SUNG KYUN KWAN UNIVERSITY
SKK GSB

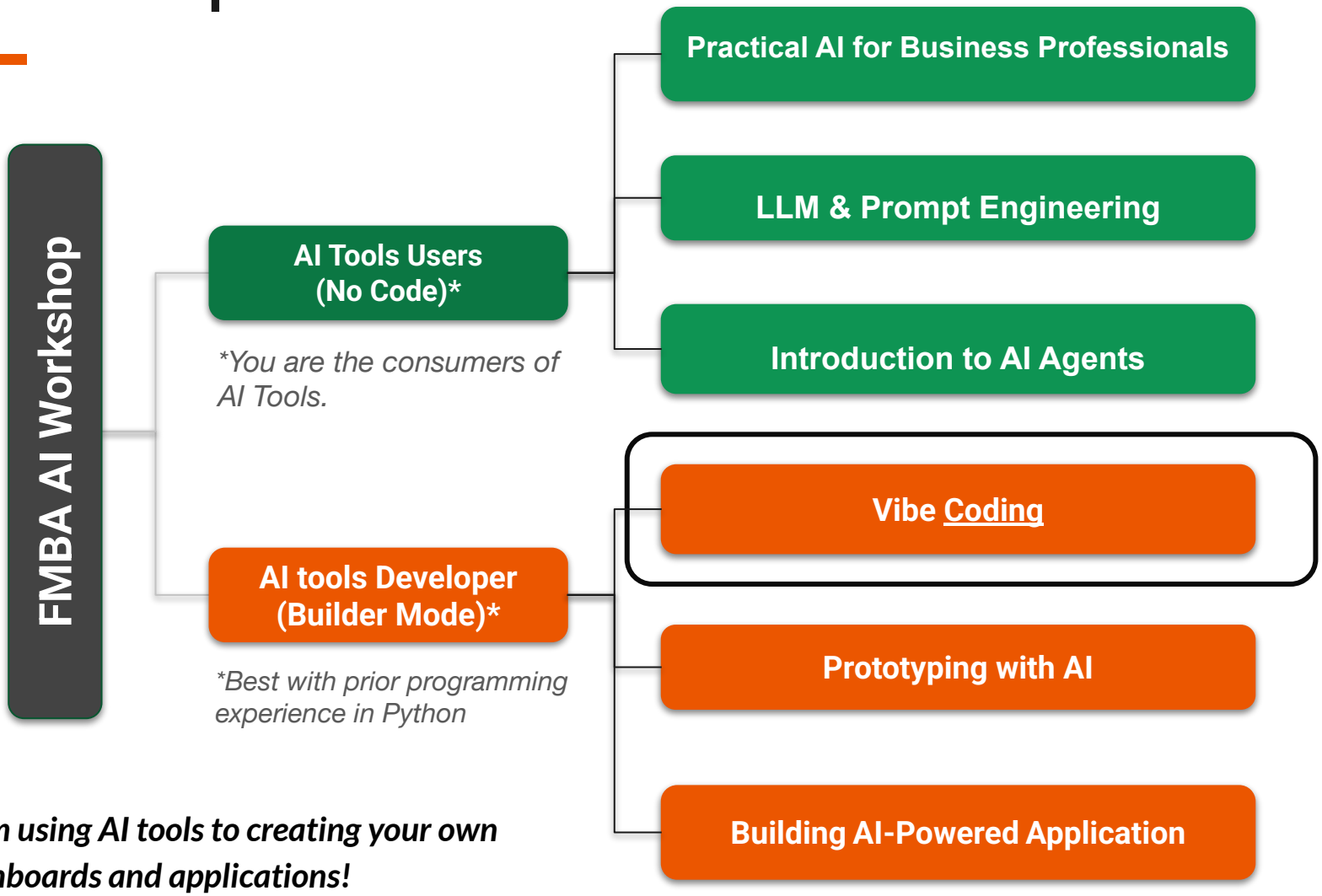
FMBA AI Workshop 4

Vibe Coding (Jan 16)
Prototyping with AI (Jan 23)
Building AI Powered Applications (Jan 30)

Dr. Yuan Tian

Please login to your Github account before the class
Workshop Information: https://github.com/tianyuan09

AI Workshop Overview



From using AI tools to creating your own dashboards and applications!

