TECHNIK NEST

INNOVATIVE MINDS, NESTING SUCCESS

Name: Hamza Badshah

Intern ID: TN/IN01/PY/001

Email ID : hamzabadshah2592@gmail.com

Task week: 06

Internship Domain: Python Development

Instructor Name: Mr. Hassan Ali

Tasks – Libraries and pip

Task 01

Use math & statistics libraries to get square roots and average.

Step-by-Step Instructions

- math.sqrt(x) returns the square root of x
- statistics.mean(list) returns the average (mean) of the list

Code Snippet

```python
import math
import statistics

Number = 49
SquareRoot = math.sqrt(Number)
print ("\n")
print(f"Square root of {Number} = {SquareRoot}")

NumbersList = [10, 20, 30, 40, 50]
Average = statistics.mean(NumbersList)
print(f"Average of {NumbersList} = {Average}")
```

Output Snippet

```
PS D:\Python Internship Tasks\week_06_tasks_with report> & "C:/Program Files/Python313/python3.13t.exe" "d:/Python Internship Tasks/week_06_tasks_with report/math_statistics (find_squareroot_average)/math_statistics_demo.py"


Square root of 49 = 7.0
Average of [10, 20, 30, 40, 50] = 30
PS D:\Python Internship Tasks\week_06_tasks_with report>
```

Learning and Challenges

- Learned how to use built-in Python modules.
- Faced a challenge using statistics.mean() on an empty list (throws error).

Solution: Make sure the list isn't empty before calling mean().

Task 02

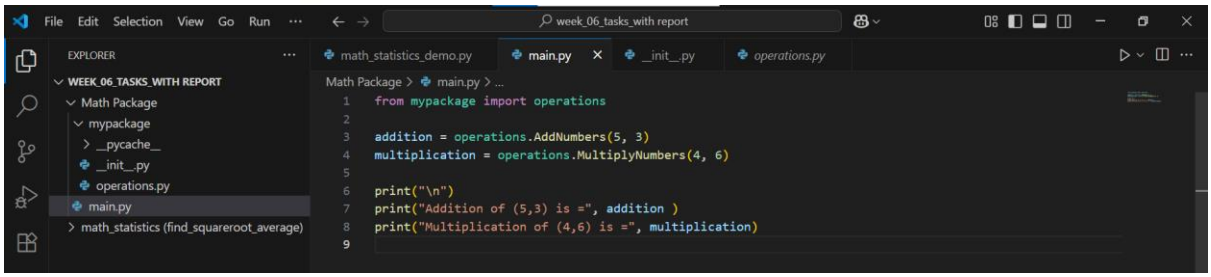
Create a custom package and import it in another script.

Step-by-Step Instructions

Folder Structure

```
my_package_example
│
├── mymath                ← Your custom package
│   ├── __init__.py       ← Makes it a package
│   └── operations.py     ← Module with your custom functions
│
└── main.py               ← Script where you import and use the package
```

Code Snippet



Output Snippet

```
TERMINAL
PS D:\Python Internship Tasks\week_06_tasks_with report> & "C:/Program Files/Python313/python3.13t.exe" "d:/Python Interns
hip Tasks/week_06_tasks_with report/Math Package/main.py"


Addition of (5,3) is = 8
Multiplication of (4,6) is = 24
PS D:\Python Internship Tasks\week_06_tasks_with report>
```

Learning and Challenges

- Learned how to create a custom Python package with __init__.py.
- Understood how to organize code using modules and packages.
- Practiced importing functions from a custom package into another script.
- Faced ImportError due to missing function references in __init__.py.
- Initially ran the script from the wrong directory, which caused module errors.
- Encountered issues with folder names containing spaces (e.g., "Week 06 tasks with report").

Tasks – Virtual Environments

Task 03

Create a virtual environment, install requests & numpy, and print their versions.

Step-by-Step Instructions

Create a virtual environment, install requests and numpy, and print their versions

1. **Create a Virtual Environment**

Open your terminal (CMD or PowerShell) and run:

python -m venv myenv

This creates a folder named myenv with your isolated environment.

