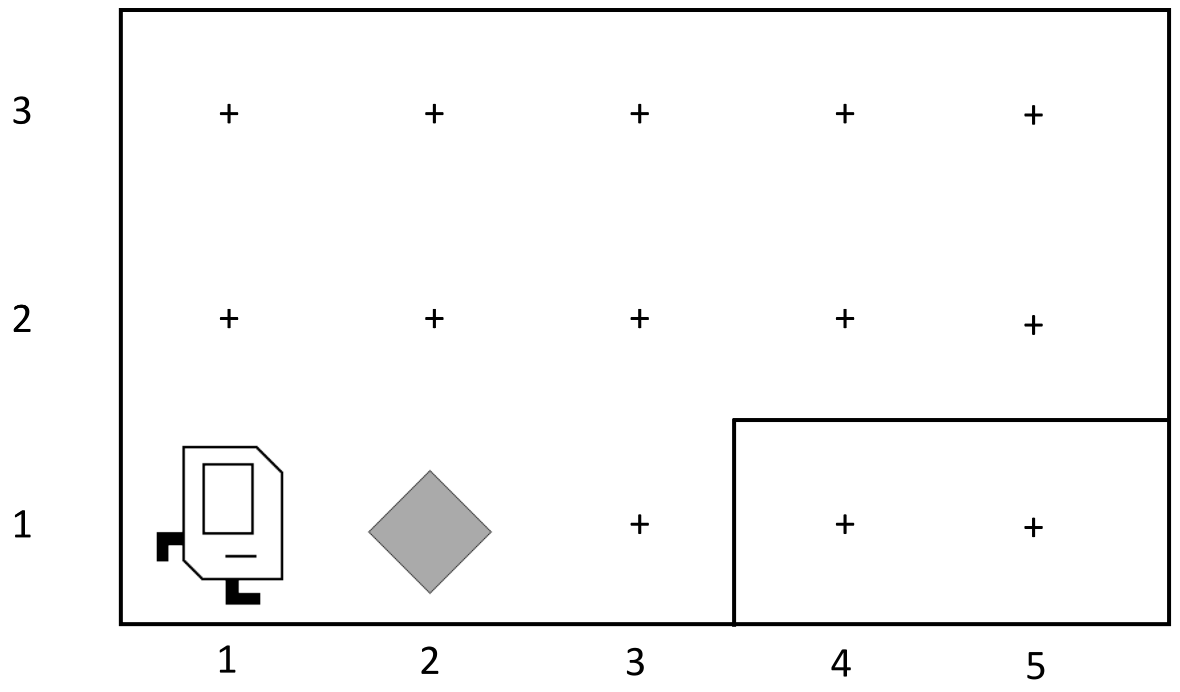
First Challenge



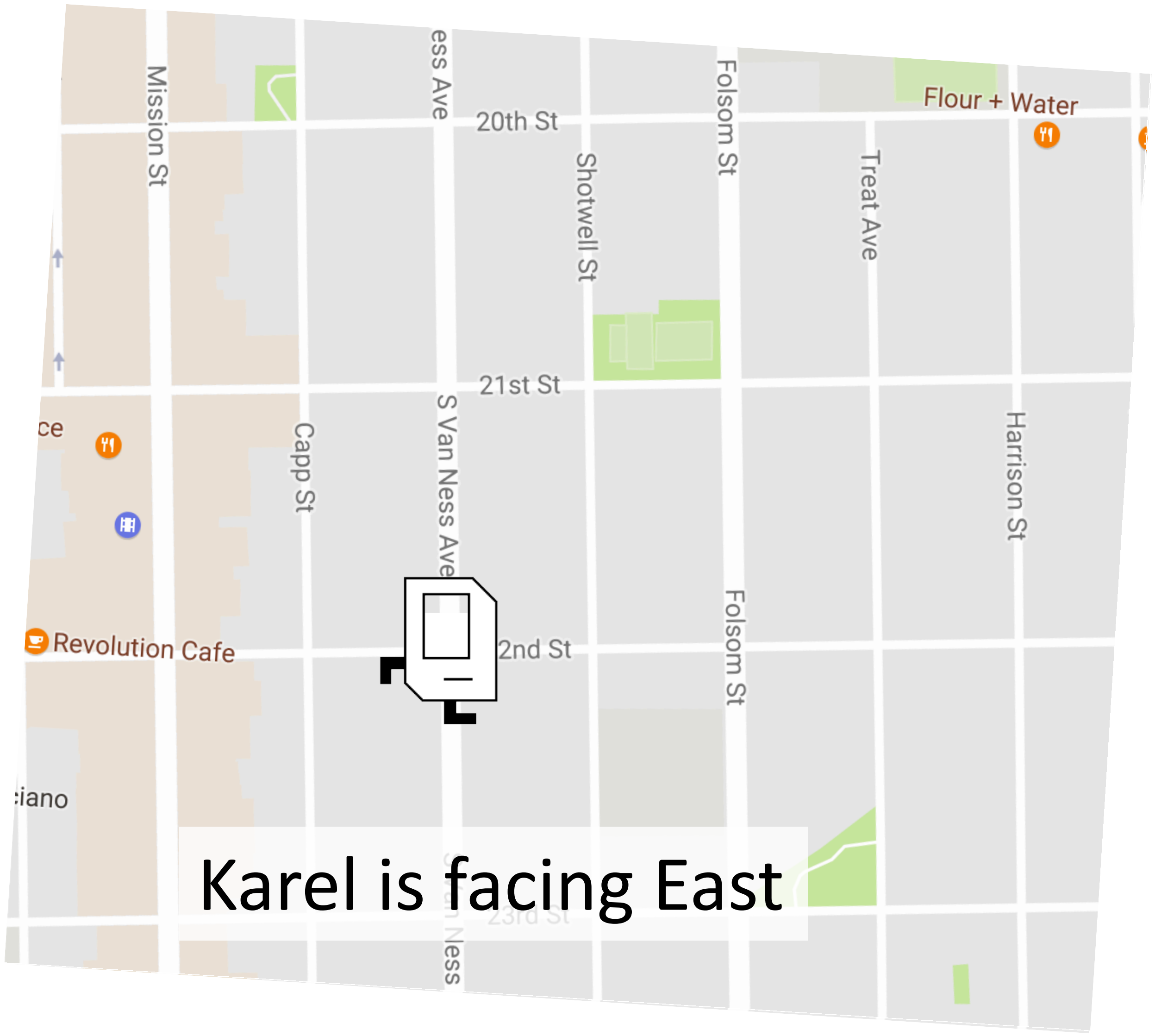
First Challenge



