Lab 1: Getting Started & Search

> *CS410: Artificial Intelligence*
> *Shanghai Jiao Tong University, Fall 2021*
> *Due Time: 2021.09.30 23:59*

Introduction

This lab explains how to use your own computer to complete assignments for SJTU CS430 and introduces some of the basics of Python.
Once these pre-requisites are met, you are required to implement simple search algorithms to find the shortest path in a maze.

If you need any help at any time through the lab, please feel free to come in office hours or post on Canvas.

Setup

Install a terminal

The terminal is a program that allows you to interact with your computer by entering commands. No matter what operating system you use (Windows, macOS, Linux), the terminal will be an essential tool for SJTU CS430.

macOS/Linux

If you're on a Mac or are using a form of Linux (such as Ubuntu), you already have a program called Terminal or something similar on your computer. Open that up and you should be good to go.

Windows

You can just skip this step and use Windows PowerShell instead of Bash (**recommended**). PowerShell comes pre-installed on Windows and requires no extra setup. You can simply launch it from the Start menu. Simple commands like `cd` and `ls` will work (python will work after the setup), which encompass most of the Bash commands you need for this course.

Install Python 3

Python 3 is the primary programming language used in this course. Use the instructions below to install the Python 3 interpreter.

> *There are many alternative ways. If you already have Python with 3.x version installed (which can be checked with `python --version`), you can skip this part as well.*

Linux

Run `sudo apt install python3` (Ubuntu), `sudo pacman -S python3` (Arch), or the command for your distro.

macOS

Download and install Python 3 (64-bit).

You may need to right-click the download icon and select "Open". After installing, please close and open your Terminal.

Windows

If you'll be using PowerShell, open the Microsoft Store and search for "python". Install Python 3.9 by the Python Software Foundation (this should be the first result). You can then skip the rest of this section. (Important: If you later decide to reinstall Python differently, uninstall it from the Microsoft Store first.)

After installing, please close and open your terminal. Try running the python command. If this doesn't work, try `python3` or `py`. Windows may use any of these to refer to Python, so you should use the one that works on your machine.

Other

Download Python from the download page.

Install a text editor

The Python interpreter that you just installed allows you to run Python code. You will also need a text editor, where you will write Python code.

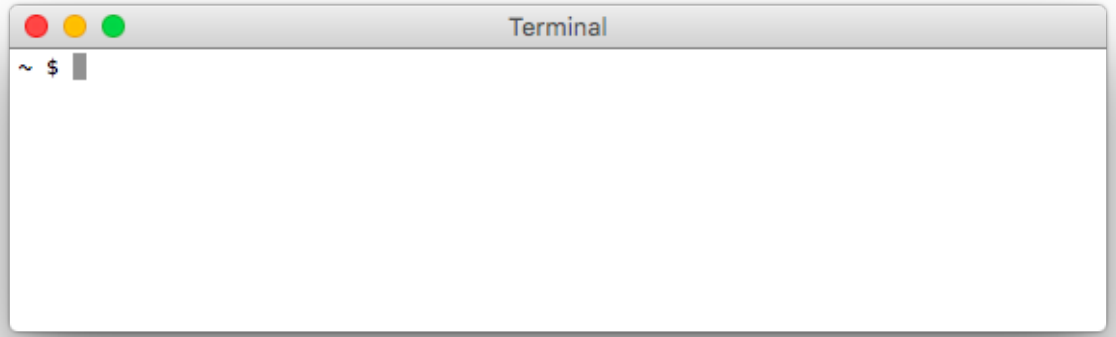
There are many editors out there, each with its own set of features. Visual Studio Code (VS Code) is the most popular choice among the staff for this course for writing Python. Some other editors that are used among staff are listed below as well. **We highly recommend for this class you use VS Code.**

> *Any other IDE (such as PyCharm and Sublime Text are welcomed, as long as you can control it.)*

Using the terminal

Let's check if everything was installed properly!

First, open a terminal window. If you're on Windows, launch *PowerShell* from the Start menu.



When you first open your terminal, you will start in the "home directory". The **home directory** is represented by the `~` symbol, which you might see at the prompt.

