Summer of Code

Artificial Intelligence

(Machine Learning & Deep Learning)

Instructor
Wajahat Ullah
- *Research Assistant* (DIP Lab)

Duration
03 Months
(September – November)

Day 02 – Linux Shell Scripting Fundamentals

Objectives:

- Introduction to Computer?
- Introduction to Programming
- Introduction to OS
- What is an Interface?
- Core Commands
- Setting Up Python and VS Code

What is a Computer?

What is a computer?
A computer is an electronic device that processes data according to a set of instructions to perform tasks automatically and efficiently.

What is programming or coding?
The process of writing instructions that a computer can understand and execute.

Why do we need programming?
To solve problems efficiently, to create software, websites, apps, and intelligent systems.

How do we program?
We use programming languages to write instructions that computers can follow. (Python, Java, C++, JavaScript, C#)

How does a Computer understand the instructions?

- **The Central Processing Unit**
  It's the brain of the computer, made up of millions (or even billions) of tiny components called transistors.

- **What is a transistor?**
  Transistors act like tiny electronic switches or logic gates. Each gate can be ON or OFF, which is represented as 1 or 0, forming the binary language, also called machine language.

- **Machine language and low-level languages**
  Writing code directly in binary (0s and 1s) is extremely difficult. This kind of programming is called low-level programming. Other examples of low-level languages include: Assembly language (uses mnemonics like MOV, ADD, JMP)

How does a Computer understand the instructions?

- **High-Level Languages**
  Programming languages which are easier and more human-readable (e.g. Python)
- **How Does the Computer Understand High-Level Code?**
  High-level code must be translated into machine code so the processor can execute it.
- **Translators**
  A Compiler translates the entire code at once into machine language (e.g., C, C++). An Interpreter translates and executes code line-by-line (e.g., Python, JavaScript).

How does a Computer understand the instructions?

Aspect	Compiled Language	Interpreted Language
Execution	Entire program is translated into machine code before execution	Code is translated line-by-line during execution
Speed	Generally faster, as it runs machine code directly	Slower, due to line-by-line interpretation
Error Handling	Errors are shown after full compilation	Errors are shown during execution
Output	Produces a separate executable file (e.g., .exe)	No separate executable: runs via interpreter
Examples	C, C++	Python, JavaScript

Introduction to Operating Systems?

01

What is an OS?
The software that manages hardware and runs your apps.

02

Linux vs Windows in AI
Linux is open-source and free. Powers most cloud servers (AWS, Google Colab). Windows is user-friendly, great for local dev with WSL (Windows Subsystem for Linux) for Linux-like experience.

03

You can learn both!
Code locally on Windows, deploy on Linux servers

OS Comparisons for AI Work

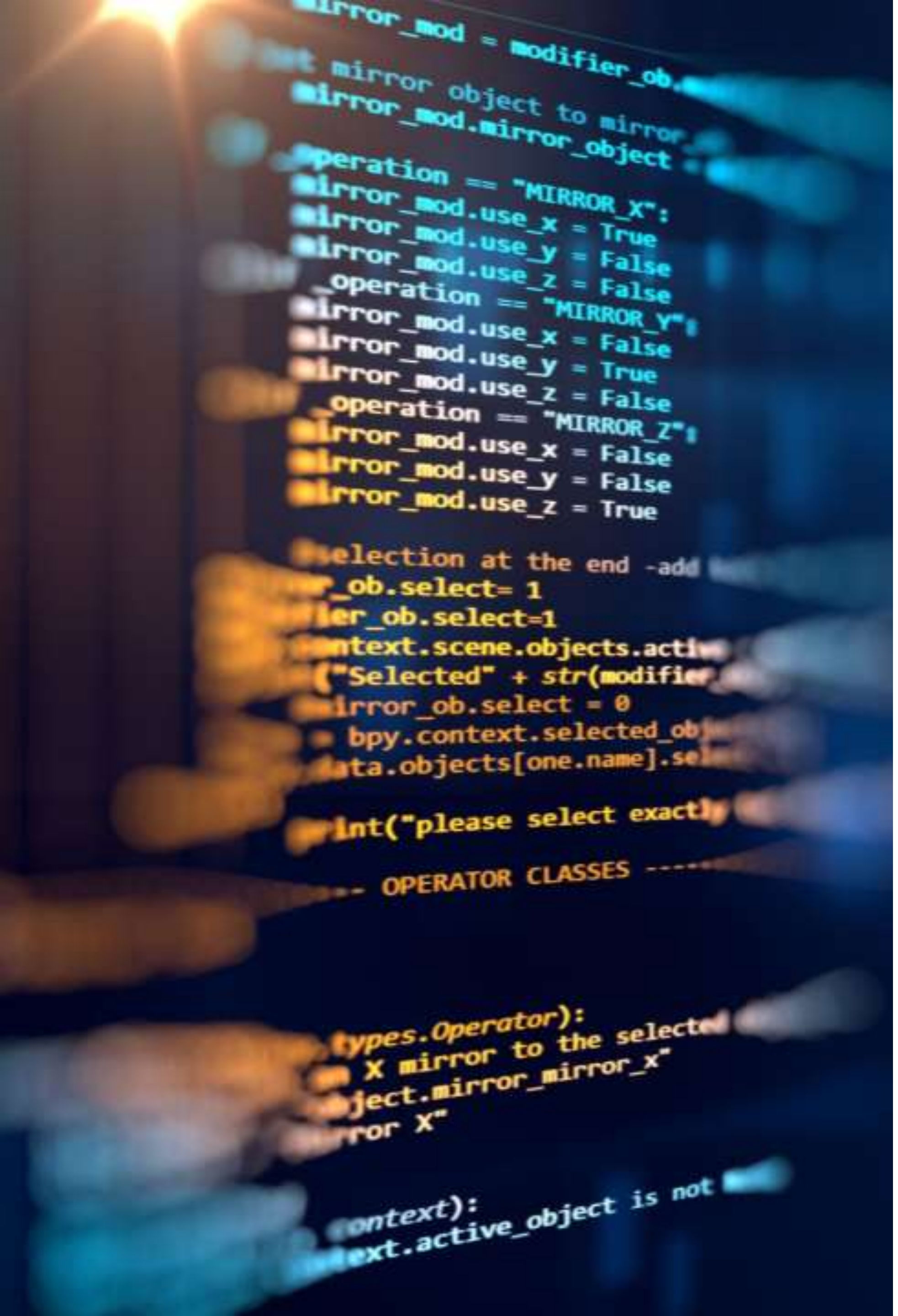
Aspect	Linux (Ubuntu/Debian)	Windows
Package Management	apt/dpkg – e.g., apt install torch	Chocolatey or winget – e.g., winget install python
Stability	High for servers; fewer crashes in long ML trains	Good for desktops; WSL bridges gap
Cost	Free	Licensed (but free for personal)
AI Tools	Native CUDA support	Requires extra setup for GPUs
Learning Curve	Steeper CLI focus	Gentler with GUIs

Interface

What is an interface?
An interface in OS is the point of interaction between the user and the system, allowing commands and feedback

GUI (Graphical User Interface)
Visual, mouse-driven comprising Icons, windows, menus for ease

Text-Based (CLI - Command Line Interface)
Type commands for precise control e.g., in Terminal/CMD.



Terminal/CMD and Shell

- **What is Terminal/CMD?**
  A text-based interface to interact with your OS – like a command center for your computer.
- **What is a Shell?**
  A program that provides direct access to the system on which it runs.
- **Prompt and Input**
  Prompt shows the current directory and user

Navigation Commands

- `pwd` **(Print Working Directory)**
  Shows current path. Linux/Windows same.
- `cd` **(Change Directory)**
  `cd /path/to/dir` or `cd ..` (up). Windows uses \ for paths.
- `ls` **(List)**
  Lists files/dirs. `ls -l` for details. Windows: `dir`.Example: `cd ~/ml_project`; ls – Navigate to your AI repo

File Creation and Management

- `touch`
  Create empty file: `touch data.csv`

- `mkdir`
  Make Directory: `mkdir models`

- `rm`
  Delete a file or directory: `rm file.txt` or `rm -r dir` (recursive). Caution: No recycle bin

- `rmdir`
  Remove empty dir: `rmdir old_dir`.

Copy & Move

- `cp` (Copy): `cp source dest` or `cp -r dir dest`.
- `mv` (Move/Rename): `mv source dest` `mv old new`.

Viewing Files

- **`cat`**
  Concatenate: `cat file.txt` – Dumps content.
- **`more`/`less`**
  Paginated view: `less huge_log.txt` (q to quit)
- **`head`**
  `head -n 10 data.csv` – First lines

Redirection

- **Redirection**
  command > file (output to file) or >> (append). AI: `python train.py > log.txt`
- **Piping (|)**
  Chain commands: `ls | grep .py` – Find Python files.