Bird's Eye View

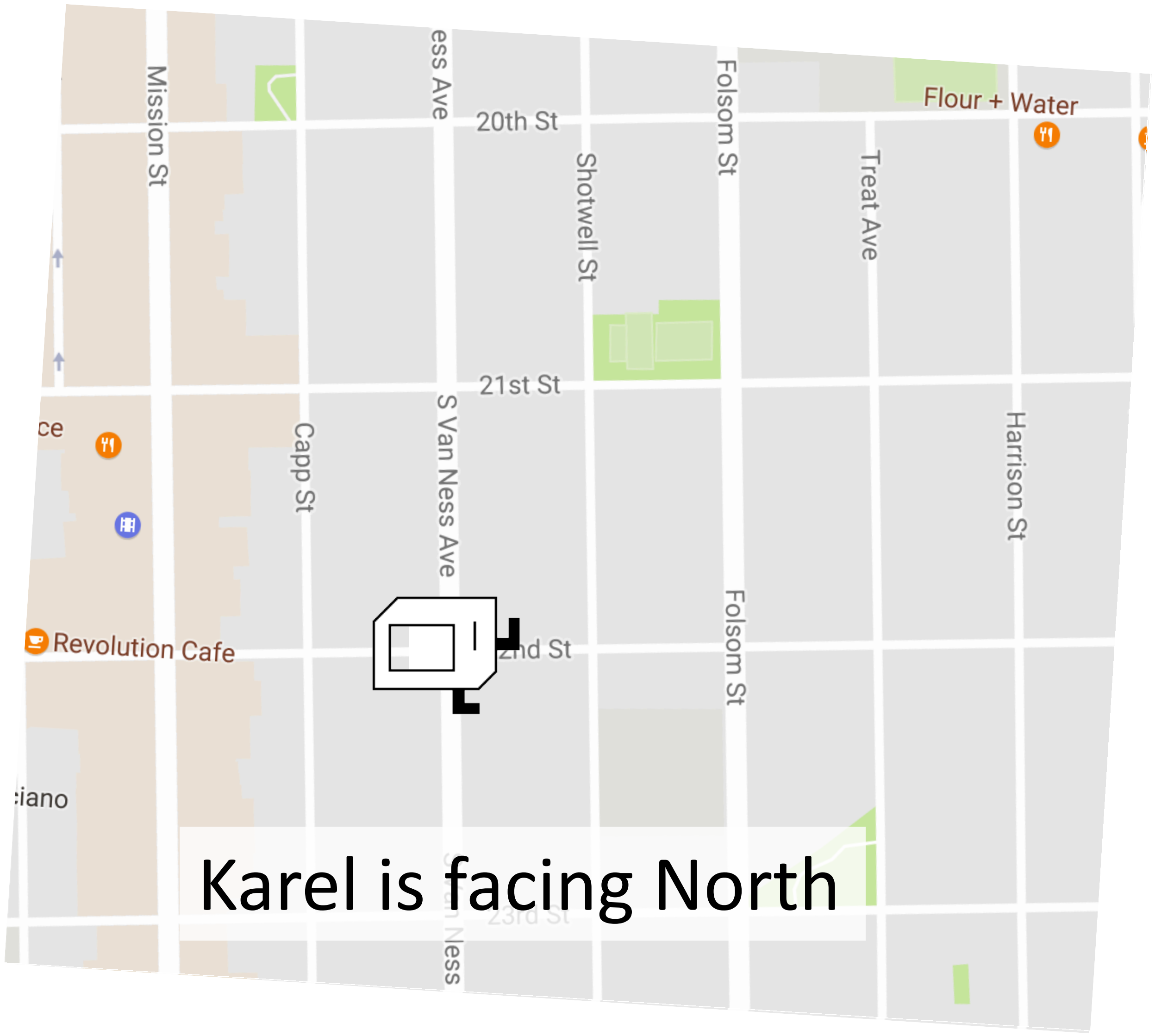


Bird's Eye View



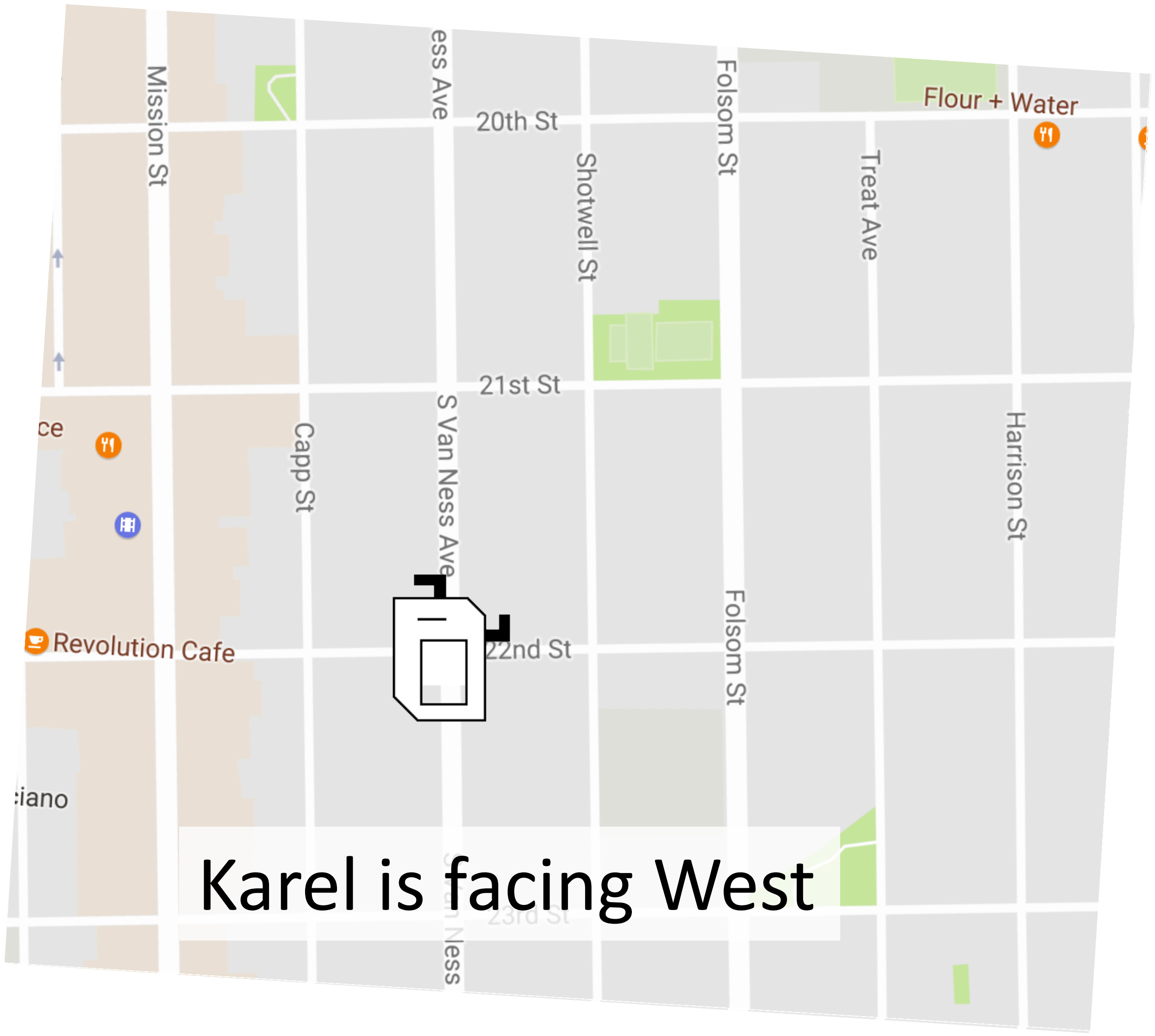
