Project Documentation (Weeks 1-3)

# Project 1: Command-Line Calculator

## Code:

import math

def cmd\_calc():

print(" Welcome to CalcHub - The Command-Line Calculator! \n")

print("Available Operations:")

print("1. Add (a)")

print("2. Subtract (s)")

print("3. Multiply (m)")

print("4. Divide (d)")

print("5. Power (p)")

print("6. Square Root (r)")

print("7. Exit (e)")

while True:

user\_choic = input("\nSelect an operation (Enter key): ").lower()

if user\_choic in {"a", "s", "m", "d", "p"}:

num1 = get\_number("Enter the first number: ")

num2 = get\_number("Enter the second number: ")

if user\_choic == "a":

result = add\_them(num1, num2)

elif user\_choic == "s":

result = subtract\_them(num1, num2)

elif user\_choic == "m":

result = multiply\_them(num1, num2)

elif user\_choic == "d":

result = divide\_them(num1, num2)

elif user\_choic == "p":

result = power\_them(num1, num2)

display\_result(result)

elif user\_choic == "r":

num = get\_number("Enter the number to find its square root: ")

result = sqrt\_them(num)

display\_result(result)

elif user\_choic == "e":

print("\nThank you for using CalcHub! Goodbye! ")

break

else:

print("Invalid input. Please try again.")

def get\_number(prompt):

while True:

try:

return float(input(prompt))

except ValueError:

print("Invalid number. Please enter a valid numeric value.")

def add\_them(a, b): return a + b

def subtract\_them(a, b): return a - b

def multiply\_them(a, b): return a \* b

def divide\_them(a, b):

return a / b if b != 0 else "Error: Division by zero is undefined."

def power\_them(a, b): return math.pow(a, b)

def sqrt\_them(a): return math.sqrt(a) if a >= 0 else "Error: Negative input for square root."

def display\_result(result):

print(f"\n🔍 Result: {result}\n" if isinstance(result, (int, float)) else f"\n❌ {result}\n")

if \_\_name\_\_ == "\_\_main\_\_":

cmd\_calc()

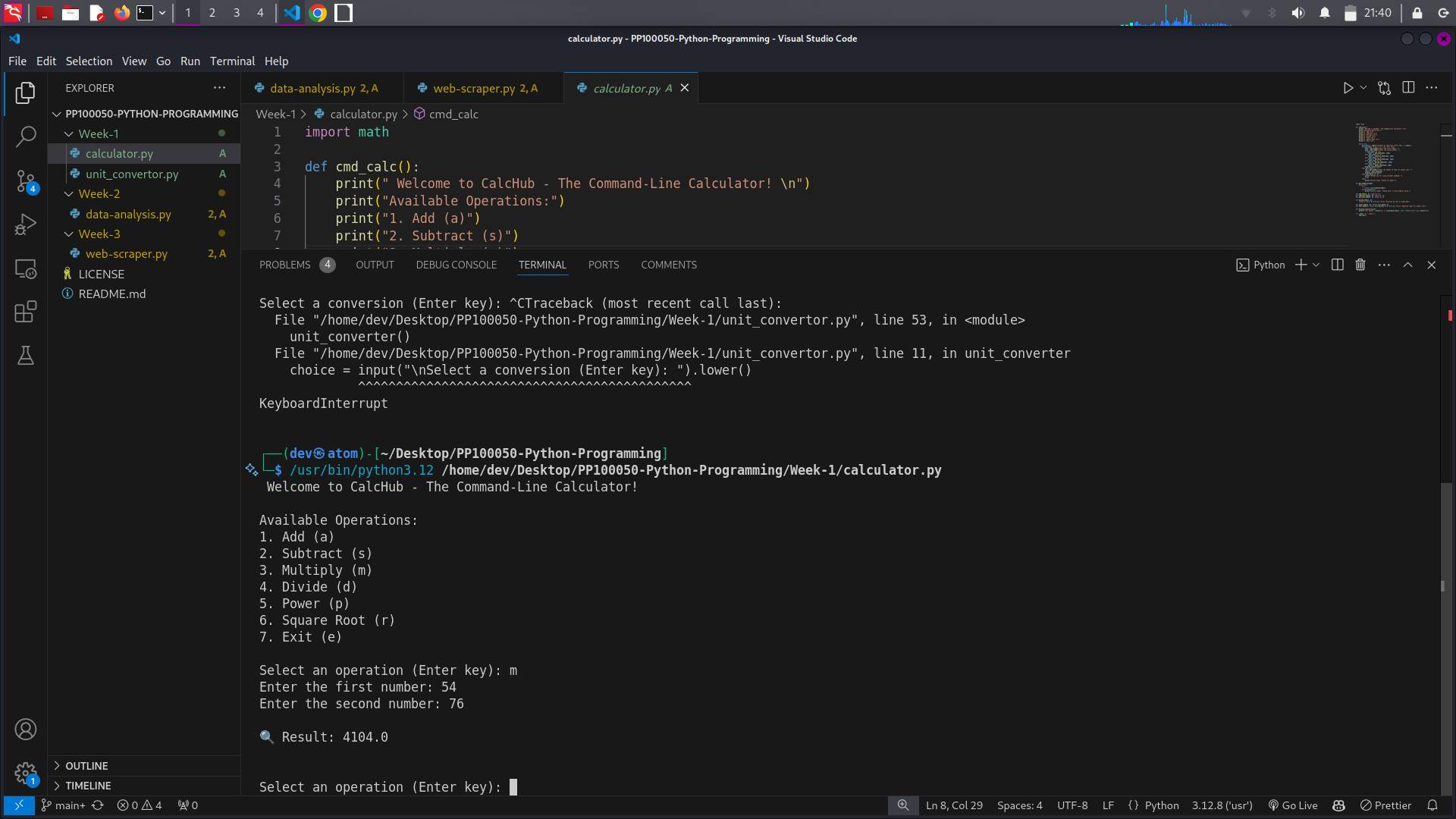
## Explanation:

