1. Overview

This project provides two main functions, `get_combined_option_chain_data` and `calculate_margin_and_premium`, to retrieve option chain data, calculate required margin, and compute premium earned for options using the Upstox API.

---

2. Functions and Their Explanations

Function 1: `get_combined_option_chain_data`

- **Purpose**: Fetches option chain data from the Upstox API for both Put (PE) and Call (CE) options, retrieves the highest bid price for PE and the highest ask price for CE, and organizes this data into a DataFrame.
- **Inputs**:
  - `instrument_name`: The name of the instrument (e.g., "Nifty 50").
  - `expiry_date`: The expiry date for the options.
- **Output**: Returns a DataFrame with columns `instrument_key`, `strike_price`, `side`, and `bid/ask`.
- **Assumptions and API Information**:
  - **Instrument Key Format**: The `instrument_key` is unique for each option and retrieved from the Upstox API.
  - **API Endpoint**: The function calls the Upstox API endpoint `https://api.upstox.com/v2/option/chain` with parameters for expiry date and option type (`PE` or `CE`).

Example API Response:


```
{
  "status": "success",
  "data": [
    {
      "expiry": "2024-11-28",
      "strike_price": 22450.0,
      "call_options": {
        "instrument_key": "NSE_FO|42853",
        "market_data": {
          "ask_price": 2177.9
        }
      },
      "put_options": {
        "instrument_key": "NSE_FO|42854",
        "market_data": {
          "bid_price": 19.5
        }
      }
    }
  ]
}
```

Function 2: `calculate_margin_and_premium`

- **Purpose**: Takes the DataFrame from `get_combined_option_chain_data`, calculates `margin_required` by querying the Upstox margin API, and computes `premium_earned`.
- **Inputs**:
    - `data`: The DataFrame output from `get_combined_option_chain_data` containing columns `instrument_key`, `strike_price`, `side`, and `bid/ask`.
- **Output**: Returns a DataFrame with additional columns `margin_required` and `premium_earned`.
- **Assumptions and API Information**:
    - **Lot Size**: Assumed default lot size is `50`, which can be modified if specific to the instrument.
    - **API Endpoint**: The Upstox margin API at `https://api.upstox.com/v2/charges/margin` is called to retrieve margin details.
    - **Example API Response**:


```
{
  "status": "success",
  "data": {
    "margins": [
      {
        "span_margin": 223816.5,
        "exposure_margin": 26686.1,
        "total_margin": 250502.6
      }
    ],
    "required_margin": 250502.6
  }
}
```


    - **Error Handling**:
        - **Missing or Incorrect `instrument_key`**: The function checks for valid `instrument_key` values before making API requests.
        - **Network Errors**: API request errors are caught with `try-except` blocks to handle network-related issues.
        - **Fallback Values**: Default values (`None` or `0`) are used for missing data.