Superuser & Permissions

- **`sudo`**
  Superuser Do: `sudo apt install package` – Run as admin

- **`chmod`**
  Change Mode: Linux/Unix command used to modify file or directory permissions, controlling who can read, write, or execute them
  `chmod [options] mode file`

- **Permission Basics**
  Permissions apply to three groups: **Owner (u)**: File creator. **Group (g)**: Users in the same group. **Others (o)**: Everyone else. Each group can have:
  - r (read, 4): View file contents.
  - w (write, 2): Edit file.
  - x (execute, 1): Run file (e.g., scripts) or enter directory.

Package Management

- **`dpkg`**
  Low-level: `dpkg -i package.deb` – Install .deb files
- **`apt` (Ubuntu/Debian)**
  `sudo apt update` (refresh list), `sudo apt upgrade` (update packages)
- **`winget` (Windows)**
  `winget install Anaconda3`

Setting Up the Python Environment

- **Install Python**
  You can install Python by downloading it from python.org.
- **Install Visual Studio Code (VS Code)**
  Download and install VS Code from the official website.
- **Install some extensions**
  Python
  Code Runner
  Jupyter

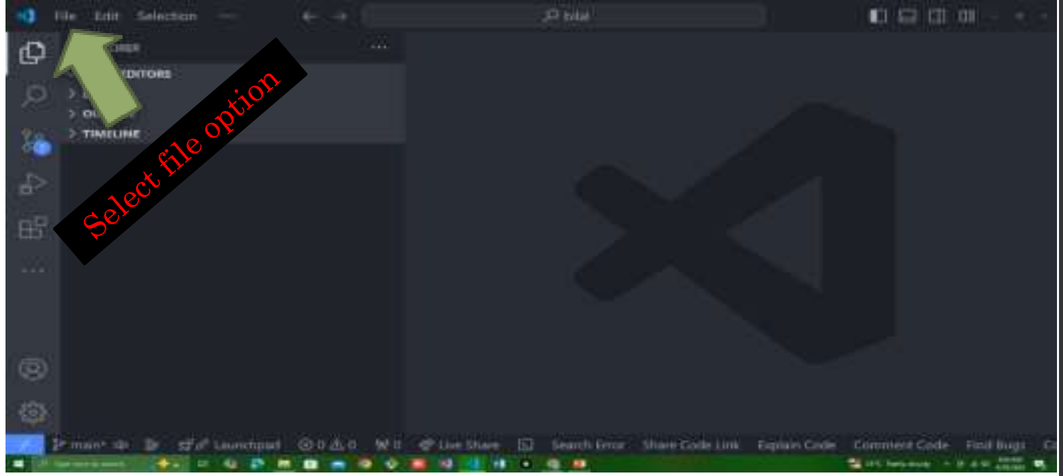
Working with Files, Folders, Kernel, and Terminal in VS Code

