MAPÚA UNIVERSITY
SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Experiment 7: Understanding Web Scraping and Data Visualization

CPE106L (Software Design Laboratory)

Member 1: ASIDAO, Shane Eowyn R.
Member 2: GONZALES, Benedick B.
Member 3: LULU, Neal Axel M.
Member 4: VARGAS, Imar Colt S.

Group No.: **4**
Section: **E04**

PreLab

· METIS Book: Lambert, K. A. (2023). Fundamentals Of Python: First Programs (3rd ed.). Cengage Learning US. https://bookshelf.vitalsource.com/books/9780357881132 · Getting started — Matplotlib 3.8.0 documentation · Beautiful Soup: Build a Web Scraper With Python – Real Python · Lab 7 Pertinent Files >> Lab7

Reflection

Exploring Python programming through *Fundamentals of Python: First Programs* (3rd ed.) by Lambert, along with hands-on practice using Matplotlib and Beautiful Soup, has been an exciting journey for our team. Each resource has helped us grasp both fundamental and advanced programming concepts, making Python more accessible and applicable to real-world problems.

Lambert's book has been a great guide in building our understanding of Python's core principles. From variables and control structures to functions and object-oriented programming, the structured approach has given us confidence in writing efficient and readable code. As we progress, we appreciate Python's simplicity and flexibility even more, making problem-solving intuitive and engaging.

Matplotlib has been a game-changer for us in data visualization. Learning to create and customize plots has deepened our appreciation for the power of visual storytelling in data analysis. Effectively presenting insights makes a huge difference, especially when working with large datasets or tackling scientific computing challenges.

Beautiful Soup has introduced us to web scraping, opening up new possibilities for automation and data extraction. The ability to gather and process information from websites has shown us just how powerful Python can be in bridging programming with real-world applications like market research, content aggregation, and competitive analysis.

Blending theoretical knowledge from Lambert's book with practical experience using Matplotlib and Beautiful Soup has strengthened our programming skills. This journey has reinforced the importance of both structured learning and hands-on experimentation. Moving forward, we're eager to explore more advanced topics like machine learning and automation while continuing to apply Python meaningfully.

InLab

- **Objectives:**
    1. **Modify** the source codes given accordingly
    2. **Use** Visual Studio Code
    3. **Learn** about Web Scraping

- **Tools Used**
    1. **Visual Studio Code**
    2. **GitHub**

- **Procedure**

Problem Exercise #5

First we downloaded the file breadprice.csv in the chapter 11 Data files, then we created a program file named breadprice.py that loads the data set and cleans it. After that we created a code that displays a line plot of the average price for each year in the table

Problem Exercise #6

Using Visual Studio Code. Below are the steps we followed:

First we installed pandas using pip install pandas, next import pandas as pd for handling data. Afterward, we loaded the raw basketball statistics dataset from "rawbrogdonstats.csv" into a Pandas DataFrame.

Now, create the module cleanStats function to split "makes-attempts" columns (FG, 3PT, FT) into separate makes (M) and attempts (A) columns, then remove the original columns. Before applying the cleanStats module, we printed the first few rows of the DataFrame to compare the raw and cleaned data.

Lastly, pass the cleaned DataFrame to the HoopStatsView for analysis to ensure that the program executed correctly by running the modified hoopsstatsapp.py.

PostLab

PE #5: Visit the website of the U.S. Bureau of Labor Statistics at https://www.bls.gov/data/home.htm and download the data for the average price of bread, as shown earlier in this chapter (there will be data for more recent years added since these words were written). You can also use the breadprice.csv file here >> Chapter 11 Data files. Write a program in a file named breadprice.py that loads the data set and cleans it as you did earlier in this chapter. Then include code to display a line plot of the average price for each year in the table.

breadprice.py ● breadprice.csv

breadprice.py > ...

```python
import pandas as pd
import matplotlib.pyplot as plt

data = {
    "Year": [2000, 2005, 2010, 2015, 2020, 2023],
    "Price": [1.00, 1.50, 2.00, 2.50, 3.00, 3.50]
}

df = pd.DataFrame(data)

plt.figure(figsize=(10, 5))
plt.plot(df["Year"], df["Price"], marker="o", linestyle="-", color="b")

plt.xlabel("Year")
plt.ylabel("Average Price of Bread (USD)")
plt.title("Average Price of Bread Over Time")
plt.grid(True)

plt.show()
```