This project implements a simple command-line calculator that performs basic arithmetic operations, including addition, subtraction, multiplication, and division. The user inputs the operation and the two numbers, and the program outputs the result. The calculator also handles division by zero with an error message.

## Tools and Technologies Used:

- Programming Language: Python  
- Libraries: None (pure Python)

## Output Result Screenshots:



# Project 2: Unit Conversion Tool

## Code:

def unit\_converter():

print("\n Welcome to UnitX - The Unit Conversion Tool!\n")

print("1. Length: Kilometers to Miles (k)")

print("2. Weight: Kilograms to Pounds (w)")

print("3. Temperature: Celsius to Fahrenheit (t)")

print("4. Exit (e)")

while True:

choice = input("\nSelect a conversion (Enter key): ").lower()

if choice == "k":

km = pos\_num("Enter distance in kilometers: ")

result = km\_to\_miles(km)

display\_result(f"{km} kilometers is {result:.2f} miles.")

elif choice == "w":

kg = pos\_num("Enter weight in kilograms: ")

result = kg\_to\_pounds(kg)

display\_result(f"{kg} kilograms is {result:.2f} pounds.")

elif choice == "t":

celsius = get\_number("Enter temperature in Celsius: ")

result = celsius\_to\_fahrenheit(celsius)

display\_result(f"{celsius}°C is {result:.2f}°F.")

elif choice == "e":

print("\nThank you for using UnitX! Goodbye! 👋")

break

else:

print("Invalid choice. Please try again.")

def get\_number(prompt):

while True:

try:

return float(input(prompt))

except ValueError:

print("Invalid input. Please enter a valid numeric value.")

def pos\_num(prompt):

while True:

number = get\_number(prompt)

if number >= 0:

return number

print("Please enter a positive value.")

def km\_to\_miles(km): return km \* 0.621371

def kg\_to\_pounds(kg): return kg \* 2.20462

def celsius\_to\_fahrenheit(c): return (c \* 9/5) + 32

def display\_result(message):

print(f"\n {message}\n")

if \_\_name\_\_ == "\_\_main\_\_":

unit\_converter()

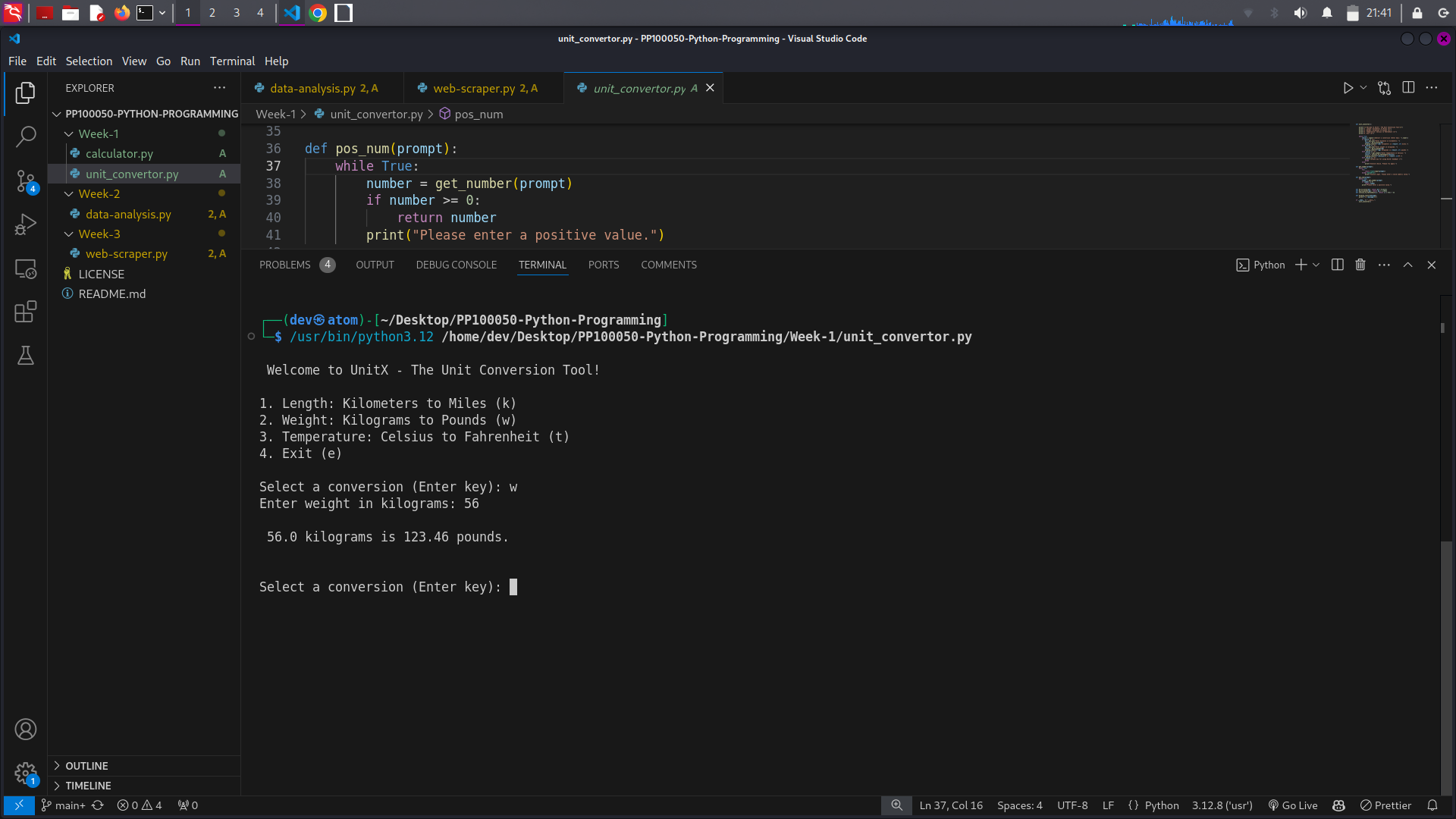
## Explanation:

This project allows users to convert between various units. Specifically, it supports converting kilometers to miles, kilograms to pounds, and Celsius to Fahrenheit. The user inputs the value, and the program returns the converted value. The menu-driven interface prompts the user for conversions and displays the result accordingly.

## Tools and Technologies Used:

- Programming Language: Python  
- Libraries: None (pure Python)

## Output Result Screenshots:



# Project 3: Basic Data Analysis

## Code:

import pandas as pd

import matplotlib.pyplot as plt

def data\_analyzer():

print("\n Welcome to DataX - The Basic Data Analysis Tool \n")

file\_path = input("Enter the path to the CSV file: ")

try:

data = pd.read\_csv(file\_path)

print("\n File loaded successfully!\n")

print(f"Columns available for analysis: {', '.join(data.columns)}")

except FileNotFoundError:

print("\n Error: File not found. Please check the file path and try again.")

return

except Exception as e:

print(f"\n Error: {e}")

return

while True:

col\_name = input("\nEnter the column name to analyze (or 'exit' to quit): ")

if col\_name.lower() == "exit":

print("\nThank you for using DataX!")

break

if col\_name not in data.columns:

print(f" Column '{col\_name}' not found. Please try again.")

continue

try:

numeric\_data = pd.to\_numeric(data[col\_name], errors="coerce").dropna()

if numeric\_data.empty:

print(f" Column '{col\_name}' does not contain valid numerical data.")

