Lab: Transitioning from Colab to Local Python Development

Introduction to Python

1 Overview

In this lab, we will move beyond the browser-based Google Colab environment and learn how to run Python code on your own computer.

Duration: 1 - 1.5 Hours

2 Prerequisites: Installation

Before we begin, you must have Python installed.

You should also install a code editor such as **VS Code** (recommended) or any other suitable editor (e.g., Sublime Text, Atom, PyCharm). VS Code can be downloaded from `https://code.visualstudio.com/`.

2.1 Video Tutorial

Please watch this video for a step-by-step guide on installing Python for Windows and Mac:

CLICK HERE: Python Tutorial for Beginners - Install and Setup (Corey Schafer)

2.2 Important Step for Windows Users

When installing Python on Windows, you will see a checkbox at the bottom of the installer that says:

“Add Python to PATH”

You MUST check this box. If you miss it, you won’t be able to run Python easily from the command prompt.

3 Topic 1: The Command Line Interface (CLI)

The terminal (or Command Prompt) is how you interact with your computer’s file system textually.

3.1 Basic Navigation Commands

Open your terminal (Command Prompt/PowerShell on Windows, Terminal on Mac/Linux) and try these:

Action	**Windows (cmd)**	**Mac / Linux**	**Description**
Check Directory	`cd` or `chdir`	`pwd`	Print Working Directory
List Files	`dir`	`ls`	List files in current folder
Change Folder	`cd foldername`	`cd foldername`	Enter a folder
Go Back	`cd ..`	`cd ..`	Go up one folder level
Make Folder	`mkdir name`	`mkdir name`	Create a new directory
Clear Screen	`cls`	`clear`	Clears text from screen

4 Topic 2: Running Python Scripts Locally

In Colab, you press ”Play”. Locally, you tell the Python interpreter to read a text file.

1. Create a folder named `lab_demo`.
2. Inside, create a file named `hello.py` with the content: `print("Hello World")`.
3. Open your terminal, navigate to the folder using `cd`.
4. Run the command:
   - **Windows:** `python hello.py`
   - **Mac/Linux:** `python3 hello.py` (sometimes just `python`)

5 Topic 3: Imports and Modular Code

Real software is split across multiple files.

5.1 The `import` system

If you have `helper.py` and `main.py` in the same folder:

helper.py:

```python
def greet(name):
    return f"Hello, {name}"
```

main.py:

```python
from helper import greet

print(greet("Student"))
```

Run `main.py` to see how imports work.

6 Topic 4: Virtual Environments (Venv)

Virtual environments isolate your project. One project might need `pandas` version 1.0, another needs version 2.0. Venv keeps them separate.

6.1 Creating and Activating

Run these commands in your project folder:

1. Create the environment (named 'venv'):

- Win/Mac/Linux: `python -m venv venv` (or `python3 ...`)

2. Activate the environment:

- **Windows (Command Prompt):** `venv\Scripts\activate`
- **Windows (PowerShell):** `venv\Scripts\Activate.ps1`
- **Mac/Linux:** `source venv/bin/activate`

When active, your terminal prompt will show `(venv)`.

7 Topic 5: Installing Packages (Pip)

`pip` is the package installer for Python.

- **Install a package:** `pip install requests`
- **List installed packages:** `pip list`
- **Save requirements:** `pip freeze > requirements.txt`
- **Install from file:** `pip install -r requirements.txt`

8 Lab Activity

Download the provided `lab_demo` folder. It contains:

- `main.py`: The entry point script.
- `my_module.py`: A helper module.
- `requirements.txt`: List of dependencies.

Task:

1. Navigate to `lab_demo` in your terminal.
2. Create a virtual environment: `python -m venv venv`.
3. Activate it.
4. Install dependencies: `pip install -r requirements.txt`.
5. Run the code: `python main.py "Your Name"`.