GitHub and Python Refresher - I

Lecture 2 – Wednesday August 27, 2025

https://github.com/Dr-AlaaKhamis/ISE518/

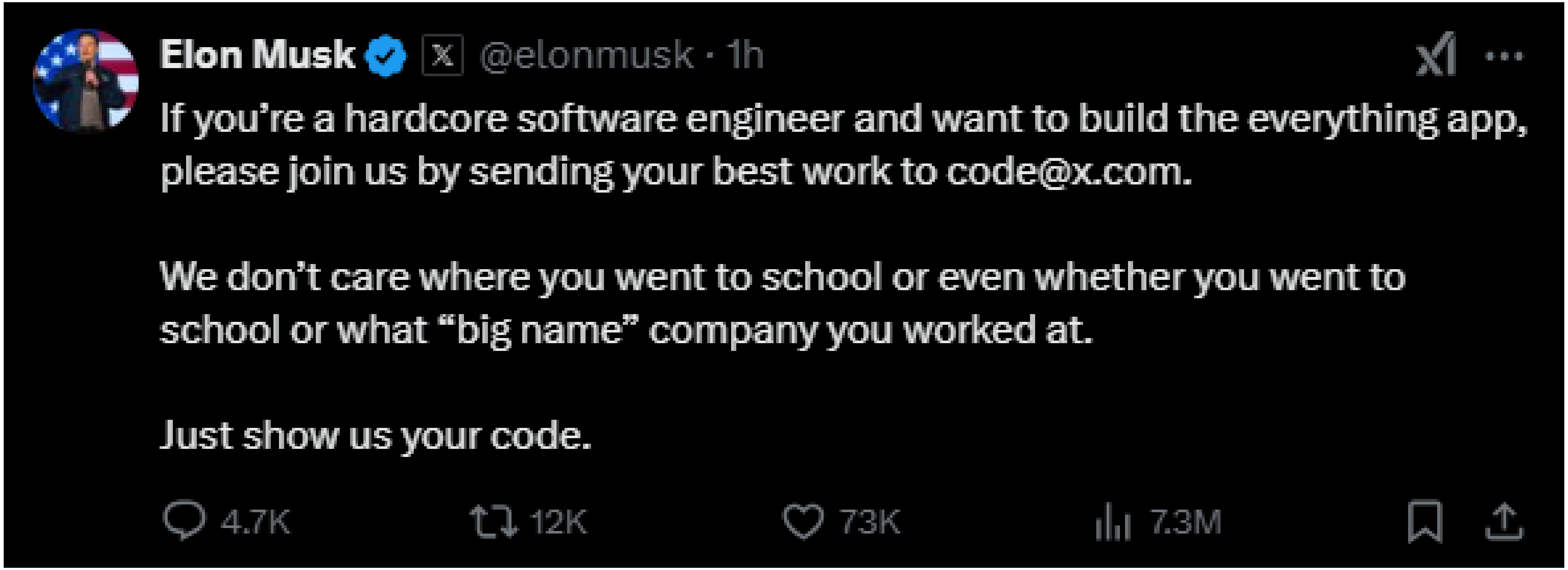
Outline

- Why Coding?
- Introduction to GitHub
- Introduction Python
- Course GitHub Demo

Outline

- **Why Coding?**
- Introduction to GitHub
- Introduction Python
- Course GitHub Demo

Why Coding?



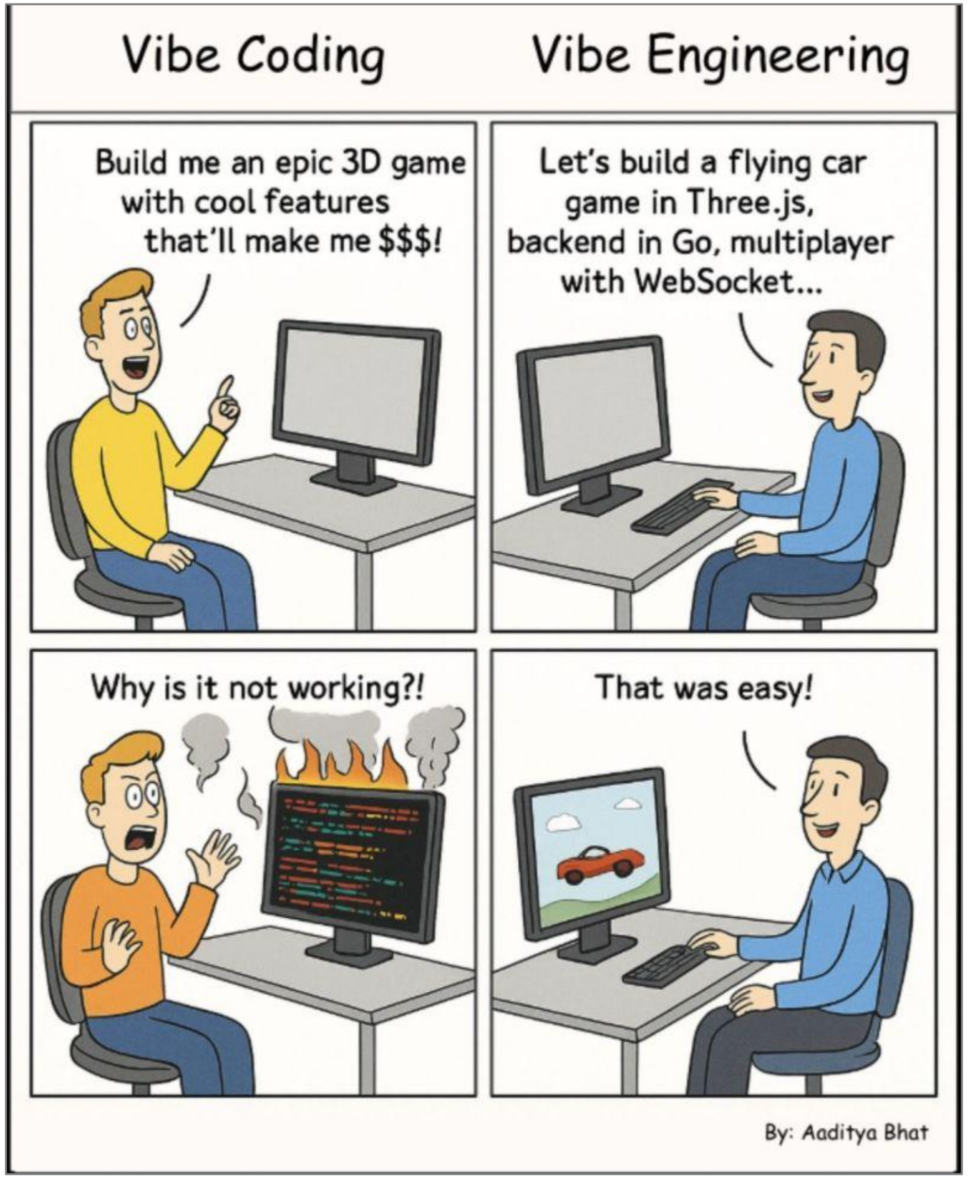
Elon Musk @elonmusk · 1h

If you're a hardcore software engineer and want to build the everything app, please join us by sending your best work to code@x.com.

We don't care where you went to school or even whether you went to school or what "big name" company you worked at.

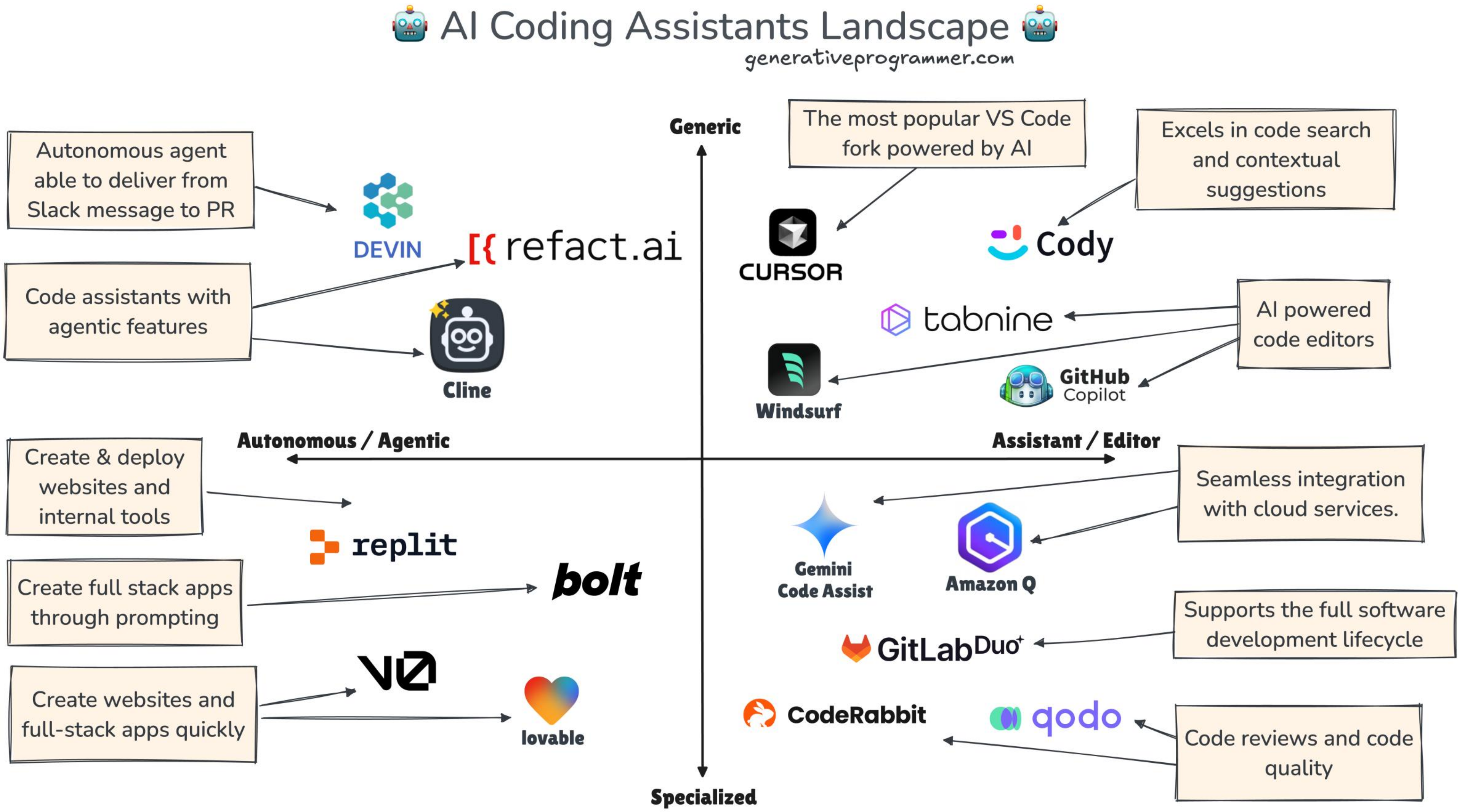
Just show us your code.

Why Coding?


Vibe Coding
Vibe Engineering
Build me an epic 3D game with cool features that'll make me $$$!
Let's build a flying car game in Three.js, backend in Go, multiplayer with WebSocket...
Why is it not working?!
That was easy!
By: Aaditya Bhat

https://www.linkedin.com/posts/lukastimm_i-believe-80-of-code-in-automotive-will-activity-7344596971258605568-c4YJ?utm_source=share&utm_medium=member_ios&rcm=ACoAAAC1tYYBWj2HpBUW6mqtmU0XgcdpiHM5TX8

Why Coding?



🤖 AI Coding Assistants Landscape 🤖

generativeprogrammer.com

Autonomous agent able to deliver from Slack message to PR

Code assistants with agentic features

Create & deploy websites and internal tools

Create full stack apps through prompting

Create websites and full-stack apps quickly

The most popular VS Code fork powered by AI

Excels in code search and contextual suggestions

AI powered code editors

Seamless integration with cloud services.

Supports the full software development lifecycle

Code reviews and code quality

Why Coding?

Automation

Eliminates repetitive manual tasks, improving efficiency.

Problem-Solving

Provides flexible tools to prototype and test engineering solutions quickly.

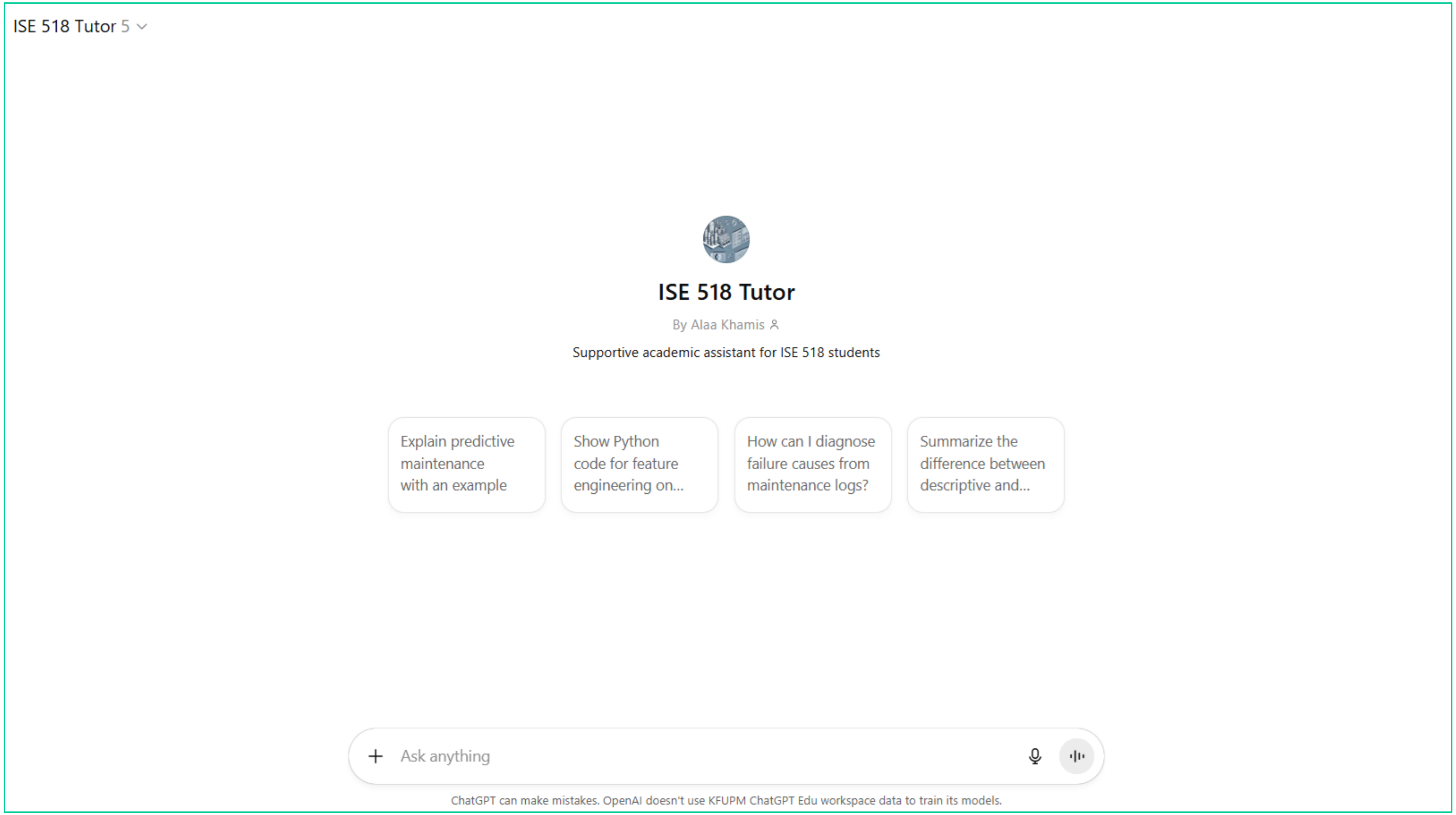
Data Analysis

Enables engineers to process, interpret, and visualize large datasets

Industry 4.0 & AI/ML

Core for smart systems, predictive maintenance, and digital twins.

Why Coding?



ISE 518 Tutor 5

ISE 518 Tutor

By Alaa Khamis

Supportive academic assistant for ISE 518 students

- Explain predictive maintenance with an example
- Show Python code for feature engineering on...
- How can I diagnose failure causes from maintenance logs?
- Summarize the difference between descriptive and...

ChatGPT can make mistakes. OpenAI doesn't use KFUPM ChatGPT Edu workspace data to train its models.

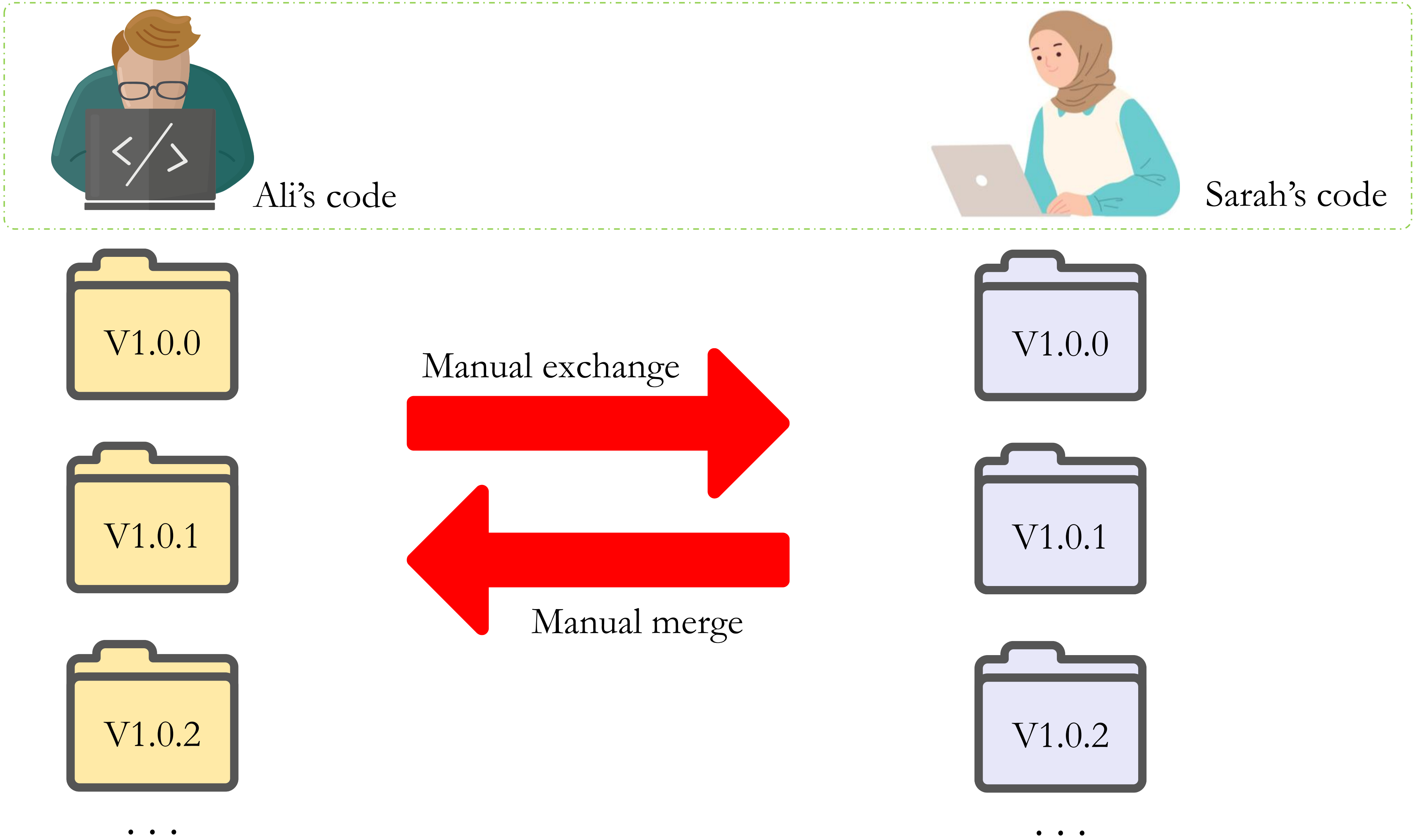
https://chatgpt.com/g/g-689c66b7a7c481918482219bff3fc80c-ise-518-tutor

Outline

- Why Coding?
- **Introduction to GitHub**
- Introduction Python
- Course GitHub Demo

Introduction to GitHub

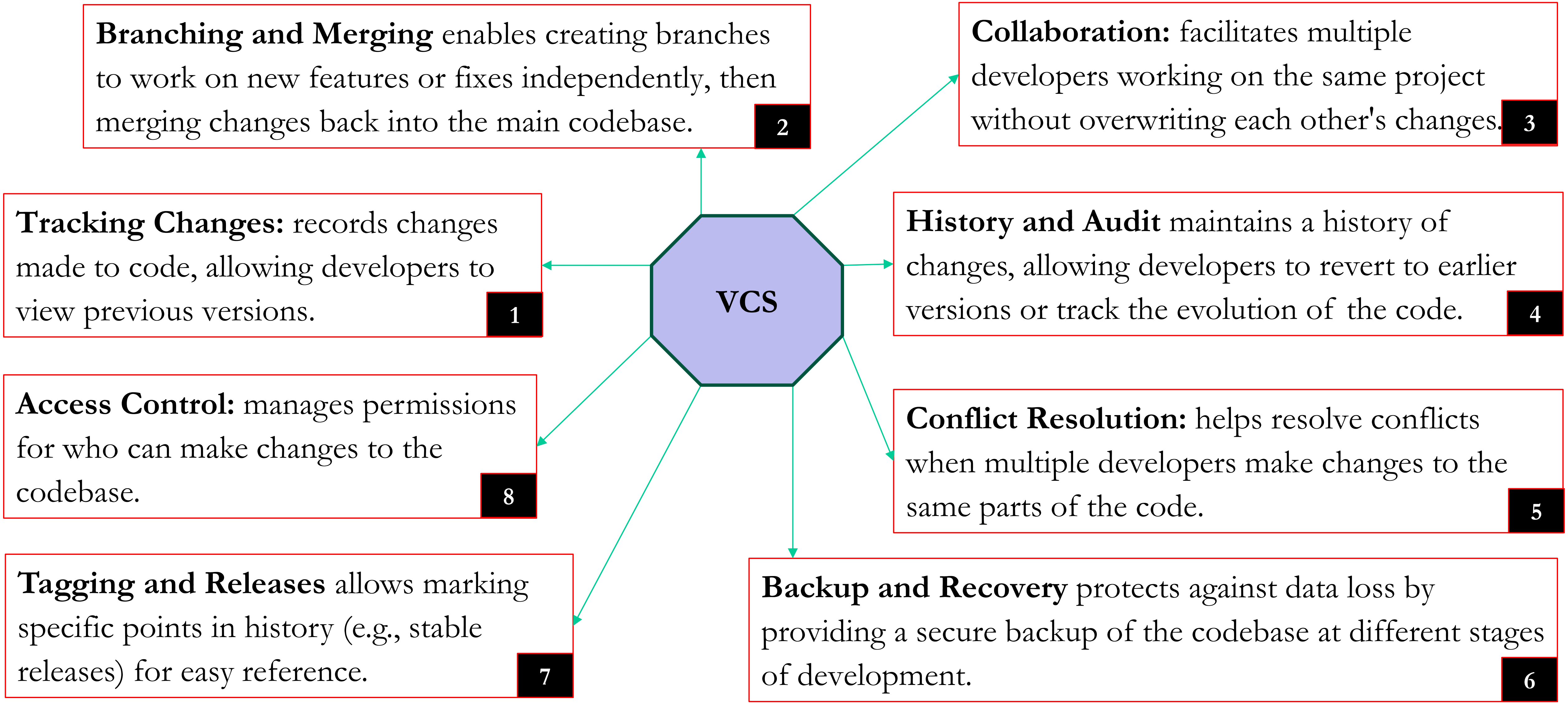
- Code version control



Project Team

Introduction to GitHub

- **Software Version Control System (VCS)**



Tracking Changes: records changes made to code, allowing developers to view previous versions. 1