2. Activate the Virtual Environment

Windows (CMD/PowerShell):

myenv\Scripts\activate

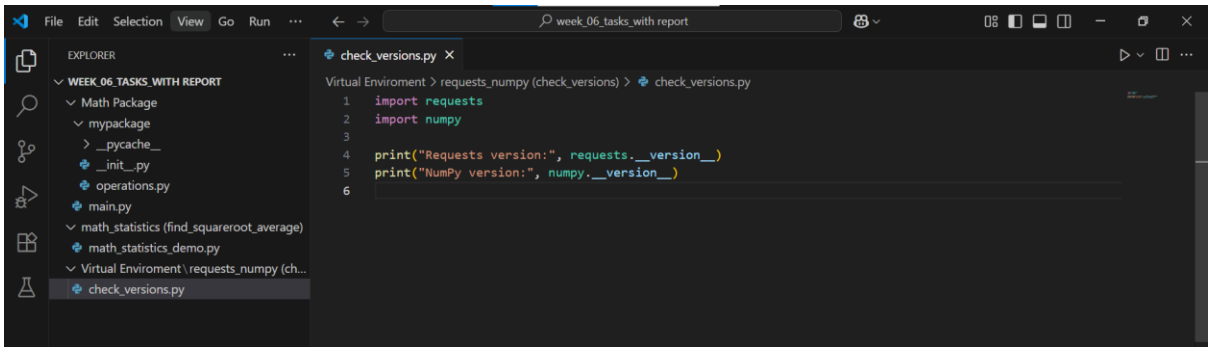
You should now see (myenv) before the command line prompt.

2. Install requests and numpy

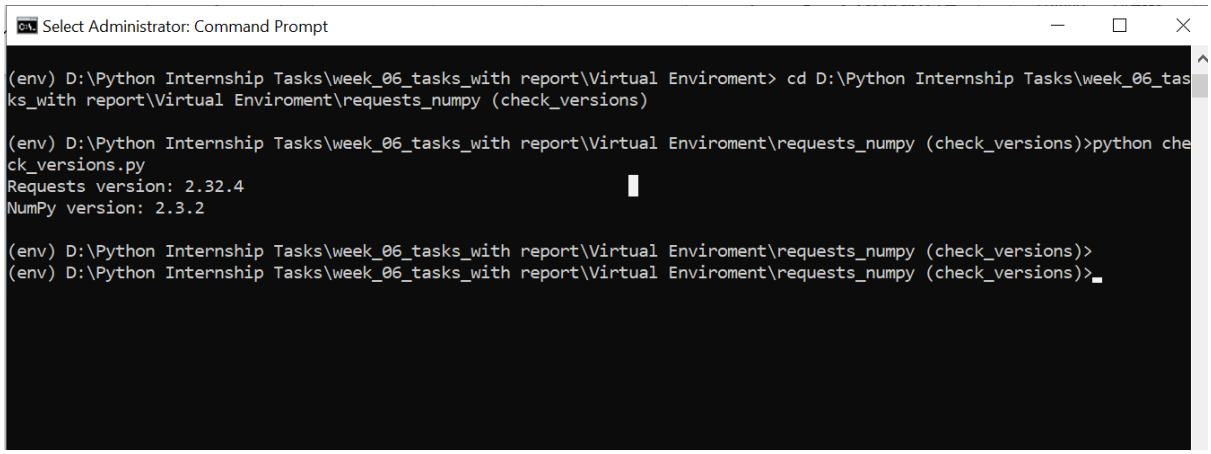
While the environment is active, run:

pip install requests numpy

Code Snippet



Output Snippet



Learning and Challenges

- Learned how to isolate Python environments using venv.
- Practiced package installation with pip.
- Discovered how to check package versions using __version__.
- Faced issue on Windows with activation solved by running terminal as administrator.

Tasks – Gradio

Task 04

Print list of all installed pip packages from Python code.

Step-by-Step Instructions

Activate Virtual Environment

- If your virtual environment is already created (e.g., myenv), activate it:

myenv\Scripts\activate

Prompt changes to:

(myenv) C:\>

Create Python Script

- Create a new file named list_installed.py and paste the code above into it. Save it inside your working folder.