What to Expect in a Workshop

Not Traditional Lectures

- **Less** talking from me, **more** doing from you.
- Hands-on learning — **less theory, more practice.**
- **Learn by experimenting**, not just listening.

Learning by Doing

- Use real AI tools to create outputs in every session.
- Practice, iterate, and learn from feedback.

Collaborative & Interactive

- You are welcome to work in pairs or small groups.
- Share ideas, test prompts, and build together.

This Workshop is *Not About...*

This workshop is NOT about deep technical knowledge; therefore, we will NOT cover the following:

- Neural networks or how they work internally.
- How to build, train or fine-tune deep learning models.
- Building predictive models using deep learning models
- Retrieval-Augmented Generation or other advanced large language models principles.
- Deep learning models architectures, frameworks, or theory.

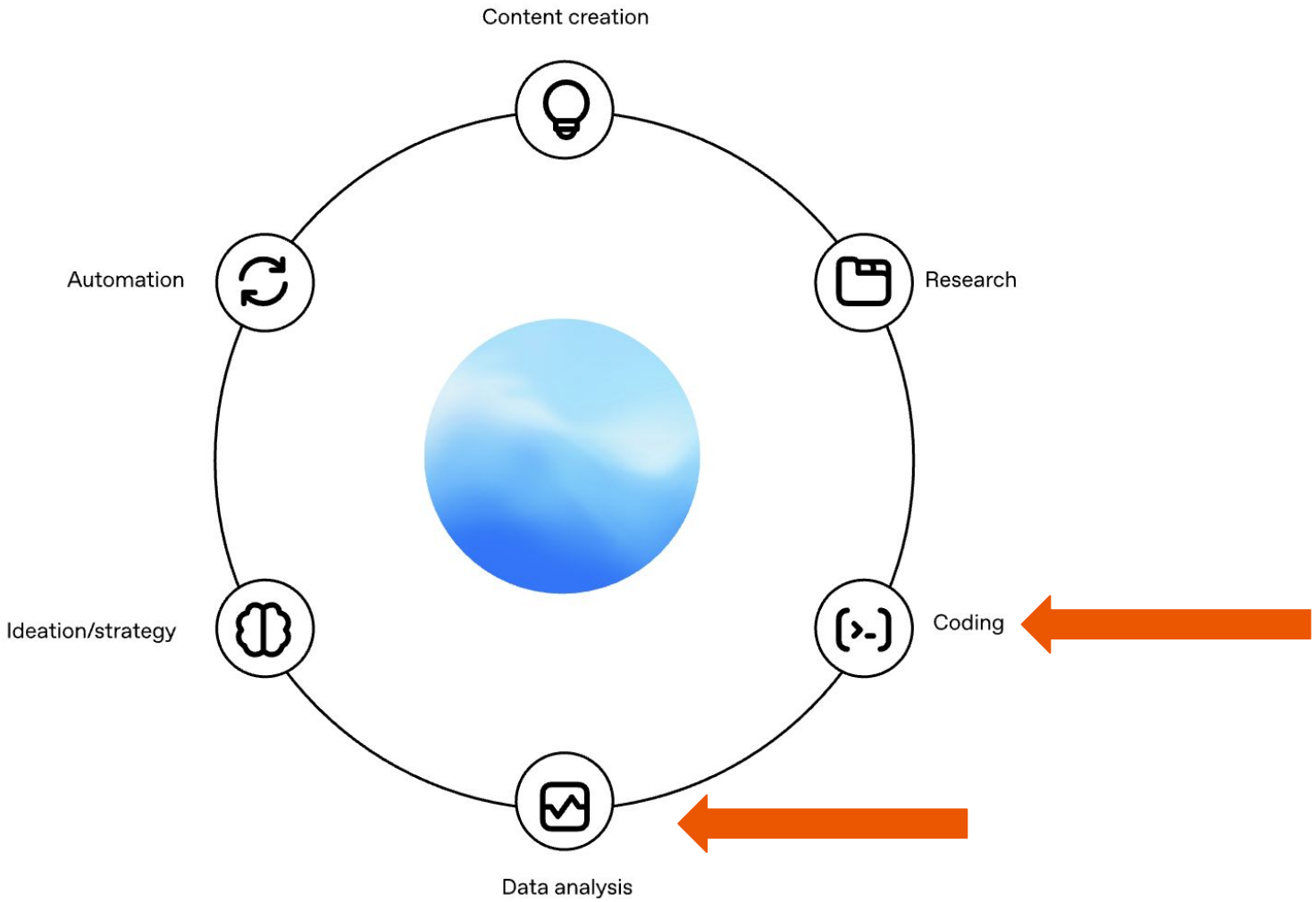
Workshop Overview

- This workshop focuses on **real-world applications** through cutting-edge tools like VS Code, Git, GitHub Copilot, and no-code/low-code AI agent platforms
- You will gain practical skills in navigating **a professional development environment** (VS Code, extensions, virtual environments, bash).
- Style: **learning by doing with AI**:
  - Interactive, practice-heavy, collaborative

Disclaimer

I have no conflict of interest in any the AI tools mentioned in this workshop.

Six Major AI Use Cases



AI Tools for Text, Voice and Image Generation

AI tools for text, voice, and image generation have become widespread. We covered many of them in the first 3 AI workshops.

Text Generation:

- ChatGPT
- Claude
- Microsoft Copilot
- Google Gemini

Image Generation:

- DALL-E
- Midjourney

Voice & Synthesis:

- NotebookLM – synthesizes and transforms information into podcasts and voice summaries.

AI Tools for Coding (Not Text Generation Chatbots)

- **GitHub Copilot** – AI pair programmer integrated into IDEs (e.g., VS Code) for real-time code suggestions and completion
- **Claude Code** – Coding-focused AI that analyzes and edits code across entire projects
- **Gemini Code / Gemini CLI** – Google's AI coding agent designed to work directly in the terminal
- **OpenAI Codex** – AI engine that writes, runs, and tests code in controlled development environments
- **Cursor** – AI-powered code editor optimized for UI and end-to-end code generation workflows