Branching and Merging enables creating branches to work on new features or fixes independently, then merging changes back into the main codebase. 2

Collaboration: facilitates multiple developers working on the same project without overwriting each other's changes. 3

History and Audit maintains a history of changes, allowing developers to revert to earlier versions or track the evolution of the code. 4

Conflict Resolution: helps resolve conflicts when multiple developers make changes to the same parts of the code. 5

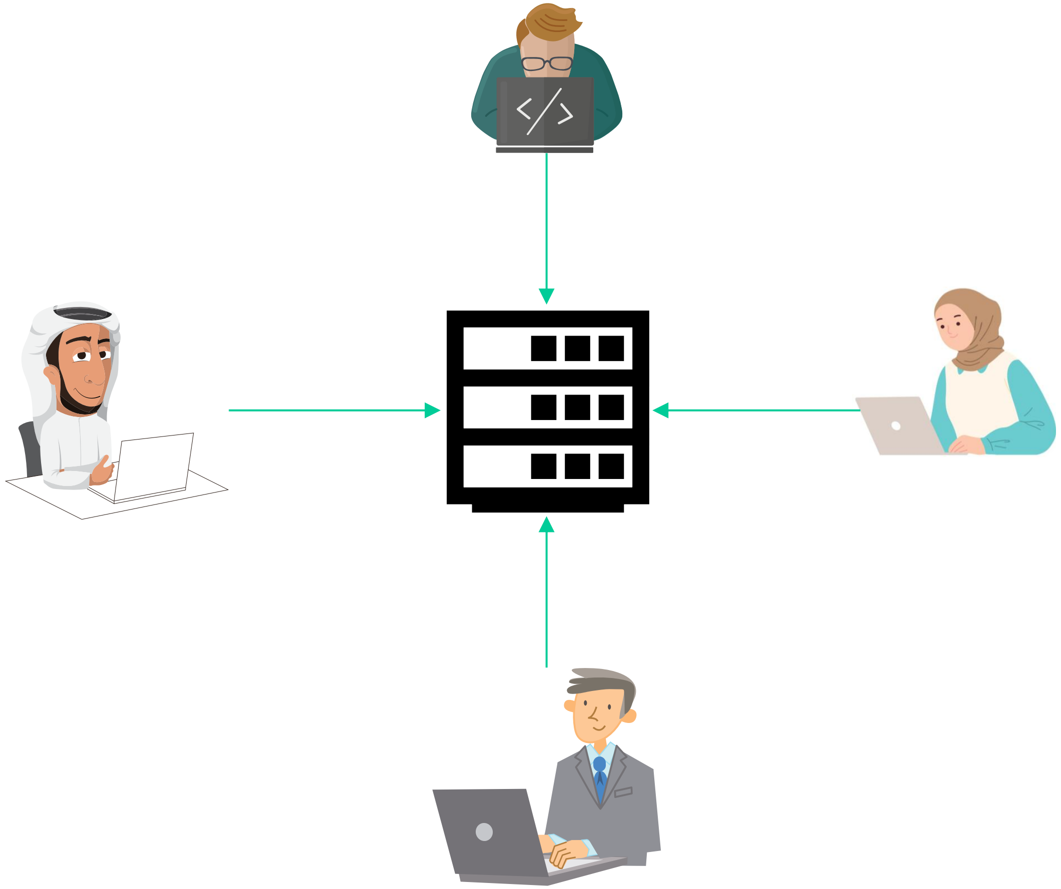
Backup and Recovery protects against data loss by providing a secure backup of the codebase at different stages of development. 6

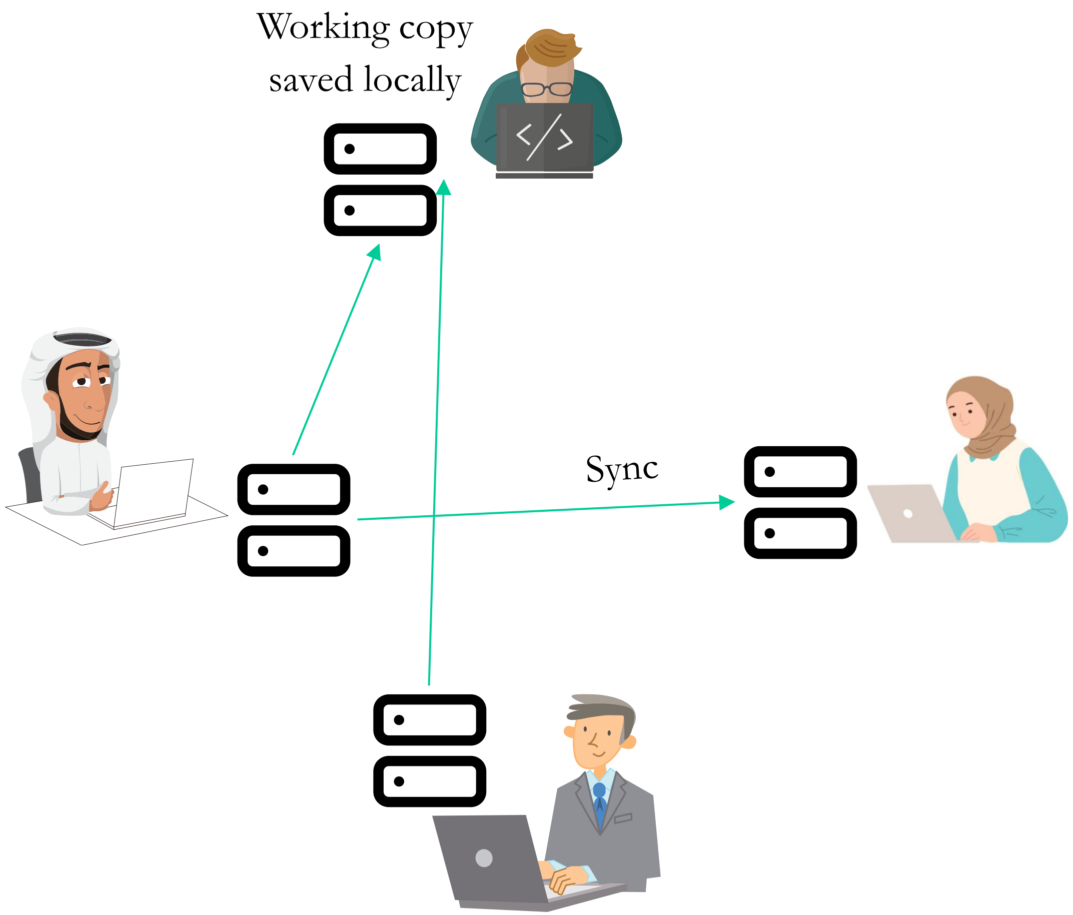
Tagging and Releases allows marking specific points in history (e.g., stable releases) for easy reference. 7

Access Control: manages permissions for who can make changes to the codebase. 8

Introduction to GitHub

- **Software Version Control System (VCS)**





Centralized VCS

(e.g., subversion and MS team foundation server)

Distributed VCS

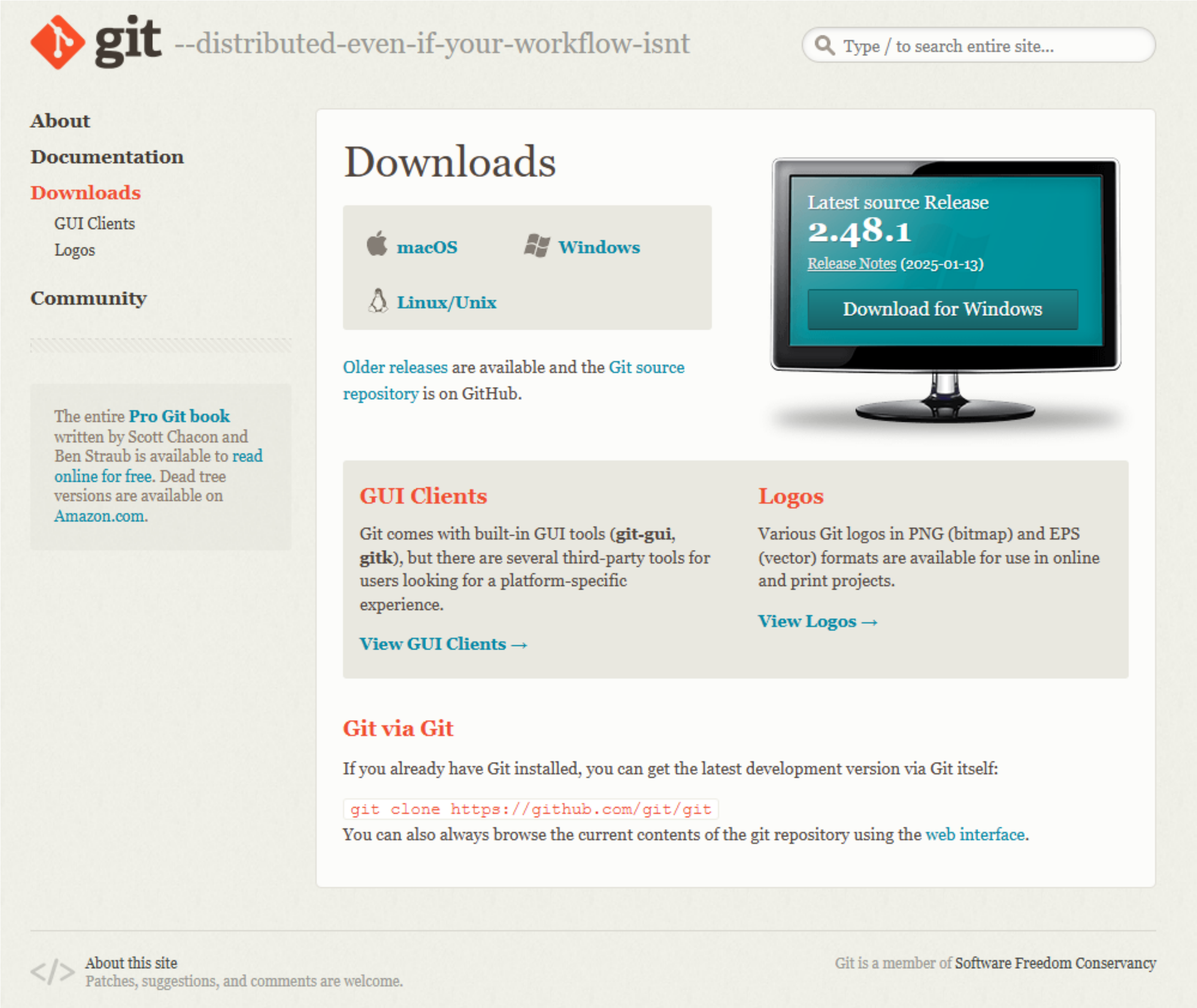
(e.g., Git and Mercurial)

Introduction to GitHub

- **What is git?**
  - git (Global Information Tracker) is an open source, **distributed version control system**.
  - It allows you to :
    - Revert files or the whole project to an earlier state
    - Compare changes over time
    - See who modified what?
    - Control modifications by collaborators with the permission of admin/owners

Introduction to GitHub

- Installing git



git --distributed-even-if-your-workflow-isnt

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Downloads

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.48.1
Release Notes (2025-01-13)

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the web interface.