Install Required Module

- If running the script gives the error:
- **ModuleNotFoundError:** No module named 'pkg_resources'
- Then install setuptools inside the virtual environment:
- pip install setuptools
- This provides access to the pkg_resources module.
- Run the Script

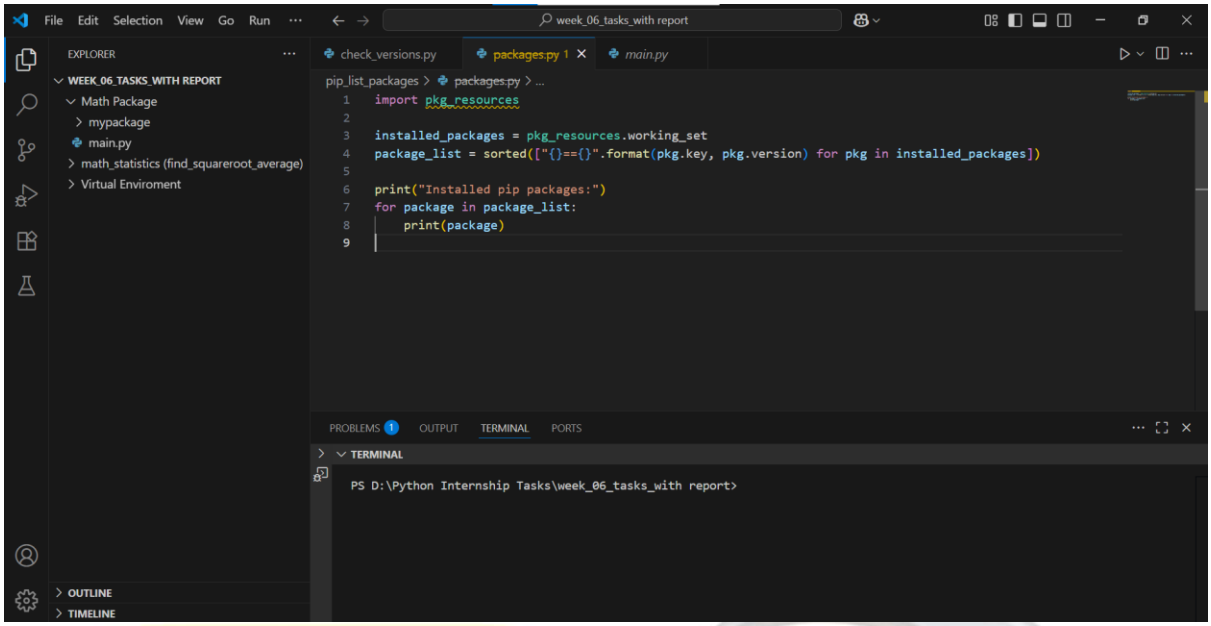
Navigate to the script's location:

cd "D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report"

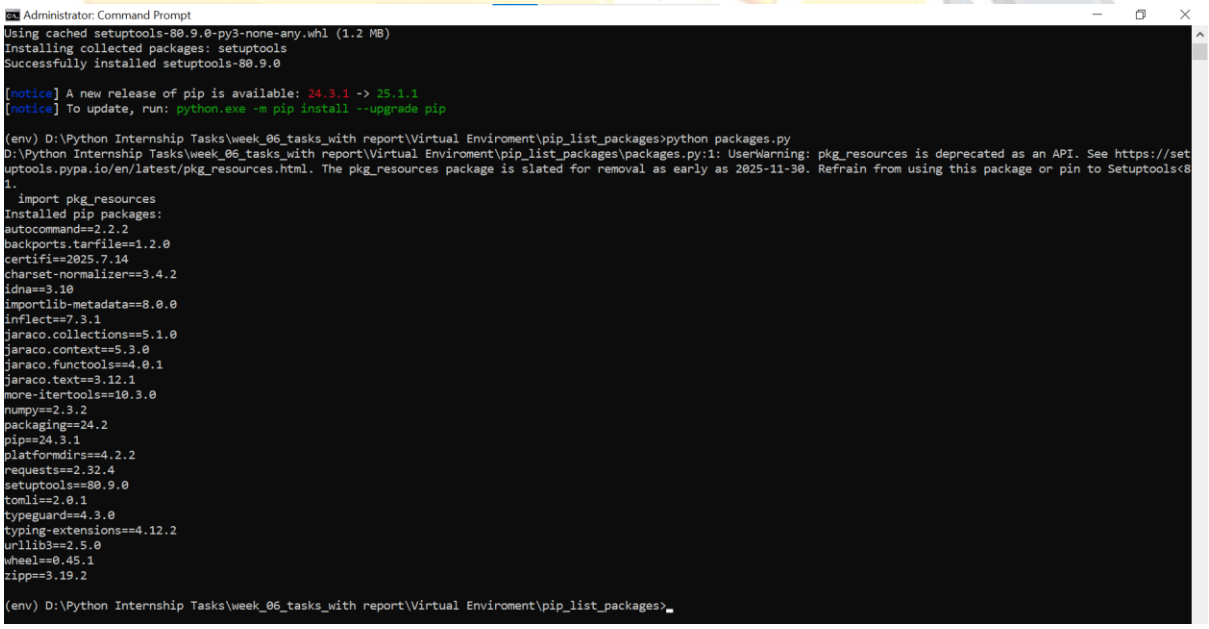
Then run:

python list_installed.py

Code Snippet



Output Screenshot



Learning and Challenges

- How to inspect installed pip packages inside a virtual environment using Python code
- pkg_resources module was not found
- Ran the script from the wrong directory
- Forgot to activate the virtual environment before running the script
- pkg_resources provides access to all installed packages

- Importance of setuptools in Python environments
- Managing and troubleshooting Python virtual environments
- Navigating between folders and using Python from the command line

Task 05

Create Gradio app that takes a number and returns its square.

Step-by-Step Instructions

- Install Gradio (if not installed)

  pip install gradio

- Save the code in a file

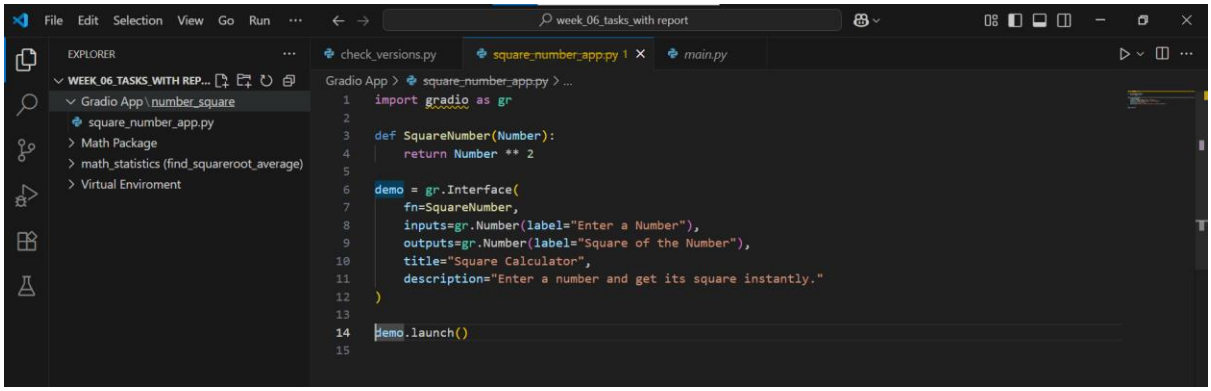
  Name the file square_numbers_app.py

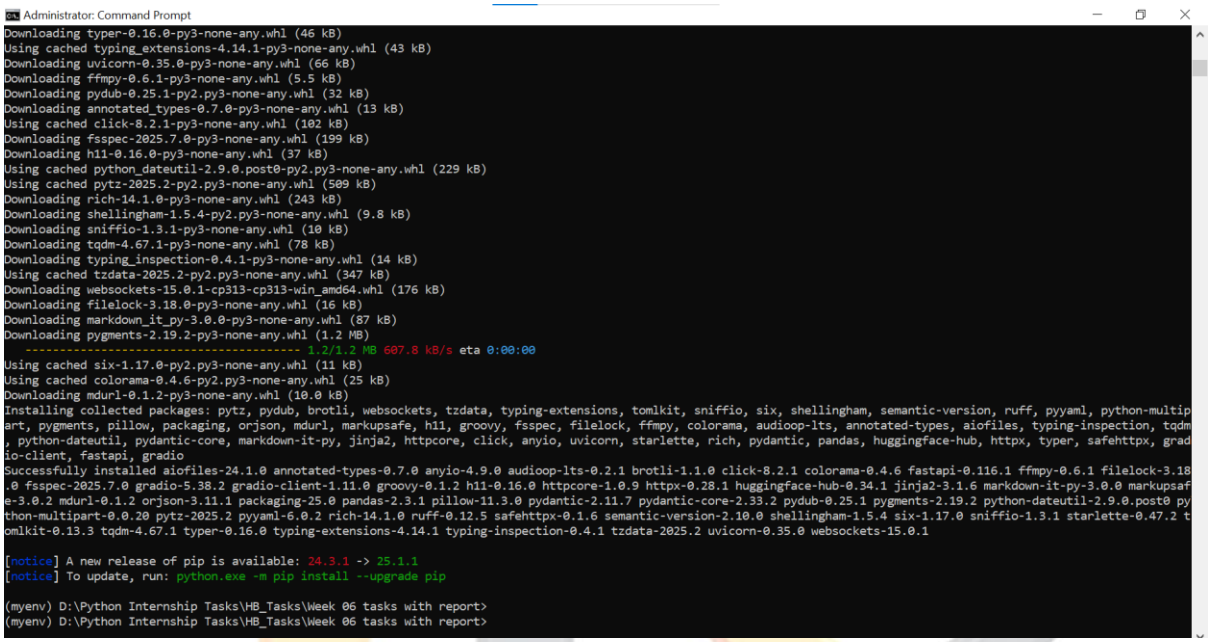
- Run the file in terminal

