

# INTRO TO PROGRAMMING WITH PYTHON

## FUNDAMENTALS: BECOMING A COMPUTER

Monday, February 10 2025 TOH210



# ANNOUNCEMENTS

Discord provides a space for students to ask questions and connect with one another. All TAs have already joined and are ready to assist students with any questions they may have

Use the QR code to the right or click the link here:

<https://discord.com/invite/F8wsBCEsM4>

## St. Olaf CS Discord Server



<https://discord.gg/F8wsBCEsM4>



# Takeaways from last time

Things you should know:

- everything in the syllabus
- how to access + use Runestone
- how to copy colab workbooks
- Moodle course



★ Introduction to Programming Spring 2024-25  
Spring 2024-25

# UPCOMING SCHEDULE

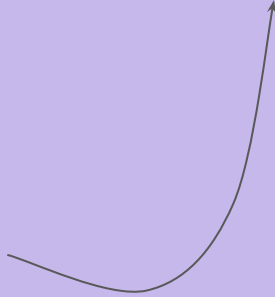
google sheet [link](#)

Week	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
	IO: Input + Output					
Remember, this includes ALL material for the week so it can take a while to complete		Due 11:59pm: Reading Assignment 1 (Runestone) + Reading 1 Discussion (Moodle)	Due 11:59pm: About Me (Moodle quiz)			Due 11:59pm: HW 1
<u>LOTS of new vocabulary</u> which means you may want to revisit to help solidify your understanding		<b>IO1:</b> inside computers, algorithms, flowcharts, pseudocode, print		<b>IO2:</b> Variables, operators, data types (int, float, string), input,  Project1, IDLE intro		<b>IO3:</b> Writing Programs, debugging  Project 1 worktime
2/10 - 2/14		Chapter 1		Chapter 2		Chapter 3

# COLAB WORKBOOK

Link: [click for access](#)

Have at least **one member of table group open this** to take notes. After we are done, make sure to share with the rest of the members of your group

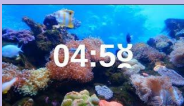
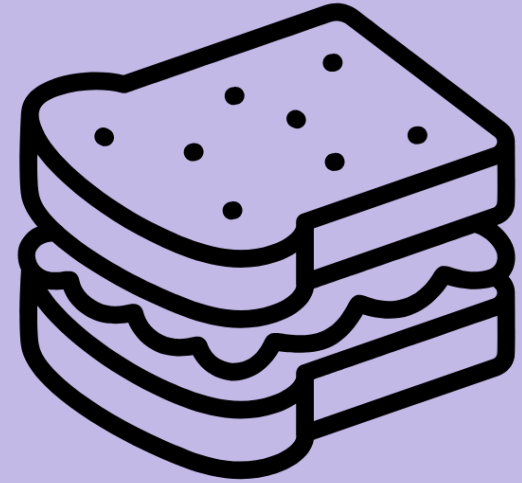




## WARM UP

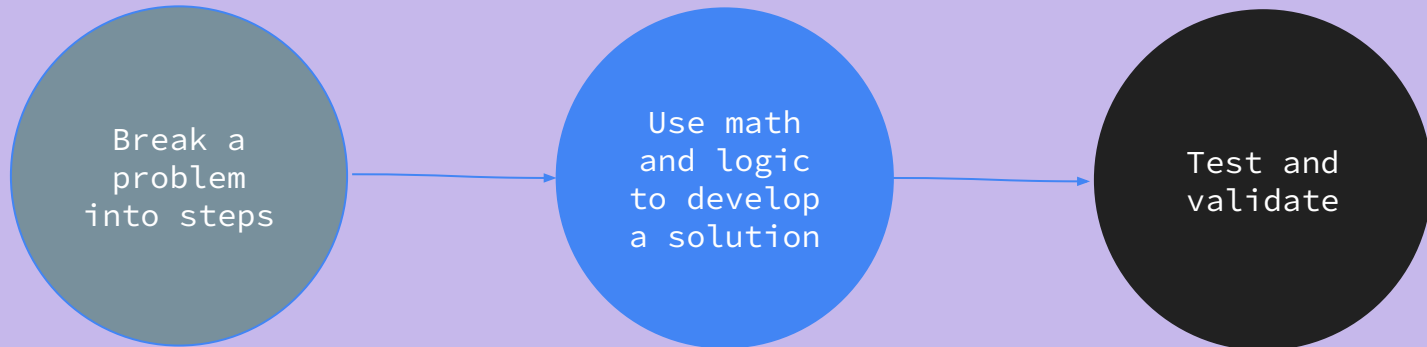
Think about the process of making a peanut butter and jelly sandwich. How would you describe the sequential steps involved in making the sandwich? **Why does the word sequential matter here?**

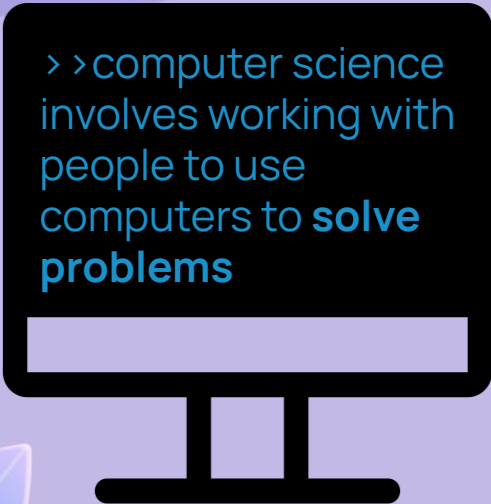
Let's say both you and a friend both wrote down the process for creating your own versions of pb+j, but have a different number of steps (you have 10 while they have 5). **How can this be?**



# The fundamental skill in computer science is problem solving

---



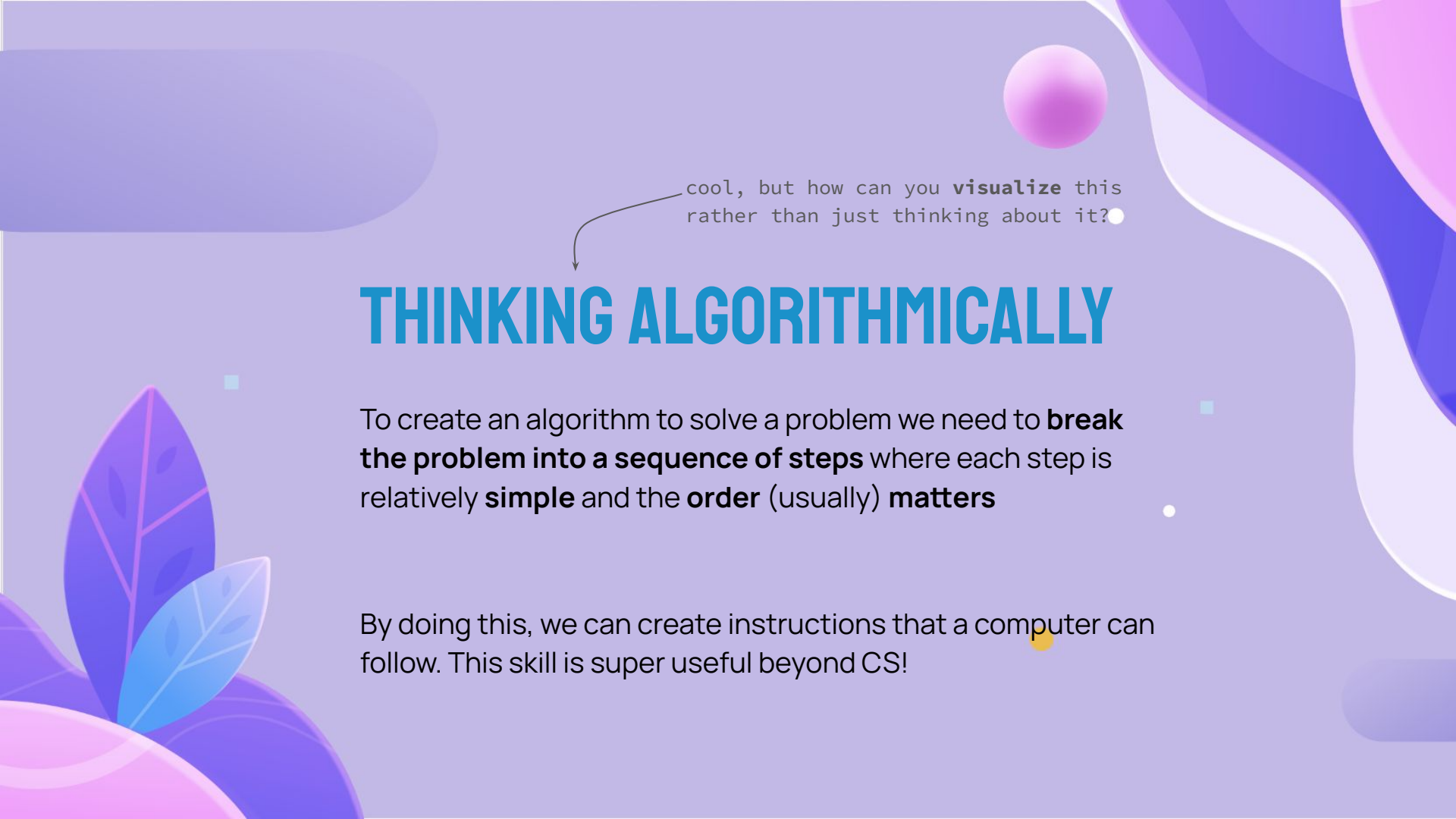


> > computer science  
involves working with  
people to use  
computers to **solve**  
**problems**

An **algorithm** is a  
step-by-step list of  
instructions for  
**solving a problem**

Learning the nuances of a computer language is only one part of coding. The other, usually more difficult, part is to understand what instructions to give to solve a given problem.





cool, but how can you **visualize** this  
rather than just thinking about it?

# THINKING ALGORITHMICALLY

To create an algorithm to solve a problem we need to **break the problem into a sequence of steps** where each step is relatively **simple** and the **order** (usually) **matters**

By doing this, we can create instructions that a computer can follow. This skill is super useful beyond CS!

Let's see an example of this!

We can plan out our **algorithm** by using  
**flowcharts** and **pseudocode**



Flowcharts are a **visual** way of using boxes and arrows to show the order of instructions

Good for **decision making** or if there are a lot of user interactions

Pseudocode is an informal algorithm using simple English **sentences** and is organized by indentations (much like a computer program)

Good for mathematical **processes**

# CAFE ORDERS



We want to create a program that sends the drink a customer selects at the automated coffee shop to the barista. The program should prompt the user to select which drink they want, if it's hot or iced, and then send the drink to the barista.