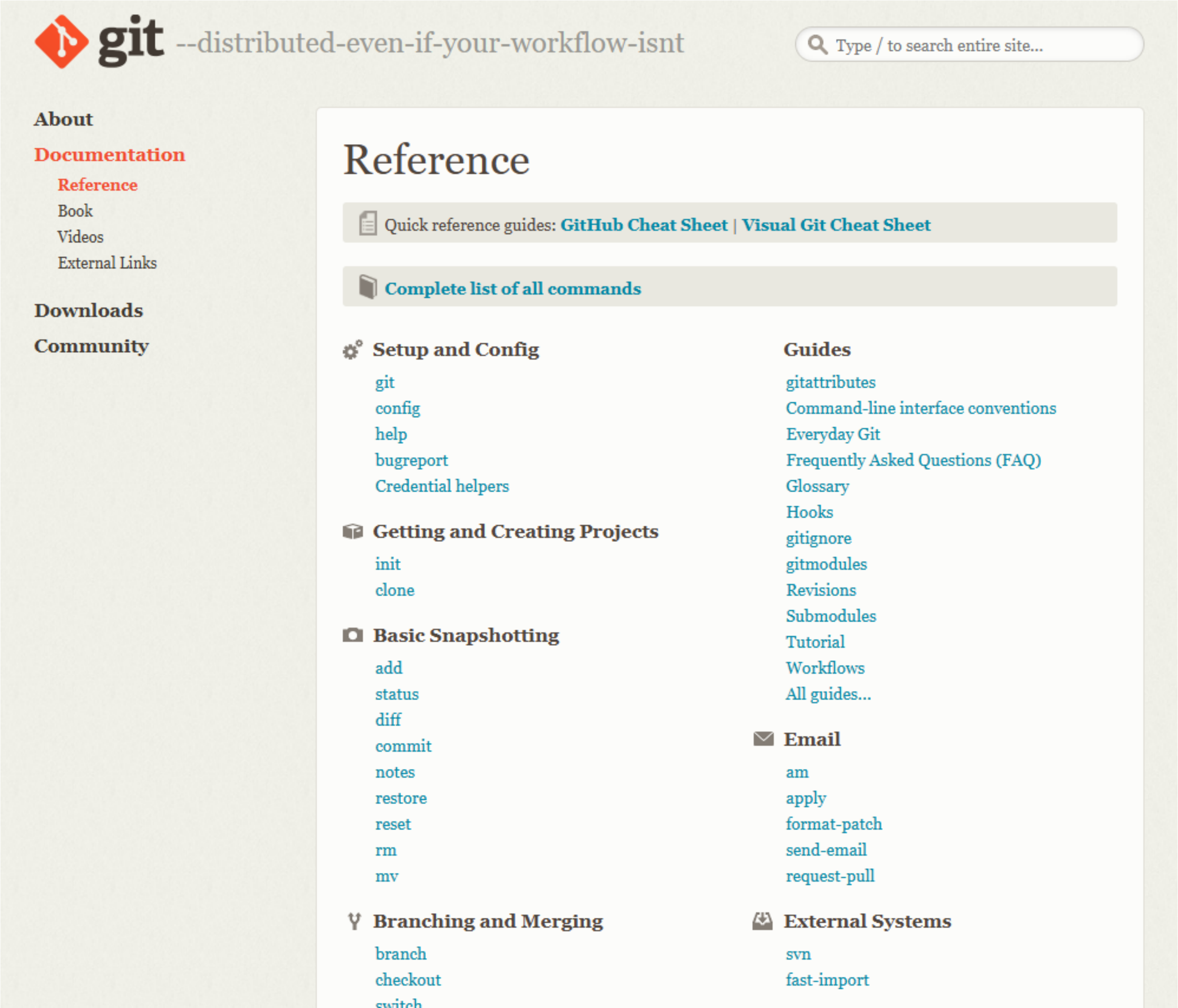
About this site
Patches, suggestions, and comments are welcome.

Git is a member of Software Freedom Conservancy

https://git-scm.com/downloads

Introduction to GitHub

- **Dive in**



git --distributed-even-if-your-workflow-isnt

Reference

Quick reference guides: GitHub Cheat Sheet | Visual Git Cheat Sheet

Complete list of all commands

Setup and Config
- git
- config
- help
- bugreport
- Credential helpers

Getting and Creating Projects
- init
- clone

Basic Snapshotting
- add
- status
- diff
- commit
- notes
- restore
- reset
- rm
- mv

Branching and Merging
- branch
- checkout
- switch

Guides
- gitattributes
- Command-line interface conventions
- Everyday Git
- Frequently Asked Questions (FAQ)
- Glossary
- Hooks
- gitignore
- gitmodules
- Revisions
- Submodules
- Tutorial
- Workflows
- All guides...

Email
- am
- apply
- format-patch
- send-email
- request-pull

External Systems
- svn
- fast-import

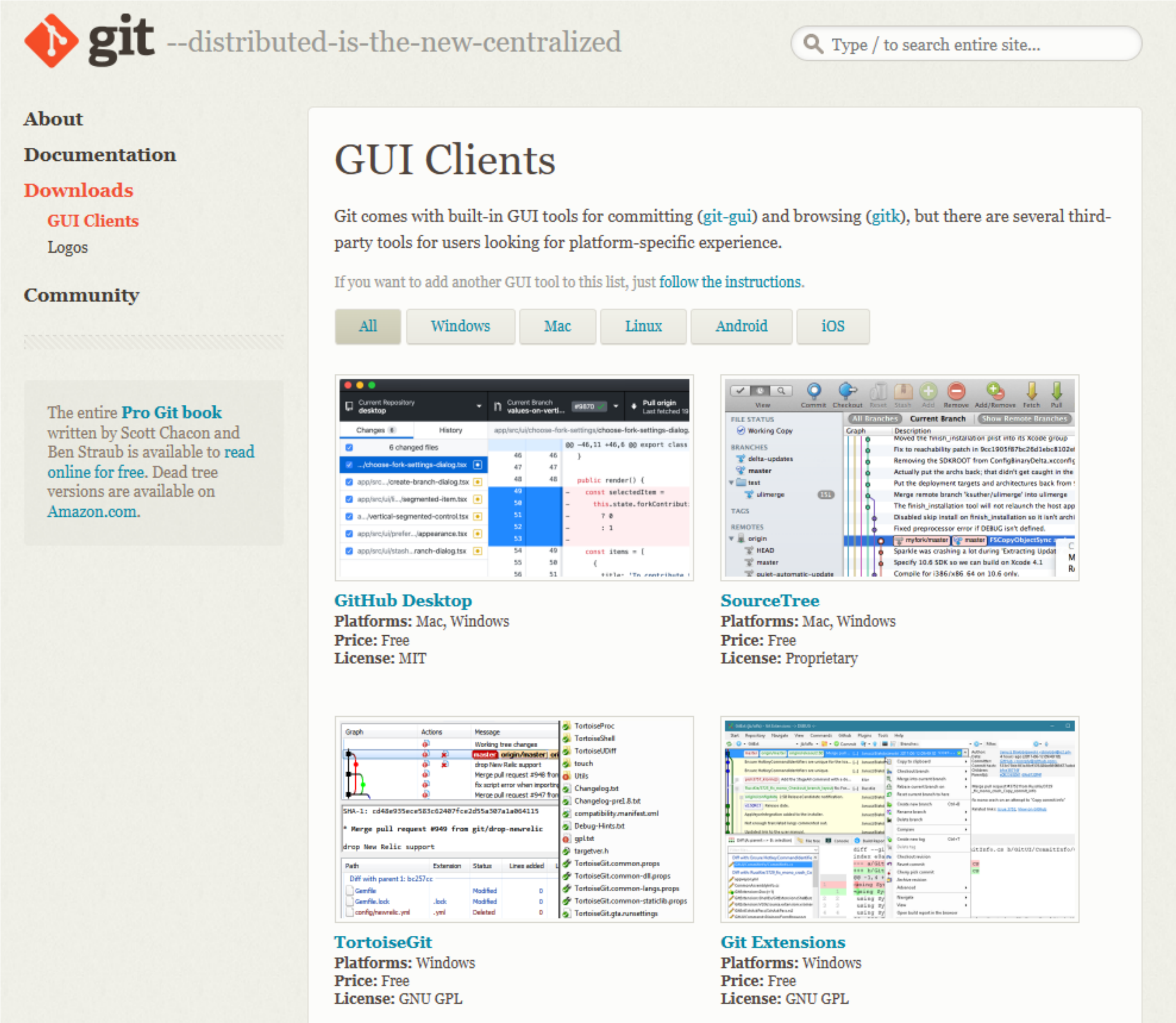
git

https://git-scm.com/docs

- Creating New Repo
- Committing
- Branching/Merging
- Inspection
- Cloning a Repo
- Updating

Introduction to GitHub

- Git Clients



git --distributed-is-the-new-centralized

Type / to search entire site...

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

GUI Clients

Git comes with built-in GUI tools for committing (git-gui) and browsing (gitk), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just follow the instructions.

All | Windows | Mac | Linux | Android | iOS

GitHub Desktop
Platforms: Mac, Windows
Price: Free
License: MIT

SourceTree
Platforms: Mac, Windows
Price: Free
License: Proprietary

TortoiseGit
Platforms: Windows
Price: Free
License: GNU GPL

Git Extensions
Platforms: Windows
Price: Free
License: GNU GPL

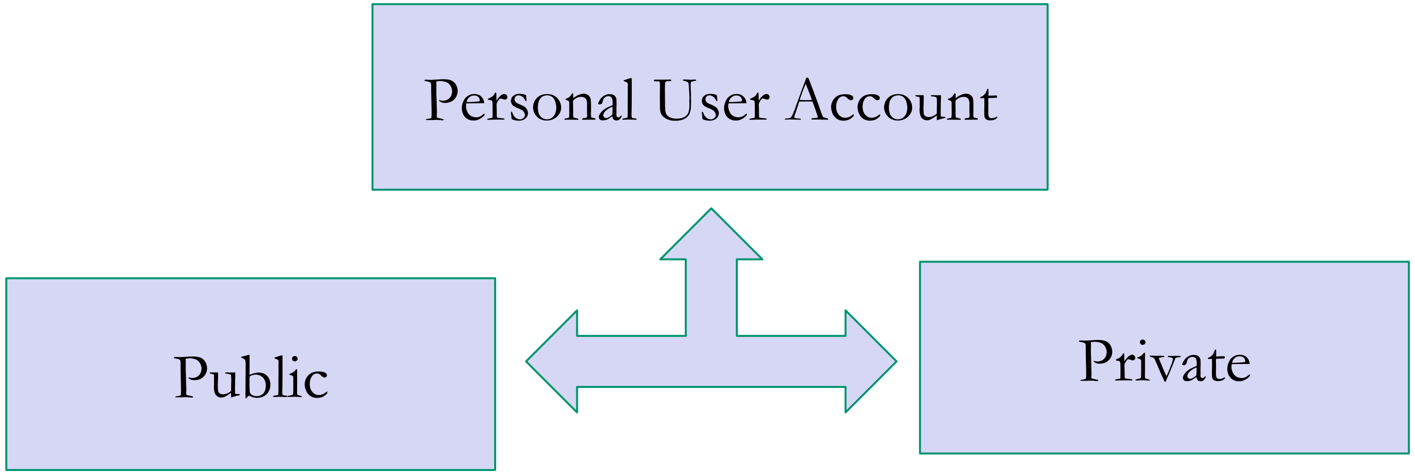
https://git-scm.com/downloads/guis

Introduction to GitHub

- **What is GitHub?**
  - GitHub is a repository hosting service for Git.
  - While Git is a command line tool, GitHub provides a web-based graphical interface that works on top of Git. It can also be treated as a social platform to share knowledge and work.
  - It also provides access control and several collaboration features, such as wikis and basic task management tools.

