****

**Data Viewer for Structure and Characteristics of Urban Areas in Rostock City**

**Project Documentation**

**Antonious Shehata - GH1028661 Mehmet Akif özkoç - GH1026518**

**Semester:** SS0324

**M604A Advanced Programming**

**Submission Date:** July 2024

**Table of Contents**

1. **Abstract .............................................................................. 2**
2. **Introduction ..................................................................... 2**
3. **Rostock and its Urban Areas …………………………….. 2**
4. **Project Overview .......................................................... 3**
5. **User Interface (UI) ....................................................... 3**
6. **Brief Description ........................................................... 4**
7. **Main Details .................................................................... 4**
8. **Interface description................................................... 5**
9. **Screen Shots for interface ................................ 5-10**
10. **Dataset integration process............................ 10-11**
11. **Code functionality ............................ 11 - 15**
12. **Conclusion .......................... 16**
13. **References ................ 16**
14. **Appendix A ........................................... 17**
    * **Appendix A: Project Source Code …………………………. 17-29**

**Data Viewer for Structure and Characteristics of Urban Areas in Rostock City**

**Abstract**

The Data Viewer for Structure and Characteristics of Urban Areas in Rostock City is a specialized tool designed to analyze and visualize demographic data. It provides insights into various aspects such as age distribution, gender demographics, household structures, and more, focusing specifically on urban areas within Rostock. This tool supports informed decision-making for policymakers and urban planners, aiming to enhance resource allocation and community development strategies.

**Introduction**

In contemporary governance and urban planning, data-driven insights are essential. The Data Viewer for Structure and Characteristics of Urban Areas in Rostock City serves as a critical instrument for understanding demographic trends within specific urban zones of Rostock. By examining key indicators like age groups, gender ratios, and household compositions, this tool aids in identifying priority areas for targeted interventions and service improvements. It empowers decision-makers to formulate effective policies that address the unique needs of urban residents, fostering sustainable urban development and improving overall quality of life.

**Rostock and Its Urban Areas**

Rostock, located on the northern coast of Germany along the Baltic Sea, is known for its rich history, vibrant culture, and economic significance. The city comprises diverse urban areas, each with unique demographic profiles shaped by factors such as historical development, economic activities, and social dynamics. The Data Viewer focuses on these urban areas, providing a detailed analysis of their demographic structure and characteristics to support local governance and urban planning initiatives.

**Project Overview**

**Components and Features**

**Data Source:**

The tool accesses demographic data from an Excel file named "Rostock.xlsx," which categorizes information into structured topics and subtopics.

**User Interface (UI):**

**Graphical Interface (GUI):**

**Developed using Python's Tkinter library, the UI features:**

Dropdown menus for selecting urban areas, main topics (e.g., Age, Gender), and subtopics (specific data points within topics).

A text area to display detailed comparative data between selected urban areas.

Graphical representations like bar graphs for visualizing trends and comparisons based on selected data.

**Features**

**Urban Area Selection:** Users can choose specific urban areas within Rostock for comparative demographic analysis.

**Topic and Subtopic Selection:** Dropdown menus enable selection of main topics (e.g., Age, Gender) and specific subtopics (e.g., Median Age, Household Structure).

**Data Display:** Upon selection, the tool generates a comparative table within the text area, providing a breakdown of data points for the selected urban areas across chosen subtopics.

**Graphical Representation:** Data is visually presented through interactive bar graphs, facilitating easy interpretation of trends and disparities between urban areas over specified periods.