AI Tools for Coding (Not Text Generation Chatbots)

- Many coding AI tools are accessed through **CLIs (command-line interfaces)** and **APIs (application programming interfaces)**, which **require basic coding and terminal skills** to use.

Coding in Professional Software Engineering

Programming languages have used to build all kinds of digital products and services around us: mobile apps, web platforms, data analytics tools, custom dashboards that solve problems in finance, healthcare, logistics, and entertainment.

Coding (or programming) remains a specialized profession in 2026. The vast majority of code is still authored, managed, and secured by **Professional Software Engineers who were trained in Computer Science.**

AI can generate code snippets, but it cannot work independently.

AI Adoption in Coding

The 2025 Stack Overflow Developer Survey collected responses from over 49,000 **professional developers** across 177 countries.

Metric	Industry Status (2026)
Daily Usage	84% of professional developers use or plan to AI coding assistants (e.g., GitHub Copilot, Cursor), up from 76% in 2024.
Volume of Code	In Google/Alphabet's Q3 2024 earnings call, Sundar Pichai said that "*more than a quarter of all **new** code at Google is generated by AI, then reviewed and accepted by engineers.*"
Productivity Gain	Engineers report a ~55% increase in task completion such as unit testing using **GitHub Copilot**

AI Adoption in Coding

The 2025 Stack Overflow Developer Survey collected responses from over 49,000 **professional developers** across 177 countries.

https://survey.stackoverflow.co/2025/ai/

- 46% developers actively distrust the accuracy of AI tools (up from 33% last year).
- Only a fraction (3%) report "highly trusting" the output.

What does it mean for the field?

- AI-generated code as a starting point, not a final product, so **human review remains essential** before code is trusted in production.
- Effective code review still **requires programming expertise.**
- It **cannot** be done by someone **without programming skills**.

What is “Vibe Coding”?

A "slang & trending" term invented in February 2025

Vibe Coding

"**Vibe coding**" is an AI-assisted **software development** technique introduced by Andrej Karpathy (a co-founder of OpenAI) in February 2025. The term quickly gained recognition among software developers, AI practitioners, and tech media.