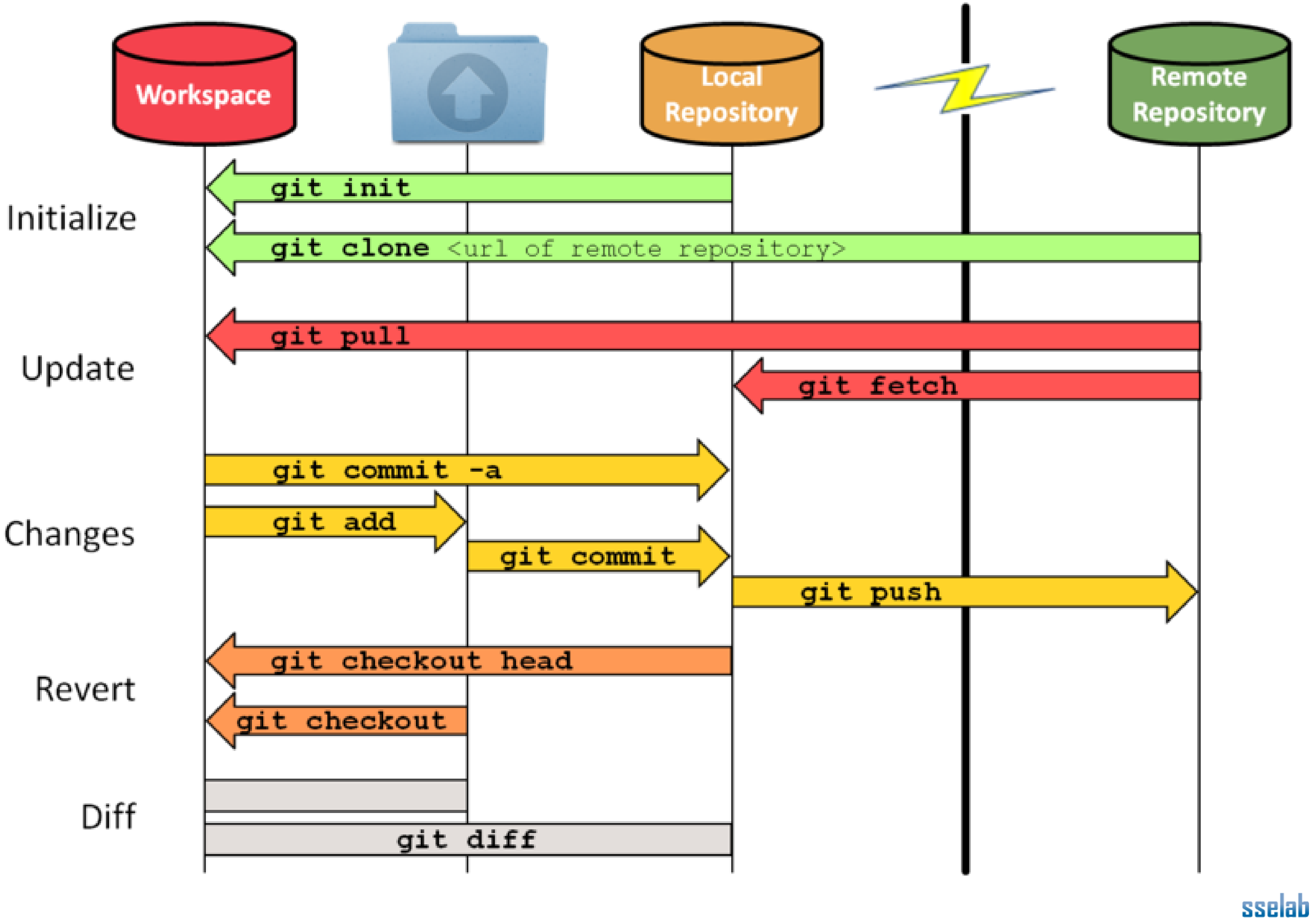
Introduction to GitHub

- **GitHub Structure**
  - Type of Project “Repository = repo”
    - Public
    - Private
  - Usage
    - Organize single project
    - It can contain folders, files, images, spreadsheets, data sets,...etc.
    - Add different collaborators for the repo
  - Remote repository vs Local repository



Introduction to GitHub

- **GitHub Workflow**

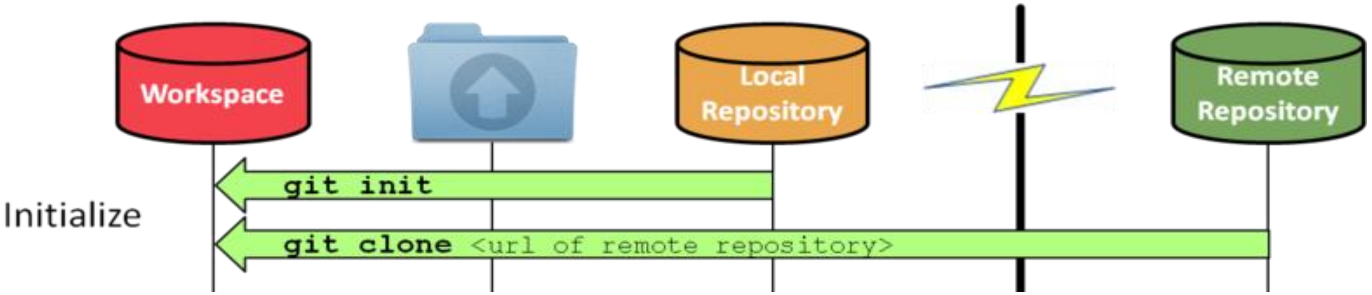


- **git init**: Initialize a Git repository in a directory.
- **git clone <URL>**: Copy a remote repository locally.
- **git pull**: Fetch and merge changes from the remote repo.
- **git fetch**: Download changes from the remote repository without merging them into your local branch.
- **git commit -m "message"**: Commit staged changes with a message.
- **git add .**: Stage all changes in the current directory.
- **git push:** Upload changes to the remote repo.
- **git checkout head**: resets a file to its last committed state (HEAD)
- **git diff**: Show changes between files or commits.

Introduction to GitHub

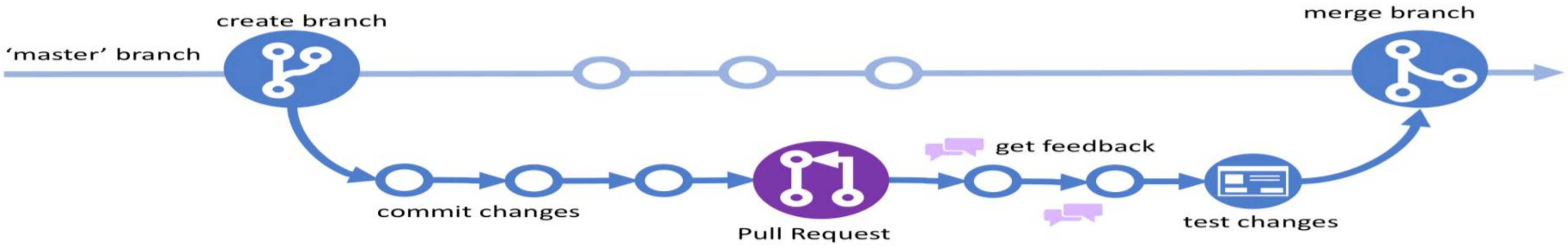
- **GitHub Workflow**

  - **Creating repository:** Creating/Initializing a repository for multiple people to work together



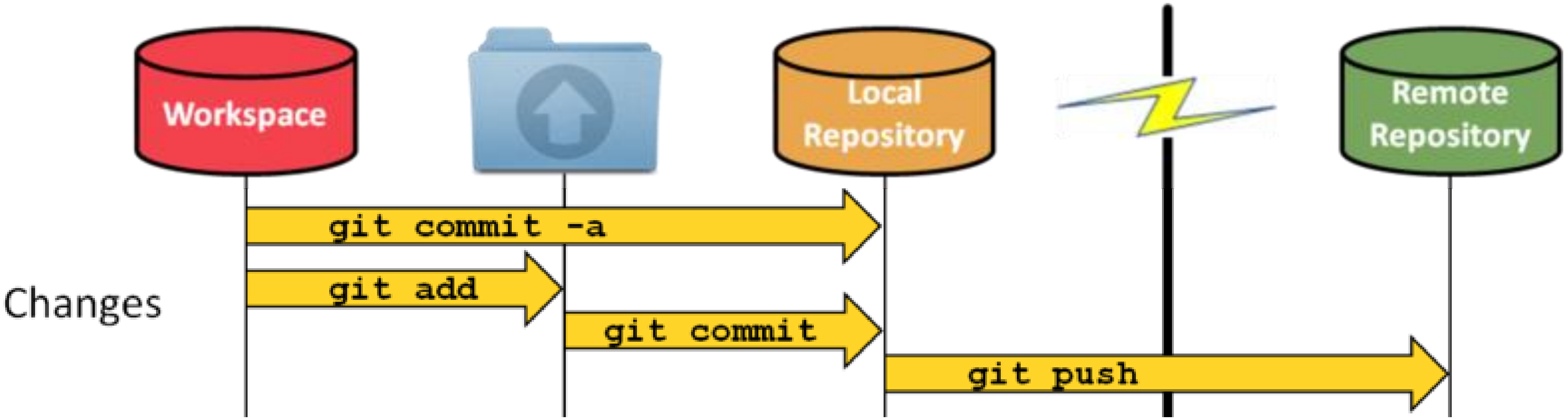
  - **Master in a repository:** This is the final version that is considered ready to use by anybody in the team

  - **Creating a branch:** Create a copy of the master branch to make new changes without affecting the master



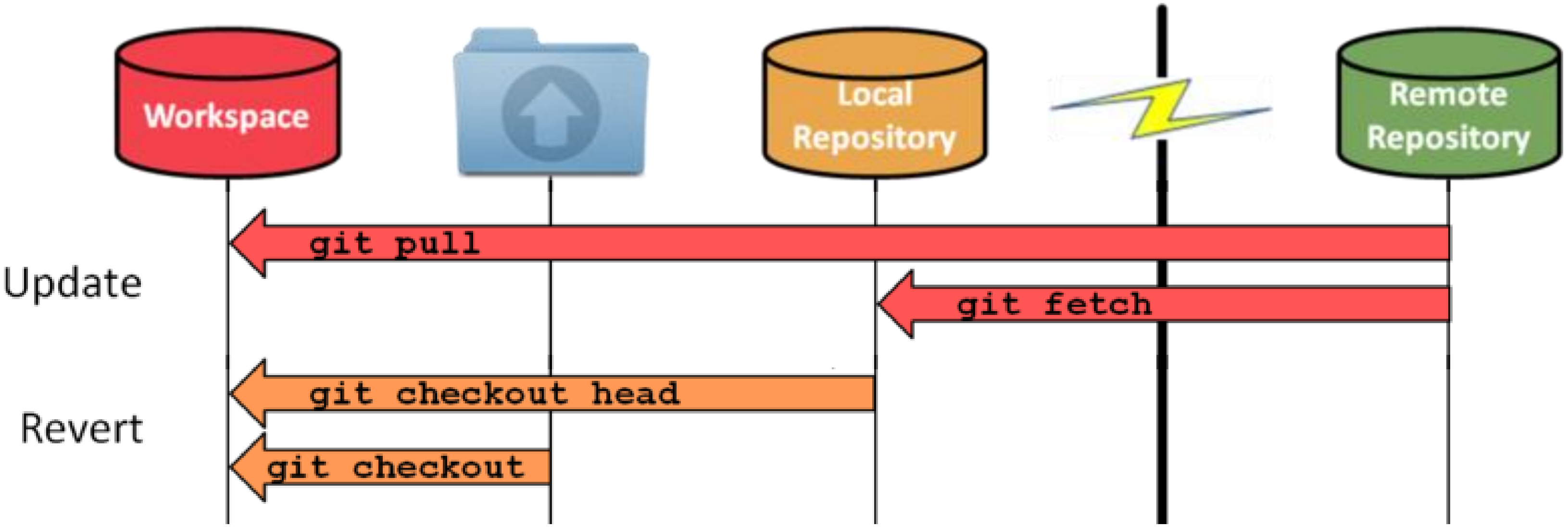
Introduction to GitHub

- **GitHub Workflow**
  - **Add:** adds your modified files to the queue to be committed later (Staging phase)
  - **Commit:** commits the files that have been added and creates a new revision with a log to the local repo. E.g. of log code: fb2d2ec5069fc6776c80b3ad6b7cbde3cade4e
  - **Push:** Push your changes to the remote repository



Introduction to GitHub

- **GitHub Workflow**
  - **Fetch:** Retrieve the latest meta-data info from the remote repo to the local repo
  - **Merge/Checkout:** Merge the new meta-data from the local repository to the workspace
  - **Pull:** Fetch + merge. Copy the data from the remote repo to the current workspace

