Setup

Installing Python 3

First, check if you already have Python 3 installed. Open up your command line and try typing `python3 -V` (or `python -V` if that doesn't work), which will tell you which version of Python you have installed.

Make sure that your Python version is **at least 3.6** or you may not be able to use all necessary libraries (personally, I recommend 3.8). If you already have a correct version Python installed, you can skip to the next section.

If you don't, then follow the instructions below to install it:

For Windows and macOS

*Warning: For Windows development in general, I personally use the Windows Subsystem for Linux (Ubuntu), which allows you to have a mostly full-functioning Linux terminal on your Windows computer. I would **strongly** recommend doing this instead of using vanilla Windows. WSL is very easy to set up, and afterwards, you can just follow the Linux instructions, instead of Windows.*

Head over to the Python downloads page and download version 3.6 or greater for your OS. Run the installer, and make sure that you check the box labeled "Add Python to your PATH." After this, you should be able to successfully run `python3 -V` (or `python -V` if that doesn't work).

If you use a package manager like `brew` or `chocolatey`, feel free to install Python that way.

For Linux

Run the following in your command line:

```
sudo apt-get update
sudo apt-get install python3.8
```

After this, you should be able to run `python3.8 -V`.

Installing Python Dependencies

`pip` is Python's package management system, used for installing Python libraries. It should come bundled with your Python 3 installation. Check that you have it by typing into your command line `pip3 -V` (or `pip -V` if that doesn't work).

Poetry is a package manager which allows you to keep dependencies separate between different projects. If you know what you're doing, you're welcome to use a different package manager like Pipenv, but I have made the switch to Poetry. Follow the instructions below to install it:

For Windows
Open an elevated Powershell window ("Run as Administrator") and run the following command:

```
(Invoke-WebRequest -Uri https://raw.githubusercontent.com/python-poetry/poetry/master/install-poetry.py -UseBasicParsing).Content | python -
```

Verify that the installation was successful by typing `poetry --version`.

For macOS and Linux
First, make sure that running `python --version` from the command line outputs the expected version. You should be able to run that command verbatim. If only `python3 --version` works, then you can fix it by running `sudo apt install python-is-python3` (on Linux) or adding the line `alias python="python3"` to your `~/.zshrc` file (on Mac).

Next, run the following command to install Poetry:

```
curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/install-poetry.py | python -
```

Verify that the installation was successful by typing `poetry --version`.

If you get an error that the `poetry` command was not found, you probably need to add Poetry to your PATH. If you don't know how to do that, try googling and reach out if you can't get it working.

Once you've successfully installed Poetry, `cd` into the folder where you will keep your materials for this course and run `poetry init` to initialize a new virtual environment. You can skip "defining your development dependencies interactively" for now.

When that's done, verify that `poetry env info` outputs the correct version of Python. If it doesn't, then you can run `poetry env run 3.8` (or whatever Python version you want to use).

Finally, you can install the dependencies we'll need for the class with this command:

```
poetry add jupyter pycosat ortools matplotlib
```

Dependencies for PycoSAT
The `pycosat` package uses bindings to pure C code, but Windows and macOS do not come with a built-in C compiler. If `pycosat` fails to install, here are some steps you can take to fix it:

For Windows:
Try installing the Visual Studio Build Tools, which includes a C compiler. If you're still having trouble, please reach out!

For macOS:
Try installing Xcode from the app store. Then run the following command to install the Xcode command line tools, which includes a C compiler.

```
xcode-select --install
```

For Linux:
Most Linux editions come pre-installed with the `gcc` compiler. However, you may need to download the C headers for Python with the following command (replace 3.8 with whatever Python version you are using, if different):

```
sudo apt-get install libpython3.8-dev
```

Installing and Configuring VS Code

VS Code is a free code editor that provides tons of powerful features out of the box without the bloat of a full IDE. It also contains a rich marketplace of extensions. This last section is optional, but I recommend you use VS Code for all the homework in this class, including Jupyter notebooks!

Download the latest stable release from the VS Code webpage and install it. Once it is installed, open it up and type `Ctrl + Shift + P` to open the Command Palette. Search for the command called `Extensions: Install Extensions`, and then install the extension simply called `Python` (published by Microsoft).

Next, from the toolbar, select `File > Open Folder`. Then open the folder where you will keep the materials for this course. Once the folder has loaded, type `Ctrl + Shift + P` again and search for the command called `Python: Select Interpreter`. Search for the name of the current folder and select the environment you created earlier with Poetry. If your folder is called CIS 189, it should be named something like `Python 3.8.5 64-bit ('CIS189-xxx-py3.8': poetry)`.

Now you should be all good to go!

Note: If you are using the Windows Subsystem for Linux, you will need to install the WSL Remote extension for VS Code, and probably add the Linux path to your Python virtual environments under Settings > Remote [WSL] > Python: Venv Path. Feel free to reach out for help with this.

VS Code Live Share

Live Share is an awesome extension for VS Code that allows you to synchronously collaborate with others on the same file over the internet. Think Google Docs, but for code.

It's a useful tool for collaborating remotely on the final project or attending remote office hours. If you come to remote office hours for debugging help, Live Share will help the TA easily navigate your code and help you pinpoint the problem. All you have to do is send them a link!

You can install it in VS Code's extensions marketplace or from the Live Share webpage. Note that you will need to log in with your Github account in order to start sharing.