continue

avg = numeric\_data.mean()

max\_val = numeric\_data.max()

min\_val = numeric\_data.min()

print(f"\n Analysis of '{col\_name}':")

print(f" ➡️ Average: {avg:.2f}")

print(f" ➡️ Maximum: {max\_val:.2f}")

print(f" ➡️ Minimum: {min\_val:.2f}")

visualize = input("Would you like to visualize this data? (y/n): ").lower()

if visualize == "y":

visualize\_col(numeric\_data, col\_name)

except Exception as e:

print(f" Error analyzing column '{col\_name}': {e}")

def visualize\_col(data, column\_name):

plt.figure(figsize=(10, 6))

plt.hist(data, bins=20, color="skyblue", edgecolor="black")

plt.title(f"Distribution of '{column\_name}'", fontsize=16)

plt.xlabel(column\_name, fontsize=12)

plt.ylabel("Frequency", fontsize=12)

plt.grid(True, linestyle="--", alpha=0.7)

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

data\_analyzer()

## Explanation:

This script analyzes a CSV file to calculate the average, maximum, and minimum values of numerical columns. The user specifies the file, and the script computes statistics for the desired column. It handles missing or invalid data gracefully using pandas. The optional Matplotlib integration allows data visualization.

## Tools and Technologies Used:

- Programming Language: Python  
- Libraries:  
 - pandas: For data manipulation and analysis  
 - matplotlib (optional): For data visualization

# Project 4: Web Scraper

## Code:

import requests

from bs4 import BeautifulSoup

import csv

import json

def web\_scraper():

print("\n The DataHarvester \n")

base\_url = input("Enter the URL of the website to scrape: ")

if not base\_url.startswith("http"):

print(" Err: Please enter a valid URL starting with 'http' or 'https'.")

return

selector = input("Enter the CSS selector for the elements to scrape: ")

pagination = input("Does the website have pagination? (y/n): ").lower()

if pagination == "y":

pagination\_param = input("Enter the pagination parameter: ")

all\_data = handle\_pagination(base\_url, selector, pagination\_param)

else:

all\_data = scrape\_single\_page(base\_url, selector)

if not all\_data:

print("No data found to scrape.")

return

save\_format = input("Save data as (CSV/JSON)? ").lower()

if save\_format == "csv":

save\_to\_csv(all\_data)

elif save\_format == "json":

save\_to\_json(all\_data)

else:

print("Invalid format. Data not saved.")

def scrape\_single\_page(url, selector):

try:

response = requests.get(url, timeout=10)

response.raise\_for\_status()

soup = BeautifulSoup(response.text, 'html.parser')

elements = soup.select(selector)

if not elements:

print(" No elements found using the provided selector.")

return []

return [element.get\_text(strip=True) for element in elements]

except requests.exceptions.RequestException as e:

print(f" Network error: {e}")

return []

def handle\_pagination(base\_url, selector, pagination\_param):

page\_number = 1

all\_data = []

while True:

paginated\_url = f"{base\_url}?{pagination\_param}={page\_number}"

print(f"Fetching page {page\_number}...")

try:

response = requests.get(paginated\_url, timeout=10)

response.raise\_for\_status()

soup = BeautifulSoup(response.text, 'html.parser')

elements = soup.select(selector)

if not elements:

print("No more data found. Stopping pagination.")

break

all\_data.extend([element.get\_text(strip=True) for element in elements])

page\_number += 1

except requests.exceptions.RequestException as e:

print(f" Error on page {page\_number}: {e}")

break

return all\_data

def save\_to\_csv(data):

file\_name = input("Enter the filename for the CSV file...: ")

try:

with open(file\_name, mode='w', newline='', encoding='utf-8') as file:

writer = csv.writer(file)

writer.writerow(["Scraped Data"])

for item in data:

writer.writerow([item])

print(f" Data successfully saved to {file\_name}")

except Exception as e:

print(f" Error saving to CSV: {e}")

def save\_to\_json(data):

file\_name = input("Enter the filename for the JSON file (e.g., data.json): ")

try:

with open(file\_name, mode='w', encoding='utf-8') as file:

json.dump(data, file, indent=4, ensure\_ascii=False)

print(f" Data successfully saved to {file\_name}")

except Exception as e:

print(f" Error saving to JSON: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

web\_scraper()

## Explanation:

The Web Scraper fetches data from a website using a specified CSS selector. The user inputs the URL and selector, and the program scrapes the corresponding data. It also supports pagination for scraping multiple pages of data. The scraped content is then saved in either CSV or JSON format, depending on the user's choice.

## Tools and Technologies Used:

- Programming Language: Python  
- Libraries:  
 - requests: For sending HTTP requests and fetching web pages  
 - BeautifulSoup: For parsing HTML and extracting content  
 - csv and json: For saving the data in different formats

## Output Result Screenshots:

(Placeholder: Capture screenshots of the scraping process and saved output file.)