



# TECHNIK NEST

INNOVATIVE MINDS, NESTING SUCCESS

**Name:** Hamza Badshah

**Intern ID:** TN/IN01/PY/001

**Email ID :** [hamzabadshah2592@gmail.com](mailto:hamzabadshah2592@gmail.com)

**Task week:** 06

**Internship Domain:** Python Development

**Instructor Name:** Mr. Hassan Ali

# TECHNIK NEST

## Tasks – Libraries and pip

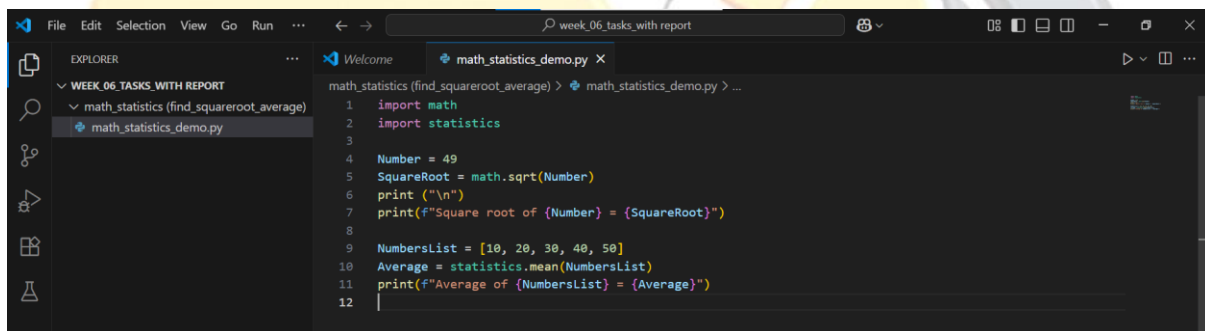
### Task 01

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

### Step-by-Step Instructions

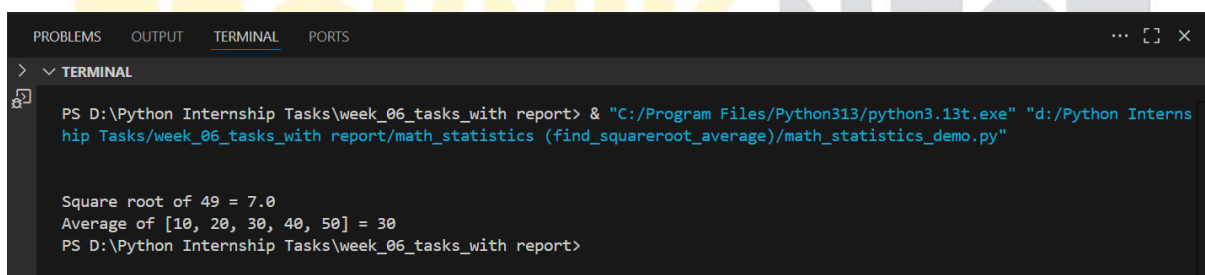
- `math.sqrt(x)` returns the square root of x
- `statistics.mean(list)` returns the average (mean) of the list

### Code Snippet



```
1 import math
2 import statistics
3
4 Number = 49
5 SquareRoot = math.sqrt(Number)
6 print("\n")
7 print(f"Square root of {Number} = {SquareRoot}")
8
9 NumbersList = [10, 20, 30, 40, 50]
10 Average = statistics.mean(NumbersList)
11 print(f"Average of {NumbersList} = {Average}")
12
```