- **Traditional Coding:** A human types: `print("Hello World")`.
- **AI-Assisted Coding:** A human types half a line, and the AI suggests the rest (auto completion).
- **Vibe Coding**: It is a **software creation approach** where **developers** describe tasks to a **large language model**, rely on it to generate code, and **evaluate results through execution** rather than reviewing or editing the code.

https://en.wikipedia.org/wiki/Vibe_coding
https://cloud.google.com/discover/what-is-vibe-coding?utm_source=chatgpt.com&hl=en

Vibe Coding: An Example

- A human says, "***I want an web app that tracks our Q3 sales goals and sends a Slack alert if we drop below 80%.***"
- *The **AI then autonomously writes the code**, sets up the server, connects the database, and fixes its own bugs.*
- *The human only need to **know how to run the code** to test whether it is working as expected.*
- *If not, the human keeps asking for revisions in **natural language** **instead of inspecting or modifying the code itself**.*

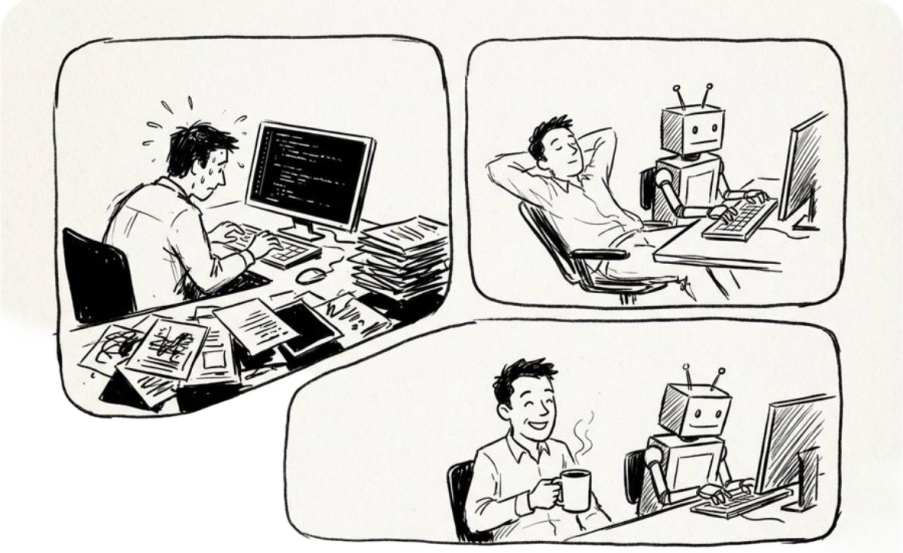
As a vibe coder, you don't read or edit code; you only run the code to check the outcome.

https://en.wikipedia.org/wiki/Vibe_coding

The Advocates about Vibe Coding

Advocates

- Enables rapid prototyping and lowers the barrier for non-experts
- Shifts focus from writing code to validating outcomes



Who Can Benefit From Vibe Coding?

- **Entrepreneurs & startup founders:** Quickly prototype app ideas and build demos for investors without hiring a full engineering team.
  - Quickly turn business ideas into working apps or prototypes
  - Build Minimum Viable Product for startups without deep engineering expertise
  - Focus on validating markets and user value, not code details
- **Product managers & business leaders:** Create working versions of product concepts fast, allowing more productive feasibility discussions with engineering teams.

Why You Still Need Coding Literacy for Vibe Coding?

Vibe coding lowers the barrier, but it does not remove the need for technical literacy:

- **You must run and test the code** — real apps don't run inside chat; they run in **terminals, servers, or browsers**
- **You need to validate outcomes** — knowing what "working" looks like requires understanding logs, errors, and outputs
- **AI-generated code can fail silently** — without basic coding knowledge, you won't know whether results are correct or misleading
- **Debugging still happens** — even if AI writes the code, humans must recognize when something breaks and why
- **Security and data risks remain** — you need enough literacy to spot unsafe behaviors (e.g., exposed keys, wrong data access)

Why You Still Need Coding Literacy for Vibe Coding

Vibe coding lowers the barrier, but it does not remove the need for technical literacy.