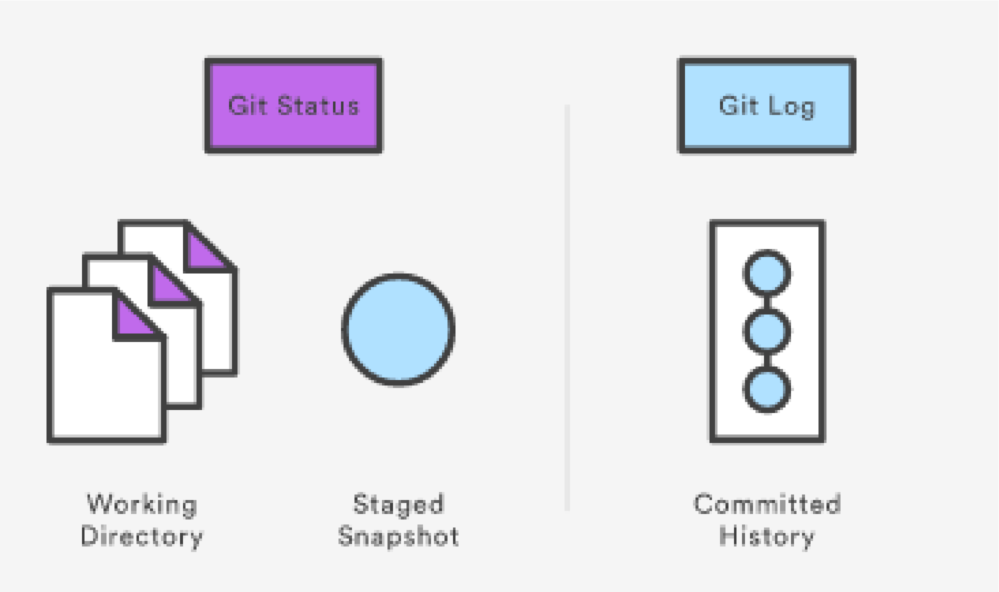


Introduction to GitHub

- **GitHub Workflow**

  - **git status:** displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show you any information regarding the committed project history.

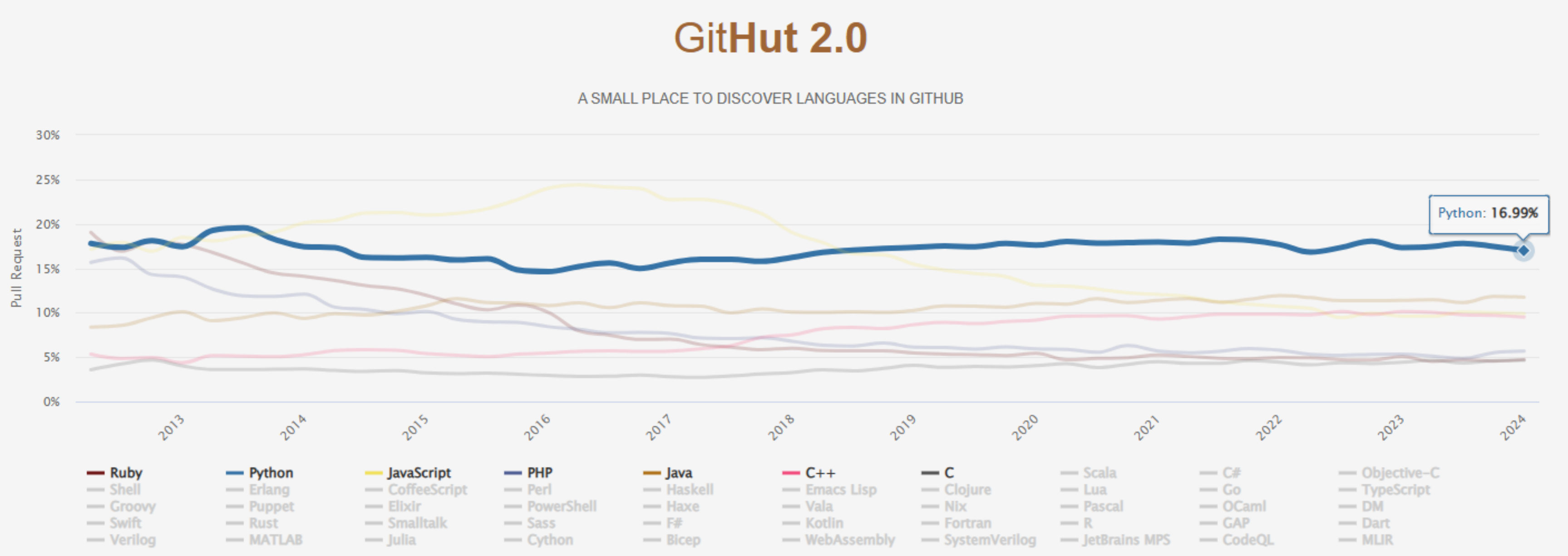
  - **git log:** displays committed snapshots. It lets you list the project history, filter it, and search for specific changes.



Outline

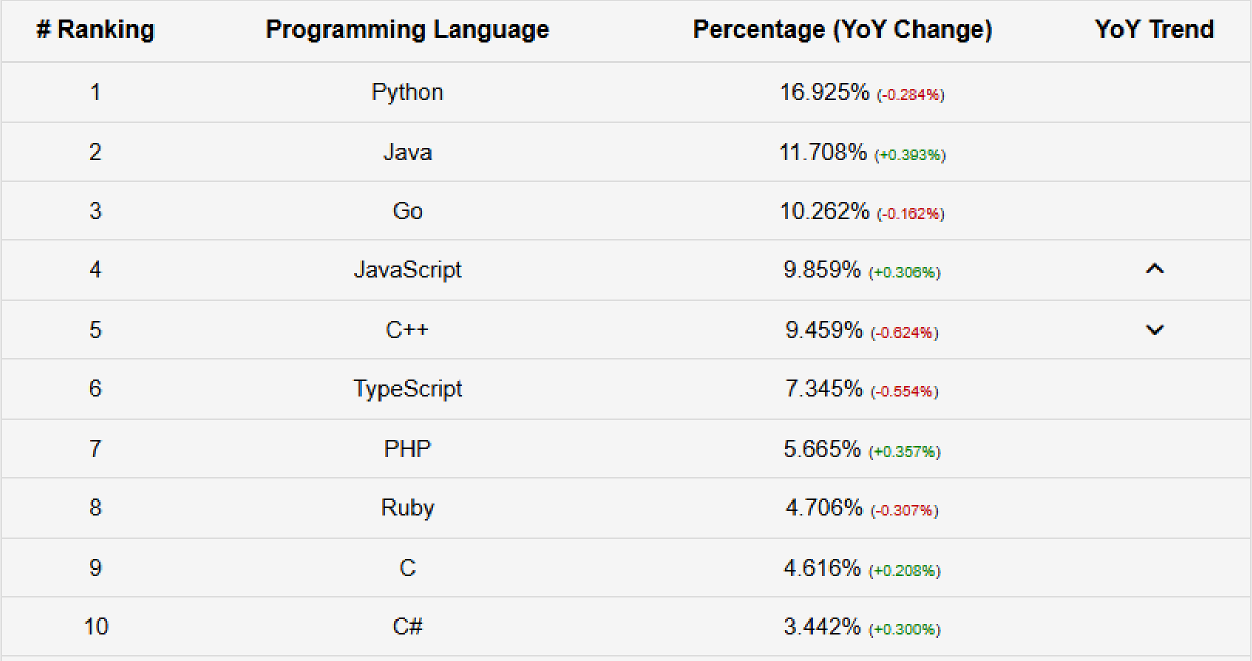
- Why Coding?
- Introduction to GitHub
- **<u>Introduction Python</u>**
- Course GitHub Demo

Introduction to Python

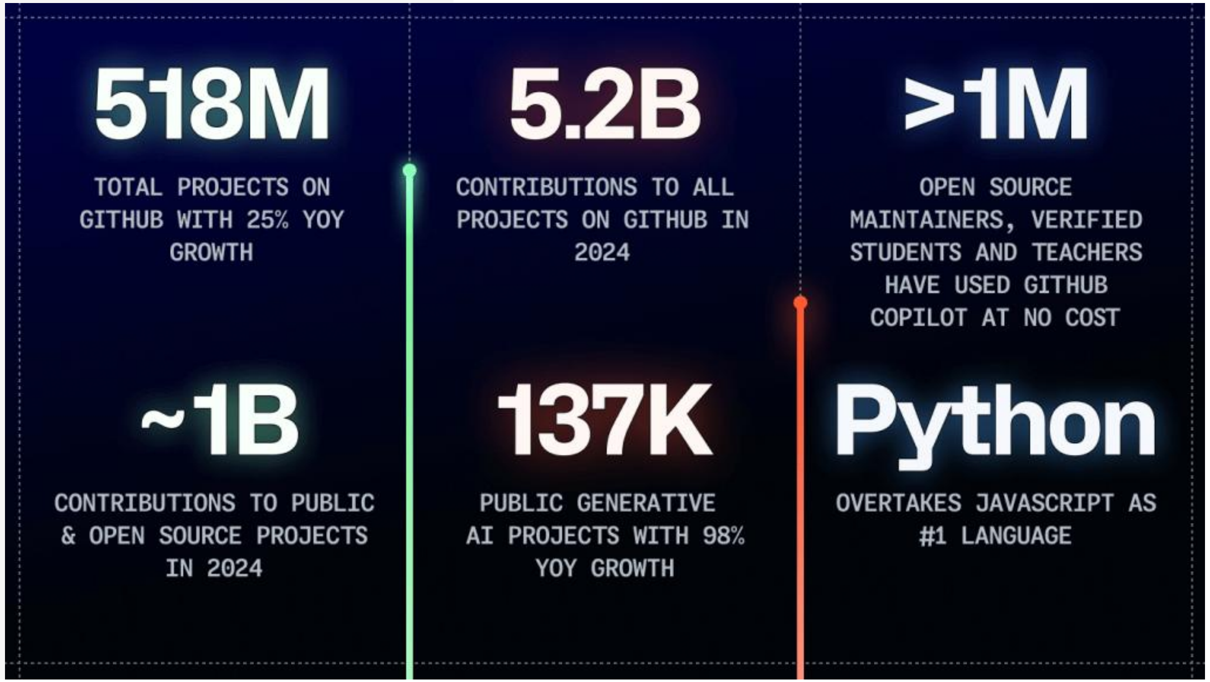


Sources:

- https://madnight.github.io/githut/#/pull_requests/2024/1
- https://github.blog/news-insights/octoverse/octoverse-2024/



# Ranking	Programming Language	Percentage (YoY Change)	YoY Trend
1	Python	16.925% (-0.284%)	
2	Java	11.708% (+0.393%)	
3	Go	10.262% (-0.162%)	
4	JavaScript	9.859% (+0.306%)	⌃
5	C++	9.459% (-0.624%)	⌄
6	TypeScript	7.345% (-0.554%)	
7	PHP	5.665% (+0.357%)	
8	Ruby	4.706% (-0.307%)	
9	C	4.616% (+0.208%)	
10	C#	3.442% (+0.300%)	



