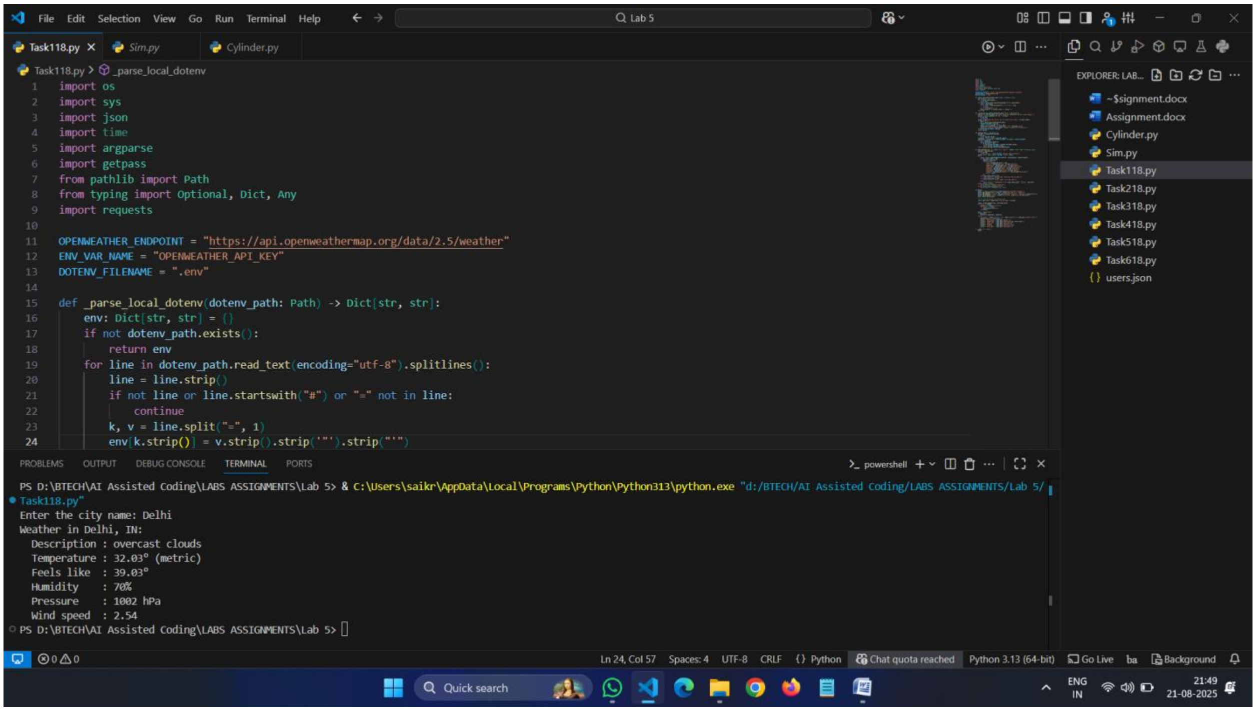
AI-Assignment-5

NAME : R.Shivani

ROLL NO : 2403a52411

BATCH:15

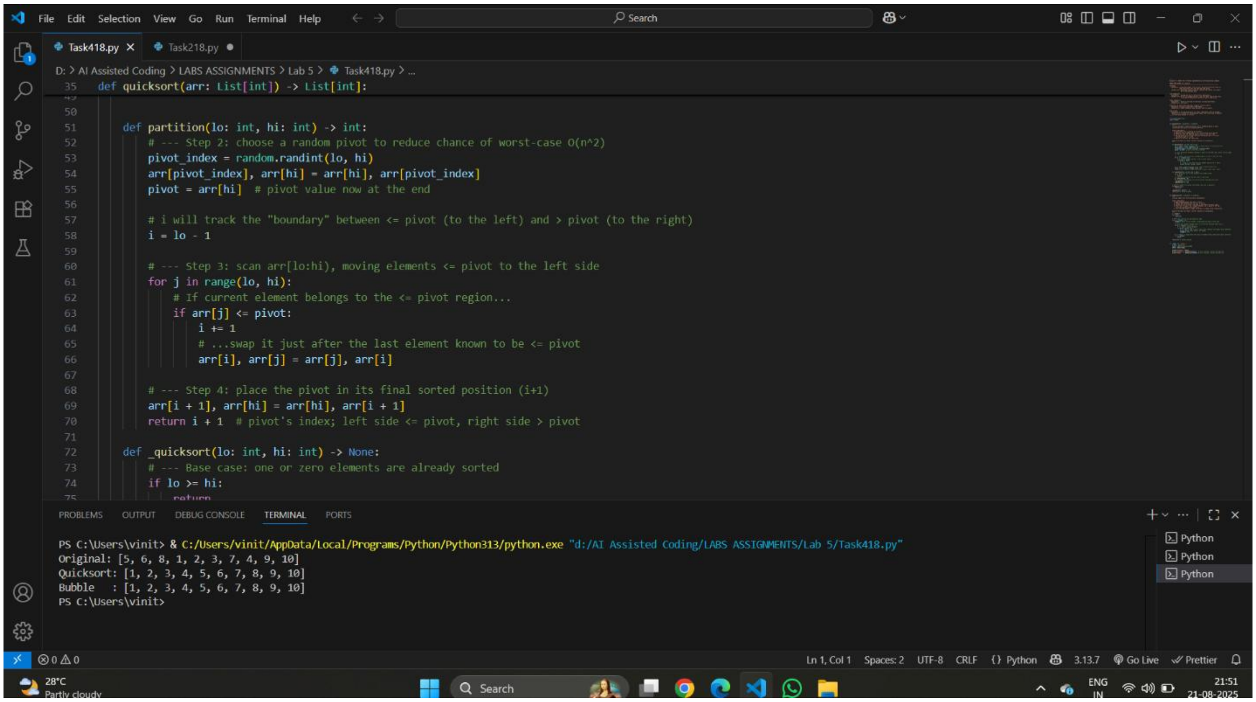
- Generate code to fetch weather data securely without exposing API keys in the code.


File Edit Selection View Go Run Terminal Help
Lab 5
Task118.py
Sim.py
Cylinder.py
Task118.py > _parse_local_dotenv
EXPLORER: LAB...
~$signment.docx
Assignment.docx
Cylinder.py
Sim.py
Task118.py
Task218.py
Task318.py
Task418.py
Task518.py
Task618.py
users.json
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Ln 24, Col 57 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached Python 3.13 (64-bit) Go Live Background
Quick search
ENG IN 21:49 21-08-2025

```python
import os
import sys
import json
import time
import argparse
import getpass
from pathlib import Path
from typing import Optional, Dict, Any
import requests

OPENWEATHER_ENDPOINT = "https://api.openweathermap.org/data/2.5/weather"
ENV_VAR_NAME = "OPENWEATHER_API_KEY"
DOTENV_FILENAME = ".env"

def _parse_local_dotenv(dotenv_path: Path) -> Dict[str, str]:
    env: Dict[str, str] = {}
    if not dotenv_path.exists():
        return env
    for line in dotenv_path.read_text(encoding="utf-8").splitlines():
        line = line.strip()
        if not line or line.startswith("#") or "=" not in line:
            continue
        k, v = line.split("=", 1)
        env[k.strip()] = v.strip().strip('"').strip("'")
```

```
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 5> & C:\Users\saikr\AppData\Local\Programs\Python\Python313\python.exe "d:/BTECH/AI Assisted Coding/LABS ASSIGNMENTS/Lab 5/Task118.py"
Enter the city name: Delhi
Weather in Delhi, IN:
  Description : overcast clouds
  Temperature : 32.03° (metric)
  Feels like  : 39.03°
  Humidity    : 70%
  Pressure    : 1002 hPa
  Wind speed  : 2.54
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 5>
```