Vibe coding reduces how much code you write, not how much responsibility you have.

Core (or Minimal) Coding Skills for Effective Vibe Coding

- **Run code in professional environments**
  - Use VS Code, GitHub Codespaces, and cloud runtimes
  - Understand how to start, stop, and rerun applications
- **Basic operating system & terminal literacy**
  - Navigate files and folders (Linux basics)
  - Use terminal commands
- **Understand core code artifacts**
  - What is a python script file?
  - Know the difference between .py files and Jupyter notebooks
- **Clear product thinking**
  - Have a concrete idea or prototype in mind before prompting
  - Break ideas into steps the AI can execute
- **Version control with Git (your regret medicine)**
  - Save working versions; roll back when the AI makes mistakes

Today's Session

- **Run code in professional environments (Github)**
  - Use VS Code, GitHub Codespaces, and cloud runtimes
  - Understand how to start, stop, and rerun applications
- **Basic operating system & terminal literacy**
  - Navigate files and folders (Linux basics)
  - Use terminal commands
- **Understand core code artifacts in Python**
  - What is a python script file?
  - Know the difference between .py files and Jupyter notebooks

Lab 1.Github Codespaces

https://tianyuan09.github.io/fmbaaiworkshop26spring/chapter2.html

Github Platform for Version Control and Sharing

- **Microsoft acquired GitHub.**
- **Code hosting platform for version control and collaboration.**
  - https://github.com/pandas-dev/pandas
  - A platform where developers want to show off their work, contribute to the open-source community, or collaborate with others.
  - **Don't put confidential data or materials on Github.** The free version will make your projects (or repository) public.
- **Codespaces -** allow you to use VS Code in a cloud environment.
- **Copilot -** an extension for VS Code to use AI for programming.

Lab 2.Command Lines and Terminal

https://tianyuan09.github.io/fmbaaiworkshop26spring/chapter2.html

Why Command-Line Skills Matter?

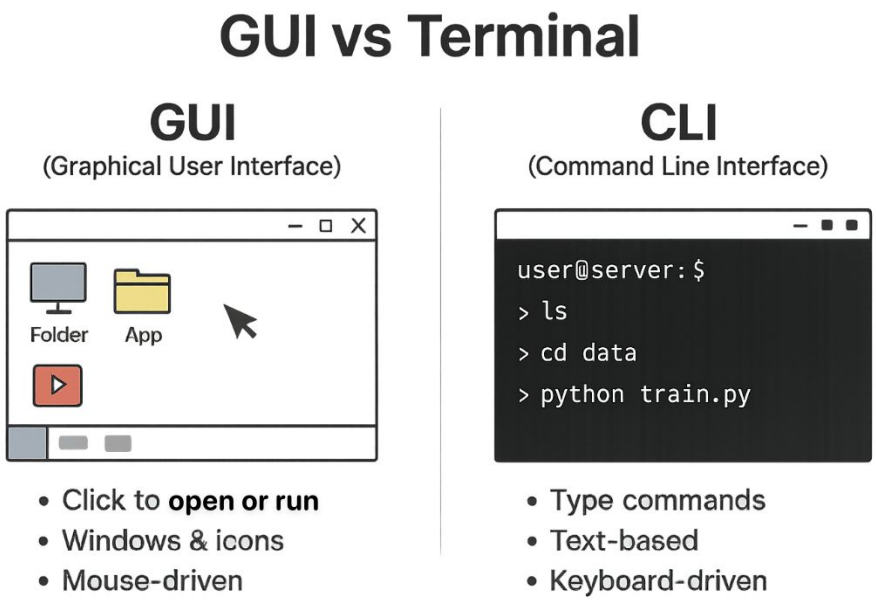
- Command-line proficiency is **a core foundation** for professional data science and engineering workflows.

- **Context:** Data science projects often run on servers or cloud platforms (e.g., AWS EC2, Azure VMs, Google Cloud Compute) which requires command lines skills to work with.