Karel is facing East

Turn Left



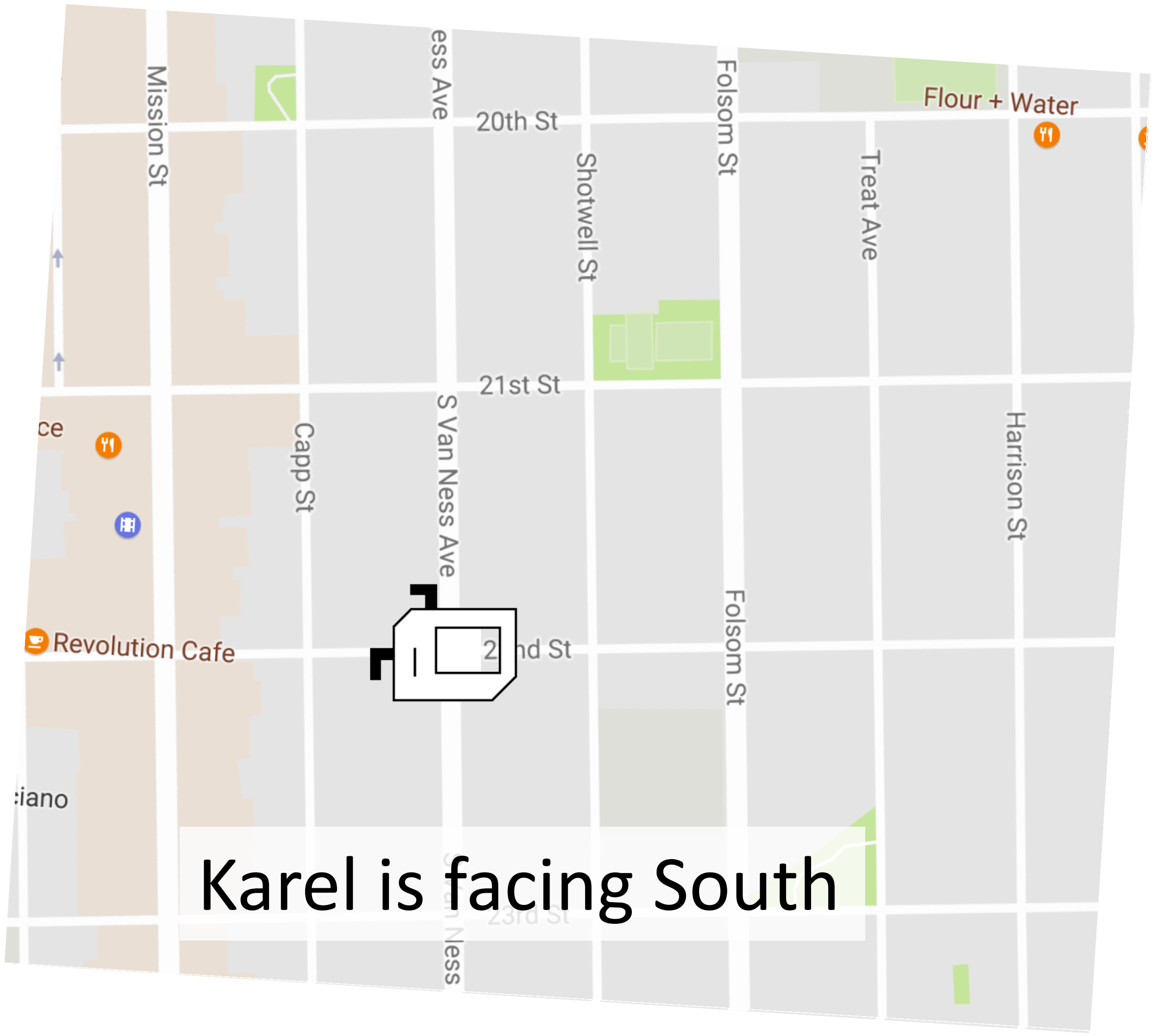
Karel is facing North

Turn Left