Before we code anything, let's create some pseudocode to figure out what we'd like this program to do!

1. ask user for **drink choice**
2. ask if drink is **hot or iced**
3. **confirm** drink is correct
4. **send** drink to barista

# DISCUSSION

1. ask user for **drink choice**      Select your drink (latte, coffee, chai): **chai**
2. ask if drink is **hot or iced**      Select hot or iced (hot, iced): **iced**
3. **confirm** drink is correct      You ordered a **iced chai**. Is that correct (yes, no): **yes**
4. **send** drink to barista      Great! One **iced chai** coming your way :)

Once you have this, grab a partner and discuss where could you **\*\*add improvements\*\***. What would you have to add so a customer can select multiple drinks? How about adding a flavoring? What happens when a user enters an invalid option?

Add any steps you think would be important to replicating the ordering process at a cafe.





## DISCUSSION

# REMEMBERING LIGHTHOUSE CODES



You are a part of a network of spies within a lighthouse organization. Your co-conspirators have left a clue for you. You see five lighthouses and want to remember which are lit (blue) and which are not.

Describe **how you would remember this information** if all you could share was a number or gesture. Think of multiple ways this could be done.



3



1 1 0 0 1

25



## Algorithm

Count all blue  
lighthouses

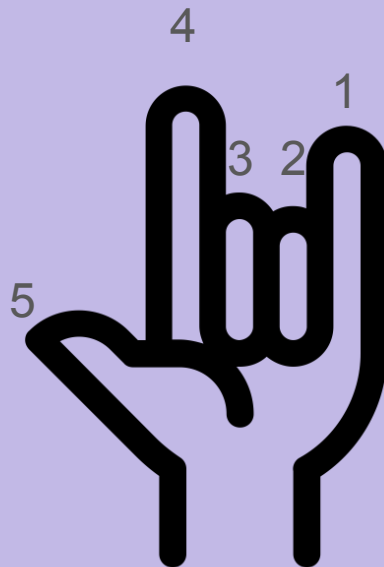
3

This tells me how  
many are lit, but not  
which ones



## Algorithm

If lit, then put finger up.  
Otherwise, keep finger down.