Operating Systems Overview

Environment	Dominant OS	Market Share / Usage Context
Personal Computers	Windows (70–75%), macOS (15–20%)	The laptops that you are using are personal computers.
Enterprise / Cloud	**Linux and Unix-like systems**	The backbone in web servers, cloud computing, AI/ML, and supercomputers.
Web Servers	Linux or Unix-like systems	77–88% of public web servers run on these systems.
Cloud Workloads	Linux-based OS	Powers **49.2%** of all global cloud workloads (as of mid-2025). Cloud providers (AWS, GCP, Azure) primarily offer Linux-based instances for AI services.
Supercomputers	**Linux (100% Monopoly)**	Since 2017, **100%** of the world's top 500 supercomputers run on Linux.
AI and ML Workloads	Linux infrastructure	**87.8%** of machine learning workloads run on Linux infrastructure (mid-2025)

Terminal, CLI, GUI



CLI (Command Line Interface)

- A text-based interface where users type commands instead of clicking. Linux servers are often managed *entirely* through the CLI.

GUI (Graphical User Interface)

- The visual part of the computer (windows, buttons, menus) that is operated with a mouse. GUIs are less efficient for automation or remote access.

Terminal in Different Operating Systems

Operating System	Primary Shell/Terminal	Details
Windows	**Command Prompt (cmd)**, **PowerShell**	Windows includes these terminal apps. It can also run **Bash** via the Windows Subsystem for Linux (WSL).
macOS	**Zsh in Terminal app**	**macOS is Unix-based**. **Zsh** is the default shell in macOS (based on Unix). **Bash** and **Zsh** are both terminals that interpret commands and **work almost the same way.**
Linux/Unix (also in cloud environments)	**Bash**	**Bash** is the default shell on most Linux systems. **Google Colab** and **GitHub Codespaces** also run on Linux systems, using the **Bash shell.**

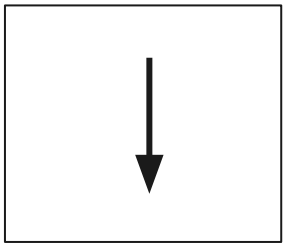
Demo of Powershell

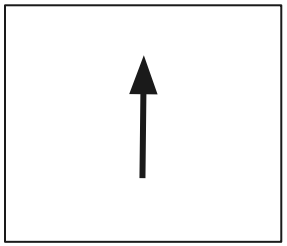
Essential Bash Commands in MacOS/Linux

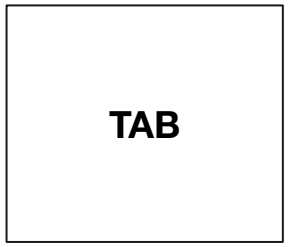
Command	Purpose	Example
`pwd`	**P**rint **W**orking/current **D**irectory/location.	`pwd`
`ls`	**L**ist files and folders in a directory.	`ls -la` (List all files in long format)
`cd`	**C**hange **D**irectory.	`cd sample_data`, `cd ..` (Parent folder), `cd ~` (Home folder)
`mkdir`	**M**ake a new **dir**ectory (folder).	`mkdir project scripts`
`touch`	Create an empty file.	`touch notes.txt`
`cp`	**C**o**p**y a file.	`cp notes.txt backup/`
`mv`	**M**o**v**e or **R**ename a file.	`mv old.txt new.txt`
`rm`	**R**e**m**ove a file.	`rm old.txt`, `rm -r backup` (Remove folder and contents)
`cat`	View contents of a file.	`cat todo.txt`
`curl`	Download a file from the internet.	`curl -O <URL>`

Bash shortcuts

Using the **Up Arrow** or **Down Arrow** in the bash to retrieve the last bash command that you have run.







Lab 3. Python Script Files Basics

Run .py scripts with Terminal

https://tianyuan09.github.io/fmbaaiworkshop26spring/chapter2.html

Python File Extensions

.ipynb file

- An IPYNB file is a notebook document created by **Jupyter Notebook (plain text files formatted using JSON)**. It offers an interactive computational environment for **data analyses** using Python.
  - Code cells in Python.
  - Text cells in Markdown.
- IPYNB notebooks are plain text files formatted using JSON.

.py file

- A PY file is a program file or script written in Python.
- Often you need the **terminal** to execute the script file.