Enrolled or on waitlist?

Welcome to

COMP 110

Not enrolled or on waitlist?

You can join the waitlist for COMP 110 (Section 004)!

Today's Goals

Introductions

What is the course about?

What are the instructional and workload expectations?

Logistics?

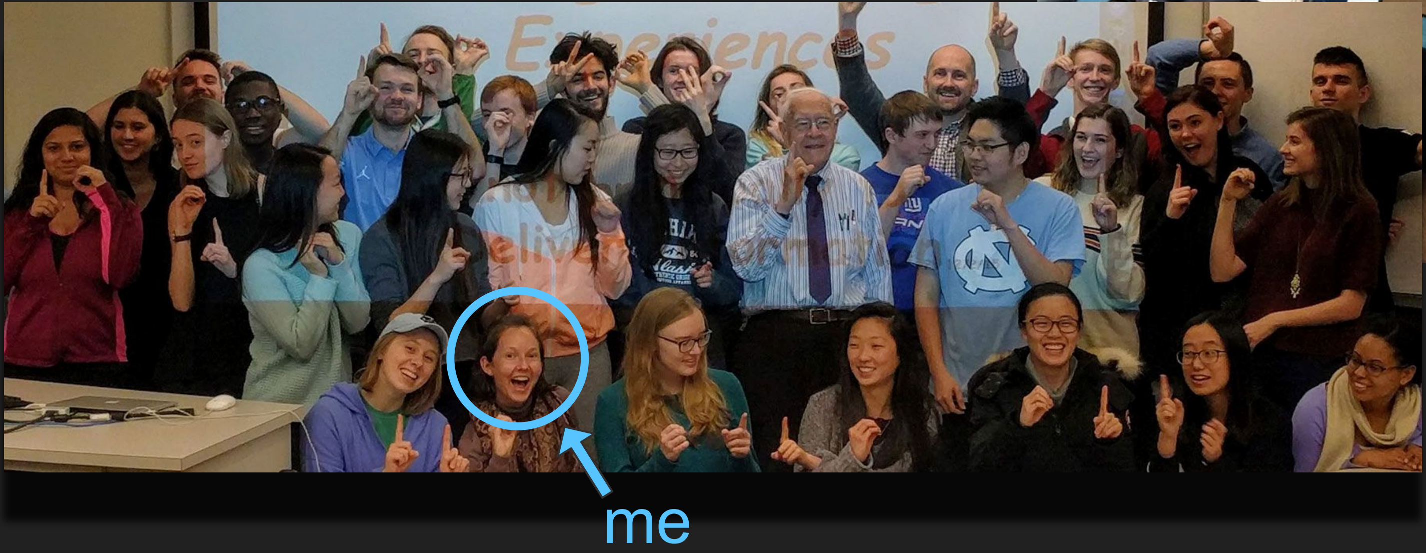
Homework

An Introduction to Coding

About me (Dr. Isabella ("Izzi") Hinks)

- Originally from Apex, NC
- Undergrad at UNC!
- COMP110 student → **UTA**

me
Control Flow (if/while/for) and Functions
Arrays (and Row-major 2D arrays)
Classes
Properties / Fields
Methods
Constructors
Scope (Global, Function Call Frame/Local, Block)
Recursion
Generics (Functions, Classes, Interfaces)
Higher-order Functions and Functional Interfaces (Filter, Map, and Reduce)
COMP 110



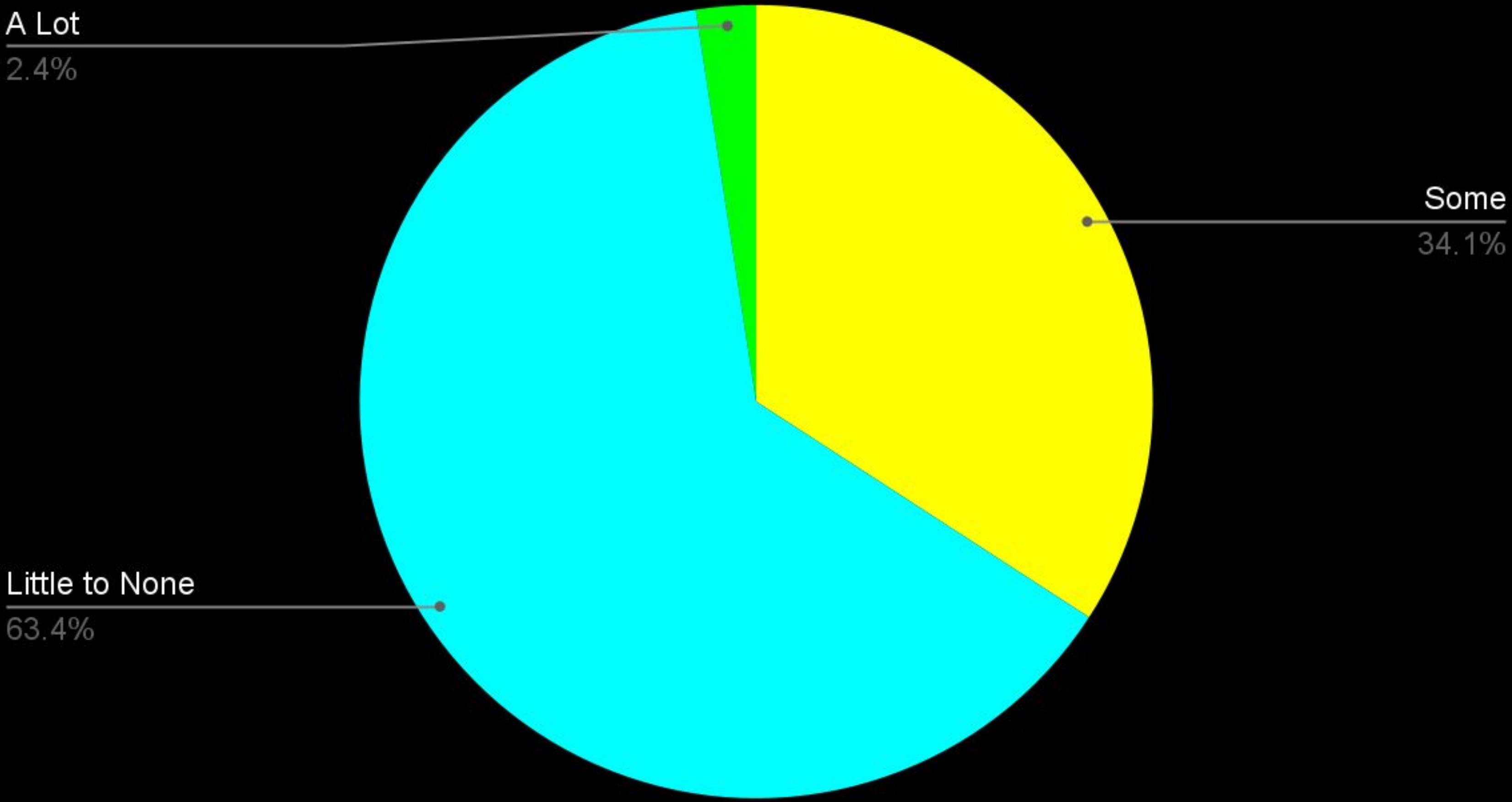
About me (Dr. Isabella ("Izzi") Hinks)

- Originally from Apex, NC
- Undergrad at UNC!
- COMP110 student → UTA → work, grad school… → **Professor**
- PhD @ NC State University

Your UTA Team!

- This course would be **impossible** for all of us, if not for them
- THE absolute best UTA team at Carolina. You will 💙 them
- This team can do it all: they'll help teach you concepts you're struggling with, guide review sessions, create study guides, build exercises, and more
- You will be assigned 2x UTAs who are your personal leads
- Drop-in, in-person office hours will be available to you for over 36 hours per week starting Monday!

TA’s coding experience before taking COMP110



You are a capable and diverse group!

- Who is new to Carolina?
- Who is coming into this course with *no programming experience*?
  - *A little experience*?
  - *A lot*?
- Who is *not* majoring in computer science?
  - Who *is* (or is considering it)?