### Output Snippet



```
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_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.

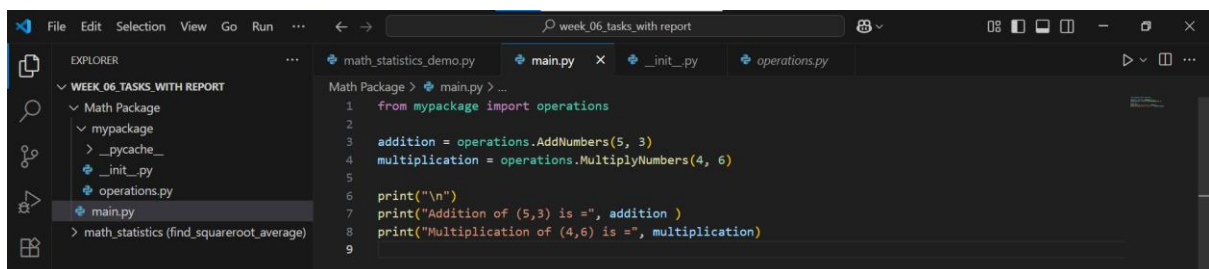
### Step-by-Step Instructions

#### 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



The screenshot shows a code editor with a project structure on the left and a code snippet in the main editor. The project structure is as follows:

- WEEK\_06\_TASKS\_WITH\_REPORT
  - Math Package
    - my\_package
      - \_\_pycache\_\_
      - \_\_init\_\_.py
      - operations.py
    - main.py
    - math\_statistics (find\_squareroot\_average)

The code snippet in the main editor is:

```
1 from mypackage import operations
2
3 addition = operations.AddNumbers(5, 3)
4 multiplication = operations.MultiplyNumbers(4, 6)
5
6 print("\n")
7 print("Addition of (5,3) is =", addition )
8 print("Multiplication of (4,6) is =", multiplication)
9
```

#### 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
```

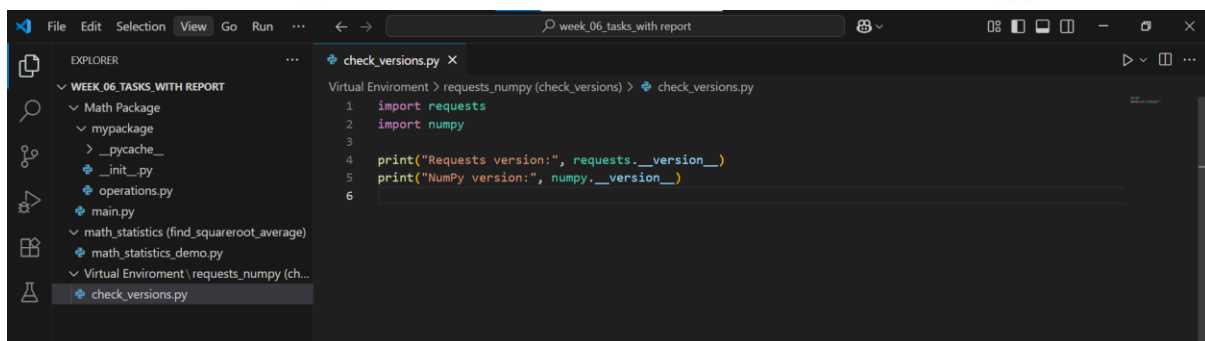
You should now see (myenv) before the command line prompt.

## 2. Install requests and numpy

While the environment is active, run:

```
pip install requests numpy
```

## Code Snippet



```
File Edit Selection View Go Run ... week_06_tasks_with_report
EXPLORER
WEEK_06_TASKS_WITH REPORT
  Math Package
  mypackage
    __pycache__
    __init__.py
    operations.py
    main.py
  math_statistics (find_squareroot_average)
  math_statistics_demo.py
  Virtual Environment \ requests_numpy (ch...
    check_versions.py

check_versions.py X
Virtual Environment > requests_numpy (check_versions) > check_versions.py
1 import requests
2 import numpy
3
4 print("Requests version:", requests.__version__)
5 print("NumPy version:", numpy.__version__)
6
```

## Output Snippet

```
Select Administrator: Command Prompt

(env) D:\Python Internship Tasks\week_06_tasks_with report\Virtual Enviroment> cd D:\Python Internship Tasks\week_06_tasks_with report\Virtual Enviroment\requests_numpy (check_versions)

(env) D:\Python Internship Tasks\week_06_tasks_with report\Virtual Enviroment\requests_numpy (check_versions)>python check_versions.py
Requests version: 2.32.4
NumPy version: 2.3.2

(env) D:\Python Internship Tasks\week_06_tasks_with report\Virtual Enviroment\requests_numpy (check_versions)>
(env) D:\Python Internship Tasks\week_06_tasks_with report\Virtual Enviroment\requests_numpy (check_versions)>_
```