---

3. Examples of Data Processing

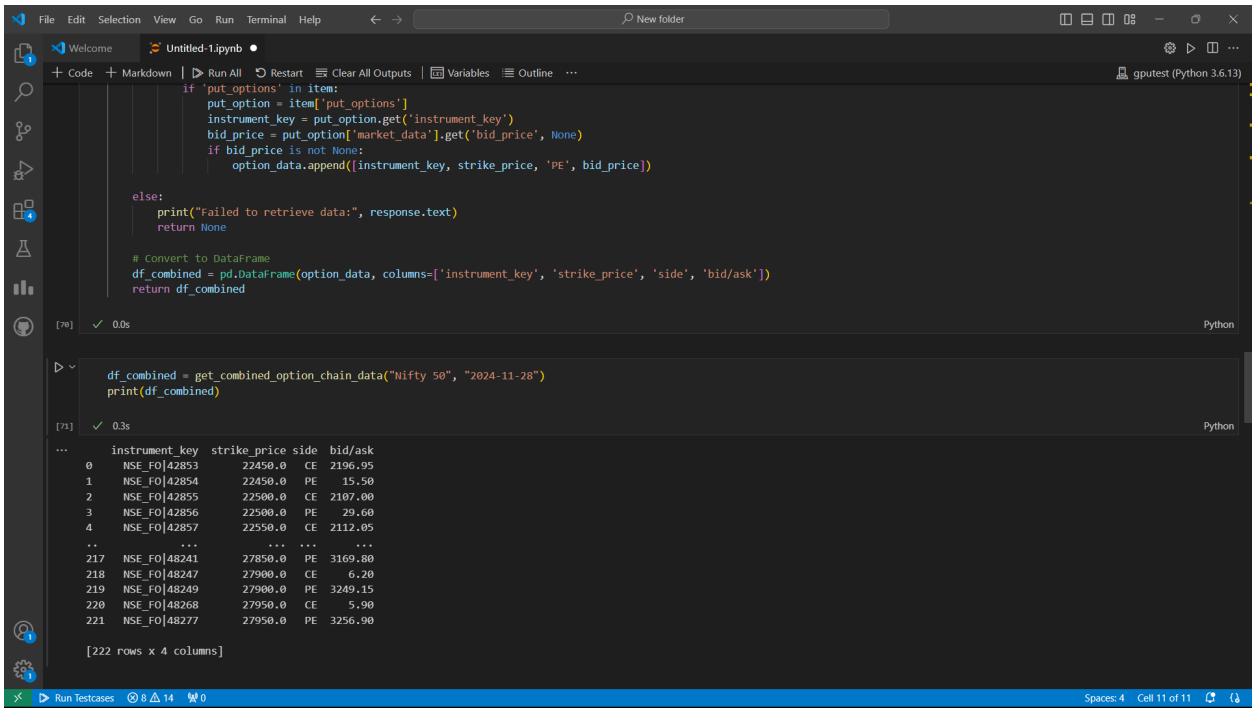
Step-by-Step Process in `calculate_margin_and_premium`:

1. **Payload Construction**: The function constructs the API payload with each option's `instrument_key`, transaction type `"SELL"`, and product type `"D"`.
2. **API Call**: It makes a POST request to fetch `required_margin`.
3. **Data Extraction**: If the response is successful and includes `required_margin`, it is extracted; otherwise, a fallback value is applied.
4. **Premium Calculation**: `premium_earned` is calculated as `bid/ask * lot size`.

4. AI Tools and Resources

- **AI Assistance**: ChatGPT was used to clarify the code structure, identify and implement effective error-handling strategies, and ensure efficient data retrieval and processing along with the understanding of financial terminology of the project.
- **Resources**:
    - **Upstox API Documentation**: Used for details on API endpoints, request payloads, and response structure.
    - **Python and Pandas Documentation**: For managing DataFrames and handling JSON data.

OUTPUTS



```
            if 'put_options' in item:
                put_option = item['put_options']
                instrument_key = put_option.get('instrument_key')
                bid_price = put_option['market_data'].get('bid_price', None)
                if bid_price is not None:
                    option_data.append([instrument_key, strike_price, 'PE', bid_price])

        else:
            print("Failed to retrieve data:", response.text)
            return None

        # Convert to DataFrame
        df_combined = pd.DataFrame(option_data, columns=['instrument_key', 'strike_price', 'side', 'bid/ask'])
        return df_combined
```

[70] ✓ 0.0s Python

```
df_combined = get_combined_option_chain_data("Nifty 50", "2024-11-28")
print(df_combined)
```