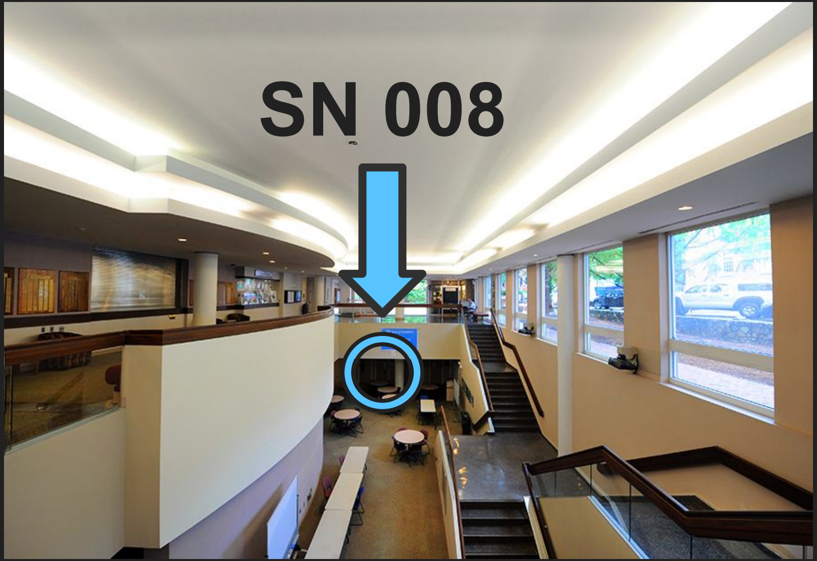
Zero Programming Experience Expected

- This course assumes *no* prior programming experience
  - (But some experience is OK!)

- COMP110 is a ***rigorous*** introduction to programming
  - 3 hours of lecture/lessons per week
  - and ~9 hours of practice/coursework

Open House this Tuesday – Friday

- 12-5 pm
- Sitterson Hall (SN) - Go downstairs to SN008
- Get help installing course software!
- Introduce yourself and meet some great people on the team!



Course Objectives

- You will learn the **fundamentals of programming**
  - Using common tools and techniques used by software engineers
  - Universal concepts that **apply to nearly all programming languages**
  - You will leave knowing what it feels like to be a programmer
- You will gain practice with **computational thinking**
  - **Thinking algorithmically** while breaking down problems step-by-step
  - Thinking at varying levels of **abstraction** by describing problems & solutions abstractly and precisely
- *Full curriculum linked in syllabus!*

Course Website

https://comp110-24f.github.io/

(Syllabus is on there!)

Grading Breakdown

- Prepare:
  - **10%** – (LS) Lesson Responses: Mult. choice re: basic concepts
- Practice:
  - **10%** – (CQ) Challenge Questions: Short-form coding questions
  - **30%** – (EX) Programming Exercises: Long-form coding projects
- Demonstrate Mastery:
  - **40%** – (QZ) 5x Quizzes: Paper and pencil
  - **10%** – (FN) Final Exam: Paper and pencil

Quizzes

Quizzes are ***in person***, with *pencil and paper*, during your section's lecture time. You are only permitted to be absent for *one quiz*.

NO MAKEUPS!

All dates are online!
For full policies, see syllabus.

CQs, Exercises, + Autograding

- You can re-submit to the autograder without penalty before the due date

- If you do not get full credit, stop and think about what might be causing a test to fail. Try again!

- Be careful to avoid a frustrating loop of "tweak one small thing, resubmit, tweak one small thing, resubmit, ..."
    1. The autograder gives you feedback – see if you can reproduce the error!
    2. If you find yourself stuck in this loop, stop by office hours (SN 008)

Use of AI

- AI tools like ChatGPT can be very useful in programming, but it takes a trained eye to use them properly!
- In this class, *you are training your eyes* to learn the fundamentals, so using AI will only hinder your understanding and won't strengthen you as a programmer!
- Considered a violation of the honor code.

Programming is a Practiced Skill

- Like playing an instrument, painting, writing cursive letters, dancing, singing, sports, wood working, quilting, and so on.... **Time spent *individually practicing* is the key to success.**
- This is *very different* from courses that are knowledge-based!
- The team and I want you to succeed in learning how to program, so we structure everything we do toward helping you practice individually.
- *Know what every line of your code is doing!*

How do *you* believe programming will be valuable toward achieving *your* personal goals?

Why are you in this course?

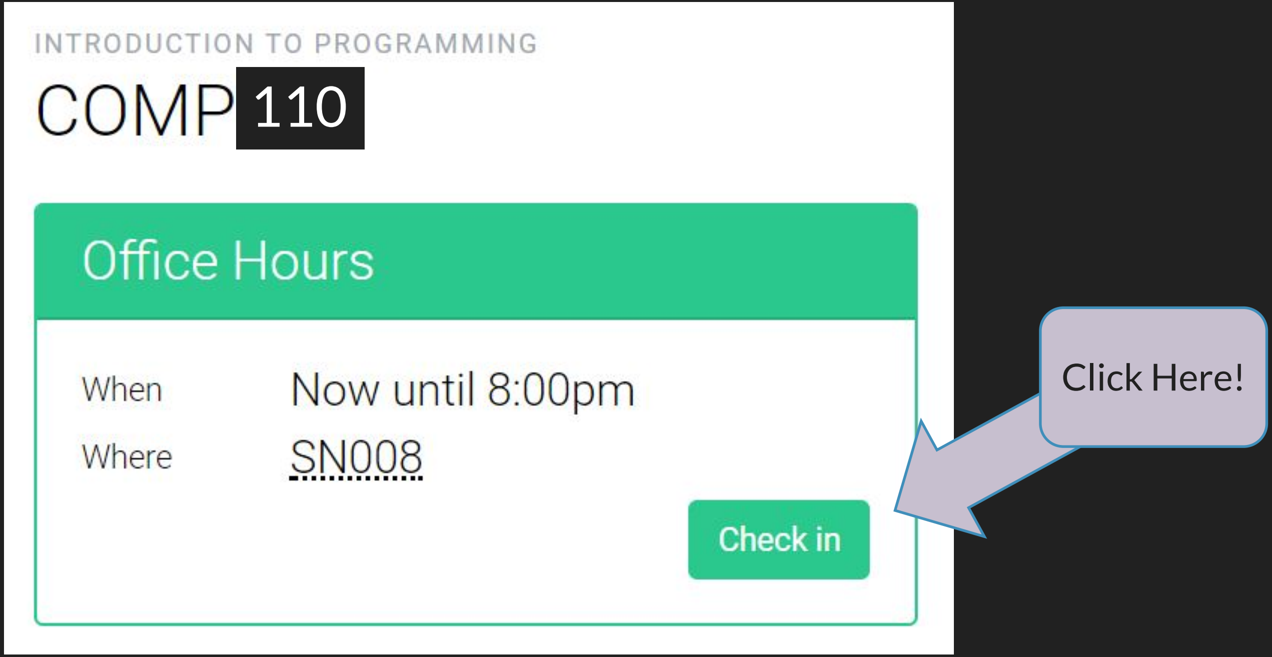
Think for a minute, introduce yourself to your neighbor(s) and discuss, then we'll share.

