Python Environments for Coursework

Jeremy Chichportich

May 16, 2025

Abstract

Google Colab may not be suitable for this coursework due to potential file size limitations, memory constraints, or session timeouts. In particular, large datasets or multiple dependencies can cause notebooks to crash or fail to execute properly in Colab.

To ensure a stable and reliable development environment, we strongly recommend installing Python 3.10 and Jupyter Notebook locally on your own computer. This document provides step-by-step instructions to help you set up a Python environment, manage packages, and run Jupyter Notebook effectively.

Contents

1 Installing Jupyter Notebook

1.1 Step 1: Install Python

Before installing Jupyter, make sure Python is installed. You can download Python from:

`https://www.python.org/downloads/`

Alternatively, you can install Python via Anaconda, which comes with Jupyter pre-installed:

`https://www.anaconda.com/products/distribution`

1.2 Step 2: Install Jupyter via pip

If you already have Python installed, open a terminal or command prompt and run:

```
pip install notebook
```

1.3 Step 3: Launch Jupyter Notebook

To start Jupyter Notebook, run:

```
jupyter notebook
```

This will open Jupyter in your web browser.

2 Managing Environments and Packages

2.1 Why Use Environments?

Python environments help isolate dependencies between projects. This avoids version conflicts and ensures reproducibility.

2.2 Creating a Virtual Environment (using `venv`)

```
# Create a new environment named myenv
python -m venv myenv

# Activate the environment
# On Windows:
myenv\Scripts\activate
# On macOS/Linux:
source myenv/bin/activate
```

2.3 Installing Packages

Once inside the environment, install packages using pip:

```
pip install numpy pandas matplotlib
```

2.4 Exporting Requirements

To create a list of installed packages (useful for sharing or deploying):

```
pip freeze > requirements.txt
```

2.5 Installing Packages from `requirements.txt`

To install packages from a requirements file in a new environment:

```
pip install -r requirements.txt
```

3 Best Practices

- Use a virtual environment for every project.
- Keep your `requirements.txt` up to date.
- Use Jupyter inside the virtual environment to ensure correct package access.

4 Optional: Using Conda (Alternative to pip + venv)

If using Anaconda, you can manage environments with Conda:

```
conda create -n myenv python=3.10
conda activate myenv
conda install jupyter numpy pandas matplotlib
conda list --explicit > environment.yml
conda env create -f environment.yml
```

5 Final Steps Before Submission

Before submitting your coursework, please follow these final instructions carefully:

1. Make sure all your source code and notebooks are saved.

2. Activate your environment:

```
conda activate your_env
```

   Replace `your_env` with the actual name of your environment.

3. Extract the list of packages into a `requirements.txt` file:

```
pip freeze > requirements.txt
```

4. Collect all your files (e.g., Python scripts, notebooks, pdf documentation, data, and `requirements.txt`) into a single folder.

5. Create a zip archive of the folder:

```
zip -r submission.zip your_folder_name
```

6. Submit the `submission.zip` file into the dedicated Imperial platform.