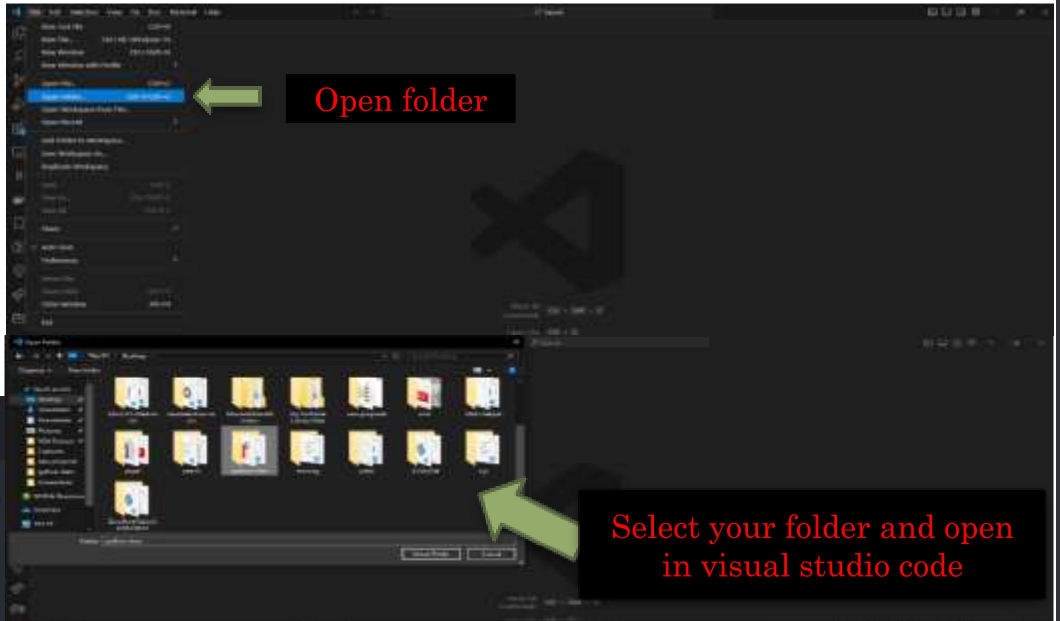
Select a File or Folder in VS Code

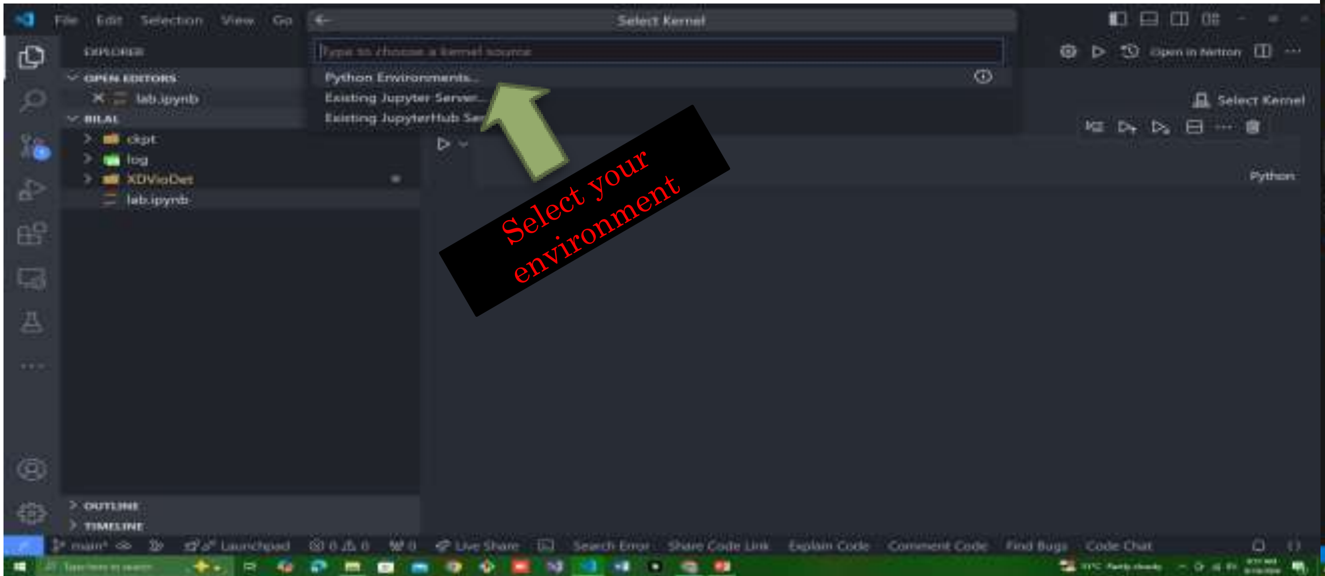
- Open VS Code.
- Click on File in the top menu.
- Select Open Folder to choose your project directory.
- Alternatively, click the Explorer icon on the left sidebar and select a file.

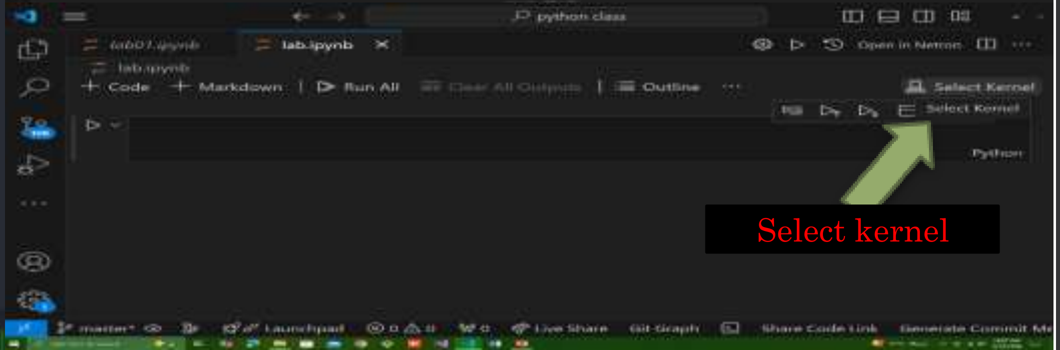
Select Jupyter Kernel

- After opening a Jupyter Notebook file (.ipynb):
- Click on the kernel name (top-right of the notebook editor).
- Choose the Python environment or kernel you want to use for running the notebook.









Happy Coding