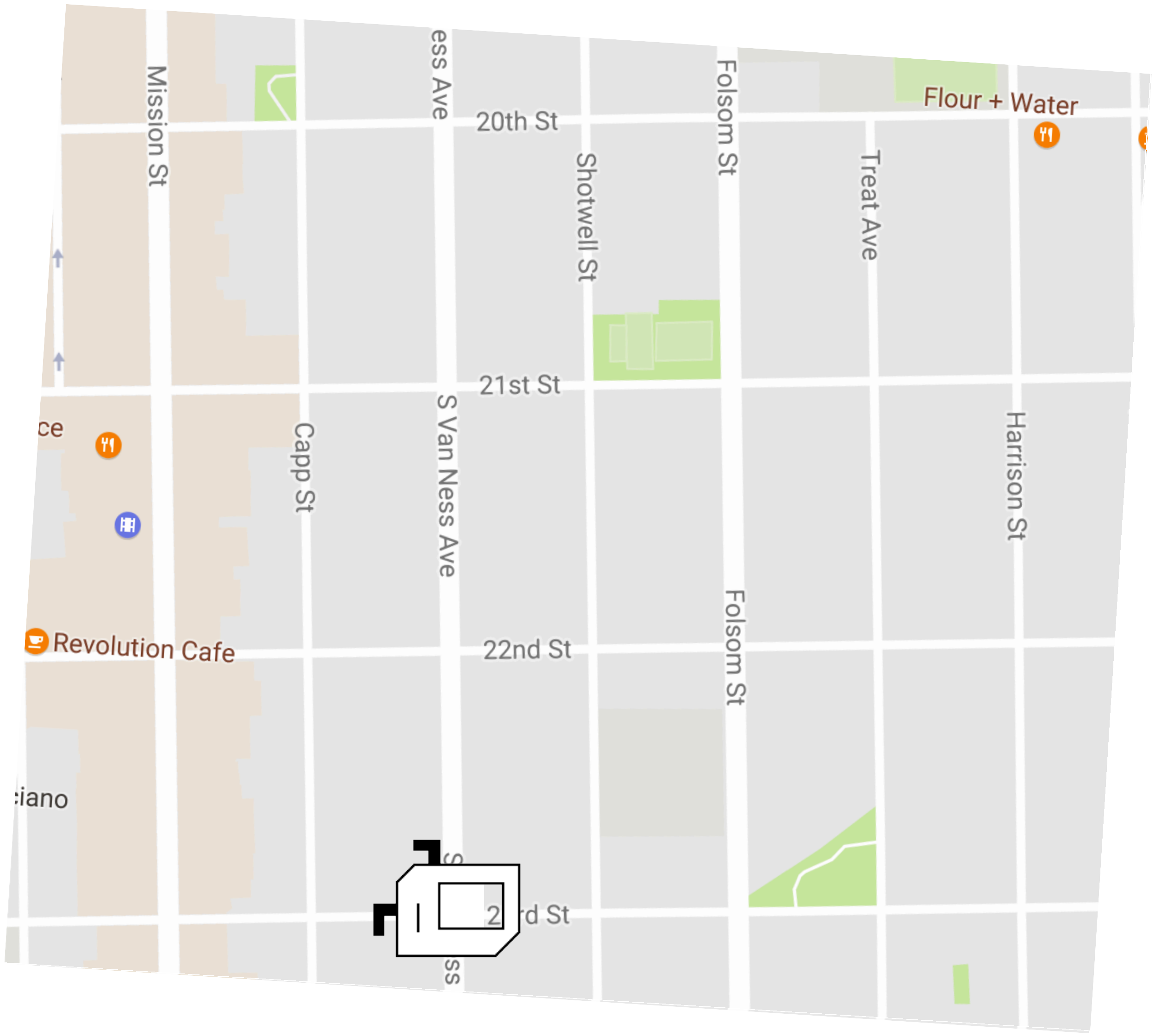
Karel is facing West

Turn Left

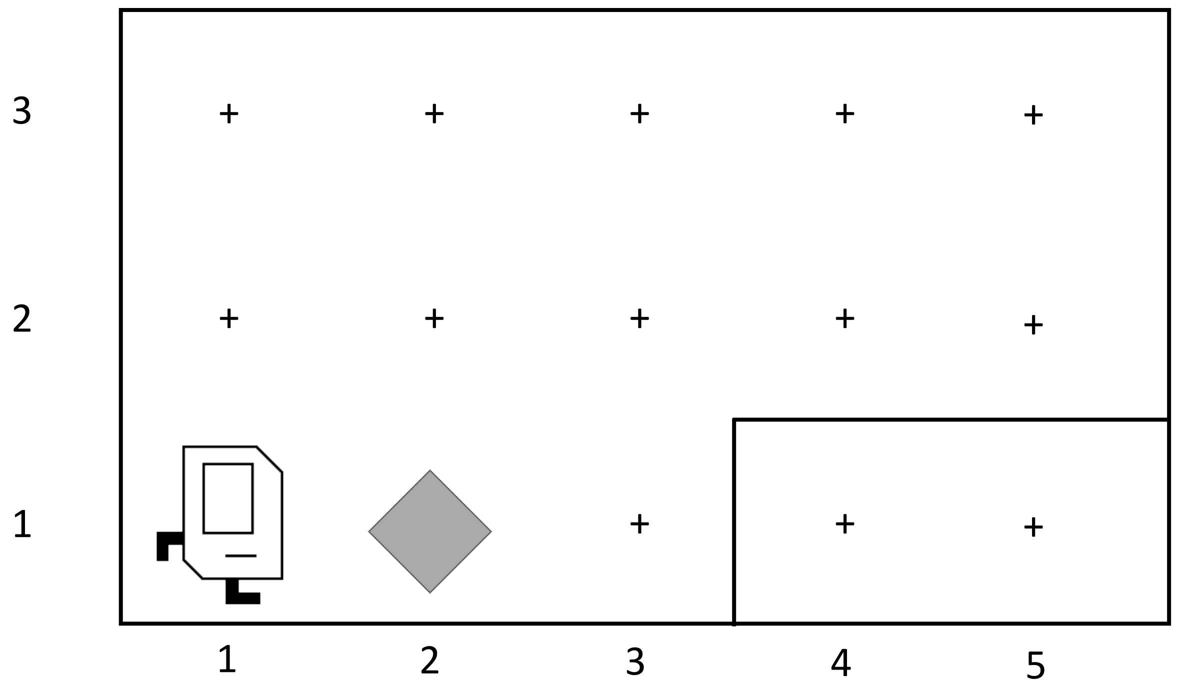


