



# TECHNIK NEST

INNOVATIVE MINDS, NESTING SUCCESS

**Name:** Musfira Ahmed

**Intern ID:** TN/IN01/PY/002

**Email ID :** [ahmedmusfira3@gmail.com](mailto:ahmedmusfira3@gmail.com)

**Task week:** 06

**Internship Domain:** Python Development

**Instructor Name:** Mr. Hassan Ali

TECHNIK NEST

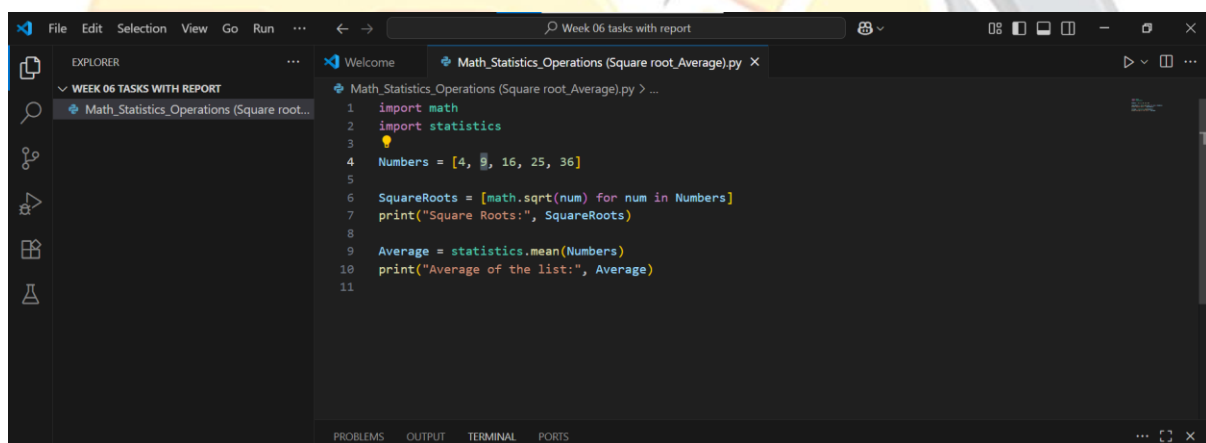
## Task 01

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

### What I Did (Step by Step)

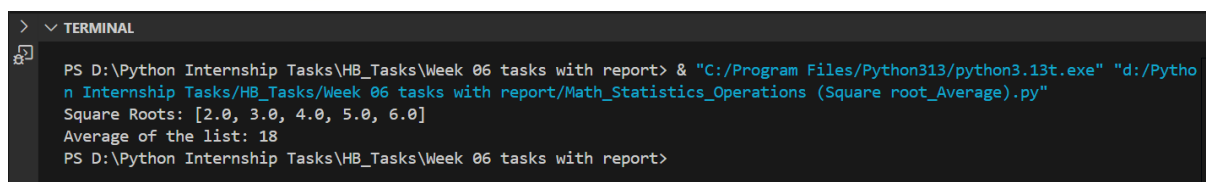
- Imported the math library to calculate square roots.
- Imported the statistics library to calculate the average of a list.
- Used user-defined data to demonstrate both functions.

### Code Screenshot



```
1 import math
2 import statistics
3
4 Numbers = [4, 9, 16, 25, 36]
5
6 SquareRoots = [math.sqrt(num) for num in Numbers]
7 print("Square Roots:", SquareRoots)
8
9 Average = statistics.mean(Numbers)
10 print("Average of the list:", Average)
11
```

### Output Screenshot



```
> TERMINAL
PS D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report> & "C:/Program Files/Python313/python3.13t.exe" "d:/Python Internship Tasks/HB_Tasks/Week 06 tasks with report/Math_Statistics_Operations (Square root_Average).py"
Square Roots: [2.0, 3.0, 4.0, 5.0, 6.0]
Average of the list: 18
PS D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>
```

### Learning and Challenges

- Learned how to use built-in Python libraries like math and statistics.

- Faced Challenge when applying square root to a list solved it using a list comprehension (**[`math.sqrt(num)` for `num` in `list`]**).
  - Understood how to calculate statistical data from a list of numbers
- 

## Task 02

Create a custom package and import it in another script.

### What I Did (Step by Step)

#### 1. Create the Package Folder

Create a folder named mypackage.

#### 2. Create `__init__.py`

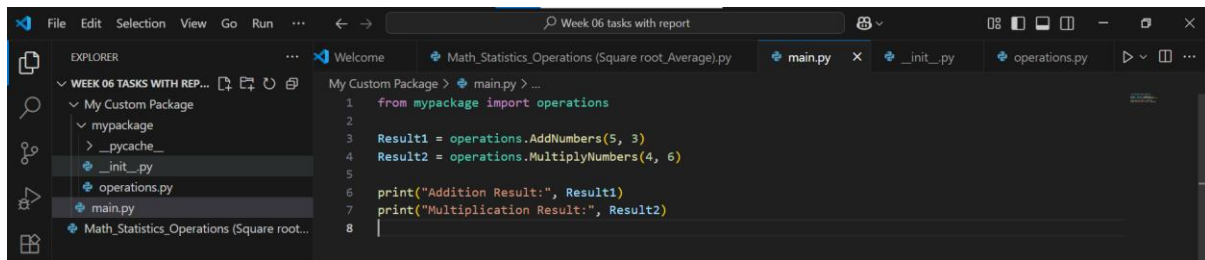
Inside mypackage/, create an empty `__init__.py` file (this marks it as a Python package).

#### 3. Create `operations.py`

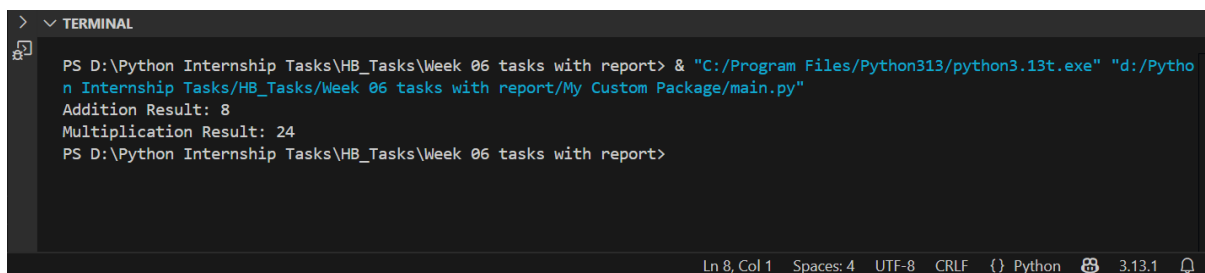
This will contain your custom functions.

**mypackage/operations.py:**

## Code Screenshot



## Output Screenshot



## Learning and Challenges

- Learned how to modularize Python code using packages.
- Understood how `__init__.py` is required to recognize folders as packages.
- Faced issue with import paths solved by ensuring folder structure is correct and script is run from the root directory.

## Task 03

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

## What I Did (Step by Step)

## Step 1: Create a Virtual Environment

Open terminal or command prompt:

```
python -m venv myenv
```

This creates a folder myenv/ containing the virtual environment.

## Step 2: Activate the Virtual Environment

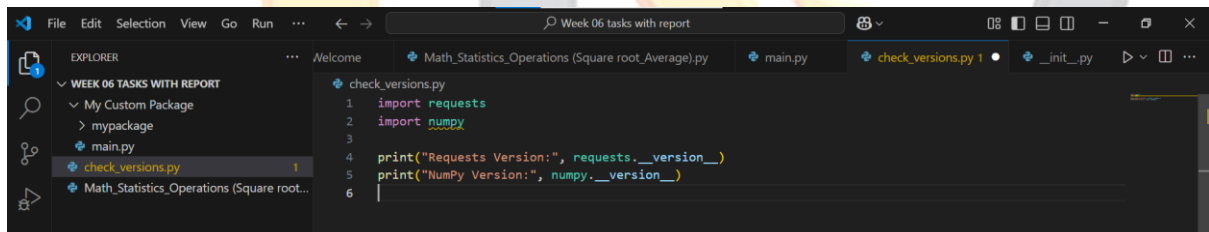
- On Windows:

```
myenv\Scripts\activate
```