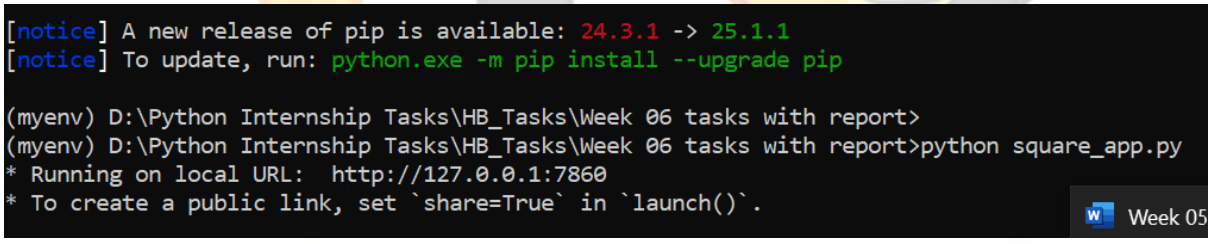
  python square_numbers_app.py

- A web browser will open with a simple UI.

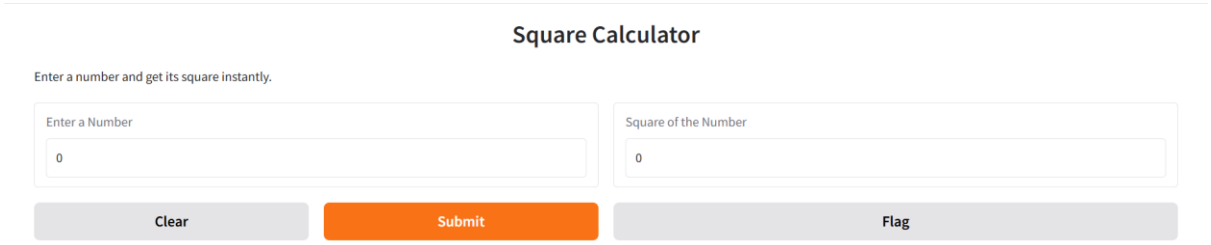
Code Snippet







Output Snippet



Learning and Challenges

- Understood how to use the Gradio library to build a quick web-based interface.
- Learned how to define a Python function that takes a number input and returns its square.
- Implemented gr.Interface() to connect the function with the user interface.
- First time using Gradio
- Entered string instead of a number (caused error)
- App didn't auto-open in browser

Task 06

Create Gradio interface that takes a sentence and returns it reversed.

Step-by-Step Instructions

1. Save the code in a file called reverse_sentence_app.py.
2. Open your terminal or VS Code terminal.
3. Navigate to the folder where your file is saved.
4. Activate your virtual environment:

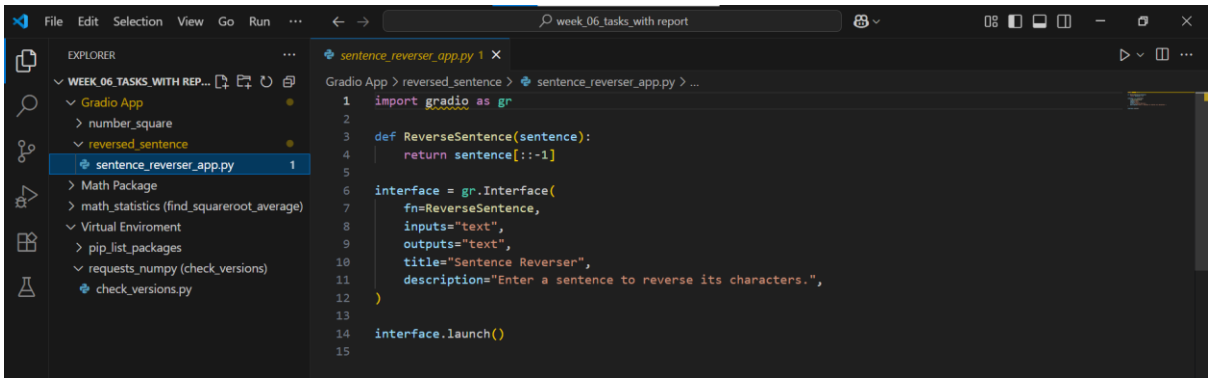
   myenv\Scripts\activate

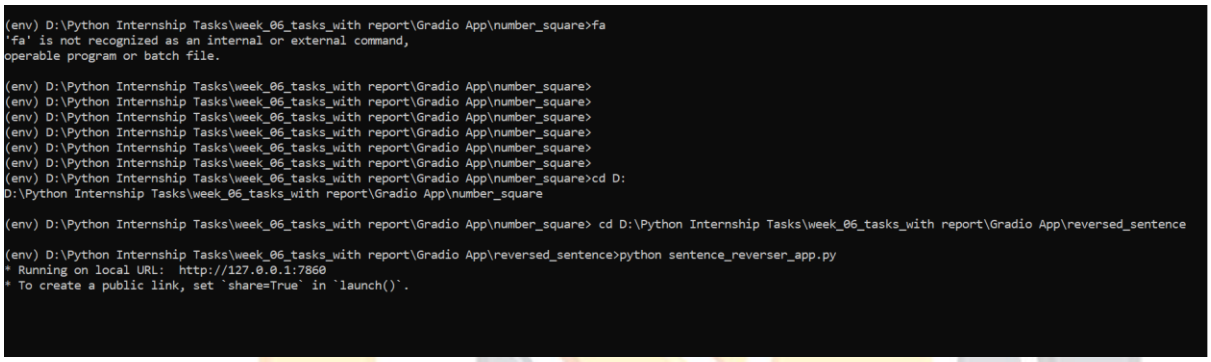
5. Run the script:

   python reverse_sentence_app.py

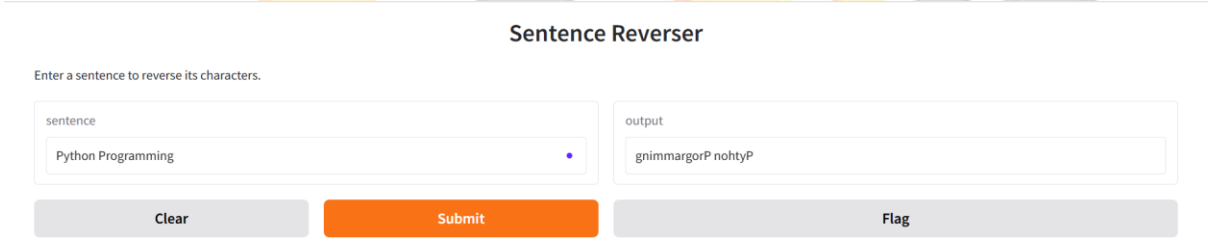
6. Open the browser and go to**: http://127.0.0.1:7860**

Code Snippet





Output Snippet



Learning and Challenges

- Learned how to use Gradio to quickly build a web interface for a Python function.
- Practiced using string slicing ([::-1]) to reverse a sentence.

- Understood how Gradio maps function inputs and outputs to a web interface.
- Learned how to run Gradio apps locally in a virtual environment.
- Gradio module not found.
- Forgetting to activate virtual environment.

Conclusion

In this task, you learned how to create and manage a virtual environment, install and inspect packages (requests, numpy), and handle common issues like missing modules or incorrect paths.

You also built a simple Gradio app to take input and display results interactively. Overall, this gave you practical experience in environment management, basic debugging, and developing user-friendly Python applications.