Karel is facing South

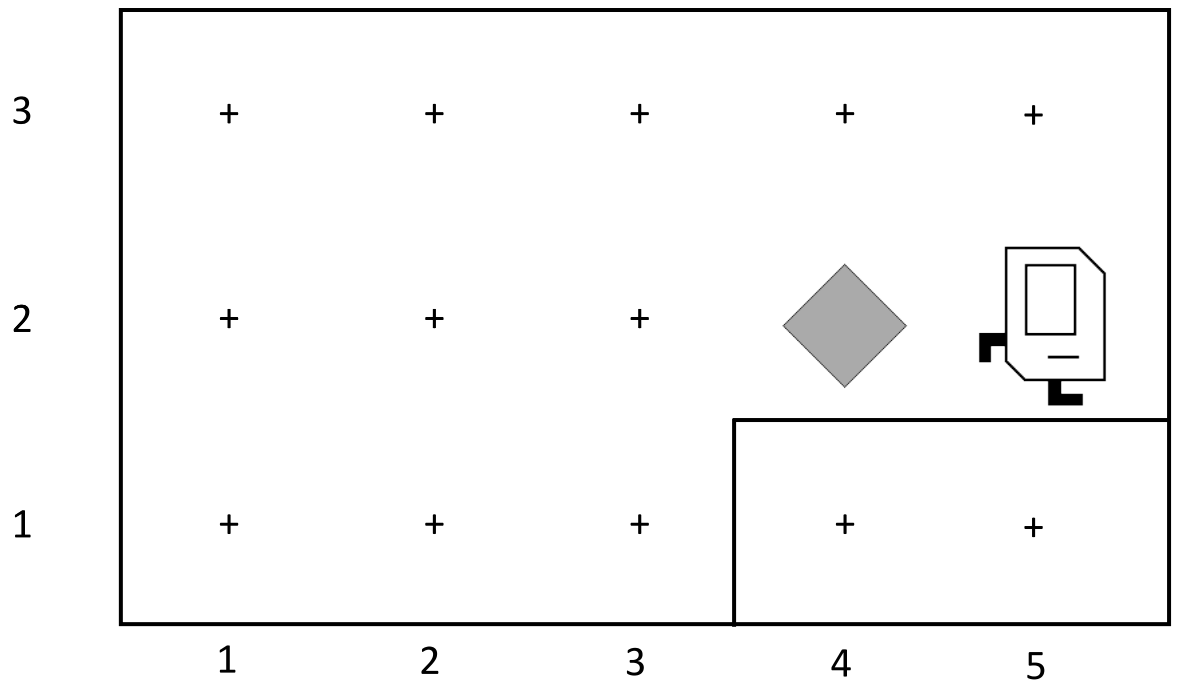
Move


Mission St
ess Ave
20th St
Folsom St
Flour + Water
Shotwell St
Treat Ave
21st St
ce
Capp St
S Van Ness Ave
Harrison St
Revolution Cafe
22nd St
Folsom St
iano
2 d St
ss