## Step 3: Install Packages

```
pip install requests numpy
```

## Code Screenshot



```
File Edit Selection View Go Run ... < -> Week 06 tasks with report
EXPLORER
WEEK 06 TASKS WITH REPORT
  My Custom Package
    mypackage
    main.py
    check_versions.py
    Math_Statistics_Operations (Square root...)
  check_versions.py
  Math_Statistics_Operations (Square root...)

check_versions.py
1 import requests
2 import numpy
3
4 print("Requests Version:", requests.__version__)
5 print("NumPy Version:", numpy.__version__)
6
```

## Output Screenshot

```
Select Administrator: Command Prompt

(myenv) C:\Windows\system32>pip install requests numpy
Collecting requests
  Using cached requests-2.32.4-py3-none-any.whl.metadata (4.9 kB)
Collecting numpy
  Downloading numpy-2.3.2-cp313-cp313-win_amd64.whl.metadata (60 kB)
Collecting charset_normalizer<4,>=2 (from requests)
  Using cached charset_normalizer-3.4.2-cp313-cp313-win_amd64.whl.metadata (36 kB)
Collecting idna<4,>=2.5 (from requests)
  Using cached idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Using cached urllib3-2.5.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Using cached certifi-2025.7.14-py3-none-any.whl.metadata (2.4 kB)
Using cached requests-2.32.4-py3-none-any.whl (64 kB)
Downloading numpy-2.3.2-cp313-cp313-win_amd64.whl (12.8 MB)
----- 12.8/12.8 MB 432.8 kB/s eta 0:00:00
Using cached certifi-2025.7.14-py3-none-any.whl (162 kB)
Using cached charset_normalizer-3.4.2-cp313-cp313-win_amd64.whl (105 kB)
Using cached idna-3.10-py3-none-any.whl (70 kB)
Using cached urllib3-2.5.0-py3-none-any.whl (129 kB)
Installing collected packages: urllib3, numpy, idna, charset_normalizer, certifi, requests
Successfully installed certifi-2025.7.14 charset_normalizer-3.4.2 idna-3.10 numpy-2.3.2 requests-2.32.4 urllib3-2.5.0

[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(myenv) C:\Windows\system32>
```

```
Administrator: Command Prompt

(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>python check_versions.py
Requests version: 2.32.4
NumPy version: 2.3.2

(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>
(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>
```

## 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.

## Task 04

Print list of all installed pip packages from Python code.

### What I Did (Step by Step)

#### Step 1: Activate Virtual Environment

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

```
myenv\Scripts\activate
```

Prompt changes to:

```
(myenv) C:\>
```

#### Step 2: Create Python Script

Create a new file named **list\_installed.py** and paste the code above into it. Save it inside your working folder.

#### Step 3: 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.

## Step 4: Run the Script

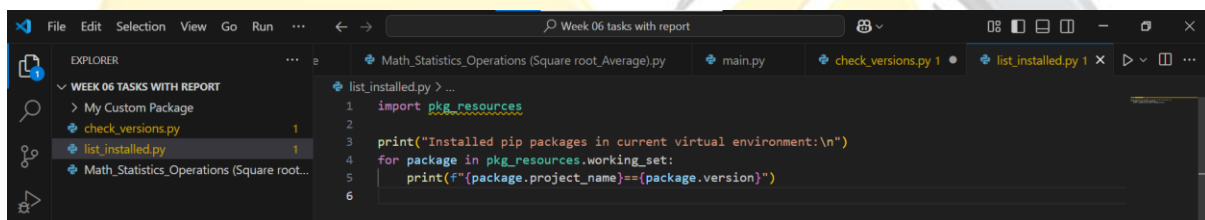
Navigate to the script's location:

**cd "D:\Python Internship Tasks\HB\_Tasks\Week 06 tasks with report"**

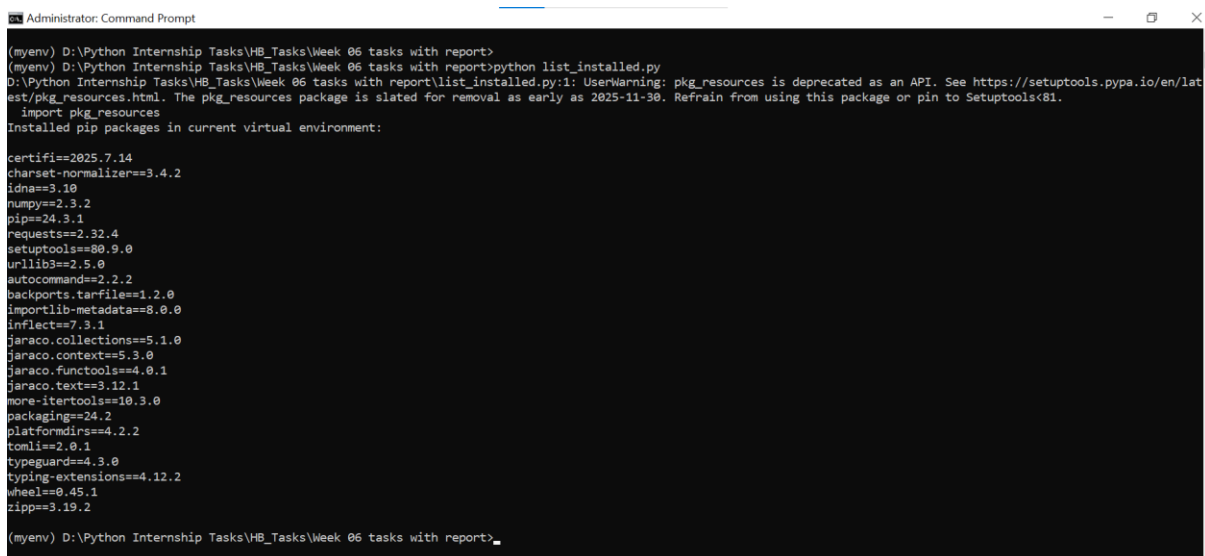
Then run:

**python list\_installed.py**

## Code Screenshot



## 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.

### What I Did (Step by Step)

1. Install Gradio (if not installed)

**pip install gradio**

2. Save the code in a file

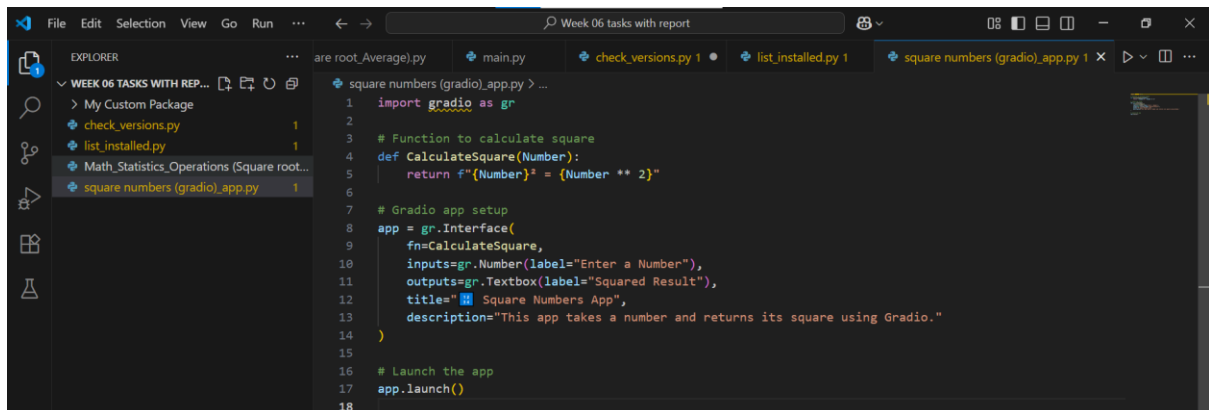
**Name the file `square_numbers_app.py`**