Office Hours

- Official Office Hours begin Monday, Aug 26
- Hours are on the website
- We use **Course.Care** (sign up info on website under “resources”!)
- General Rules:
  - Must submit a ticket to be seen
  - Limited to 15 minutes and one specific question per appointment
  - Completely lost? *Try tutoring!*

Office Hours Check-in Process - Starting MONDAY (8/26)

Click "Get Help" on the course home page



INTRODUCTION TO PROGRAMMING

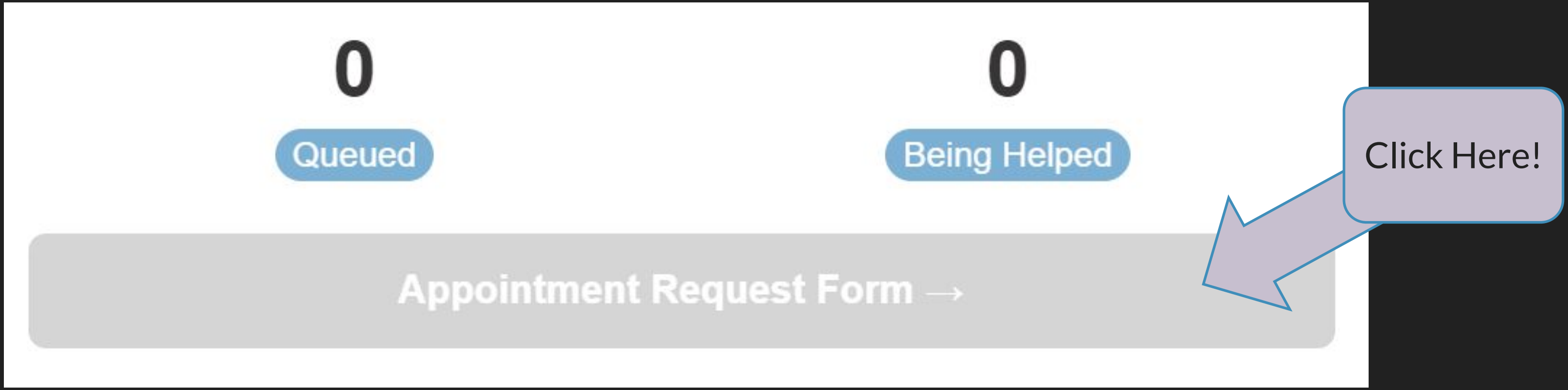
COMP 110

Office Hours

When Now until 8:00pm

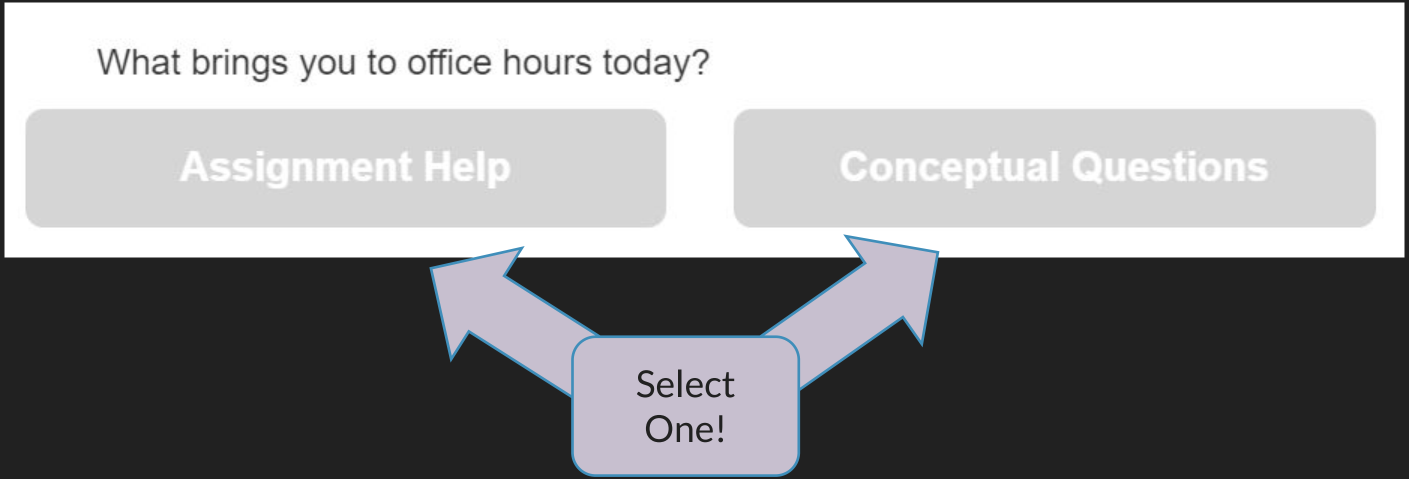
Where SN008

Office Hours Check-in Process

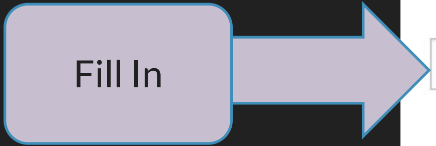


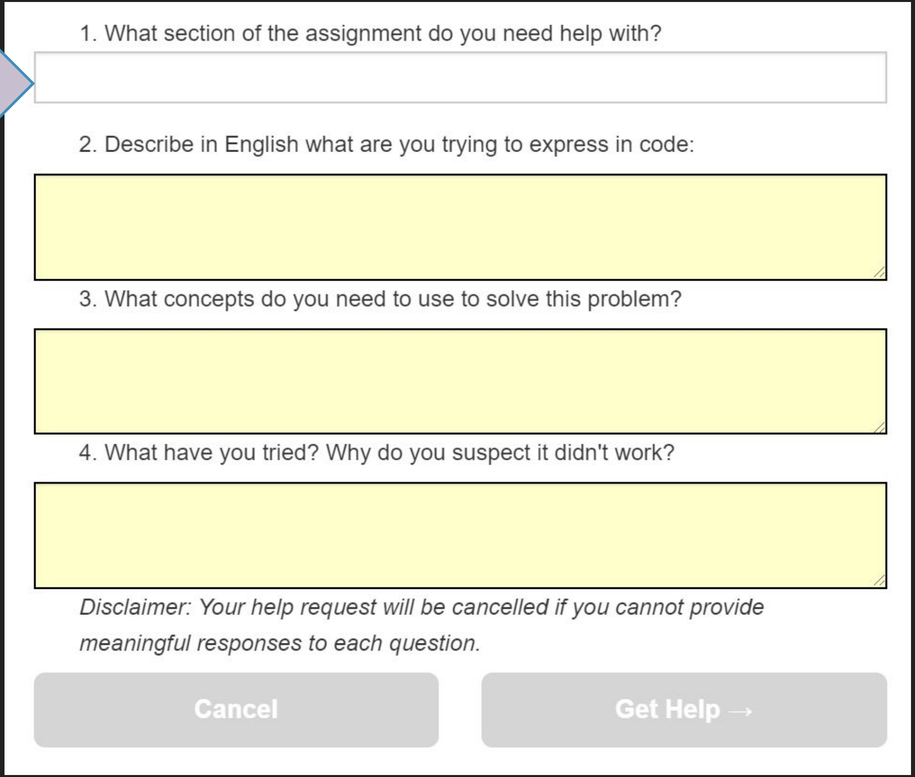
You can see how many people are currently being helped waiting to be helped.

Office Hours Check-in Process


What brings you to office hours today?

Office Hours Check-in Process




1. What section of the assignment do you need help with?

2. Describe in English what are you trying to express in code:

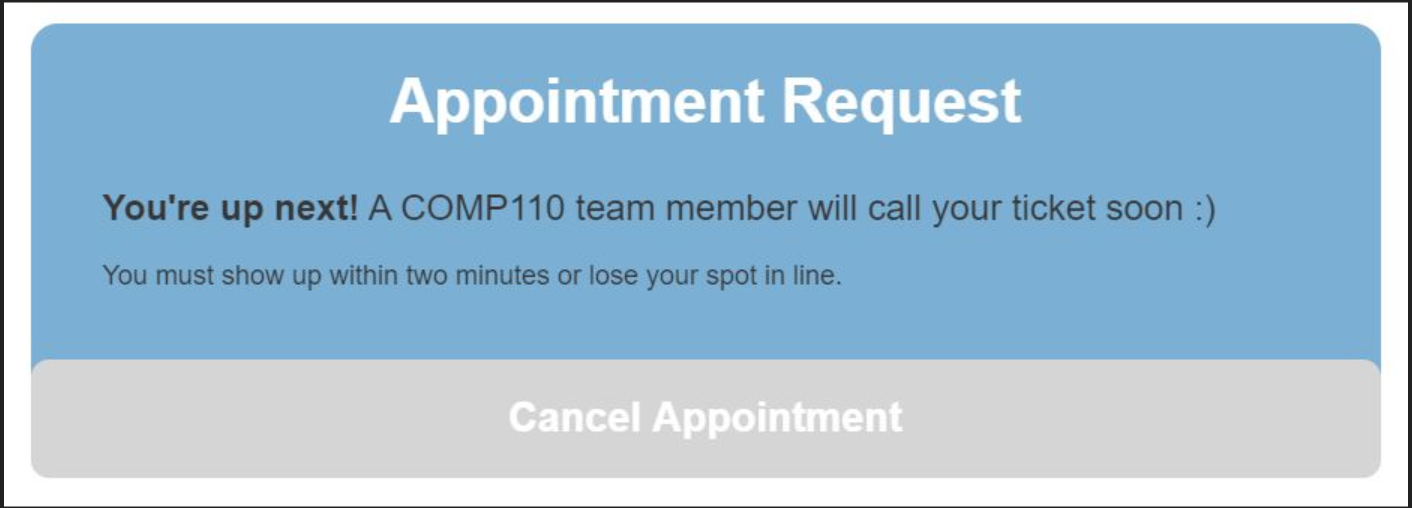
3. What concepts do you need to use to solve this problem?