## 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

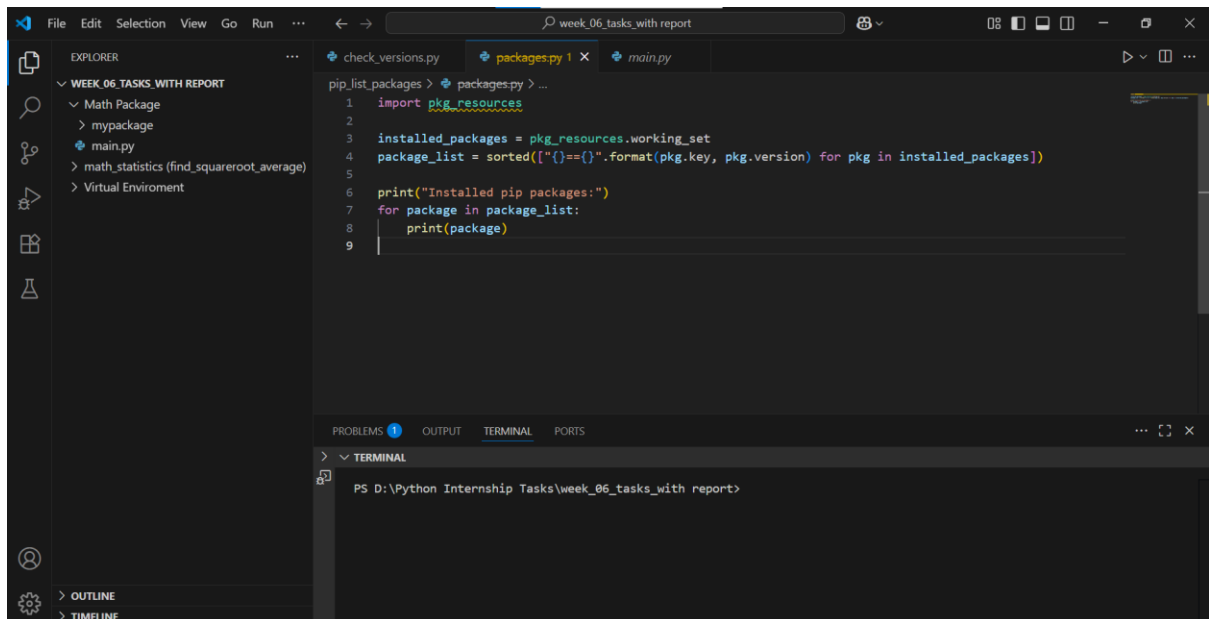
Navigate to the script's location:

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

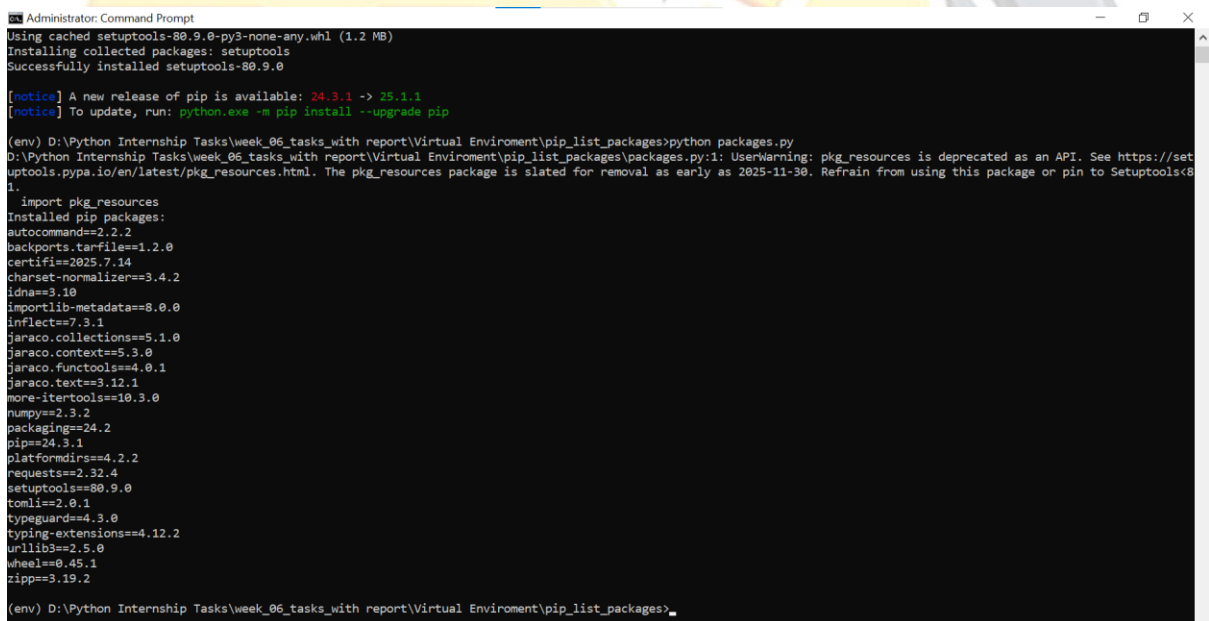
Then run:

**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

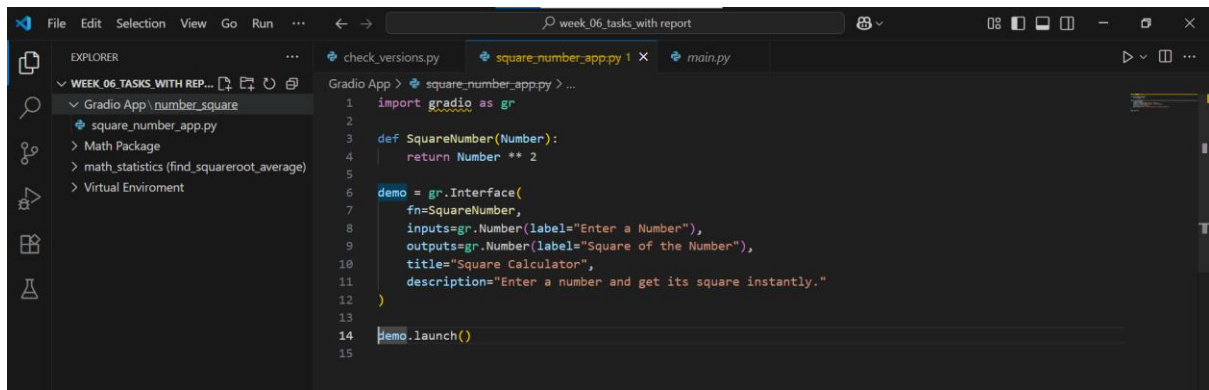
**Name the file `square_numbers_app.py`**