First Challenge



First Challenge



Learn By Doing

PC PyCharm

The Python IDE for Professional Developers

DOWNLOAD

Full-fledged Professional or Free Community

Function Definition

```
def name():
    function statements
```

This adds a new command to Karels vocabulary

Anatomy of a Program

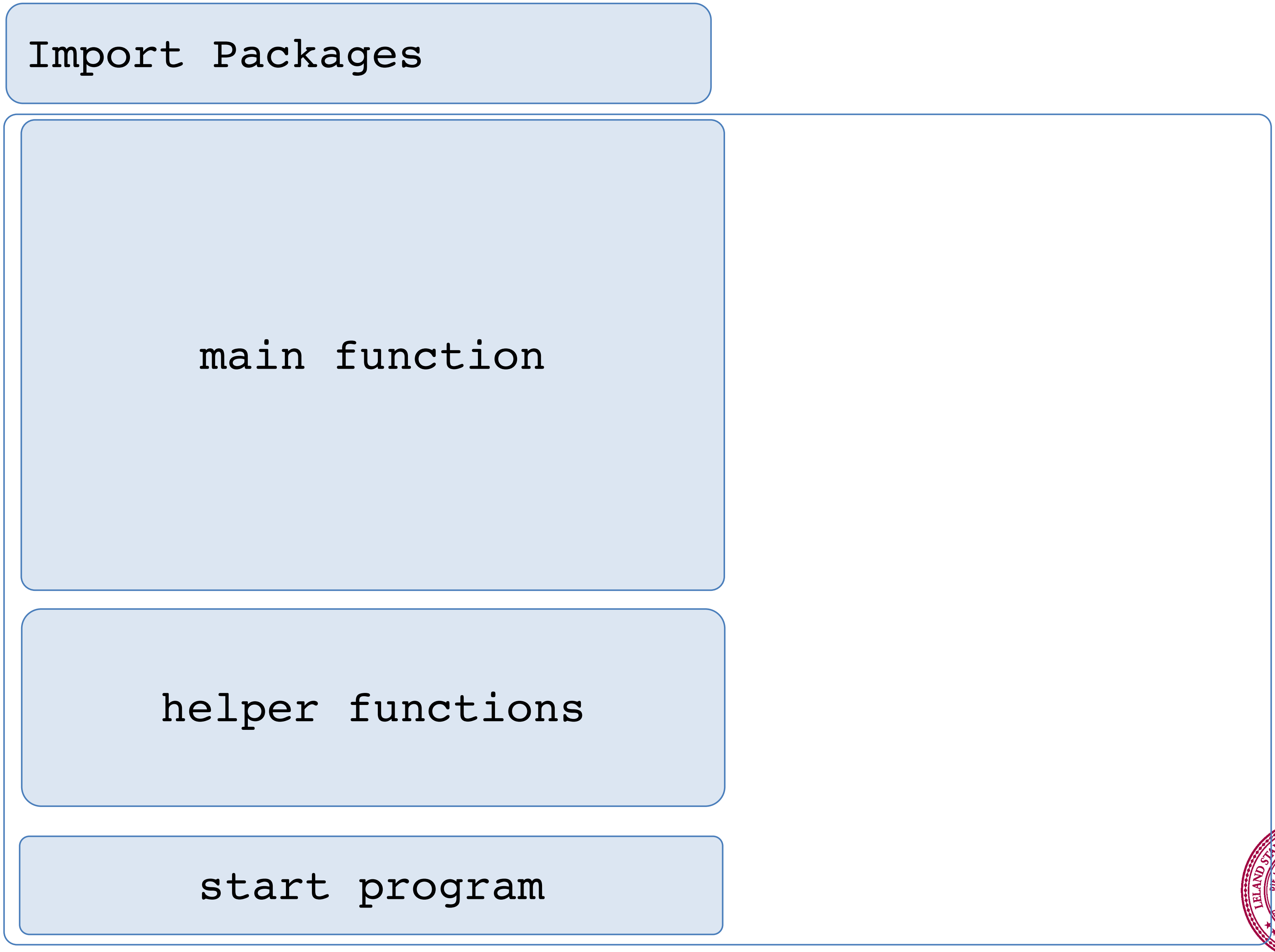
Import Packages

Program

Anatomy of a Program

```
Import Packages
```

Anatomy of a Program



Anatomy of a Program

```
Import Packages
```

```
def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

        helper functions

          start program
```

Anatomy of a Program

```
Import Packages
```

```
def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()
```

```
start program
```

Anatomy of a Program

```
Import Packages
```

```
def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

Anatomy of a Program

```
from karel.stanfordkarel import *
```

```
def main():
   move()
   pick_beeper()
   move()
   turn_left()
   move()
   turn_right()
   move()
   put_beeper()
   move()

def turn_right():
   turn_left()
   turn_left()
   turn_left()

if __name__ == "__main__":
    run_karel_program()
```

Anatomy of a Program

```
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

Anatomy of a Program

```
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

This piece of the program's ***source code*** is called a ***function***.

Anatomy of a Program

```
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

This line of code gives the ***name*** of the function (here, run)

Anatomy of a Program

```
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

This line of code gives the ***name*** of the function (here, turn_right)

Anatomy of a Program

```
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

This is called a ***code block***

Anatomy of a Program

```
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

This is called a ***code block***

Anatomy of a Program

```
from karel.stanfordkarel import *

def main():
    move()
    pick_beeper()
    move()
    turn_left()
    move()
    turn_right()
    move()
    put_beeper()
    move()

def turn_right():
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```

This is called a ***code block***

Why Study CS?

