

# Assignment-5

NAME : K.NAVYA SRI.

ENROLL NO : 2503A52L12

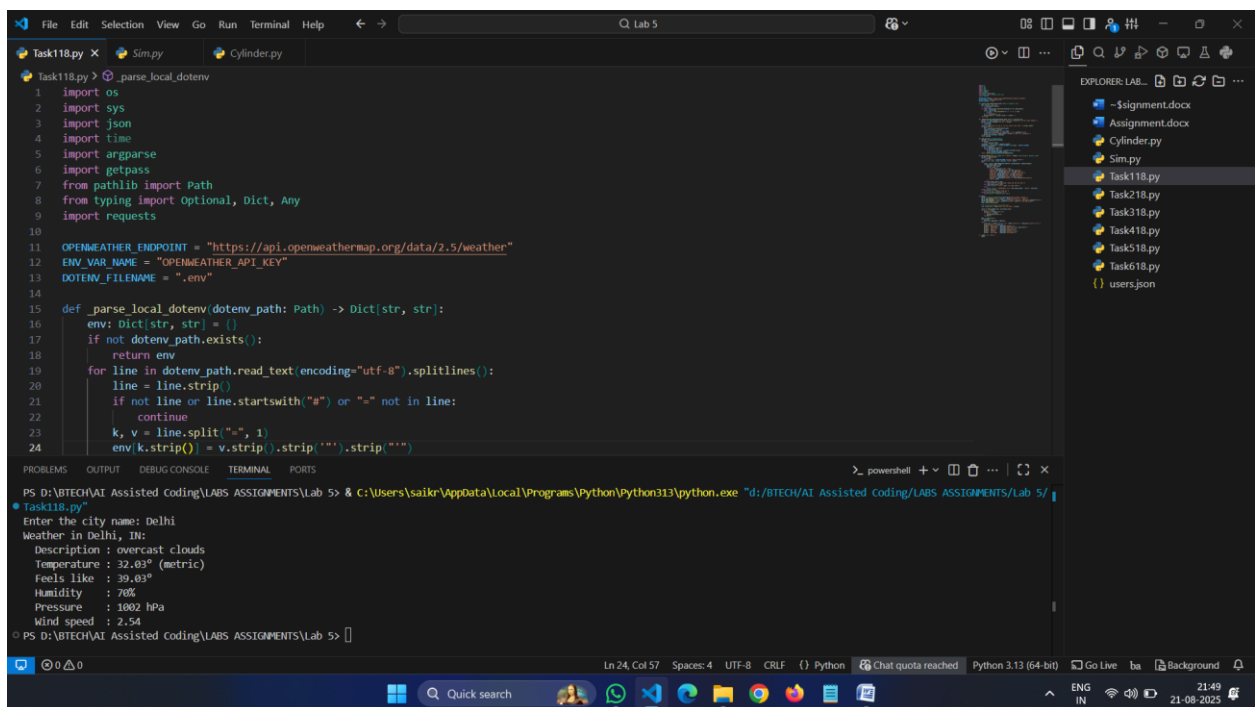
BATCH : 16

**Task 1 :** Use an AI tool to generate a Python program that connects to a weather API.

**Prompt:**

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

**CODE&OUTPUT:**



```
File Edit Selection View Go Run Terminal Help
Task118.py x Sim.py Cylinder.py
Task118.py > _parse_local_dotenv
1 import os
2 import sys
3 import json
4 import time
5 import argparse
6 import getpass
7 from pathlib import Path
8 from typing import Optional, Dict, Any
9 import requests
10
11 OPENWEATHER_ENDPOINT = "https://api.openweathermap.org/data/2.5/weather"
12 ENV_VAR_NAME = "OPENWEATHER_API_KEY"
13 DOTENV_FILENAME = ".env"
14
15 def _parse_local_dotenv(dotenv_path: Path) -> Dict[str, str]:
16     env: Dict[str, str] = {}
17     if not dotenv_path.exists():
18         return env
19     for line in dotenv_path.read_text(encoding="utf-8").splitlines():
20         line = line.strip()
21         if not line or line.startswith("#") or "=" not in line:
22             continue
23         k, v = line.split("=", 1)
24         env[k.strip()] = v.strip().strip('"').strip("'")
25
26 PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 5> & C:\Users\saiar\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>
```

**Task:** Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).

**Prompt:**

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

**CODE&OUTPUT:**

```

D: > AI Assisted Coding > LABS ASSIGNMENTS > Lab 5 > Task418.py > ...
35 def quicksort(arr: List[int]) -> List[int]:
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51 def partition(lo: int, hi: int) -> int:
52     # --- Step 2: choose a random pivot to reduce chance of worst-case O(n^2)
53     pivot_index = random.randint(lo, hi)
54     arr[pivot_index], arr[hi] = arr[hi], arr[pivot_index]
55     pivot = arr[hi] # pivot value now at the end
56
57     # i will track the "boundary" between <= pivot (to the left) and > pivot (to the right)
58     i = lo - 1
59
60     # --- Step 3: scan arr[lo:hi], moving elements <= pivot to the left side
61     for j in range(lo, hi):
62         # If current element belongs to the <= pivot region...
63         if arr[j] <= pivot:
64             i += 1
65             # ...swap it just after the last element known to be <= pivot
66             arr[i], arr[j] = arr[j], arr[i]
67
68     # --- Step 4: place the pivot in its final sorted position (i+1)
69     arr[i + 1], arr[hi] = arr[hi], arr[i + 1]
70     return i + 1 # pivot's index; left side <= pivot, right side > pivot
71
72 def _quicksort(lo: int, hi: int) -> None:
73     # --- Base case: one or zero elements are already sorted
74     if lo >= hi:
75         return
76
77 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>

