Overview

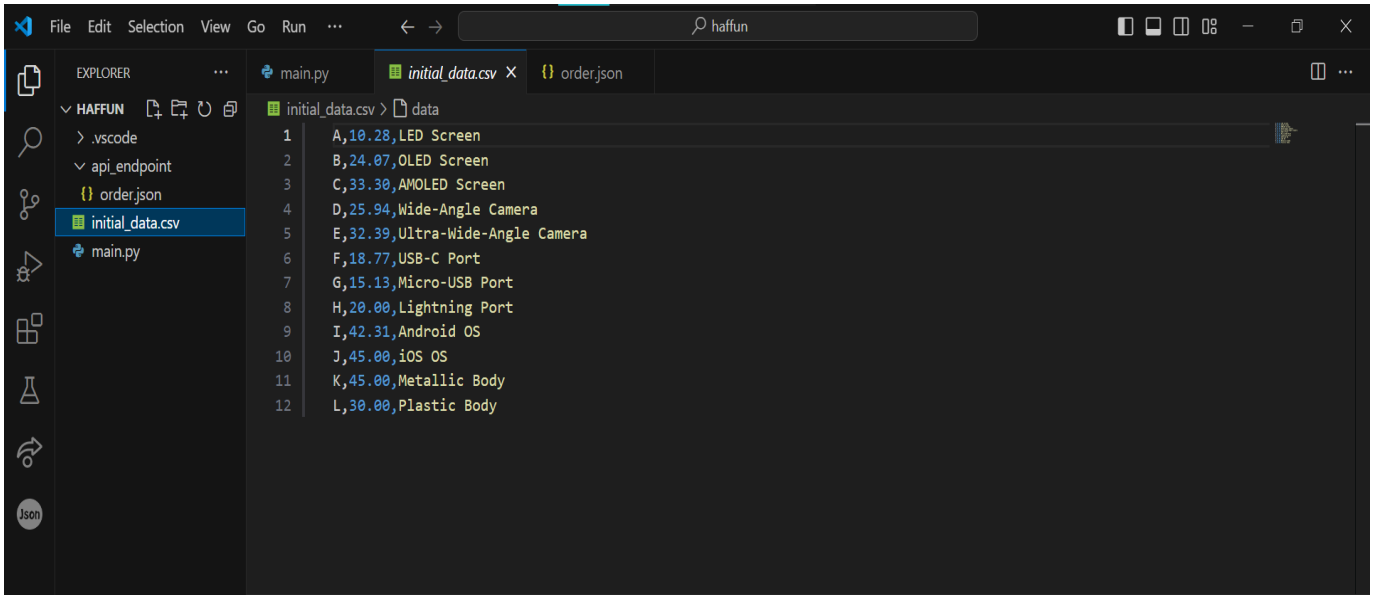
This Python script processes component orders for a hypothetical product, ensuring each order adheres to specified component types and calculating the total cost of the order. It reads initial data from a CSV file, processes incoming HTTP POST requests to append new orders to a JSON file, and ensures order IDs are incremented correctly.

Prerequisites

- Python 3.x
- A CSV file named initial_data.csv in the same directory
- Basic understanding of Python and JSON

Files

- script.py: The main script containing the logic for processing orders.
- initial_data.csv: The CSV file containing initial component data in the format component,price,part.
- CSV Format
- The initial_data.csv file should have the following format:



Usage

1. Ensure the initial_data.csv file is present in the same directory as the script.
2. Run the script:
   main.py
3. Input the HTTP POST command as prompted:
   HTTP POST /order { "components": ['I', 'A', 'D', 'F', 'K'] }

How the Script Works

1.CSV to Map Conversion:
The script reads initial_data.csv and converts it into a dictionary (processed_data) mapping component codes to their prices and parts.

2.Component Type Validation:
The script defines valid component types in the types list, ensuring that each order includes exactly one component from each category.

3.No Duplicate Validation:
The noDupes function checks that the order contains one valid component from each category and ensures no duplicates.

4.Processing Orders:
The processedObject function creates an order object with the total price and list of parts based on the provided components.

5.Handling HTTP POST Request:

- The httpPostRequest function:
  Reads existing data from api_endpoint/order.json.
- Increments the ORDER_ID based on the last entry.
- Processes the new order and appends it to the existing data.
- Writes the updated data back to api_endpoint/order.json.

6.Main Input Handling:
The script simulates handling an HTTP POST command by parsing the input and invoking httpPostRequest with the extracted components.

The script will:

- Validate that the components are from the specified categories.
- Calculate the total cost.
- Create an order object.
- Append the order to the existing data in api_endpoint/order.json.
- Increment the ORDER_ID.

Detailed Steps

CSV Data Conversion:
The csv_to_map function reads initial_data.csv and creates a dictionary (processed_data).

Component Validation:
The noDupes function ensures each component in the order is from the correct category defined in types.

Order Processing:
The processedObject function creates an order with the total cost and parts list if the components are valid.

POST Request Handling:
The httpPostRequest function reads existing orders from the JSON file, calculates the new ORDER_ID, processes the new order, and writes it back.
Main Script Execution:

The script reads input, parses the command, extracts components, and calls httpPostRequest to handle the order.

File Edit Selection View Go Run ··· haffun

EXPLORER

HAFFUN

- .vscode
- api_endpoint
  - order.json
- initial_data.csv
- main.py

main.py | initial_data.csv | order.json

api_endpoint > order.json > ...

```
2    {
3        "order_id": 1,
4        "total": 142.3,
5        "parts": [
6            "LED Screen",
7            "Wide-Angle Camera",
8            "USB-C Port",
9            "Android OS",
10           "Metallic Body"
11       ]
12   },
13   {
14       "order_id": 2,
15       "total": 144.99,
16       "parts": [
17           "LED Screen",
18           "Wide-Angle Camera",
19           "USB-C Port",
20           "iOS OS",
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\velur\OneDrive\Desktop\haffun> & C:/Users/velur/AppData/Local/Programs/Python/Python312/python.exe c:/Users/velur/OneDrive/Desktop/haffun/main.py
HTTP POST /order { "components": ['I', 'A', 'D', 'F', 'K'] }
Updated data: [{'order_id': 1, 'total': 142.3, 'parts': ['LED Screen', 'Wide-Angle Camera', 'USB-C Port', 'Android OS', 'Metallic Body']}, {'order_id': 2, 'total': 144.99, 'parts': ['LED Screen', 'Wide-Angle Camera', 'USB-C Port', 'iOS OS', 'Metallic Body']}, {'order_id': 3, 'total': 142.3, 'parts': ['LED Screen', 'Wide-Angle Camera', 'USB-C Port', 'Android OS', 'Metallic Body']}]
201
 {'order_id': 3, 'total': 142.3, 'parts': ['LED Screen', 'Wide-Angle Camera', 'USB-C Port', 'Android OS', 'Metallic Body']}
PS C:\Users\velur\OneDrive\Desktop\haffun>
```

Python
Python
Python

OUTLINE
TIMELINE
LIVE SHARE

0 0 0 Shared 0 Live Share Chat: 2 new Ln 1, Col 2 Spaces: 4 UTF-8 CRLF {} Json