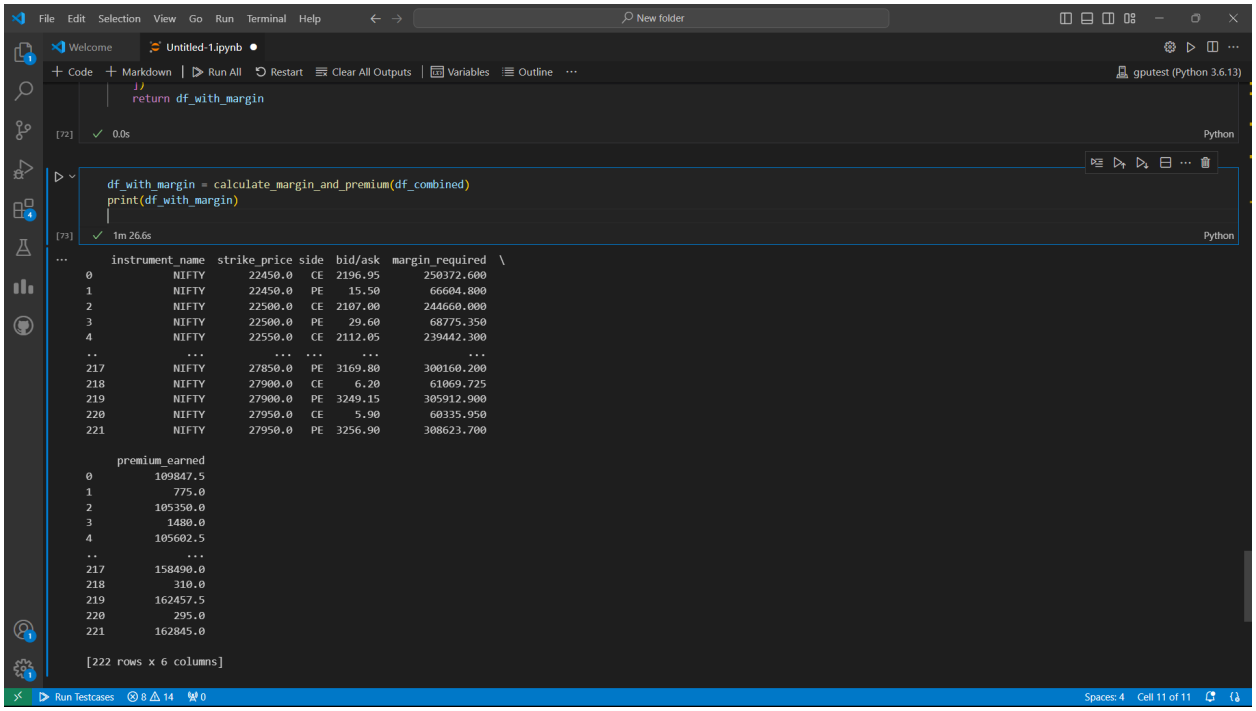
[71] ✓ 0.3s Python

```
     instrument_key  strike_price side  bid/ask
0     NSE_FO|42853       22450.0   CE  2196.95
1     NSE_FO|42854       22450.0   PE    15.50
2     NSE_FO|42855       22500.0   CE  2107.00
3     NSE_FO|42856       22500.0   PE    29.60
4     NSE_FO|42857       22550.0   CE  2112.05
..             ...           ...  ...      ...
217   NSE_FO|48241       27850.0   PE  3169.80
218   NSE_FO|48247       27900.0   CE     6.20
219   NSE_FO|48249       27900.0   PE  3249.15
220   NSE_FO|48268       27950.0   CE     5.90
221   NSE_FO|48277       27950.0   PE  3256.90

[222 rows x 4 columns]
```

Run Testcases ⊗ 8 △ 14 0 Spaces: 4 Cell 11 of 11

Part 1: Retrieve Option Chain Data



```
    return df_with_margin
```

[72] ✓ 0.0s Python

```
df_with_margin = calculate_margin_and_premium(df_combined)
print(df_with_margin)
```

[73] ✓ 1m 26.6s Python

```
     instrument_name  strike_price side  bid/ask  margin_required  \
0              NIFTY       22450.0   CE  2196.95       250372.600
1              NIFTY       22450.0   PE    15.50        66604.800
2              NIFTY       22500.0   CE  2107.00       244660.000
3              NIFTY       22500.0   PE    29.60        68775.350
4              NIFTY       22550.0   CE  2112.05       239442.300
..               ...           ...  ...      ...              ...
217            NIFTY       27850.0   PE  3169.80       300160.200
218            NIFTY       27900.0   CE     6.20        61069.725
219            NIFTY       27900.0   PE  3249.15       305912.900
220            NIFTY       27950.0   CE     5.90        60335.950
221            NIFTY       27950.0   PE  3256.90       308623.700

     premium_earned
0          109847.5
1             775.0
2          105350.0
3            1480.0
4          105602.5
..              ...
217        158490.0
218          310.0
219        162457.5
220          295.0
221        162845.0

[222 rows x 6 columns]
```

Run Testcases ⊗ 8 △ 14 0 Spaces: 4 Cell 11 of 11

Part 2: Calculate Margin and Premium Earned