```

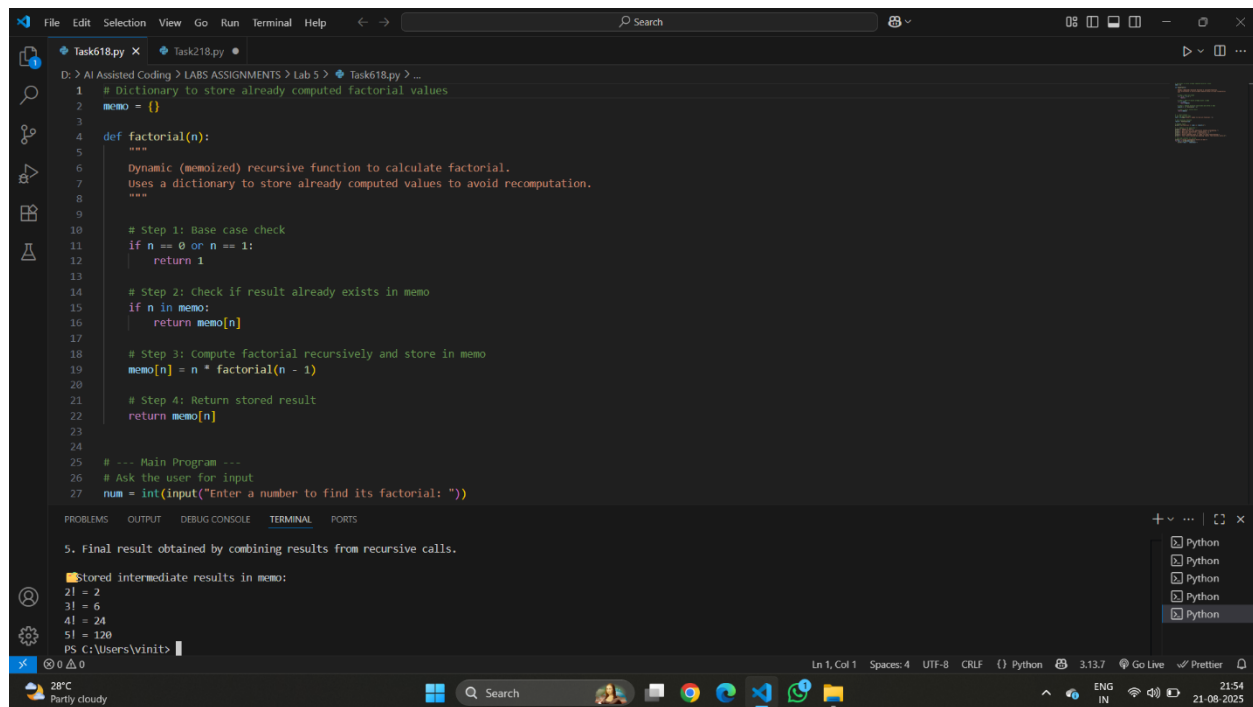
- Generate a recommendation system that also provides reasons for each suggestion

```

Task518.py > ...
1 from sklearn.metrics.pairwise import cosine_similarity
2 from sklearn.feature_extraction.text import TfidfVectorizer
3
4 # Sample product dataset
5 products = [
6     {"id": 1, "name": "iPhone 14", "category": "Smartphone", "brand": "Apple", "price": 799, "rating": 4.5},
7     {"id": 2, "name": "Samsung Galaxy S23", "category": "Smartphone", "brand": "Samsung", "price": 799, "rating": 4.3},
8     {"id": 3, "name": "MacBook Air", "category": "Laptop", "brand": "Apple", "price": 999, "rating": 4.7},
9     {"id": 4, "name": "Dell XPS 13", "category": "Laptop", "brand": "Dell", "price": 899, "rating": 4.2},
10    {"id": 5, "name": "iPad Pro", "category": "Tablet", "brand": "Apple", "price": 799, "rating": 4.6},
11    {"id": 6, "name": "Samsung Galaxy Tab", "category": "Tablet", "brand": "Samsung", "price": 699, "rating": 4.1}
12 ]
13
14 # Convert product category + brand into text features
15 corpus = [p["category"] + " " + p["brand"] for p in products]
16
17 vectorizer = TfidfVectorizer()
18 X = vectorizer.fit_transform(corpus)
19
20 # Compute similarity
21 similarity_matrix = cosine_similarity(X)
22
23 def recommend_products(product_name, top_n=3):
24     # Find the product index
25     product_idx = None
26     for idx, p in enumerate(products):
27         if p["name"].lower() == product_name.lower():
28             product_idx = idx
29             break
30     if product_idx is None:
31         return f"❌ Product '{product_name}' not found."
32
33     scores = list(enumerate(similarity_matrix[product_idx]))
34     scores = sorted(scores, key=lambda x: x[1], reverse=True)
35
36     recommendations = []
37     for idx, score in scores[1:top_n+1]:
38
39 PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 5> & C:\Users\sakir\AppData\Local\Programs\Python\Python313\python.exe "d:/BTECH/ AI Assisted Coding/LAB 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>

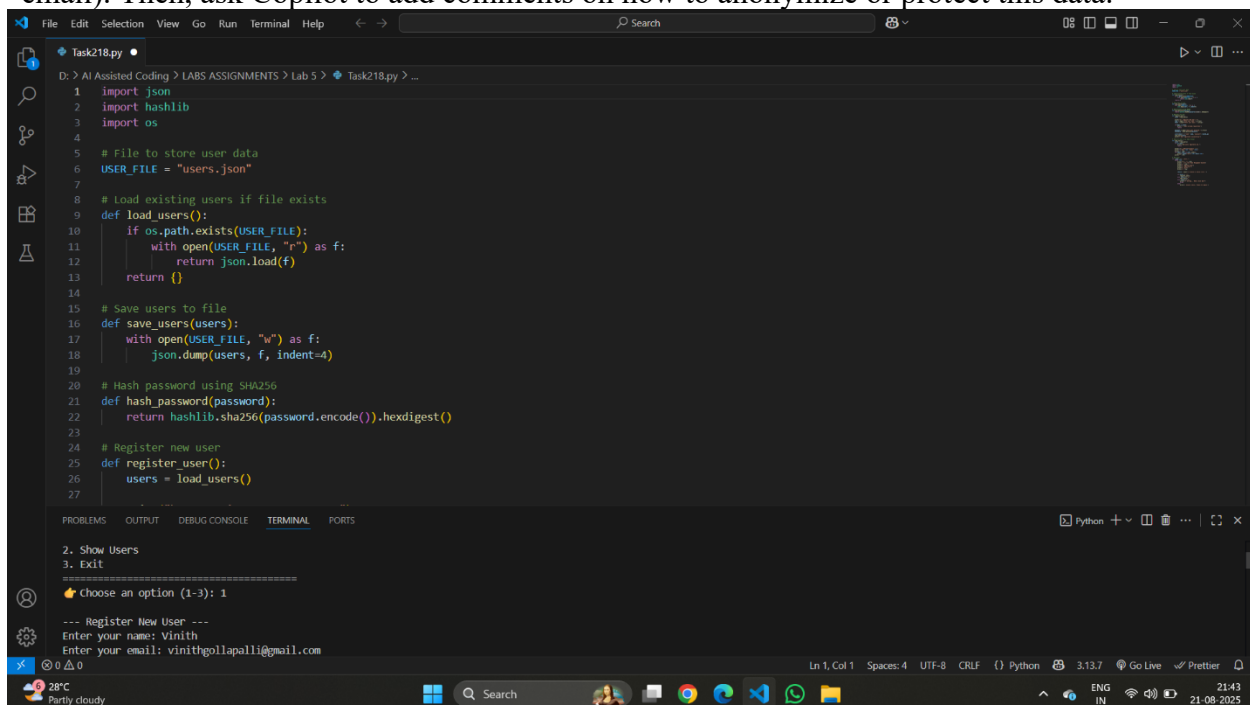
```