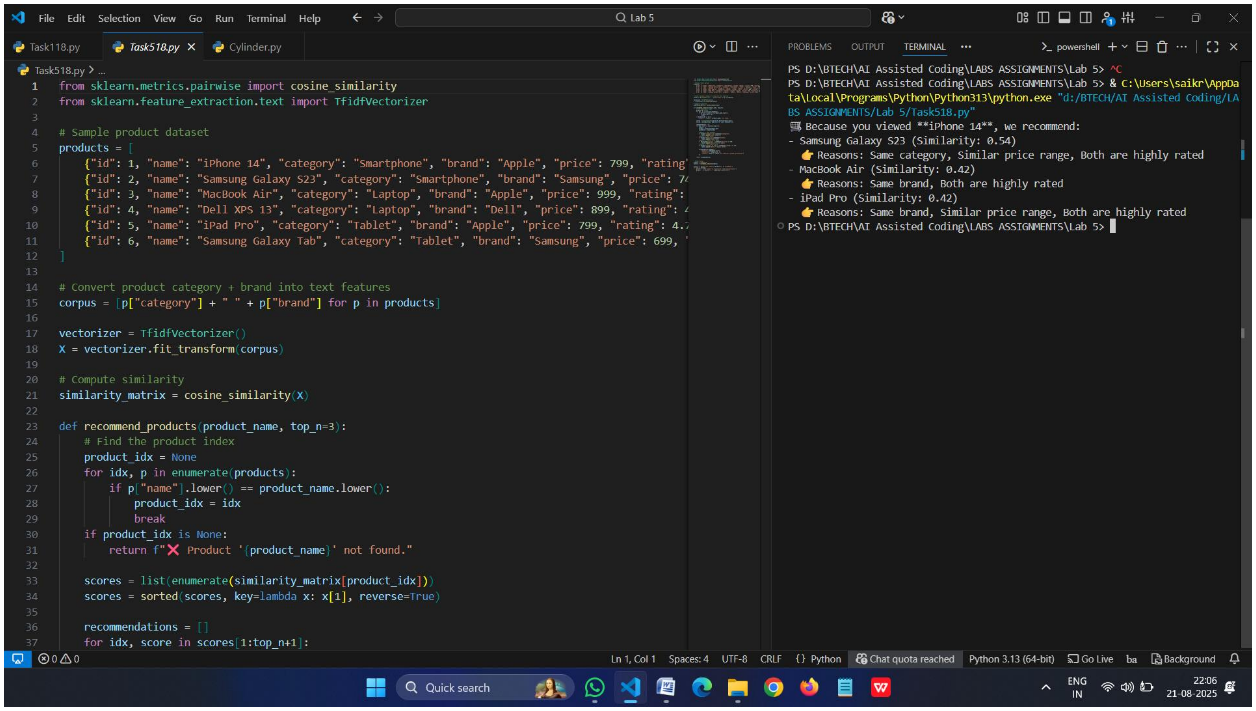
- Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ.



```python
def quicksort(arr: List[int]) -> List[int]:

    def partition(lo: int, hi: int) -> int:
        # --- Step 2: choose a random pivot to reduce chance of worst-case O(n^2)
        pivot_index = random.randint(lo, hi)
        arr[pivot_index], arr[hi] = arr[hi], arr[pivot_index]
        pivot = arr[hi]  # pivot value now at the end

        # i will track the "boundary" between <= pivot (to the left) and > pivot (to the right)
        i = lo - 1

        # --- Step 3: scan arr[lo:hi), moving elements <= pivot to the left side
        for j in range(lo, hi):
            # If current element belongs to the <= pivot region...
            if arr[j] <= pivot:
                i += 1
                # ...swap it just after the last element known to be <= pivot
                arr[i], arr[j] = arr[j], arr[i]

        # --- Step 4: place the pivot in its final sorted position (i+1)
        arr[i + 1], arr[hi] = arr[hi], arr[i + 1]
        return i + 1  # pivot's index; left side <= pivot, right side > pivot

    def _quicksort(lo: int, hi: int) -> None:
        # --- Base case: one or zero elements are already sorted
        if lo >= hi:
            return
```

```
PS C:\Users\vinit> & C:/Users/vinit/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Assisted Coding/LABS ASSIGNMENTS/Lab 5/Task418.py"
Original: [5, 6, 8, 1, 2, 3, 7, 4, 9, 10]
Quicksort: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Bubble   : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
PS C:\Users\vinit>
```