518M
TOTAL PROJECTS ON GITHUB WITH 25% YOY GROWTH

5.2B
CONTRIBUTIONS TO ALL PROJECTS ON GITHUB IN 2024

>1M
OPEN SOURCE MAINTAINERS, VERIFIED STUDENTS AND TEACHERS HAVE USED GITHUB COPILOT AT NO COST

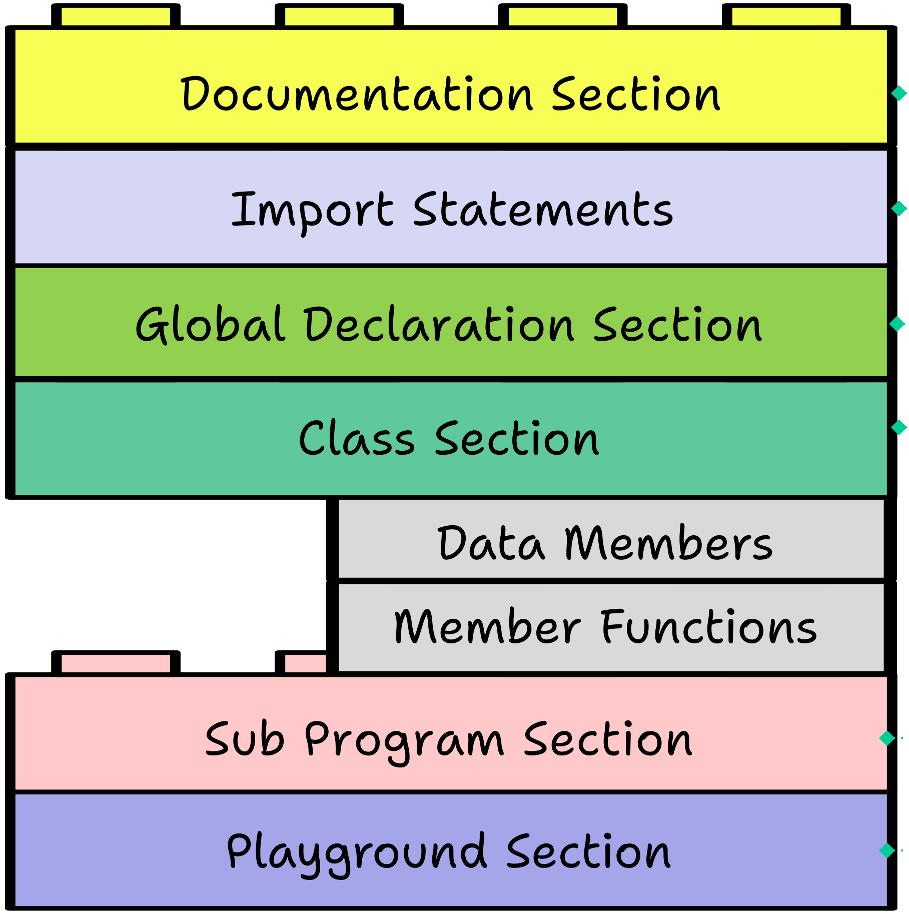
~1B
CONTRIBUTIONS TO PUBLIC & OPEN SOURCE PROJECTS IN 2024

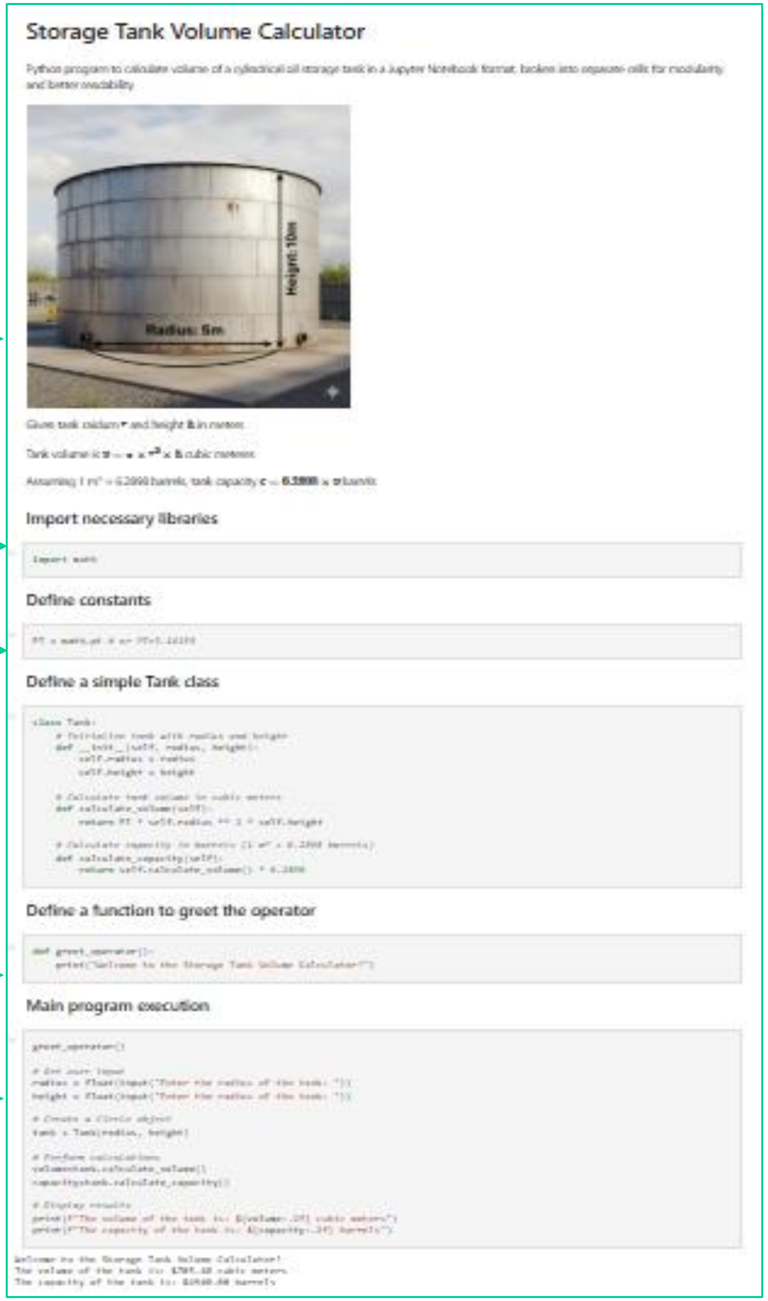
137K
PUBLIC GENERATIVE AI PROJECTS WITH 98% YOY GROWTH

Python
OVERTAKES JAVASCRIPT AS #1 LANGUAGE

Introduction to Python

- Structure of a Python Program





Storage Tank Volume Calculator

Python program to calculate volume of a cylindrical oil storage tank in a Jupyter Notebook format, broken into separate cells for modularity and better readability

Given tank radius r and height h in meters

Tank volume is $v = \pi \times r^2 \times h$ cubic meters

Assuming 1 m³ = 6.2898 barrels, tank capacity $c = 6.2898 \times v$ barrels

Import necessary libraries

```
import math
```

Define constants

```
PI = math.pi # or PI=3.14159
```

Define a simple Tank class

```
class Tank:
    # Initialize tank with radius and height
    def __init__(self, radius, height):
        self.radius = radius
        self.height = height

    # Calculate tank volume in cubic meters
    def calculate_volume(self):
        return PI * self.radius ** 2 * self.height

    # Calculate capacity in barrels (1 m³ = 6.2898 barrels)
    def calculate_capacity(self):
        return self.calculate_volume() * 6.2898
```

Define a function to greet the operator

```
def greet_operator():
    print("Welcome to the Storage Tank Volume Calculator!")
```

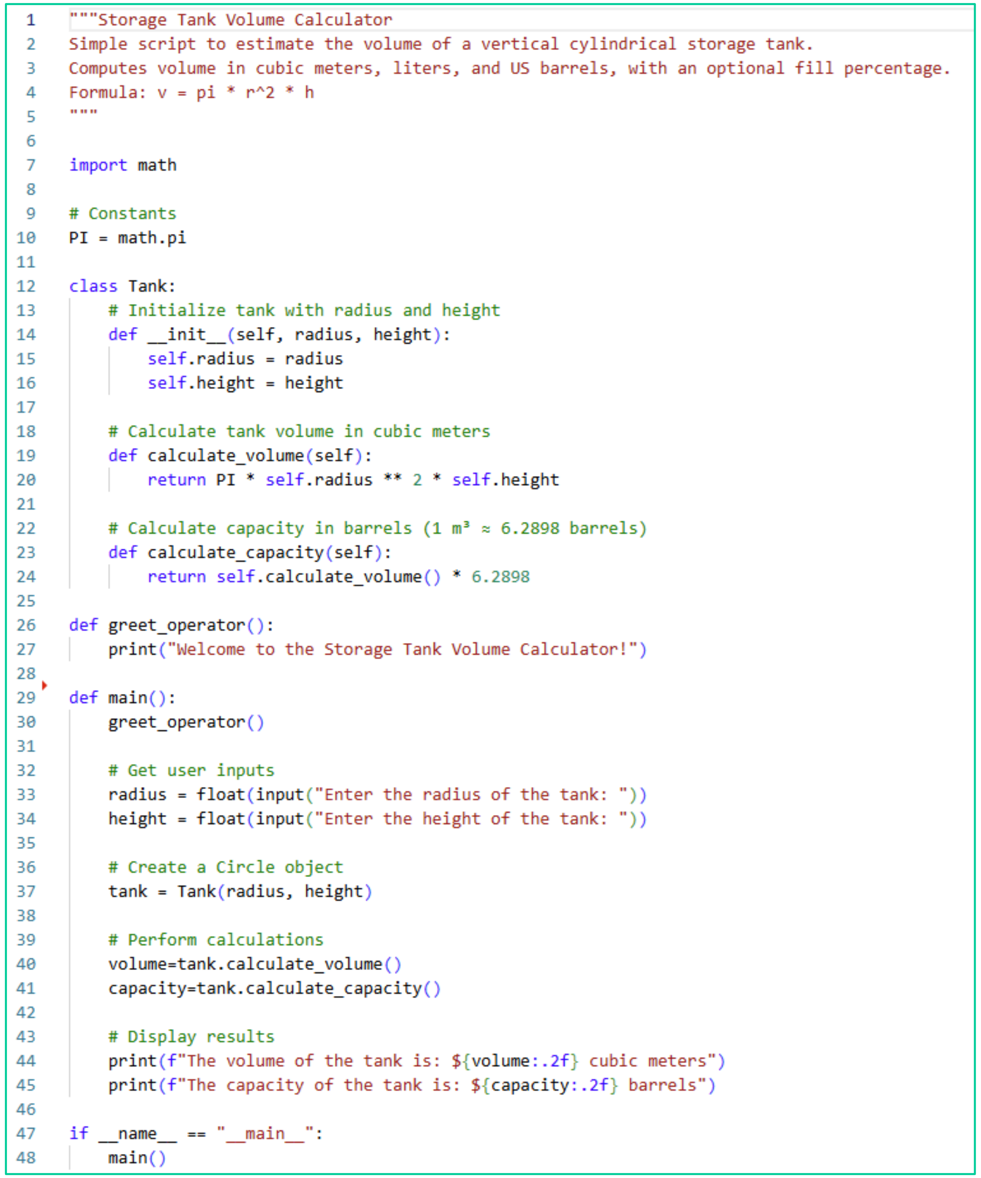
Main program execution

```
greet_operator()

# Get user input
radius = float(input("Enter the radius of the tank: "))
height = float(input("Enter the radius of the tank: "))

# Create a Circle object
tank = Tank(radius, height)

# Perform calculations
volume=tank.calculate_volume()
capacity=tank.calculate_capacity()

# Display results
print(f"The volume of the tank is: {volume:.2f} cubic meters")
print(f"The capacity of the tank is: {capacity:.2f} barrels")
```

```
Welcome to the Storage Tank Volume Calculator!
The volume of the tank is: 785.40 cubic meters
The capacity of the tank is: [illegible] barrels
```

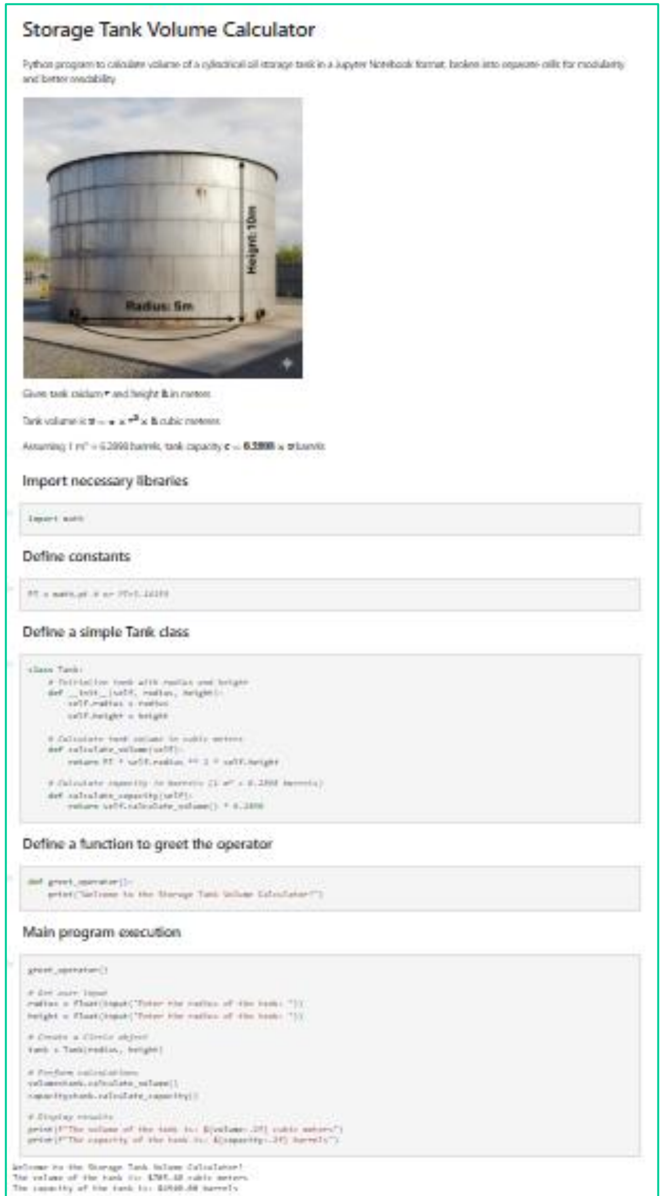
Introduction to Python

- Python Script



```python
"""Storage Tank Volume Calculator
Simple script to estimate the volume of a vertical cylindrical storage tank.
Computes volume in cubic meters, liters, and US barrels, with an optional fill percentage.
Formula: v = pi * r^2 * h
"""

import math

# Constants
PI = math.pi

class Tank:
    # Initialize tank with radius and height
    def __init__(self, radius, height):
        self.radius = radius
        self.height = height

    # Calculate tank volume in cubic meters
    def calculate_volume(self):
        return PI * self.radius ** 2 * self.height

    # Calculate capacity in barrels (1 m³ ≈ 6.2898 barrels)
    def calculate_capacity(self):
        return self.calculate_volume() * 6.2898

def greet_operator():
    print("Welcome to the Storage Tank Volume Calculator!")

def main():
    greet_operator()

    # Get user inputs
    radius = float(input("Enter the radius of the tank: "))
    height = float(input("Enter the height of the tank: "))

    # Create a Circle object
    tank = Tank(radius, height)

    # Perform calculations
    volume=tank.calculate_volume()
    capacity=tank.calculate_capacity()

    # Display results
    print(f"The volume of the tank is: ${volume:.2f} cubic meters")
    print(f"The capacity of the tank is: ${capacity:.2f} barrels")

if __name__ == "__main__":
    main()
```

- Python Juypter



Storage Tank Volume Calculator

Python program to calculate volume of a cylindrical oil storage tank in a Jupyter Notebook format, broken into separate cells for modularity and better readability

Given tank radius r and height h in meters

Tank volume is $v = \pi \times r^2 \times h$ cubic meters

Assuming 1 m³ ≈ 6.2898 barrels, tank capacity $c = 6.2898 \times v$ barrels

Import necessary libraries

```python
import math
```

Define constants

```python
PI = math.pi # or PI=3.14159
```

Define a simple Tank class

```python
class Tank:
    # Initialize tank with radius and height
    def __init__(self, radius, height):
        self.radius = radius
        self.height = height

    # Calculate tank volume in cubic meters
    def calculate_volume(self):
        return PI * self.radius ** 2 * self.height

    # Calculate capacity in barrels (1 m³ ≈ 6.2898 barrels)
    def calculate_capacity(self):
        return self.calculate_volume() * 6.2898
```

Define a function to greet the operator

```python
def greet_operator():
    print("Welcome to the Storage Tank Volume Calculator!")
```

Main program execution

```python
greet_operator()

# Get user input
radius = float(input("Enter the radius of the tank: "))
height = float(input("Enter the radius of the tank: "))

# Create a Circle object
tank = Tank(radius, height)

# Perform calculations
volume=tank.calculate_volume()
capacity=tank.calculate_capacity()

# Display results
print(f"The volume of the tank is: ${volume:.2f} cubic meters")
print(f"The capacity of the tank is: ${capacity:.2f} barrels")
```

```
Welcome to the Storage Tank Volume Calculator!
The volume of the tank is: $785.40 cubic meters
The capacity of the tank is: $4940.00 barrels
```

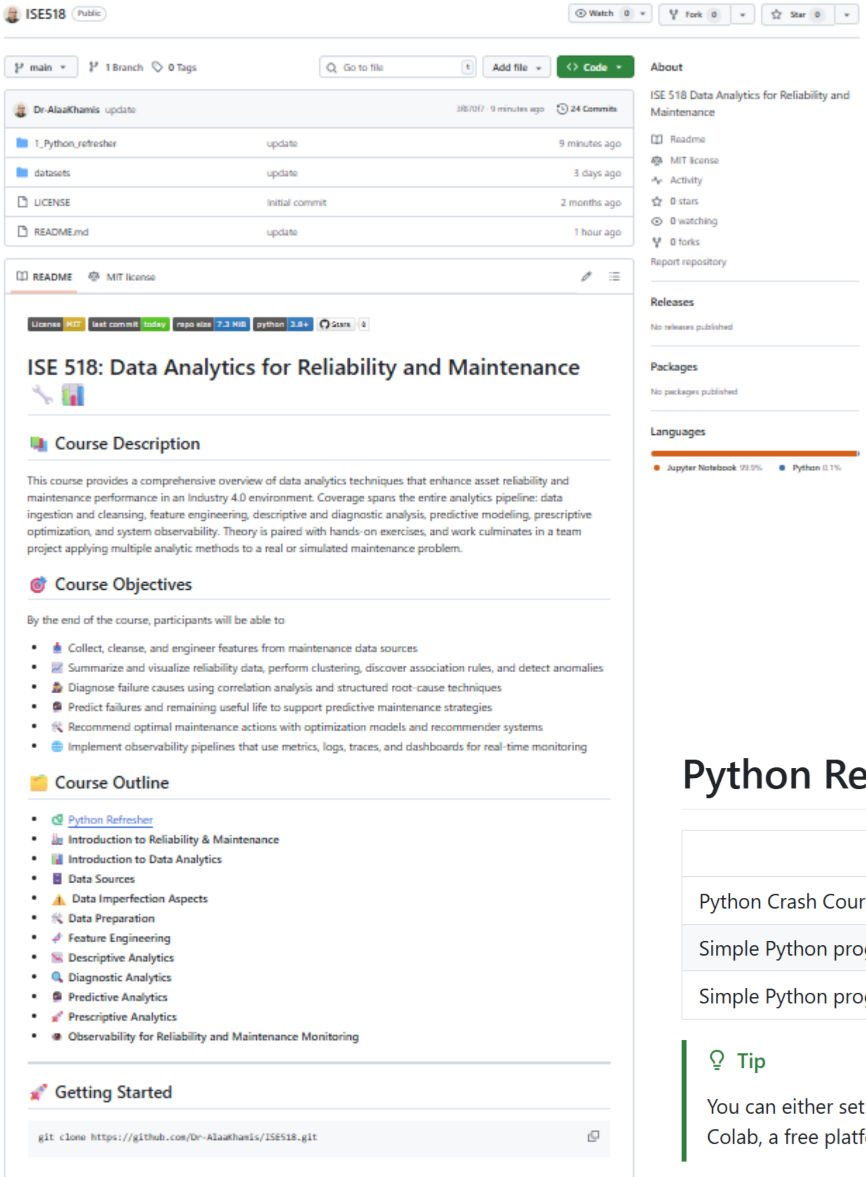
Source: https://github.com/Dr-AlaaKhamis/ISE518/tree/main/1_Python_refresher

Outline

- Why Coding?
- Introduction to GitHub
- Introduction Python
- **Course GitHub Demo**

Course GitHub Demo

https://github.com/Dr-AlaaKhamis/ISE518



Dr-AlaaKhamis update 3f8701f · 9 minutes ago 24 Commits

1_Python_refresher	update	9 minutes ago
datasets	update	3 days ago
LICENSE	Initial commit	2 months ago
README.md	update	1 hour ago

About

ISE 518 Data Analytics for Reliability and Maintenance

- Readme
- MIT license
- Activity
- 0 stars
- 0 watching
- 0 forks

Report repository

Releases

No releases published

Packages

No packages published

Languages

- Jupyter Notebook 99.9%
- Python 0.1%

ISE 518: Data Analytics for Reliability and Maintenance 🔧 📊

Course Description

This course provides a comprehensive overview of data analytics techniques that enhance asset reliability and maintenance performance in an Industry 4.0 environment. Coverage spans the entire analytics pipeline: data ingestion and cleansing, feature engineering, descriptive and diagnostic analysis, predictive modeling, prescriptive optimization, and system observability. Theory is paired with hands-on exercises, and work culminates in a team project applying multiple analytic methods to a real or simulated maintenance problem.

🎯 Course Objectives

By the end of the course, participants will be able to

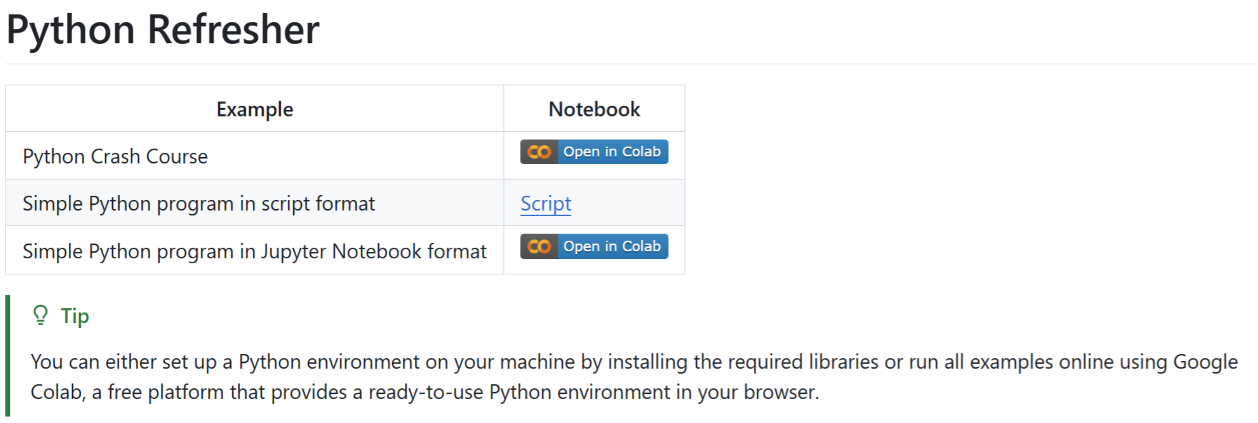
- Collect, cleanse, and engineer features from maintenance data sources
- 📈 Summarize and visualize reliability data, perform clustering, discover association rules, and detect anomalies
- Diagnose failure causes using correlation analysis and structured root-cause techniques
- Predict failures and remaining useful life to support predictive maintenance strategies
- 🛠 Recommend optimal maintenance actions with optimization models and recommender systems
- 🌐 Implement observability pipelines that use metrics, logs, traces, and dashboards for real-time monitoring

Course Outline

- Python Refresher
- Introduction to Reliability & Maintenance
- 📊 Introduction to Data Analytics
- Data Sources
- ⚠️ Data Imperfection Aspects
- 🛠 Data Preparation
- Feature Engineering
- 📉 Descriptive Analytics
- 🔍 Diagnostic Analytics
- Predictive Analytics
- 🚀 Prescriptive Analytics
- Observability for Reliability and Maintenance Monitoring

🚀 Getting Started

```
git clone https://github.com/Dr-AlaaKhamis/ISE518.git
```



Python Refresher

Example	Notebook
Python Crash Course	Open in Colab
Simple Python program in script format	Script
Simple Python program in Jupyter Notebook format	Open in Colab

💡 Tip

You can either set up a Python environment on your machine by installing the required libraries or run all examples online using Google Colab, a free platform that provides a ready-to-use Python environment in your browser.