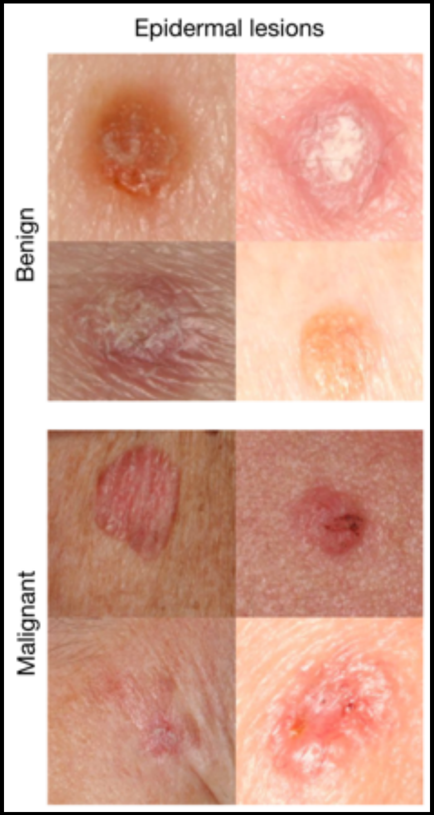
Joy of Building

Interdisciplinary

Closest Thing To Magic

Now is the Time

Now is the Time

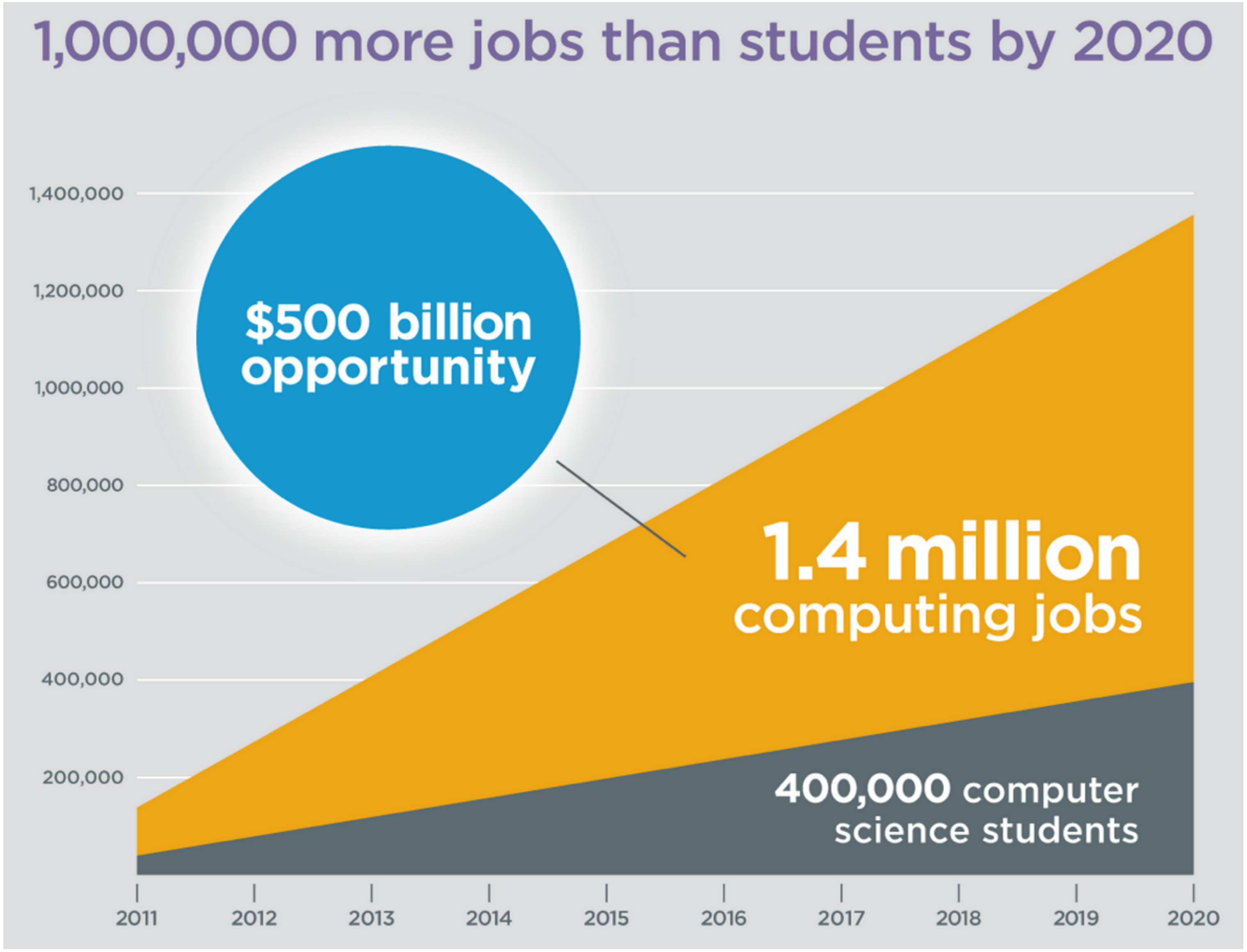


A machine learning algorithm performs **better than** the best dermatologists.

Developed this year, at Stanford.

Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115-118.

Oh and Its Useful



Code.org

Everyone is Welcome

The End

Actually, the beginning!