# Intro

can code communities

# Introductions

Who is CanCode Communities?

Who are your instructors?

Icebreaker:

- Why are you taking this course?
- Have you programmed before? If so what languages?
- What is your favorite hobby?

# Expectations for the Course

- You are expected to participate in class exercises.
- Ask questions if you have them.
- Reach out if you need additional help.
- You should be completing any exercises assigned.

Please refer to your Welcome Guide for more detailed information on the expectations.

# Outline of the Course

# Intro to Python

- Python is a programming language. It is very popular in the computer science industry.
- The syntax of Python is straightforward.
- Python is a great beginner language to get into programming.
- Because of its popularity, Python has many tools available to use within and for it.
- What are some things you would like to do with Python?

# How to set up your IDE

# How to use a .py file

# Example

Let’s create a Python program that prints “Hello World!”

# How to Code

How would you define programming, in your own words?

# How to Code

Programming is the process of writing commands for a computer to follow.

An algorithm is a series of instructions to achieve a specific goal.

When developing code, we start by writing our algorithm in pseudocode.

- Pseudocode is the steps for our algorithm, in no particular language, and easy to read for people.

Then we try to implement our algorithm in Python, and test it.

We will have to make changes to our code, and potentially our algorithm, and repeat.

# Example

Let’s write an algorithm for driving a car in pseudocode.

# Exercise

Write an algorithm for making a peanut butter and jelly sandwich in pseudocode.

# Syntax vs Semantics

Syntax and semantics mean similar things in programming to what they mean in any language.

**Syntax** is the rules and regulations for writing a statement in code.

- It doesn't have anything to do with meaning.
- It refers to the grammar and structure of the language.
- A statement is syntactically correct if it follows all the rules.

**Semantics** is the meaning of a statement.

- A statement is semantically correct if it does what you intend it to do.

# Interpretation vs Compilation

Programs have to be translated into machine code so the computer can understand them.

- Some programming languages are **compiled**, which means you have to put the code through a compiler that translates it into machine code before you can run it.
- Some programming languages are **interpreted**, which means your code is automatically interpreted into machine code as you're writing it.

Python is an interpreted language.

# Binary

All the code you write is translated into **binary** so your computer can understand it.

The numbers we use are decimal (base 10), so there are 10 possible digits.

In contrast, binary is base 2, so there are 2 possible digits: 1 and 0.

A single digit in binary is called a **bit.**

# Binary Example

Think about the number 102.

Decimal (Base 10):

| 1 | 0 | 2 |
|---|---|---|
| 10^2 | 10^1 | 10^0 |
| (100) | (10) | (1) |

1*100 + 0*10 + 2*1 =
100 + 2 = 102

Binary (Base 2):

| 1 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|
| 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 |
| (64) | (32) | (16) | (8) | (4) | (2) | (1) |

1*64 + 1*32 + 0*16 + 0*8 + 1*4 + 1*2 + 0*2 =
64 + 32 + 4 + 2 = 96 + 6 = 102

# Exercise

How would you represent the number 57 in binary?

# Different Bases

You can have a number system using any base.

Binary uses base 2, decimal uses base 10.

Other common systems are hexadecimal (base 16) and octal (base 8).

They all have different uses. For example, hexadecimal is used for color codes.