3. Run the file in terminal

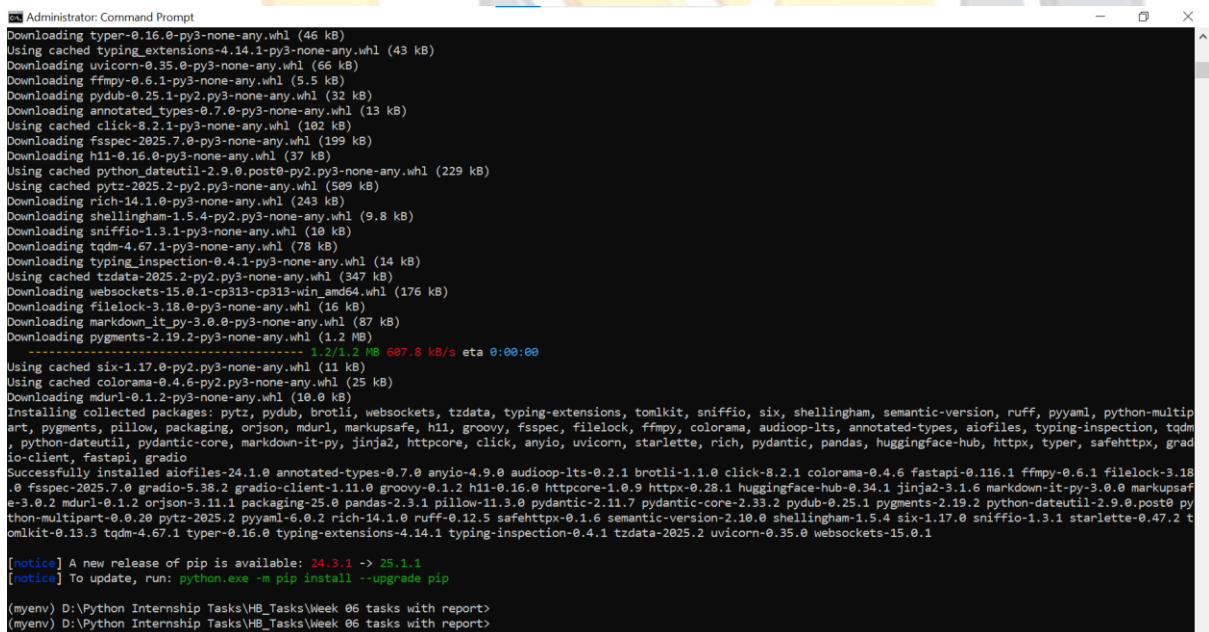
## python square\_numbers\_app.py

4. A web browser will open with a simple UI.

### Code Screenshot



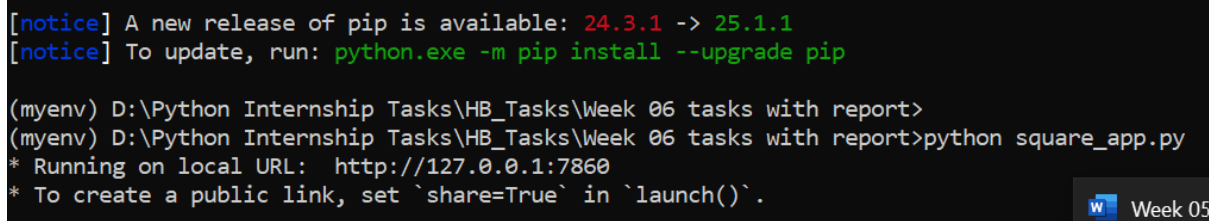
```
1 import gradio as gr
2
3 # Function to calculate square
4 def CalculateSquare(Number):
5     return f"{Number}2 = {Number ** 2}"
6
7 # Gradio app setup
8 app = gr.Interface(
9     fn=CalculateSquare,
10    inputs=gr.Number(label="Enter a Number"),
11    outputs=gr.Textbox(label="Squared Result"),
12    title="Square Numbers App",
13    description="This app takes a number and returns its square using Gradio."
14)
15
16 # Launch the app
17 app.launch()
18
```



```
Administrator: Command Prompt
Downloading typer-0.16.0-py3-none-any.whl (46 kB)
Using cached typing_extensions-4.14.1-py3-none-any.whl (43 kB)
Downloading uvicorn-0.35.0-py3-none-any.whl (66 kB)
Downloading ffmpeg-0.6.1-py3-none-any.whl (5.5 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Downloading annotated_types-0.7.0-py3-none-any.whl (13 kB)
Using cached click-8.2.1-py3-none-any.whl (102 kB)
Downloading fsspec-2025.7.0-py3-none-any.whl (199 kB)
Downloading h11-0.16.0-py3-none-any.whl (37 kB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Using cached pytz-2025.2-py2.py3-none-any.whl (509 kB)
Downloading rich-14.1.0-py3-none-any.whl (243 kB)
Downloading shellingham-1.5.4-py2.py3-none-any.whl (9.8 kB)
Downloading sniffio-1.3.1-py3-none-any.whl (10 kB)
Downloading tqdm-4.67.1-py3-none-any.whl (78 kB)
Downloading typing_inspection-0.4.1-py3-none-any.whl (14 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Downloading websockets-15.0.1-cp313-cp313-win_amd64.whl (176 kB)
Downloading filelock-3.18.0-py3-none-any.whl (16 kB)
Downloading markdown_it_py-3.0.0-py3-none-any.whl (87 kB)
Downloading pygments-2.19.2-py3-none-any.whl (1.2 MB)
----- 1.2/1.2 MB 607.8 kB/s eta 0:00:00
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: pytz, pydub, brotli, websockets, tzdata, typing-extensions, tomlkit, sniffio, six, shellingham, semantic-version, ruff, pyyaml, python-multipart, pygments, pillow, packaging, orjson, mdurl, markupsafe, h11, groovy, fsspec, filelock, ffmpeg, colorama, audiop-lts, annotated-types, aiofiles, typing-inspection, tqdm, python-dateutil, pydantic-core, markdown-it-py, Jinja2, httpcore, click, anyio, uvicorn, starlette, rich, pydantic, pandas, huggingface-hub, httpx, typer, safehttpx, gradio-client, fastapi, gradio
Successfully installed aiofiles-24.1.0 annotated-types-0.7.0 anyio-4.9.0 audiop-lts-0.2.1 brotli-1.1.0 click-8.2.1 colorama-0.4.6 fastapi-0.116.1 ffmpeg-0.6.1 filelock-3.18.0 fsspec-2025.7.0 gradio-5.38.2 gradio-client-1.11.0 groovy-0.1.2 h11-0.16.0 httpcore-1.0.9 httpx-0.28.1 huggingface-hub-0.34.1 Jinja2-3.1.6 markdown-it-py-3.0.0 markupsafe-3.0.2 mdurl-0.1.2 orjson-3.11.1 packaging-25.0 pandas-2.3.1 pillow-11.3.0 pydantic-2.11.7 pydantic-core-2.33.2 pydub-0.25.1 pygments-2.19.2 python-dateutil-2.9.0.post0 python-multipart-0.0.20 pytz-2025.2 pyyaml-6.0.2 rich-14.1.0 ruff-0.12.5 safehttpx-0.1.6 semantic-version-2.10.0 shellingham-1.5.4 six-1.17.0 sniffio-1.3.1 starlette-0.47.2 tomlkit-0.13.3 tqdm-4.67.1 typer-0.16.0 typing-extensions-4.14.1 typing-inspection-0.4.1 tzdata-2025.2 uvicorn-0.35.0 websockets-15.0.1

[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>
(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>
```




```
[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>
(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>python square_app.py
* Running on local URL: http://127.0.0.1:7860
* To create a public link, set `share=True` in `launch()`.

Week 05
```