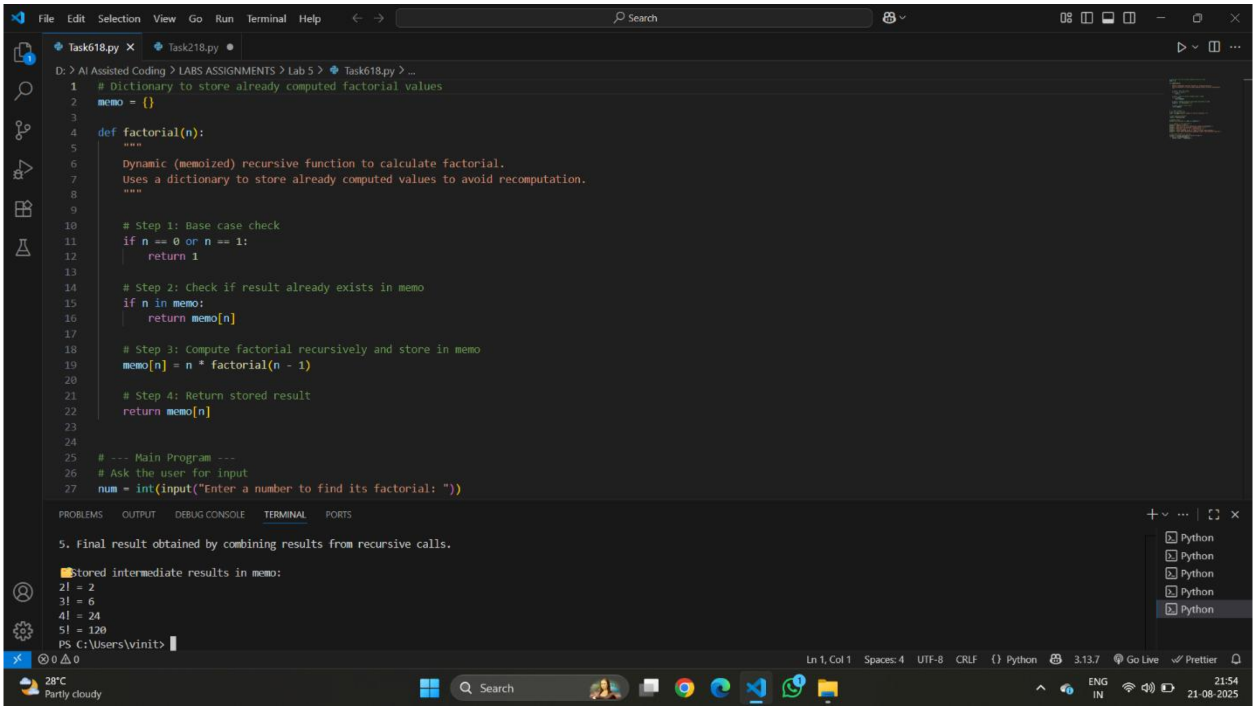
- Generate a recommendation system that also provides reasons for each suggestion



```python
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample product dataset
products = [
    {"id": 1, "name": "iPhone 14", "category": "Smartphone", "brand": "Apple", "price": 799, "rating"
    {"id": 2, "name": "Samsung Galaxy S23", "category": "Smartphone", "brand": "Samsung", "price": 74
    {"id": 3, "name": "MacBook Air", "category": "Laptop", "brand": "Apple", "price": 999, "rating":
    {"id": 4, "name": "Dell XPS 13", "category": "Laptop", "brand": "Dell", "price": 899, "rating": 4
    {"id": 5, "name": "iPad Pro", "category": "Tablet", "brand": "Apple", "price": 799, "rating": 4.7
    {"id": 6, "name": "Samsung Galaxy Tab", "category": "Tablet", "brand": "Samsung", "price": 699, '
]

# Convert product category + brand into text features
corpus = [p["category"] + " " + p["brand"] for p in products]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)

# Compute similarity
similarity_matrix = cosine_similarity(X)

def recommend_products(product_name, top_n=3):
    # Find the product index
    product_idx = None
    for idx, p in enumerate(products):
        if p["name"].lower() == product_name.lower():
            product_idx = idx
            break
    if product_idx is None:
        return f"❌ Product '{product_name}' not found."

    scores = list(enumerate(similarity_matrix[product_idx]))
    scores = sorted(scores, key=lambda x: x[1], reverse=True)

    recommendations = []
    for idx, score in scores[1:top_n+1]:
```

```
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 5> ^C
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 5> & C:\Users\saikr\AppData\Local\Programs\Python\Python313\python.exe "d:/BTECH/AI Assisted Coding/LABS ASSIGNMENTS/Lab 5/Task518.py"
🛒 Because you viewed **iPhone 14**, we recommend:
- Samsung Galaxy S23 (Similarity: 0.54)
  👉 Reasons: Same category, Similar price range, Both are highly rated
- MacBook Air (Similarity: 0.42)
  👉 Reasons: Same brand, Both are highly rated
- iPad Pro (Similarity: 0.42)
  👉 Reasons: Same brand, Similar price range, Both are highly rated
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 5>
```