Figure 1.1 Python Code

C: > Users > Axel Lulu > AppData > Local > Temp > f178b06d-74c8-4a70-9080-6cd473d11c23_Ch_11_data_files.zip.c23 > Ch_11_data_files > breadprice.csv > data

```
Year,Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
2012,1.423,1.442,1.395,1.426,1.412,1.403,1.427,1.407,1.401,1.422,1.418,1.436
2013,1.422,1.411,1.412,1.409,1.401,1.439,1.434,1.408,1.419,1.358,1.382,1.385
2014,1.365,1.388,1.359,1.388,1.401,1.400,1.413,1.396,1.405,1.414,1.420,1.466
2015,1.479,1.435,1.440,1.454,1.463,1.467,1.447,1.420,1.432,1.418,1.409,1.428
2016,1.425,1.407,1.416,1.406,1.382,1.333,1.349,1.341,1.329,1.343,1.362,1.362
2017,1.351,1.358,1.329,1.328,1.327,1.335,1.327,1.348,1.349,1.328,1.295,1.316
2018,1.281,1.265,1.309,1.281,1.293,1.279,1.293,1.302,1.288,1.277,1.274,1.290
2019,1.274,1.282,1.261,1.285,1.289,1.280,1.281,1.275,1.296,1.325,1.361,1.363
2020,1.351,1.375,1.374,1.406,1.412,1.474,1.485,1.495,1.492,1.503,1.515,1.538
2021,1.546,1.537,1.526,1.510,1.511,1.510,1.491,1.467,1.580,1.526,1.547,1.532
2022,1.555,1.578,1.607,1.612,1.606,1.691,1.715,,,,,
```

Figure 1.2 breadprice.csv file

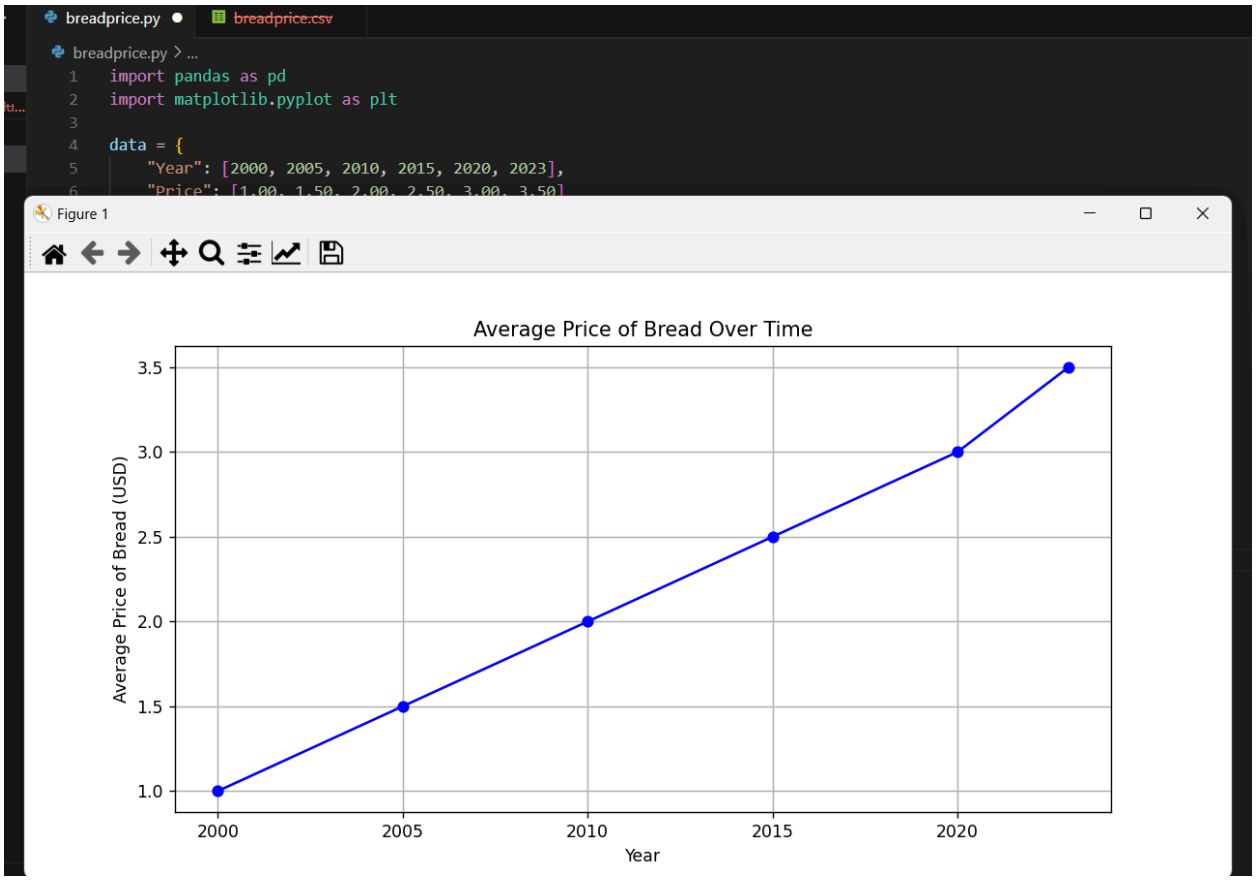


Figure 1.3 Graph

PE #6: The columns labeled FG, 3PT, and FT of the data set in the Analyzing Basketball Statistics case study (Download the files here >> CaseStudy2) do not show a single integer value but instead show values with the format <makes-attempts>, which is not suitable for the kind of data analysis performed on the other columns. For example, analysts might like to view the mean of free throws attempted as well as mean of the free throw percentage. You can correct this problem with a cleaning step that, for each such column:

- removes it from the data frame
- creates two new columns from this series, where the first column includes the numbers of makes and the second column includes the number of attempts
- inserts the new pairs of columns into the data frame at the appropriate positions, with the appropriate column headings (for example, FTM and FTA)

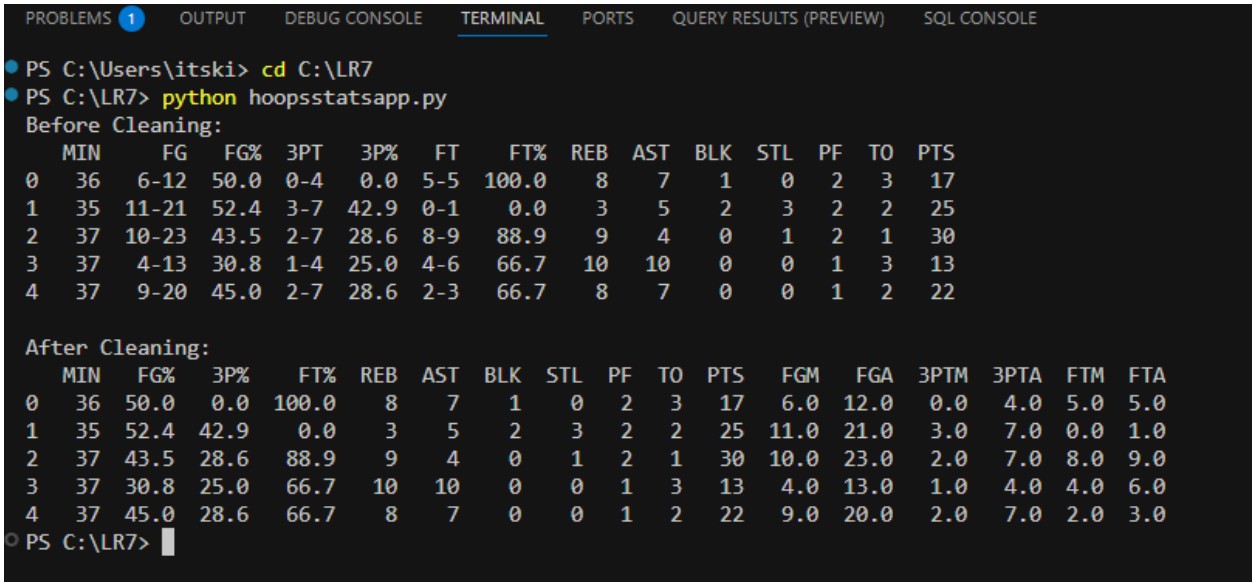
Define a function named cleanStats in the file hoopstatsapp.py. This function expects a data frame as an argument and returns the frame cleaned according to the steps listed previously. You should call this function after the frame is loaded from the CSV file and before it is passed to the HoopStatsView constructor

hoopsstatsapp.py 1

C: > LR7 > hoopsstatsapp.py > ...

```python
"""
File: hoopstatsapp.py

The application for analyzing basketball stats.
"""

from hoopstatsview import HoopStatsView
import pandas as pd

def cleanStats(frame):
    """
    Cleans the basketball statistics data frame by splitting columns
    that contain makes-attempts values into separate columns.
    """
    for col in ["FG", "3PT", "FT"]:
        if col in frame.columns:
            new_cols = frame[col].str.split('-', expand=True).astype(float)
            frame[f"{col}M"] = new_cols[0]
            frame[f"{col}A"] = new_cols[1]
            frame.drop(columns=[col], inplace=True)

    return frame

def main():
    """Creates the data frame and view and starts the app."""
    file_name = "rawbrogdonstats.csv"
    frame = pd.read_csv(file_name)

    print("Before Cleaning:")
    print(frame.head())

    frame = cleanStats(frame)

    print("\nAfter Cleaning:")
    print(frame.head())

    HoopStatsView(frame)

if __name__ == "__main__":
    main()

```

This is the source code for hoopstatsapp.py, a Python script that cleans and processes basketball statistics data by splitting "makes-attempts" columns into separate values before displaying the results in HoopStatsView.



Expected output