## Output Screenshot

---

 **Square Numbers App**

This app takes a number and returns its square using Gradio.

<div>Enter a Number</div> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;">4</div>	<div>Squared Result</div> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"><math>4^2 = 16</math></div>
---	---

Clear

Submit

Flag

## 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.

## What I Did (Step by Step)

### How to Run:

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:

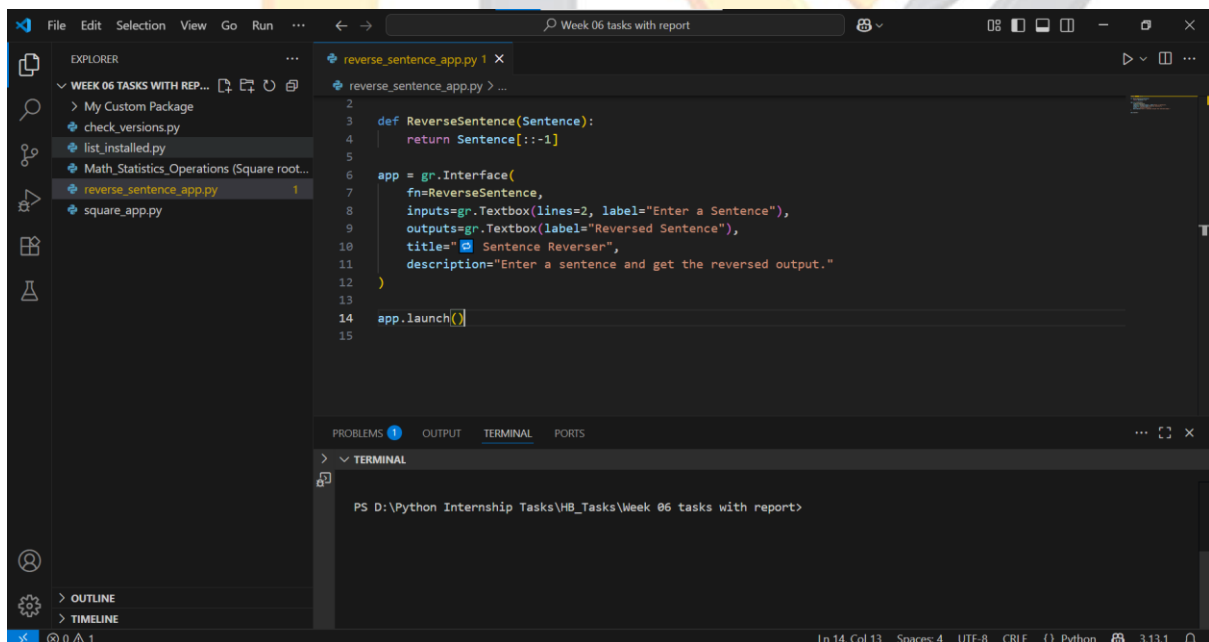
**myenv\Scripts\activate    # For Windows**