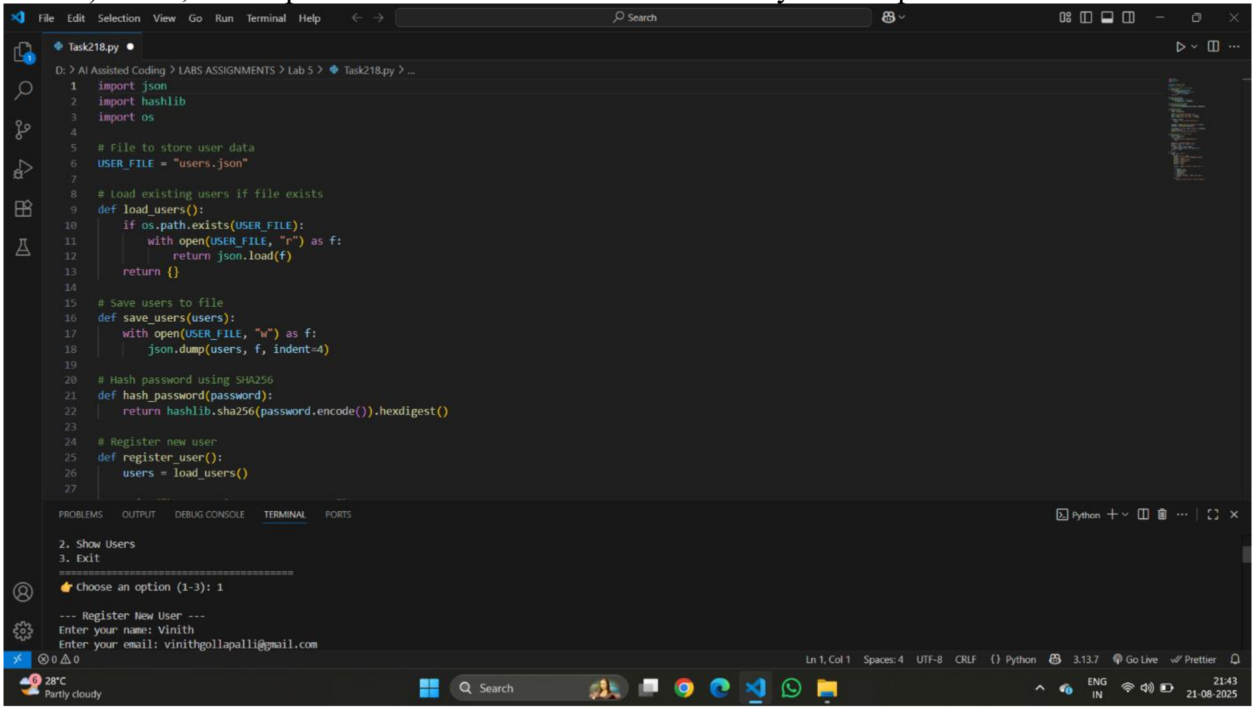
- Generate a recursive factorial function with comments that explain each line and a final summary of the algorithm's flow


File Edit Selection View Go Run Terminal Help
Search
Task618.py
Task218.py
D: > AI Assisted Coding > LABS ASSIGNMENTS > Lab 5 > Task618.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Python
Python
Python
Python
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live Prettier
28°C
Partly cloudy
Search
ENG
IN
21:54
21-08-2025

```python
# Dictionary to store already computed factorial values
memo = {}

def factorial(n):
    """
    Dynamic (memoized) recursive function to calculate factorial.
    Uses a dictionary to store already computed values to avoid recomputation.
    """

    # Step 1: Base case check
    if n == 0 or n == 1:
        return 1

    # Step 2: Check if result already exists in memo
    if n in memo:
        return memo[n]

    # Step 3: Compute factorial recursively and store in memo
    memo[n] = n * factorial(n - 1)

    # Step 4: Return stored result
    return memo[n]


# --- Main Program ---
# Ask the user for input
num = int(input("Enter a number to find its factorial: "))
```

```
5. Final result obtained by combining results from recursive calls.

📁Stored intermediate results in memo:
2! = 2
3! = 6
4! = 24
5! = 120
PS C:\Users\vinit> 
```

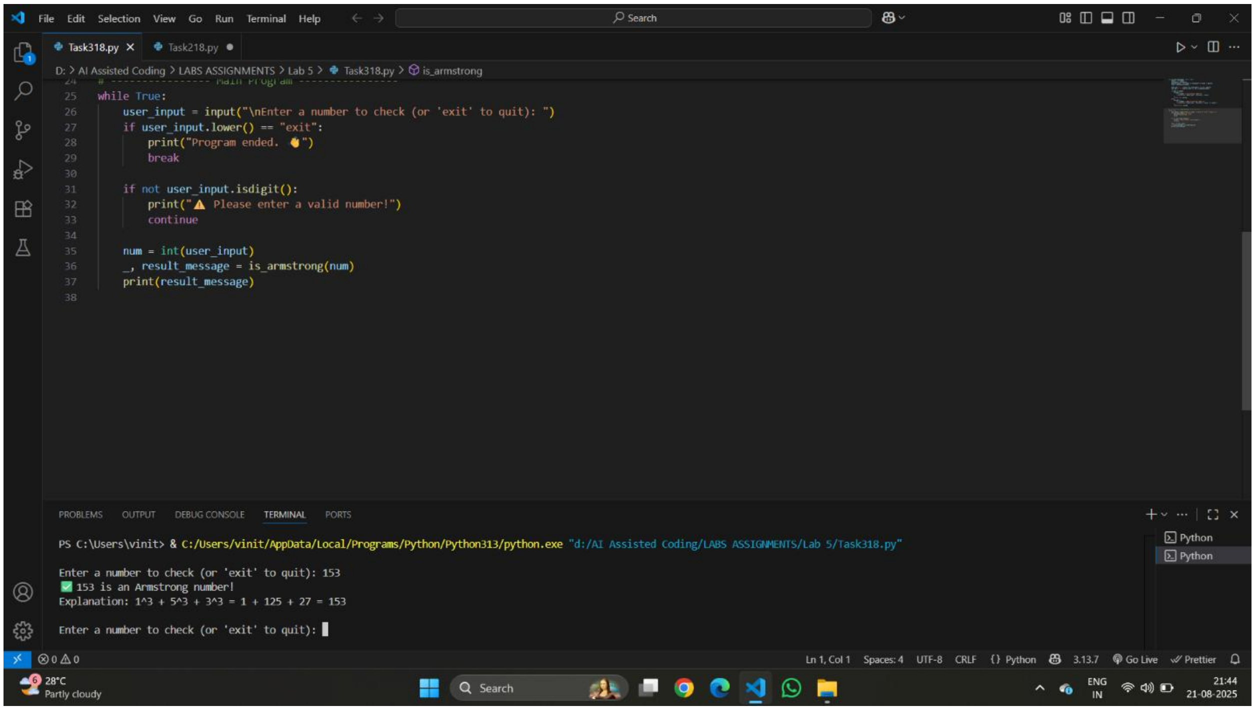
• Prompt GitHub Copilot to generate a Python script that collects user data (e.g., name, age, email). Then, ask Copilot to add comments on how to anonymize or protect this data.


File Edit Selection View Go Run Terminal Help
Search
Task218.py
D: > AI Assisted Coding > LABS ASSIGNMENTS > Lab 5 > Task218.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live Prettier
28°C
Partly cloudy
Search
ENG
IN
21:43
21-08-2025

```python
import json
import hashlib
import os

# File to store user data
USER_FILE = "users.json"

# Load existing users if file exists
def load_users():
    if os.path.exists(USER_FILE):
        with open(USER_FILE, "r") as f:
            return json.load(f)
    return {}

# Save users to file
def save_users(users):
    with open(USER_FILE, "w") as f:
        json.dump(users, f, indent=4)

# Hash password using SHA256
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

# Register new user
def register_user():
    users = load_users()

```

```
2. Show Users
3. Exit
========================================
👉 Choose an option (1-3): 1

--- Register New User ---
Enter your name: Vinith
Enter your email: vinithgollapalli@gmail.com
```

- Generate an Armstrong number checking function with comments and explanation.


File Edit Selection View Go Run Terminal Help
Search
Task318.py
Task218.py
D: > AI Assisted Coding > LABS ASSIGNMENTS > Lab 5 > Task318.py > is_armstrong
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Python
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live Prettier
28°C
Partly cloudy
Search
ENG
IN
21:44
21-08-2025

```python
# ---------------- Main Program ----------------
while True:
    user_input = input("\nEnter a number to check (or 'exit' to quit): ")
    if user_input.lower() == "exit":
        print("Program ended. 👋")
        break

    if not user_input.isdigit():
        print("⚠ Please enter a valid number!")
        continue

    num = int(user_input)
    _, result_message = is_armstrong(num)
    print(result_message)
```

```
PS C:\Users\vinit> & C:/Users/vinit/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Assisted Coding/LABS ASSIGNMENTS/Lab 5/Task318.py"

Enter a number to check (or 'exit' to quit): 153
✅ 153 is an Armstrong number!
Explanation: 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153

Enter a number to check (or 'exit' to quit):
```