- Run the file in terminal

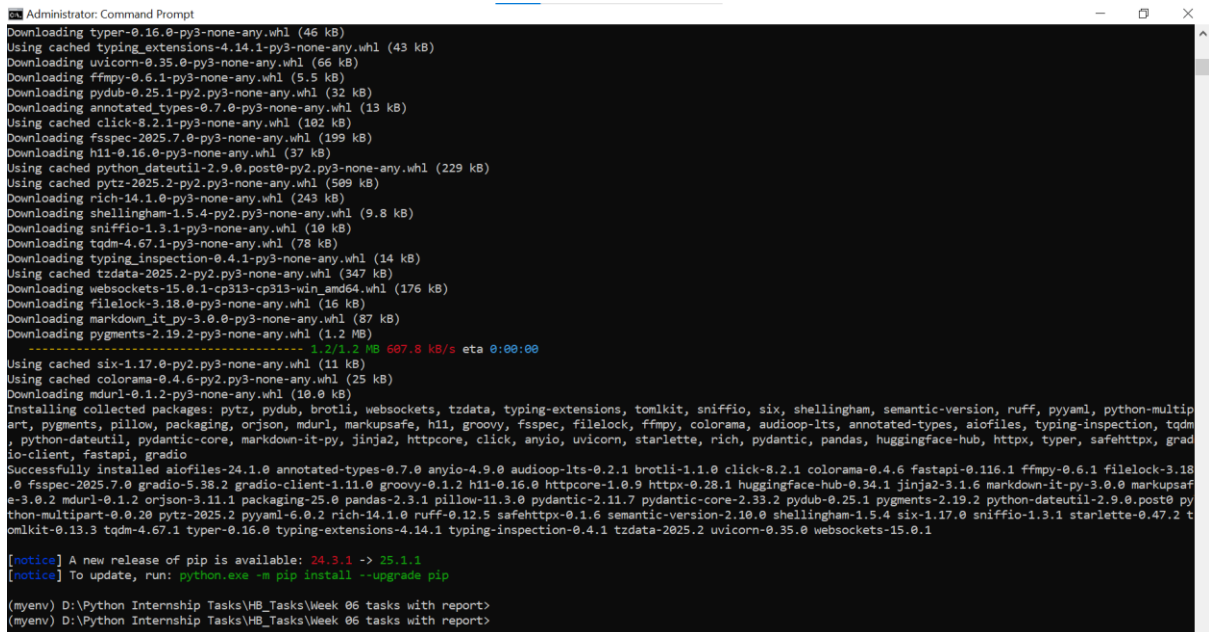
**`python square_numbers_app.py`**

- A web browser will open with a simple UI.

#### **Code Snippet**



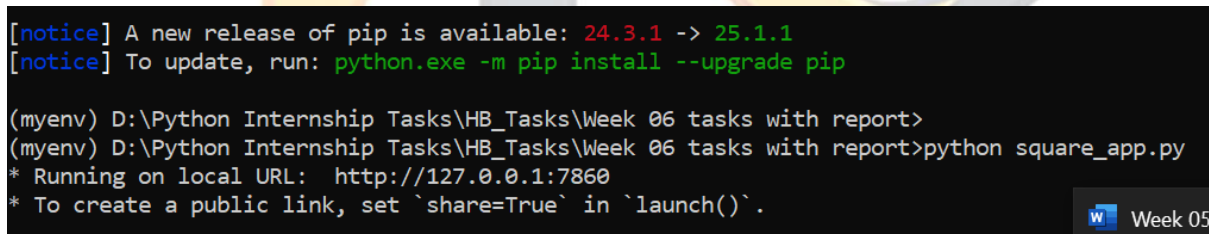
```
1 import gradio as gr
2
3 def SquareNumber(Number):
4     return Number ** 2
5
6 demo = gr.Interface(
7     fn=SquareNumber,
8     inputs=gr.Number(label="Enter a Number"),
9     outputs=gr.Number(label="Square of the Number"),
10    title="Square Calculator",
11    description="Enter a number and get its square instantly."
12)
13
14 demo.launch()
```



```
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 (499 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 667.8 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-multip
art, 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, grad
io-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 markupsaf
e-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 py
thon-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 t
omlkit-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 Snippet

Square Calculator

Enter a number and get its square instantly.

Enter a Number

0

Clear

Square of the Number

0

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.**

## 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

```
(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square>fa
'fa' is not recognized as an internal or external command,
operable program or batch file.

(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square>
(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square>
(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square>
(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square>
(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square>
(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square>cd D:
D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square

(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\number_square> cd D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\reversed_sentence

(env) D:\Python Internship Tasks\week_06_tasks_with report\Gradio App\reversed_sentence>python sentence_reverser_app.py
* Running on local URL:  http://127.0.0.1:7860
* To create a public link, set 'share=True' in 'launch()'.
```

## 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.

The logo for Technik NEST features a large, stylized 'TN' in the background. The 'T' is yellow with a grey shadow, and the 'N' is grey with a yellow shadow. Below this, the word 'TECHNIK' is written in yellow and 'NEST' in grey, both in a bold, sans-serif font.

**TECHNIK NEST**