5. Run the script:

**python reverse\_sentence\_app.py**

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

## Code Screenshot



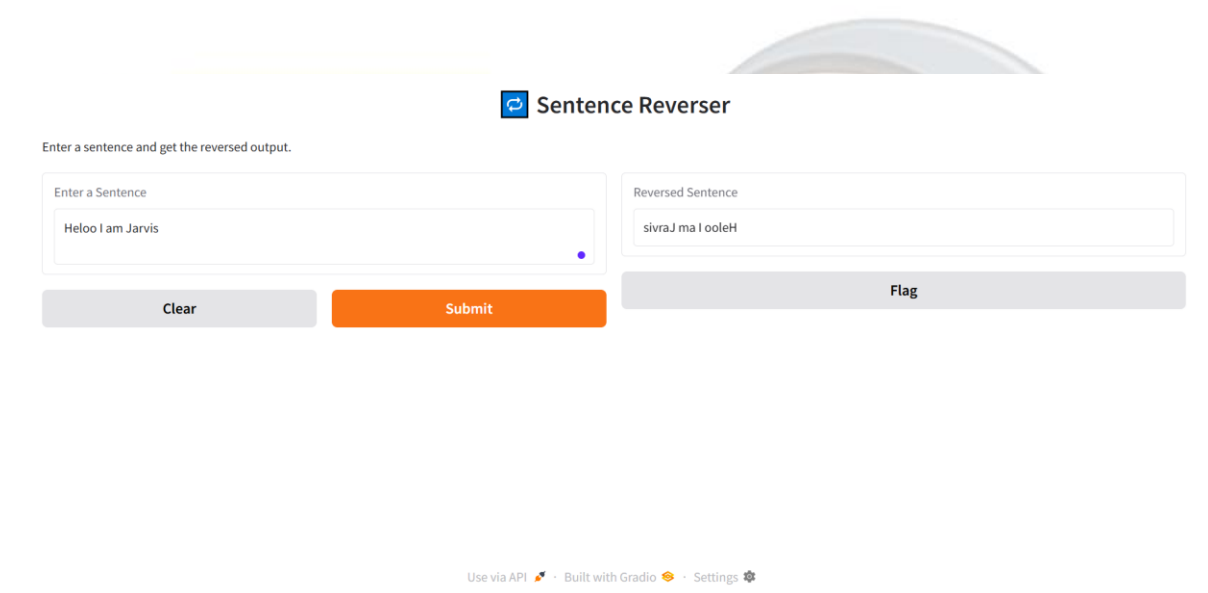
## Output Screenshot

```
Administrator: Command Prompt - python reverse_sentence_app.py

(myenv) C:\Windows\system32>D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report\myenv\Scripts\python.exe
'D:\Python' is not recognized as an internal or external command,
operable program or batch file.

(myenv) C:\Windows\system32>:

(myenv) D:\Python Internship Tasks\HB_Tasks\Week 06 tasks with report>python reverse_sentence_app.py
* Running on local URL:  http://127.0.0.1:7860
* To create a public link, set `share=True` in `launch()`.
```



## 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.
- App not loading at **127.0.0.1**.