4. What have you tried? Why do you suspect it didn't work?

Disclaimer: Your help request will be cancelled if you cannot provide meaningful responses to each question.

<u>IMPORTANT</u>: You must demonstrate *effort and thought* in these fields. If you do not, the TAs are instructed to cancel your request so you can try again.

Office Hours Check-in Process

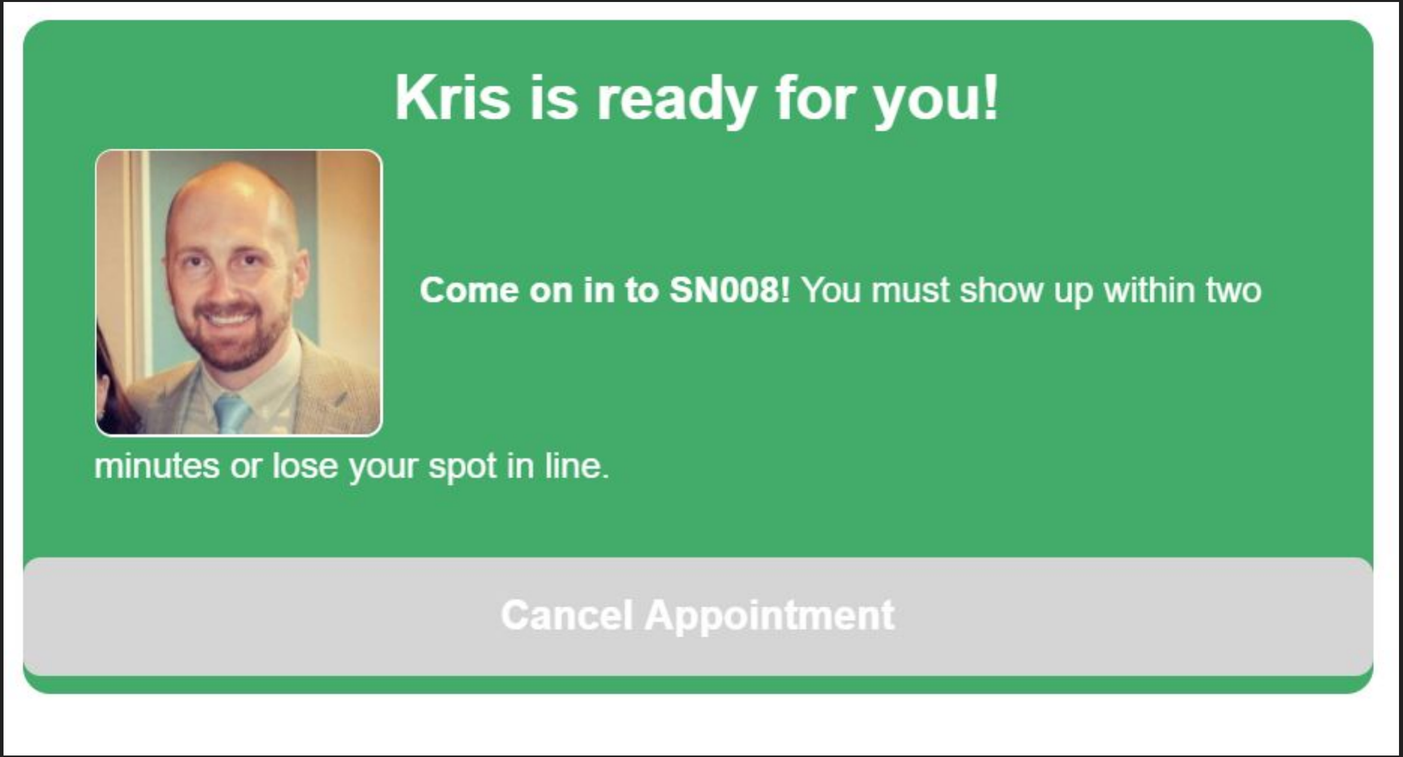


Appointment Request

You're up next! A COMP110 team member will call your ticket soon :)

You must show up within two minutes or lose your spot in line.

Office Hours Check-in Process



Kris is ready for you!

Come on in to SN008! You must show up within two minutes or lose your spot in line.

Tutoring

- Best for longer-form help (> 15 mins) and conceptual questions
- Official days/times will be announced on the course site

Feedback + Help

Feedback is always welcome!

For help, you can post your questions on EdStem or email comp110help@gmail.com

COMP 110

CL01: An Introduction to Coding

Computational Thinking

- Strategic thought and problem-solving
- Can help perform a task better, faster, cheaper, etc.
- Examples:
  - Meal prepping
  - Making your class schedule
  - “Life Hacks”

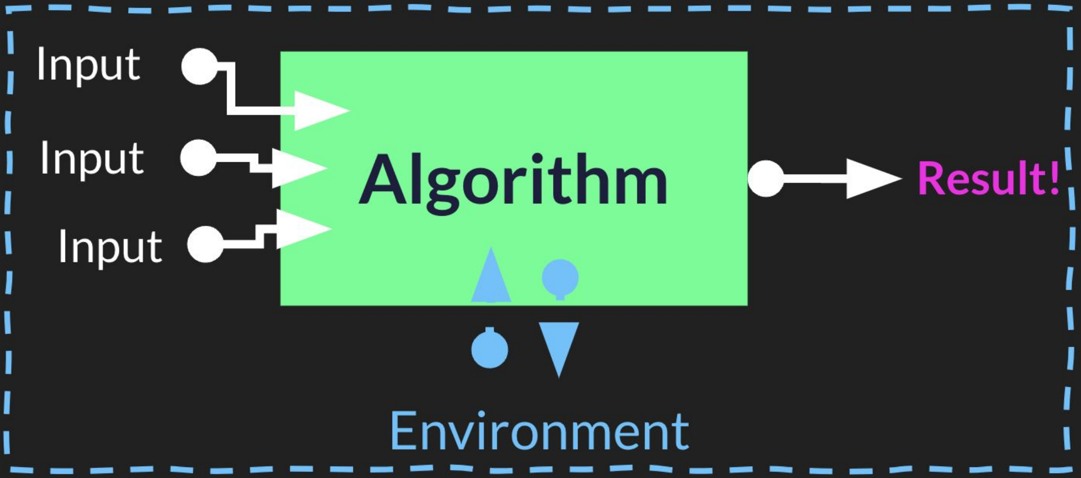
Algorithms

Input is data given to an algorithm

An **algorithm** is a series of steps

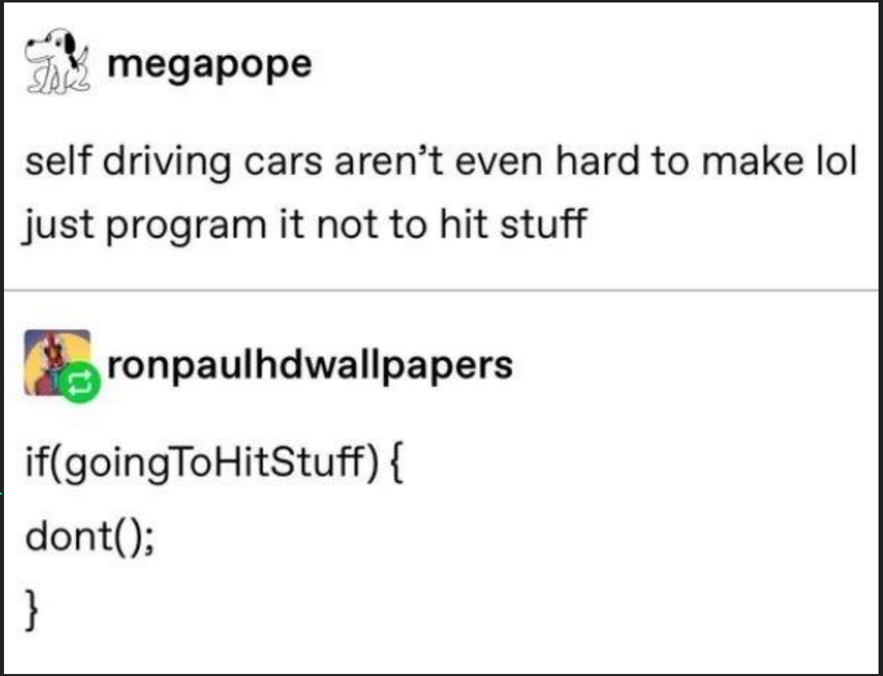
An algorithm **returns** some **result**

An algorithm *may* be influenced by its **environment** and it *may* produce side-effects which influence its environment.



