In [10]:
```python
import pandas as pd # importing

# Load and reshape CSV
df = pd.read_excel(r"C:\Users\ASUS\Downloads\Financial_Summary_Microsoft_Tesla_Apple_2021_2023.xlsx") # path to excel file i have created in my device
csv_path = 'Financial_Summary_Microsoft_Tesla_Apple_2021_2023.csv' # Converted in csv
df.to_csv(csv_path, index=False)
df = pd.read_csv(csv_path)
df.columns = df.columns.str.strip()

# Reshape data
companies = ['Microsoft', 'Tesla', 'Apple']
metrics = ['Revenue', 'Net_Income', 'Assets', 'Liabilities', 'Cash_Flow']

long_df = pd.DataFrame()
for company in companies: # For Loops
    temp = pd.DataFrame()
    temp['Year'] = df['Year']
    temp['Company'] = company
    for metric in metrics:
        col_name = f"{company}_{metric}"
        temp[metric.replace('_', ' ')] = df[col_name]
    long_df = pd.concat([long_df, temp], ignore_index=True)

# Lists for guidance
available_companies = long_df['Company'].unique().tolist()
available_years = sorted(long_df['Year'].unique())
available_metrics = [m.replace('_', ' ') for m in metrics]

# Create basic chatbot knowledge base
chatbot_data = {
    "What is the total revenue of Microsoft in 2023?": long_df.loc[(long_df['Company'] == 'Microsoft') & (long_df['Year'] == 2023), 'Revenue'].values[0],
    "What is the net income of Apple in 2022?": long_df.loc[(long_df['Company'] == 'Apple') & (long_df['Year'] == 2022), 'Net Income'].values[0],
    "How has Tesla's net income changed from 2021 to 2023?": (
        long_df.loc[(long_df['Company'] == 'Tesla') & (long_df['Year'] == 2023), 'Net Income'].values[0]
        - long_df.loc[(long_df['Company'] == 'Tesla') & (long_df['Year'] == 2021), 'Net Income'].values[0]
    ),
    "What is the cash flow of Microsoft in 2021?": long_df.loc[(long_df['Company'] == 'Microsoft') & (long_df['Year'] == 2021), 'Cash Flow'].values[0],
}
```

In [7]:
```python
# Smarter rule-based chatbot
def simple_chatbot(user_query):
    user_query = user_query.strip().lower() # User can write it in lower cases also

    # Predefined queries
    if user_query in [q.lower() for q in chatbot_data]:
        key = next(k for k in chatbot_data if k.lower() == user_query)
        if "changed" in key:
            change = chatbot_data[key]
            trend = "increased" if change > 0 else "decreased"
            return f"Tesla's net income has {trend} by ${abs(change):,.2f} from 2021 to 2023."
        else:
            return f"{key} Answer: ${chatbot_data[key]:,.2f}"

    # List companies
    if "companies" in user_query or "company" in user_query: # you can use this for asking like Companies it will show details of companies
        return f"The available companies are: {', '.join(available_companies)}"

    # List years
    if "years" in user_query or "year" in user_query:
        return f"The available years are: {', '.join(map(str, available_years))}"

    # List metrics
    if "metrics" in user_query or "data" in user_query or "available fields" in user_query:
        return f"Available financial metrics are: {', '.join(available_metrics)}"

    # Generic query parser (e.g., "revenue of tesla in 2022")
    for company in available_companies:
        if company.lower() in user_query:
            for metric in available_metrics:
                if metric.lower() in user_query:
                    for year in available_years:
                        if str(year) in user_query:
                            value = long_df.loc[
                                (long_df['Company'].str.lower() == company.lower()) &
                                (long_df['Year'] == year),
                                metric
                            ]
                            if not value.empty:
                                return f"{metric} of {company} in {year} is ${value.values[0]:,.2f}"
                            else:
                                return "Sorry, that data is not available."

    # Fallback
    return "Sorry, I can only answer questions about Company Financials. Try asking about revenue, net income, or list companies."
```

In [12]:
```python
# Run chatbot
print("\n--- Simple Enhanced Chatbot Ready :) ---") # Some Commands you can ask for Testing Purposes!! I know you will :)
print("You can ask me Like: \n1)What is the total revenue of Microsoft in 2023?")
print("\n 2)How has Tesla's net income changed from 2021 to 2023?")
print("\n 3)Companies, Years, Metrics")
print("\n -- Type 'exit' to quit Enhanced Chatbot :( -- ")
while True:
    user_input = input("You: ")
    if user_input.lower() == 'exit': # here while loop will stop after typing exit
        print("Chatbot: Goodbye :)")
        break
    response = simple_chatbot(user_input)
    print("Chatbot:", response)
```

```
--- Simple Enhanced Chatbot Ready :) ---
You can ask me Like:
1)What is the total revenue of Microsoft in 2023?

 2)How has Tesla's net income changed from 2021 to 2023?

 3)Companies, Years, Metrics

 -- Type 'exit' to quit Enhanced Chatbot :( --
Chatbot: The available years are: 2021, 2022, 2023
Chatbot: The available companies are: Microsoft, Tesla, Apple
Chatbot: Net Income of Tesla in 2021 is $5,519.00
Chatbot: Goodbye :)
```

In [ ]:

Financial Chatbot Documentation

Overview

This rule-based chatbot responds to predefined financial queries using data from Microsoft, Tesla, and Apple (2021–2023).

Technologies Used

- Python
- Pandas
- CSV format for structured data

Predefined Queries Supported

- What is the total revenue of Microsoft in 2023?
- What is the net income of Apple in 2022?
- How has Tesla's net income changed from 2021 to 2023?
- What is the operating income of Apple in 2021?
- List Companies, years, metrics.
- You can ask it lower cases and upper cases so the Chatbot can write the Answers.

How It Works

- Financial data from Excel is converted to CSV and loaded using pandas.
- Each query is mapped to a response using `if-else` logic.
- User input is matched against predefined questions to return answers.
- Handles unrecognized queries gracefully with a fallback message.

Limitations

- Only supports exact, predefined queries.
- No natural language flexibility.

- Does not handle real-time data or user history (state).

In [ ]: