Chapter 0: Getting Started

How to read this manual

This handbook provides you with the specific instructions for this project. Because there is such a large number of groups, we cannot allow you to be creative here. We need you to follow some standards that make it easier for us to grade your work.

This handbook will also teach you everything you need to know to complete the project. You should remember most of it from previous courses.

Between instructions and code snippets you will encounter three types of boxes in this manual.

Critical Instructions

You absolutely have to follow these instructions.

Frequent Mistakes and Strong Recommendations

You should be aware of these.

Optional Insights

You can skip these or read them if you want to learn more.

Installing Python

The very first step is to install python and verify our installation. You only need to do this once.

On Windows

We recommend you install Python through the Microsoft Store

The steps below are an alternative if the Microsoft Store doesn't work for you.

1. Visit python.org and click on "Downloads"
2. Download the latest Python 3.x version for Windows
3. **IMPORTANT:** During installation, check the box that says 'Add Python to PATH'
4. Click 'Install Now' and wait for installation to complete
5. Verify installation by opening Command Prompt and typing: `python --version`

On Mac

1. Visit python.org and click on "Downloads"
2. Download the latest Python 3.x version for macOS

3. Double-click the downloaded `.pkg` file
4. Follow the installation wizard
5. Verify installation by opening Terminal and typing: `python3 --version`

Setting up your development environment

These are the steps you will need to repeat for each assignment (each chapter) to set up your project environment.

We strongly recommend that you use Visual Studio Code for these assignments

You can download the installer from code.visualstudio.com.

Creating the project folder structure

You can create a new folder and navigate to it by typing the following into the Windows command prompt or Mac terminal:

```
mkdir assignment_01 (1)
cd assignment_01 (2)
```

(1) `mkdir` stands for "make directory", `assignment_01` can be whatever you want the folder to be named.

(2) `cd` stands for "change directory", you have to put the same folder name here as you did in the first line.

Working in virtual environments

A virtual environment is an isolated Python environment that keeps your project's dependencies separate from other projects. This is essential because different projects may require different versions of the same package. Think of it as a container that holds all the specific tools your project needs without affecting other projects on your computer

Why use virtual environments?

- Keeps project dependencies isolated
- Prevents version conflicts between projects
- Makes it easy to share your project with others
- Allows you to replicate the exact environment on different machine

Creating a virtual environment

This command creates a new folder called 'venv' containing a complete Python environment.

```
python -m venv venv (1)
```

(1) `-m` stands for "make", the first `venv` stands for "virtual environment", the second `venv` can be whatever you want to name your environment.

On Mac, remember always use `python3` instead of `python` for all commands.

Activating the virtual environment

Before installing any packages or running your Flask app, you must activate the virtual environment:

- **on Windows:** `venv\Scripts\activate`
- **on Mac:** `source venv/bin/activate`

Confirm that the venv is active

If the activation was successful, you will now see `(venv)` at the beginning of your command line.

When you are done working in the environment, you can deactivate it by typing `deactivate`.

Installing required packages

You will need to install some packages for each assignment. You should do that with the python package manager `pip`:

```
pip install package-name (1)
```

(1) Replace `package-name` with whatever package you want to install

Only install packages while your virtual environment is active!

Verifying your package installation

You can verify that your installation worked by listing all installed packages with `pip list`. This will likely show more than just the one you installed because a package might install its dependencies as well. For example, installing Flask (which we will introduce in the next chapter) automatically installs dependencies such as Wekzeug and Jinja.

You can usually find documentation and example snippets for these packages at `https://pypi.org/`!

Version control with git

TODO

Final folder structure

Your projects must follow the folder structure below!

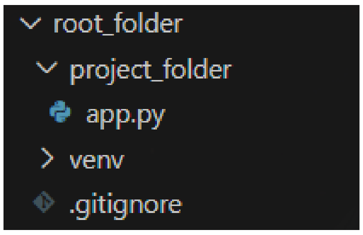
This is very important to help us review and grade your code. We will deduct points if your `venv` is in the

```
root_folder/ # (1)!
├── .gitignore # (2)!
├── venv/
└── project_folder/ # (3)!
    └── Your code goes here (e.g., 'app.py')
```

1. You are free to name this root folder whatever you want (e.g., `assignment_01` instead of `root_folder`)
2. The `.gitignore` file is only needed if you are using git to push your code to github.
3. You are free to name this project folder whatever you want(e.g., `my_app` instead of `project_folder`).

Understanding this folder structure notation

We are going to use the above notation for folder structures throughout this manual. Each line is a file (if it includes a `.`) or a folder. A folder contains all the elements that are directly below it and further indented. The above structure would look as follows in Windows:



```
root_folder
  project_folder
    app.py
  venv
  .gitignore
```