- 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
Task618.py Task218.py
D:\AI Assisted Coding > LABS ASSIGNMENTS > Lab 5 > Task618.py > ...
1 # Dictionary to store already computed factorial values
2 memo = {}
3
4 def factorial(n):
5     """
6     Dynamic (memoized) recursive function to calculate factorial.
7     Uses a dictionary to store already computed values to avoid recomputation.
8     """
9
10    # Step 1: Base case check
11    if n == 0 or n == 1:
12        return 1
13
14    # Step 2: Check if result already exists in memo
15    if n in memo:
16        return memo[n]
17
18    # Step 3: Compute factorial recursively and store in memo
19    memo[n] = n * factorial(n - 1)
20
21    # Step 4: Return stored result
22    return memo[n]
23
24
25 --- Main Program ---
26 # Ask the user for input
27 num = int(input("Enter a number to find its factorial: "))
28
29
30 # Final result obtained by combining results from recursive calls.
31
32 # stored intermediate results in memo:
33 2! = 2
34 3! = 6
35 4! = 24
36 5! = 120
37
38 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
Task218.py
D:\AI Assisted Coding > LABS ASSIGNMENTS > Lab 5 > Task218.py > ...
1 import json
2 import hashlib
3 import os
4
5 # File to store user data
6 USER_FILE = "users.json"
7
8 # Load existing users if file exists
9 def load_users():
10     if os.path.exists(USER_FILE):
11         with open(USER_FILE, "r") as f:
12             return json.load(f)
13     return {}
14
15 # Save users to file
16 def save_users(users):
17     with open(USER_FILE, "w") as f:
18         json.dump(users, f, indent=4)
19
20 # Hash password using SHA256
21 def hash_password(password):
22     return hashlib.sha256(password.encode()).hexdigest()
23
24 # Register new user
25 def register_user():
26     users = load_users()
27
28
29 2. Show Users
30 3. Exit
31
32 Choose an option (1-3): 1
33
34 --- Register New User ---
35 Enter your name: Vinit
36 Enter your email: vinitgollapalli@gmail.com
```

## PROMPT:

Generate an Armstrong number checking function with comments and explanation.

FileEditSelectionViewGoRunTerminalHelp<-->Search

Task318.pyTask218.py

D:\> AI Assisted Coding > LABS ASSIGNMENTS > Lab 5 > Task318.py is\_armstrong

```
25 while True:
26     user_input = input("\nEnter a number to check (or 'exit' to quit): ")
27     if user_input.lower() == "exit":
28         print("Program ended. 🍀")
29         break
30
31     if not user_input.isdigit():
32         print("⚠️ Please enter a valid number!")
33         continue
34
35     num = int(user_input)
36     _, result_message = is_armstrong(num)
37     print(result_message)
38
```

PROBLEMSOUTPUTDEBUG CONSOLETERMINALPORTS

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):

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live Prettier

28°C  
Partly cloudy

Search

21:44  
21-08-2025