> *Don't worry if your terminal window doesn't look exactly the same. The important part is that the prompt shows `$` (indicating Bash), `%` (indicating zsh), or `PS` (PowerShell). If you see `C:\Users\XXXX>` but no `PS` before it, you're in the Command Prompt! Do **not** use that! Close it, and launch PowerShell instead.*

Try running `echo "$HOME"`. Verify that it displays the path to your home directory.

Python Interpreter

We can use the terminal to check if your Python 3 interpreter was installed correctly. Try the following command:

```
python
```

If the installation worked, you should see some text printed out about the interpreter followed by `>>>` on its own line. This is where you can type in Python code. Try typing some expressions you saw in lecture, or just play around to see what happens! You can type `exit()` or `Ctrl-D` to return to your command line.

> *For Python beginners, get started with official tutorial or runoob tutorial (in Chinese).*

Installing NumPy

NumPy is a powerful and widely-used library that supports scientific computing with Python.

> *In this lab, we are only leveraging the data structure `numpy.ndarray` to represent mazes. However, it is **highly recommended** to get familiar with NumPy as it is everywhere for academic research.*

We recommend to install NumPy with `pip`, which is already installed with Python:

```
pip install numpy
```

Downloading and extracting the assignment

If you haven't already, download the zip archive `lab1.zip` from Canvas, which contains all the files that you'll need for this lab. Once you've done that, let's find the downloaded file. On most computers, `lab1.zip` is probably located in a directory called Downloads in your home directory. Use the ls command to check:

```
ls ~/Downloads
```

You must expand the zip archive before you can work on the lab files. Different operating systems and different browsers have different ways of unzipping. If you don't know how, you can search online.

> *Using a terminal, you can unzip the zip file from the command line. First, `cd` into the directory that contains the zip file:*
>
> ```
> cd ~/Downloads
> ```
>
> *Now, run the unzip command (non-PowerShell) with the name of the zip file:*
>
> ```
> unzip lab1.zip
> ```
>
> *If you're using PowerShell (already in Windows 10), you can instead run:*
>
> ```
> Expand-Archive -DestinationPath . -Force lab1.zip
> ```
>
> *You might also be able to extract files without using the terminal by double clicking (or right-clicking) them in your OS's file explorer.*

Assignment

Once everything is set up, you are ready to do some exercises!

Open a terminal, and `cd` to the folder of this lab (i.e., `lab1`). The overall structure of the folder looks like this:

```
.
├── README.md
├── algorithm.py            # Modify me!
├── assets
|   └── terminal.png
├── examples
|   ├── testMaze.lay        # Modify me!
|   └── tinyMaze.lay
├── layout.py
├── main.py
├── problem.py
└── util.py                 # Modify me!
```

If everything goes right, you should be able to find the path to the destination in a small maze world by typing the following:

```
python main.py
```

The above command solves the path-finding problem with a deterministic solver (see `tiny_maze_search` in `algorithm.py`).

> *In the following exercises, please do **NOT** modify code outside the scope (specified by `""" YOUR CODE HERE """`).*

Exercise 1: Stack & Queue

Implement `Stack` and `Queue` in `util.py`.

Exercise 2: Depth first search

Implement `depth_first_search` in `algorithm.py` with your `Stack` data structure. You can evaluate your algorithm with:

```
python main.py --algo_name dfs
```

To customize your own test case, modify `examples/testMaze.lay` and evaluate with:

```
python main.py --algo_name dfs --layout_name testMaze
```

> *Follow the same format as* `examples/tinyMaze.lay` *to customize your own test cases.*

Exercise 3: Breadth first search

Implement `breadth_first_search` in `algorithm.py` with your `Queue` data structure.

Exercise 4: Improvement

Design a maze world where DFS (or BFS) fails. Provide some discussions about why the algorithm fails.

Submission

There are **only** four files you need to submit:

- `util.py` where your basic data structures are implemented;
- `algorithm.py` where your search algorithms are implemented;
- `testMaze.lay` which stores your designed maze for Exercise 4;
- `report.pdf` for your **brief** report.

Name your **zip** file containing the above four as `xxxxxxxxxxx.zip` with your student ID, and submit it on Canvas.