**Challenge 1: Greeting Script**

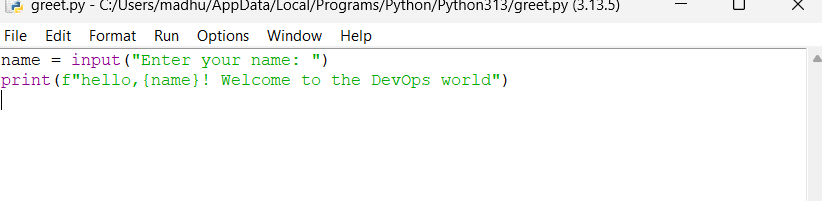
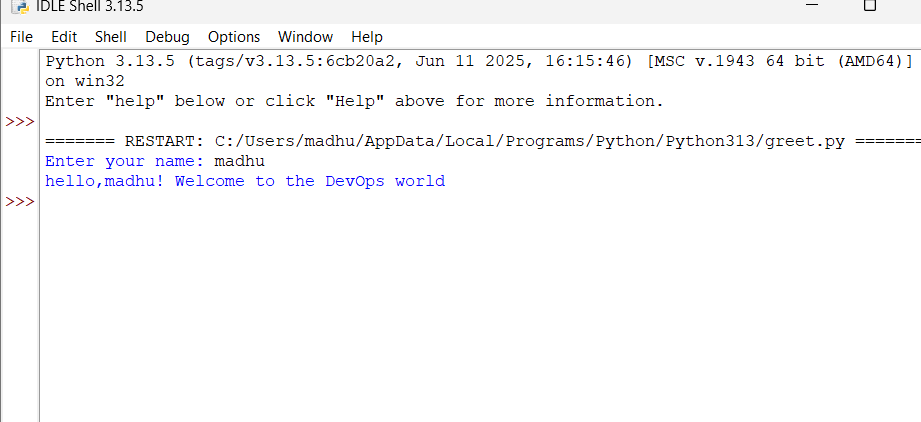
# greet.py

name = input("Enter your name: ")

print(f"Hello, {name}! Welcome to the DevOps world! 🚀")

**Run:**

python greet.py



**🌟 Challenge 2: Word Count in File**

# word\_count.py

file\_name = input("Enter the file name: ")

try:

with open(file\_name, "r") as file:

content = file.read()

word\_count = len(content.split())

print(f"✅ The file '{file\_name}' contains {word\_count} words.")

except FileNotFoundError:

print(f"❌ Error: The file '{file\_name}' was not found.")

except Exception as e:

print(f"❌ An error occurred: {e}")

**Run:**

echo "DevOps is about automation, collaboration, and efficiency." > sample.txt

python word\_count.py

**Bonus:**

file\_name = input("Enter the file name: ")

try:

word\_count = 0

with open(file\_name, "r") as file:

for line in file:

word\_count += len(line.split())

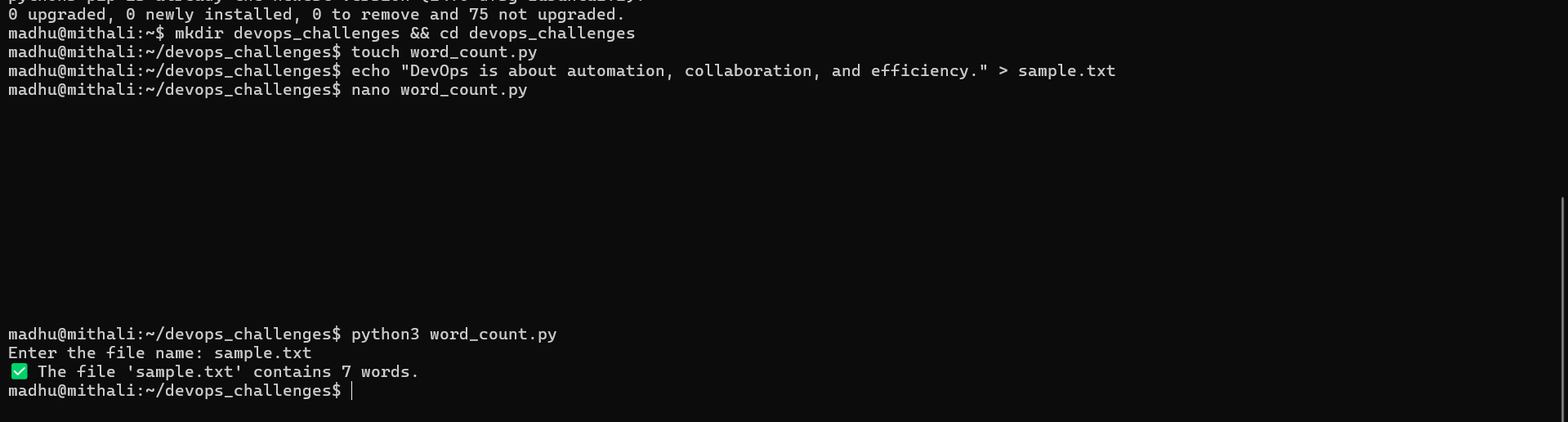
print(f"✅ The file '{file\_name}' contains {word\_count} words.")