Example: Self-driving cars

Algorithm


megapope

self driving cars aren't even hard to make lol just program it not to hit stuff

ronpaulhdwallpapers

```
if(goingToHitStuff) {
dont();
}
```

What is an algorithm?

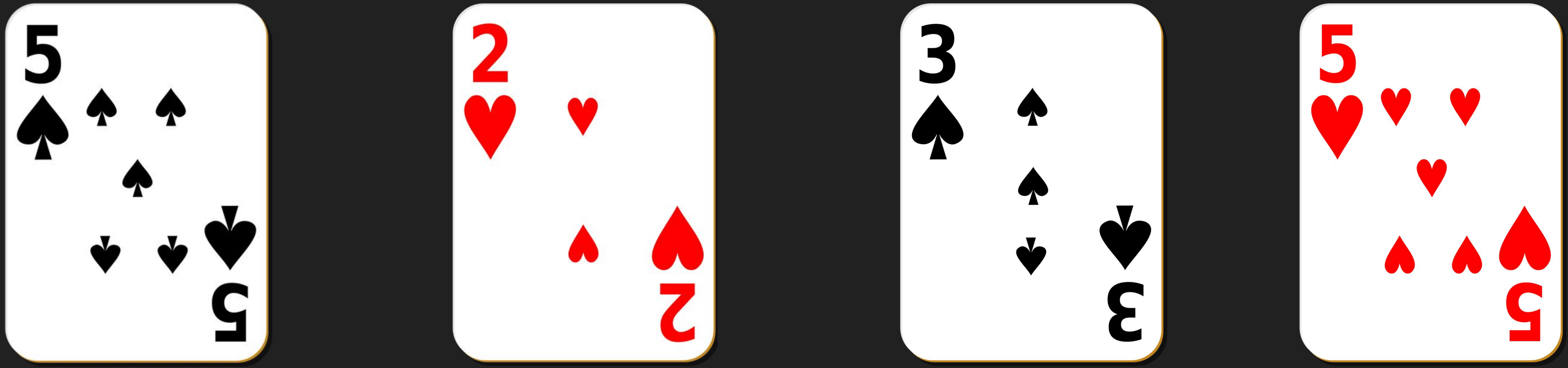
- A set of steps to solve a general problem
- Finite
- Can handle a problem of arbitrary size

Discussion

What are examples of computational thinking that you use day to day?

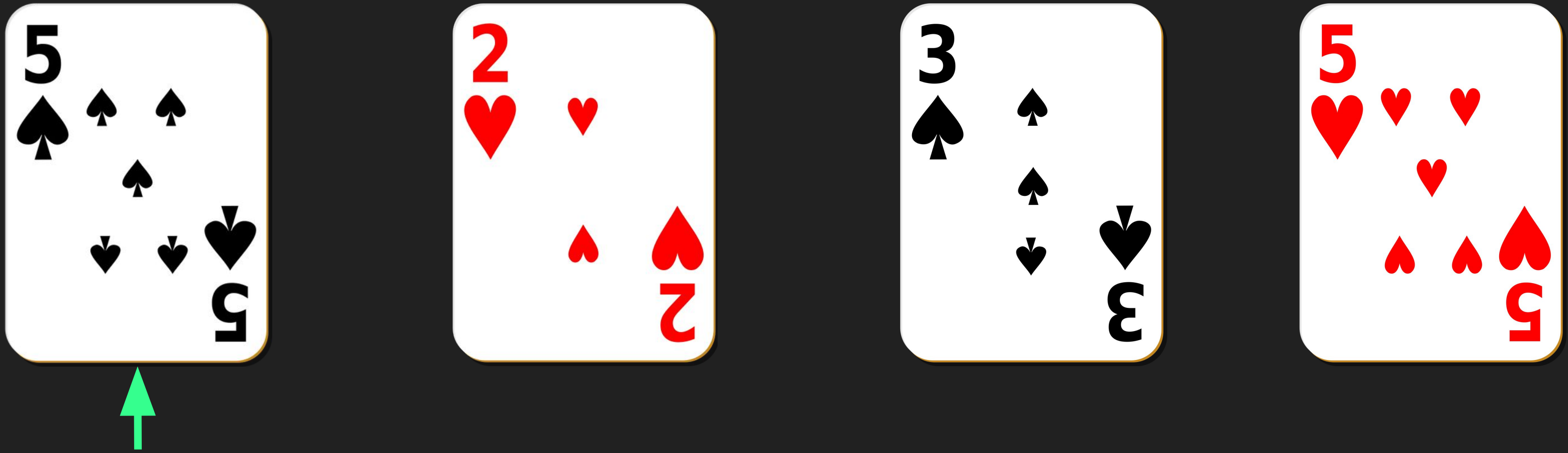
What kind of algorithms do you use to implement these ideas?

Finding the Lowest Card in a Deck

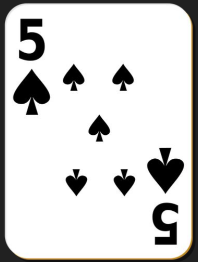


- Go from left to right
- Remember the lowest card you’ve seen *so far* and compare it to the next cards

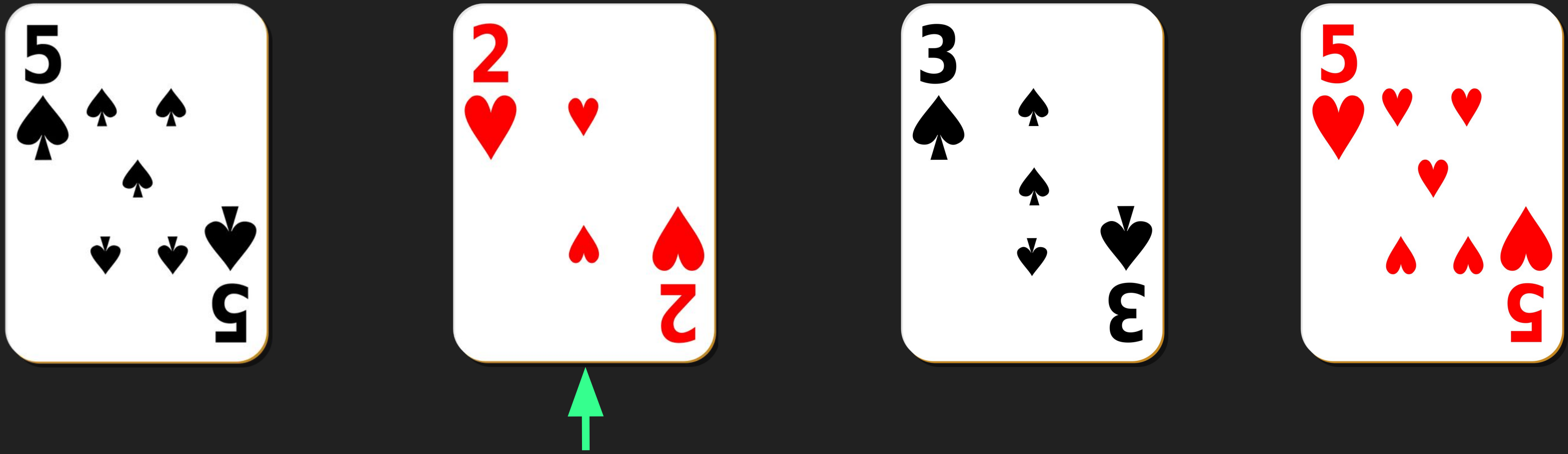
Finding the Lowest Card



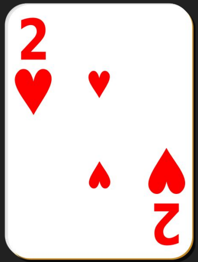
Low card:



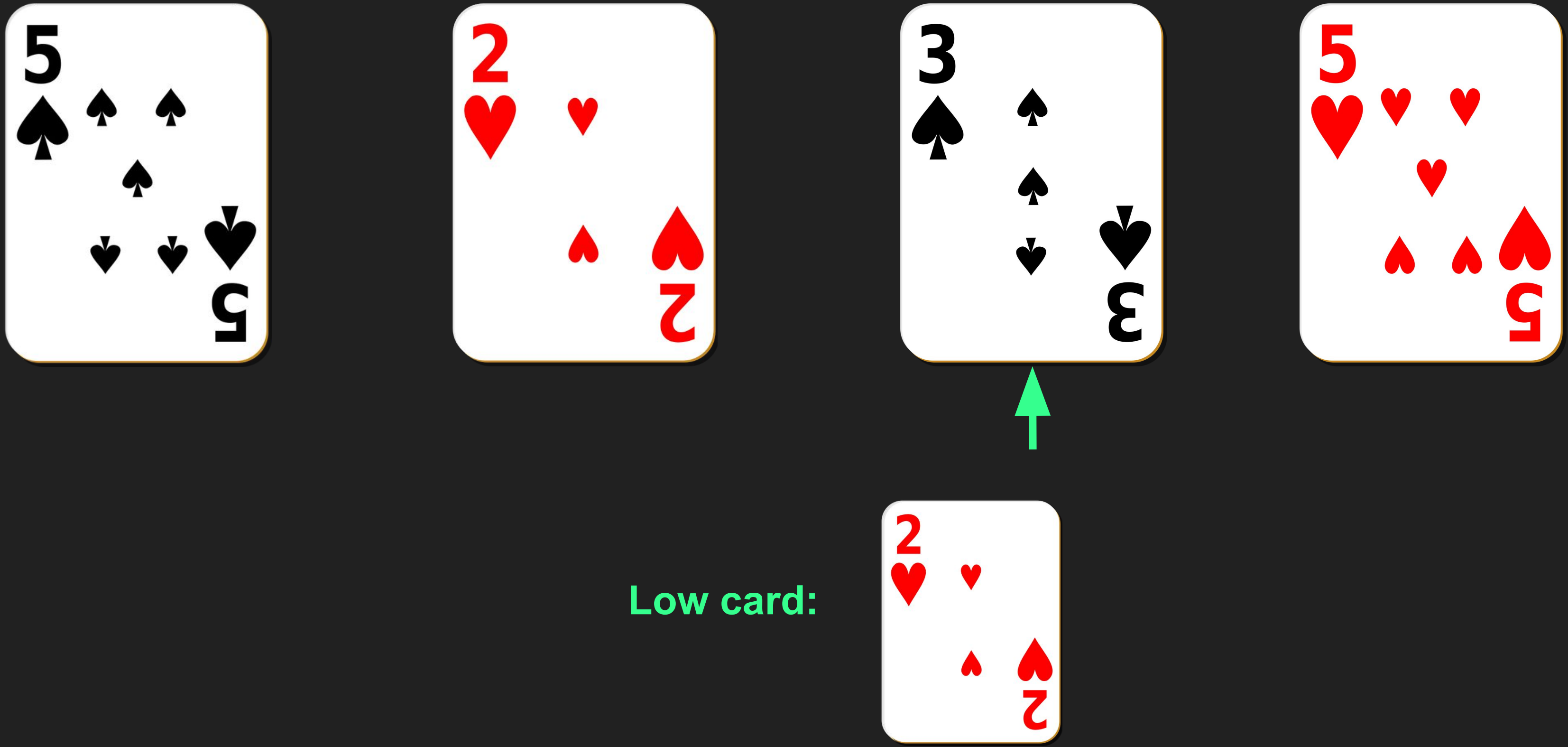
Finding the Lowest Card



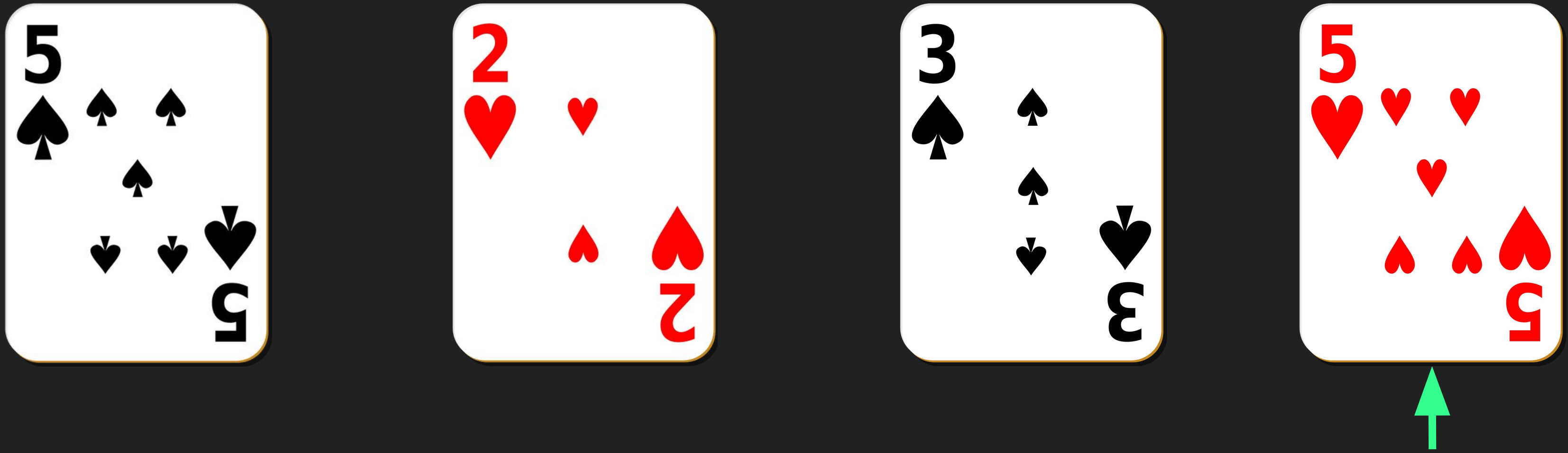
Low card:



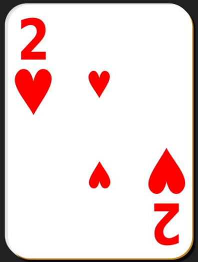
Finding the Lowest Card



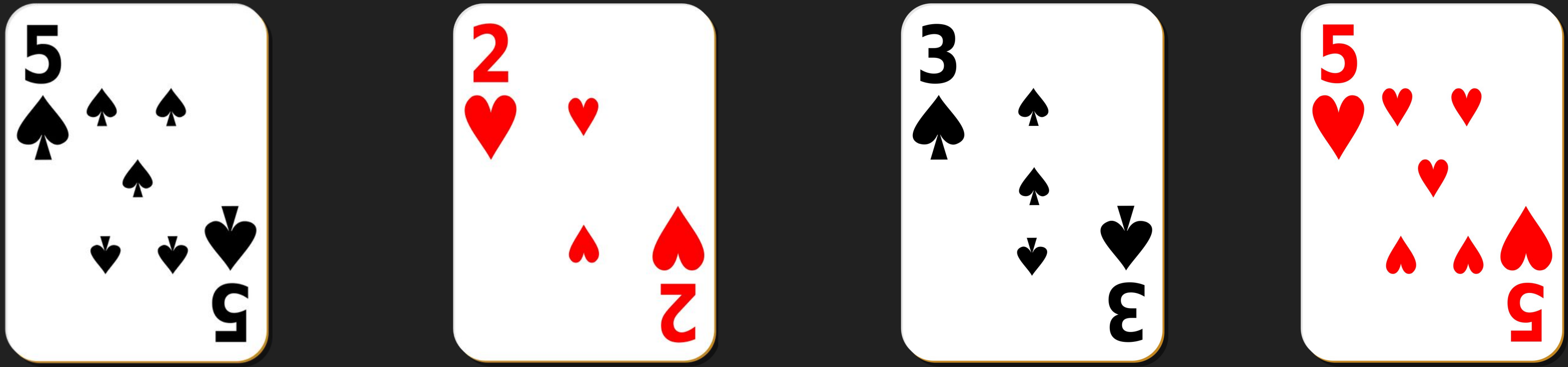
Finding the Lowest Card



Low card:

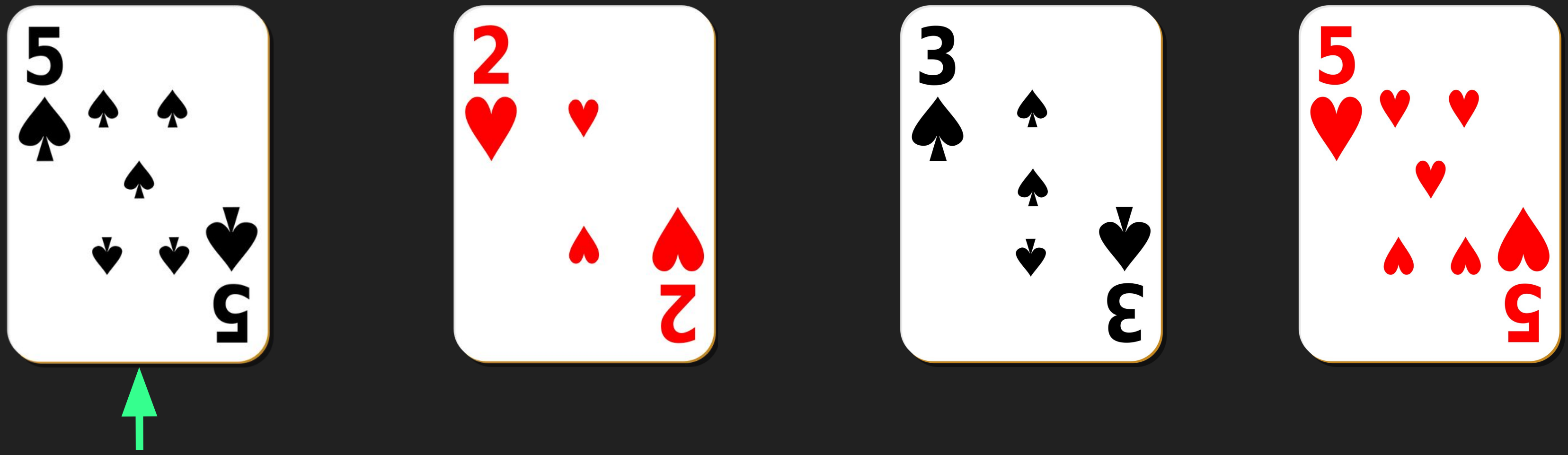


Finding the Lowest Card

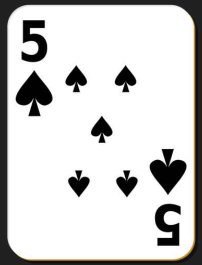


Low card:

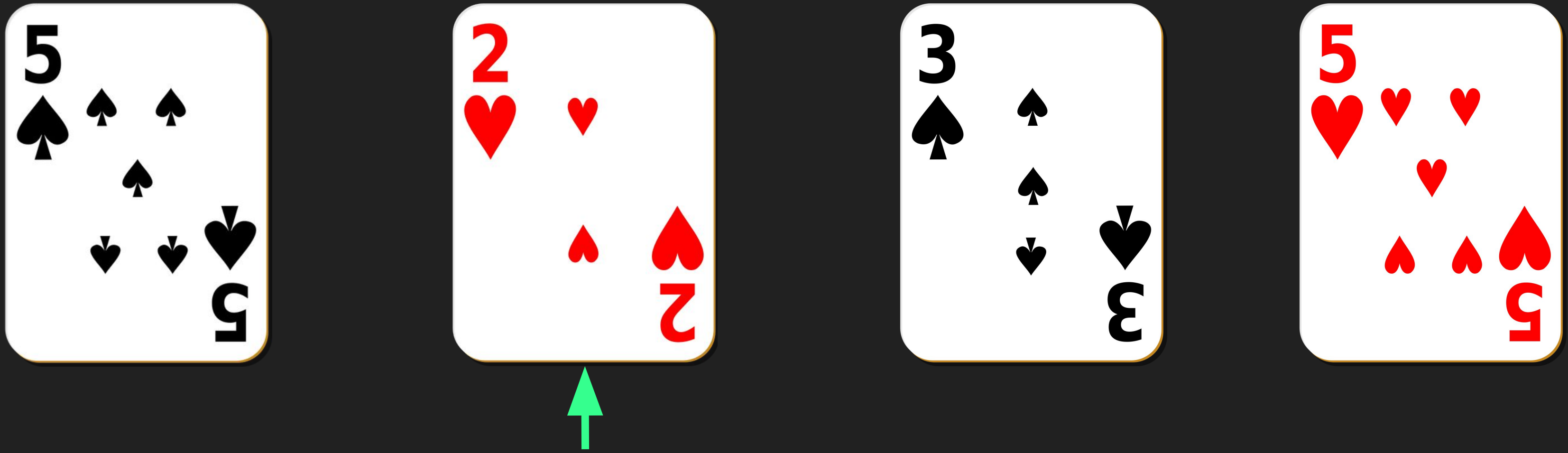
Finding the Lowest Card



Low card:

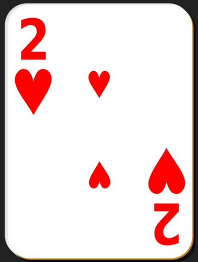


Finding the Lowest Card

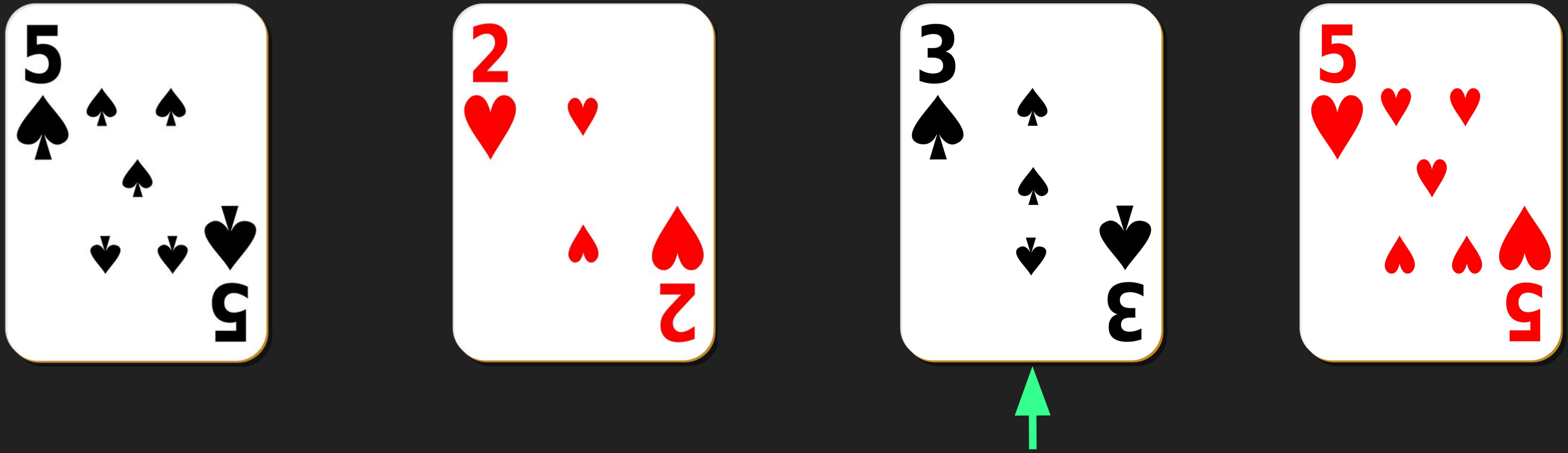


$2 < 5?$ ✓

Low card:

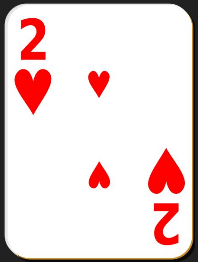


Finding the Lowest Card

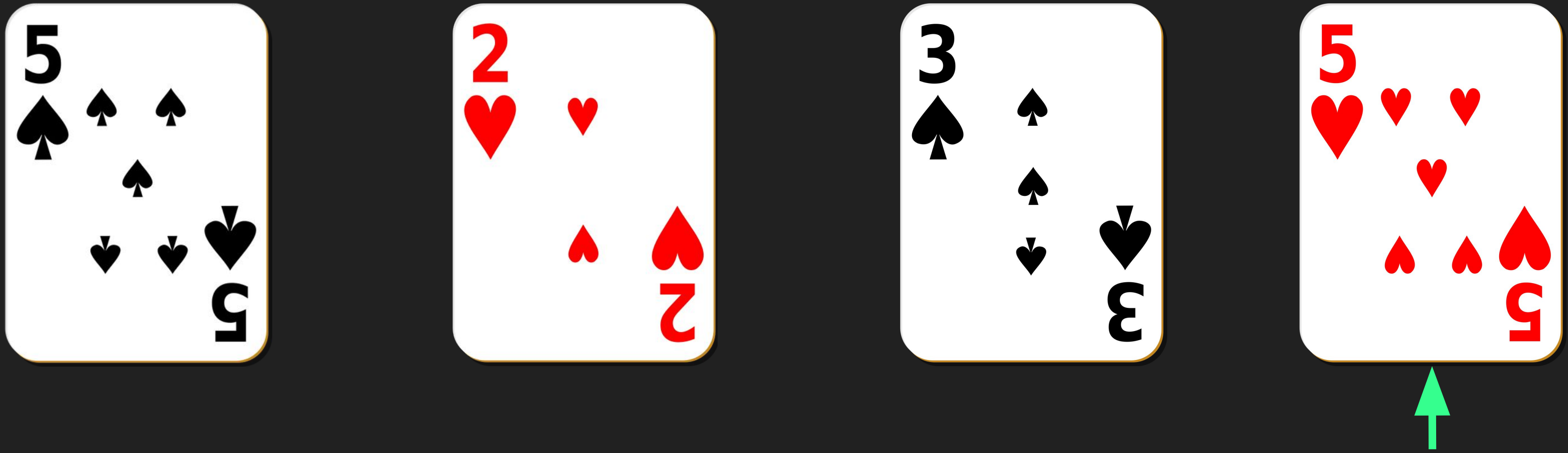


3 < 2? ❌

Low card:

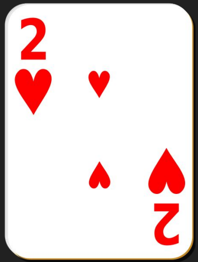


Finding the Lowest Card

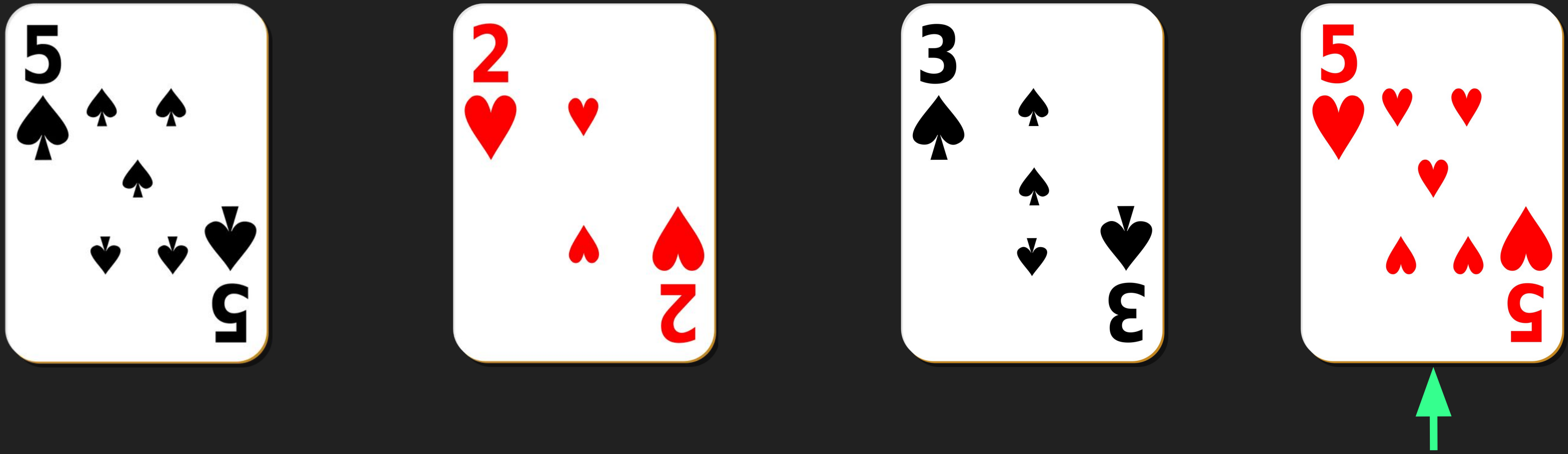


$5 < 2?$ ✗

Low card:



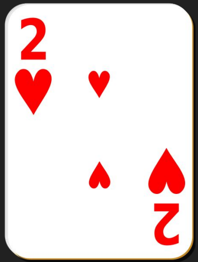
Finding the Lowest Card



$5 < 2?$

Relational Operator

Low card:



Pseudocode

Looks like code, but simplified and readable.

Not meant to run on a computer.

Helps you outline what your algorithm is going to look like.

You should be able to expand on your pseudocode to help you write actual code!

```
If (going to hit stuff):
    dont()
```

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you've seen *so far* and compare it to the next cards

Pseudocode:

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you've seen *so far* and compare it to the next cards

Pseudocode:

```
lowest_card = first card in deck
```

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you’ve seen *so far* and compare it to the next cards

Pseudocode:

```
lowest_card = first card in deck
```

$$\underset{\textbf{Assignment}}{\uparrow}$$

Assignment

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you’ve seen *so far* and compare it to the next cards

Pseudocode:

```
lowest_card = first card in deck
Repeatedly until end of deck:
    if current_card < lowest_card:
        lowest_card = current_card
```

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you've seen *so far* and compare it to the next cards

Pseudocode:

```
lowest_card = first card in deck
Repeatedly until end of deck:
    if current_card < lowest_card:
        lowest_card = current_card
```

Loop

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you've seen *so far* and compare it to the next cards

Pseudocode:

```
lowest_card = first card in deck
Repeatedly until end of deck:
    if current_card < lowest_card:
        lowest_card = current_card
```

Conditional

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you've seen *so far* and compare it to the next cards

Pseudocode:

```
lowest_card = first card in deck
Repeatedly until end of deck:
    if current_card < lowest_card:
        lowest_card = current_card
```

Relational Operator

Finding the Lowest Card Pseudocode

- Go from left to right
- Remember the lowest card you've seen *so far* and compare it to the next cards

find_lowcard(deck)

```
lowest_card = first card in deck
Repeatedly until end of deck:
    if current_card < lowest_card:
        lowest_card = current_card
```

Function

Takeaways

- Pseudocode: simple and readable version of algorithm that resembles code
- Assignment Operator: Assigns a variable some value
- Loop Statement: Repeatedly performs an action a fixed number of times
- Relational Operator: Compares two values
- Conditional Statement: A statement that only performs an action under certain conditions
- Function: Generalizes code to work for a generic input

Again, you don't need to know these right now, but I want you to have a point of reference when you do learn them!

Homework!

- Read Syllabus and Support on Course Page
- Respond to Lesson 00 (LS00) Gradescope Questions
  - Due Wednesday at 11:59pm
- Course Setup + EX00 (due August 27 at 11:59pm)
  - Come to open house for help!