This tells me how  
many and which  
position they are in

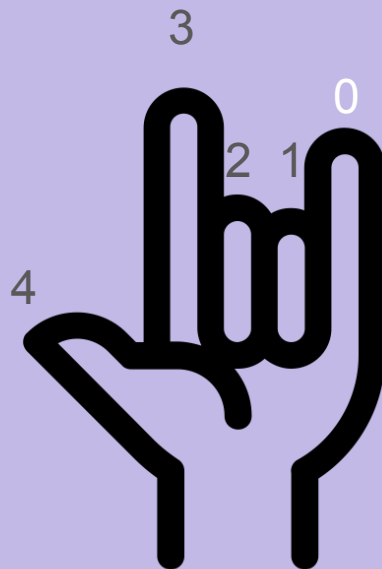
Maybe **541** or **145**  
could also work





## Algorithm

If lit, then put finger up.  
Otherwise, keep finger down.



Fun fact: in CS  
we often start  
counting at 0!

Maybe **430** or **034**  
could also work



## Algorithm

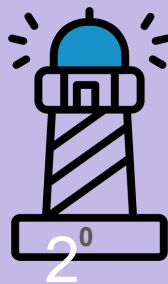
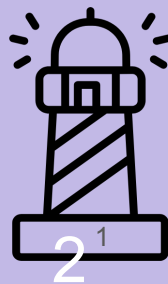
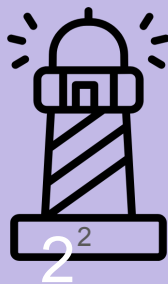
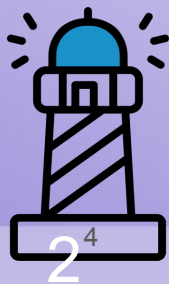
If lit, then 1.

Otherwise, 0.

1 1 0 0 1

This tells me how many and which position they are in

This is called **binary** and it's the language that computers speak



## Algorithm

If lit, then add up  
corresponding power of 2.  
Otherwise, add 0.

$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
16	8	4	2	1

$$16 + 8 + 0 + 0 + 1 = 25$$

This is the decimal  
number we use to  
represent the binary  
number **11001**

**HOW DO WE TRANSLATE HUMAN LANGUAGE  
TO COMPUTER LANGUAGE?**

# Programming Languages

Thousand of programming languages have been created. Which one you should use depends on a number of factors such as the type of program being written and which kind of computer it will run on.

Here are some popular ones, as well as some really old ones.

## BASIC

Designed in 1964 at Dartmouth College, very popular when home computers first became available.

## Fortran

Designed in 1954 at IBM, very popular for calculations on large computers. Still used for **weather forecasting** at NASA.

## COBOL

Designed in 1959, still being used in **banking**.

## C

One of the most popular and used for programming **hardware**

## C++

Based on C with some flair. Used in programs that need to be **fast** like console games

## Objective-C

Based on C with some flair. Popular because of its use in Apple and **iOS**

## MATLAB

Ideal for programs that carry out loads of **calculations**

## Python

Very **versatile**. Used in backend of YouTube and Google, data science, among others.

## Java

Very **versatile**. Often used for coding on **Android OS**

## JavaScript

Used for programs that run in the **browser**, such as simple games

## Ruby

Automatically turns lots of information into **web** pages

## Ada

Used to control **spacecraft**, satellites, and airplanes

## R

Used in **statistical** courses, biomedical research, data science.

## GENERAL PURPOSE

can program almost anything like web dev, scientific computing, data analysis, AI

## MOST POPULAR

lots of existing programs are written in it

## FUN NAME

it's creator was a big fan of Monty Python (British sketch comedy group)

## HIGH LEVEL

Syntax is designed to be relatively simple, so it is easier and faster to program in than other programs

# WHY PYTHON?

Languages are **designed** for **different** types of programming, but they share a bit of the **same** basic **structure**

displays information to users by  
taking in an **argument** (input)

arguments to print can be  
**strings** or **numbers** or **variables**

# print function

```
print("hiya! :)")
```

```
print(42)
```

```
text = "hiya! :)"  
print(text)
```

```
number = 42  
print(number)
```

# WORKBOOK TIME!

## BEFORE NEXT TIME

Finish workbook

Reading 1 + discussion

About Me

Take a look at HW 1