except FileNotFoundError:

print(f"❌ Error: The file '{file\_name}' was not found.")

except Exception as e:

print(f"❌ An error occurred: {e}")



**🌟 Challenge 3: Random Password Generator**

# generate\_password.py

import secrets

import string

def generate\_password(length=12):

characters = string.ascii\_letters + string.digits + string.punctuation

password = ''.join(secrets.choice(characters) for \_ in range(length))

return password

print(f"🔐 Your secure password: {generate\_password()}")

**Bonus:**

def generate\_password(length):

if length < 8:

print("❌ Password length should be at least 8 characters for security.")

return None

characters = string.ascii\_letters + string.digits + string.punctuation

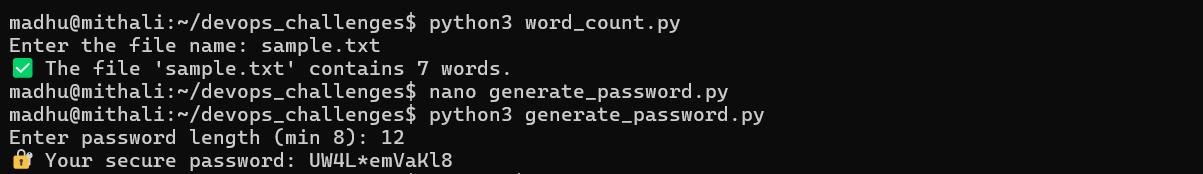
return ''.join(secrets.choice(characters) for \_ in range(length))

length = int(input("Enter password length (min 8): "))

password = generate\_password(length)

if password:

print(f"🔐 Your secure password: {password}")



**🌟 Challenge 4: Prime Number Checker**

# check\_prime.py

def is\_prime(n):

if n < 2:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

try:

num = int(input("Enter a number: "))

if is\_prime(num):

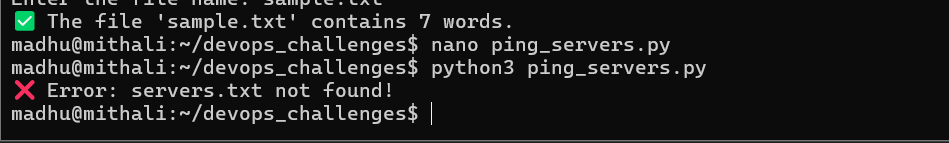
print(f"✅ {num} is a prime number.")

else:

print(f"❌ {num} is not a prime number.")

except ValueError:

print("❌ Please enter a valid integer.")



**🌟 Challenge 5: Ping Servers from File**

# ping\_servers.py

import os

import platform

def ping\_server(server):

param = "-n 1" if platform.system().lower() == "windows" else "-c 1"

response = os.system(f"ping {param} {server} > /dev/null 2>&1")

return response == 0

try:

with open("servers.txt", "r") as file:

servers = [line.strip() for line in file.readlines()]

for server in servers:

if server:

status = "✅ Reachable" if ping\_server(server) else "❌ Unreachable"

print(f"{server}: {status}")

except FileNotFoundError:

print("❌ Error: servers.txt not found!")

except Exception as e:

print(f"❌ An error occurred: {e}")

**Bonus - Log Results:**

with open("ping\_results.log", "w") as log\_file:

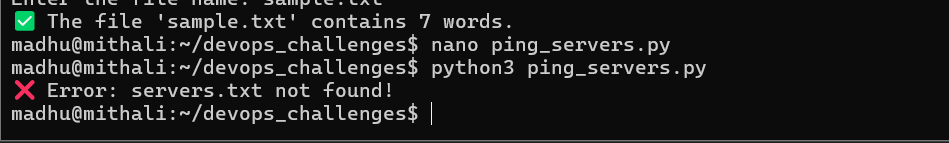
for server in servers:

if server:

status = "✅ Reachable" if ping\_server(server) else "❌ Unreachable"

log\_file.write(f"{server}: {status}\n")

print(f"{server}: {status}")



**🌟 Challenge 6: CRUD Operations with Requests**

# crud\_call.py

import requests

API\_URL = "https://jsonplaceholder.typicode.com/posts"

# Define CRUD functions...

# get\_post, create\_post, update\_post, delete\_post as described earlier

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

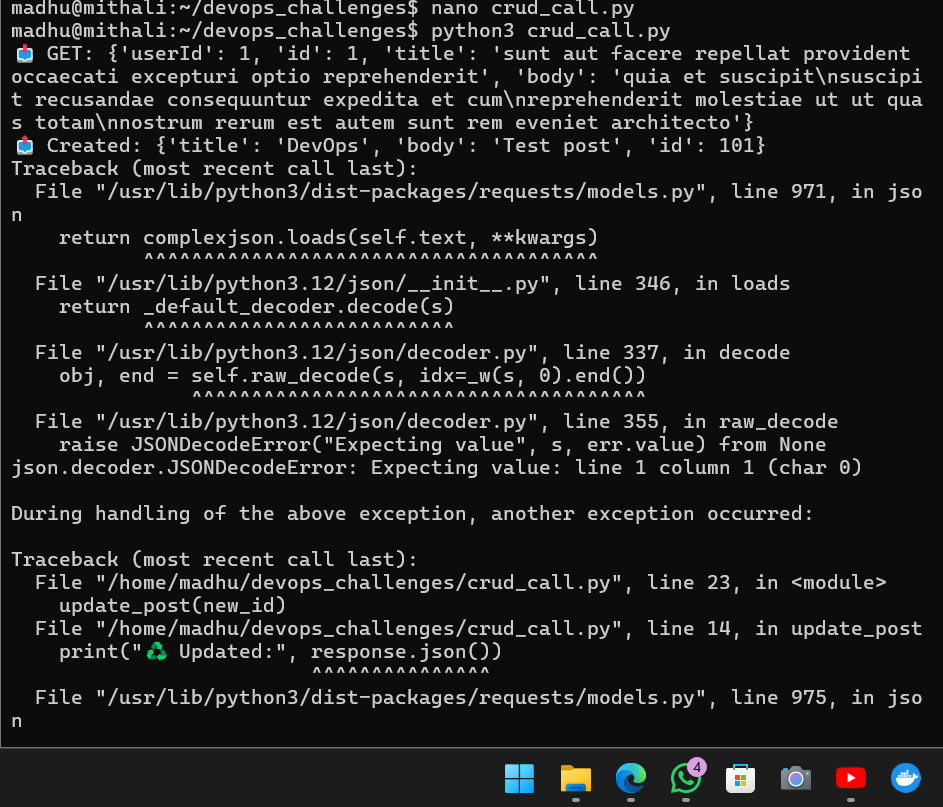
get\_post(1)

new\_post\_id = create\_post()

if new\_post\_id:

update\_post(new\_post\_id)

delete\_post(new\_post\_id)



**🌟 Challenge 7: Bulk Rename Files**

# rename\_files.py

import os

directory = "test\_files"

prefix = "renamed\_"

if not os.path.exists(directory):

print(f"❌ Error: Directory '{directory}' not found!")

exit(1)

for count, filename in enumerate(os.listdir(directory), start=1):

old\_path = os.path.join(directory, filename)

if os.path.isfile(old\_path):

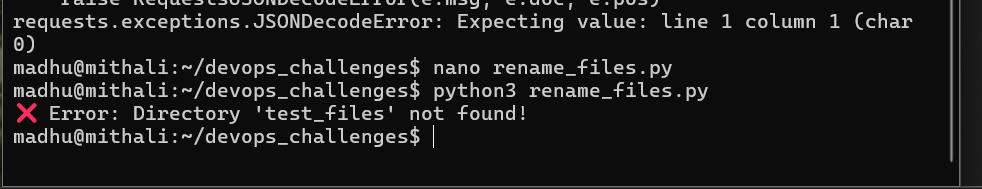
new\_name = f"{prefix}{count}.txt"

new\_path = os.path.join(directory, new\_name)

os.rename(old\_path, new\_path)

print(f"✅ Renamed: {filename} → {new\_name}")

print("🎉 Renaming completed!")



**🌟 Challenge 8: Monitor CPU & Memory**

# monitor\_resources.py

import psutil

import time

def monitor\_resources(interval=5):

print("📊 Monitoring CPU and Memory usage... (Press Ctrl+C to stop)\n")

try:

while True:

cpu\_usage = psutil.cpu\_percent(interval=1)

memory\_info = psutil.virtual\_memory()

memory\_usage = memory\_info.percent

print(f"🖥️ CPU Usage: {cpu\_usage}% | 🧠 Memory Usage: {memory\_usage}%")

time.sleep(interval - 1)

except KeyboardInterrupt:

print("\n❌ Monitoring stopped.")

**Bonus - Log to File:**

with open("resource\_usage.log", "a") as log\_file:

while True:

... # Write log entries with timestamps

**🌟 Challenge 9: Create Linux User with Subprocess**

# create\_user.py

import subprocess

def create\_user(username):

try:

subprocess.run(["id", username], check=True, stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)

print(f"✅ User '{username}' already exists.")

return

except subprocess.CalledProcessError:

pass

try:

subprocess.run(["sudo", "useradd", "-m", "-s", "/bin/bash", username], check=True)

print(f"✅ User '{username}' created successfully.")

except subprocess.CalledProcessError:

print(f"❌ Failed to create user '{username}'. Ensure you have sudo privileges.")

return

try:

subprocess.run(["id", username], check=True)

print(f"✅ Verification successful: '{username}' exists in the system.")

except subprocess.CalledProcessError:

print(f"❌ Verification failed: '{username}' does not exist.")

username = input("Enter the username to create: ")

create\_user(username)