**Brief Description:**

**Project Name:**

Data Viewer for Structure and Characteristics of Urban Areas in Rostock City

**Developing Language and consul used:**

Python, visual studio code

**Python Version to be installed:**

python 3.11.8 64 bit

**Libraries to be installed:**

tkinder, PIL, matplotlib, openpyxl,numpy, os

**Import from tkinder:**

ttk

**Import from PIL:**

Image, ImageTK

**Import from matplotlib:**

FigureCanvasTkAgg, matloplotlib.pyplot

**Import from matplotlib.figure:**

Figure

**Project Link:**

<https://github.com/AntoniousShehata/Rostock.git>

**Main Details:**

**Main important files:**

Rostock.xlsx as a dataset, Reports folder with Rostock.xlsx

**Number of pages:**

Three Pages

**Name of pages:**

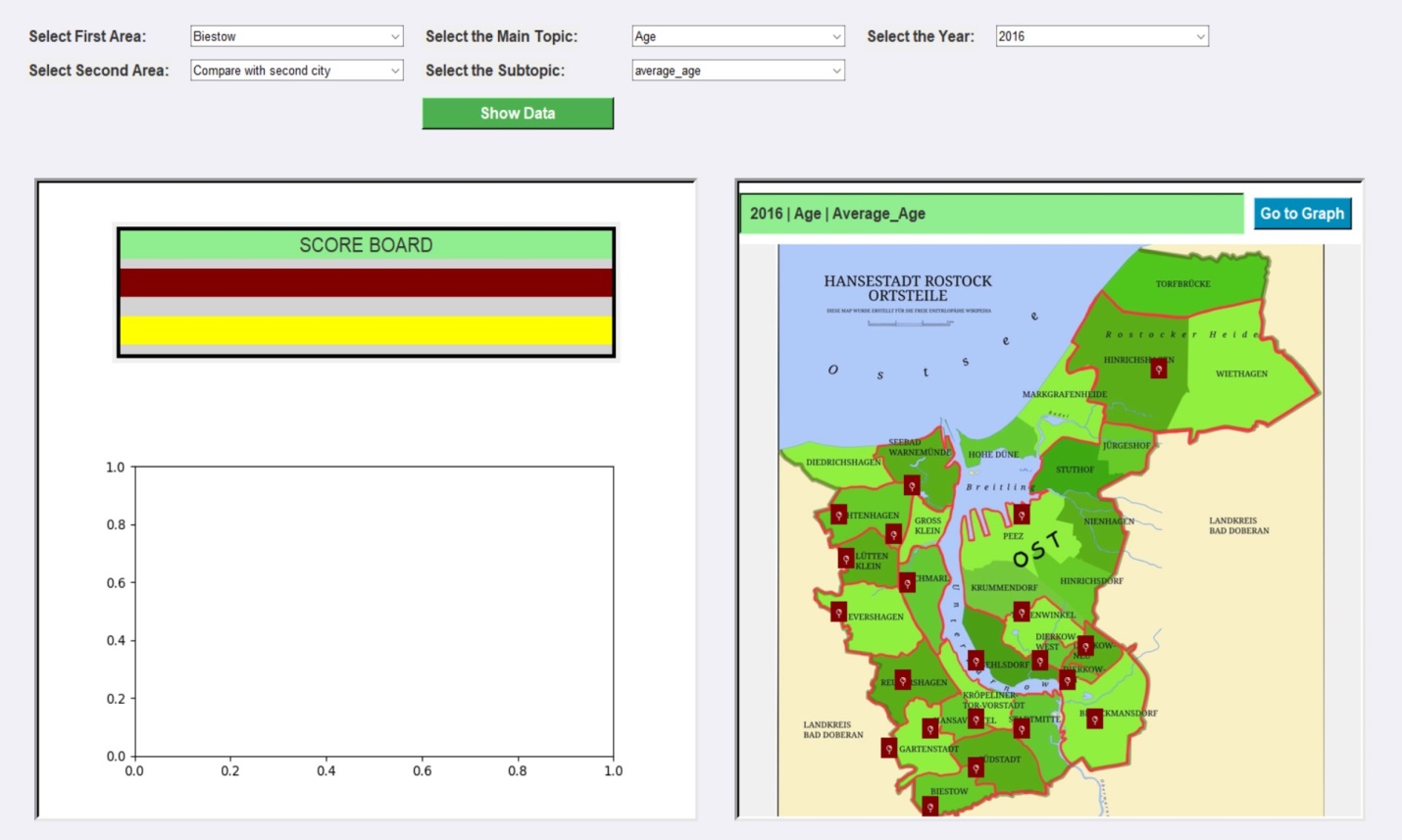
Home.py, dataload.py, styles.py, mergeprocess.py

**Interface description:**

**Main screen:**

Frame named **“Data Viewer for Structure and Characteristics of Urban Areas in Rostock City”**

**Screen Shots for interface:**

****

**Consists of:**

7 Labels, 5 dropdowns, 2 buttons, Score board, interactive map.

**7 Labels:**

**“Select First Area”** label points to its drowndown to choose first area (needed).

**“Select Second Area”** label points to its drowndown to choose Second area (Optional).

**“Select The Main Topic”** label points to its drowndown to choose Main Topic (needed).

**“Select The Subtopic”** label points to its drowndown to choose Subtopic (needed).

**“Select The Year”** label points to its drowndown to choose Year (map filter).

**“Score Board”** Green label points to data displayed highest and lowest values (statistics).

**“2016 | Age | Average\_Age”** label above the map it is a dynamic title for the displayed data.

**5 Dropdowns:**

**“Select First Area”** drowndown to choose first area 1 of 21 items (needed).

**“Select Second Area”** drowndown to choose Second area 1 of 21 items (Optional).

* default value “Compare with second city” means no second city chosen.

**“Select The Main Topic”** drowndown to choose Main Topic 1 of 9 items (needed).

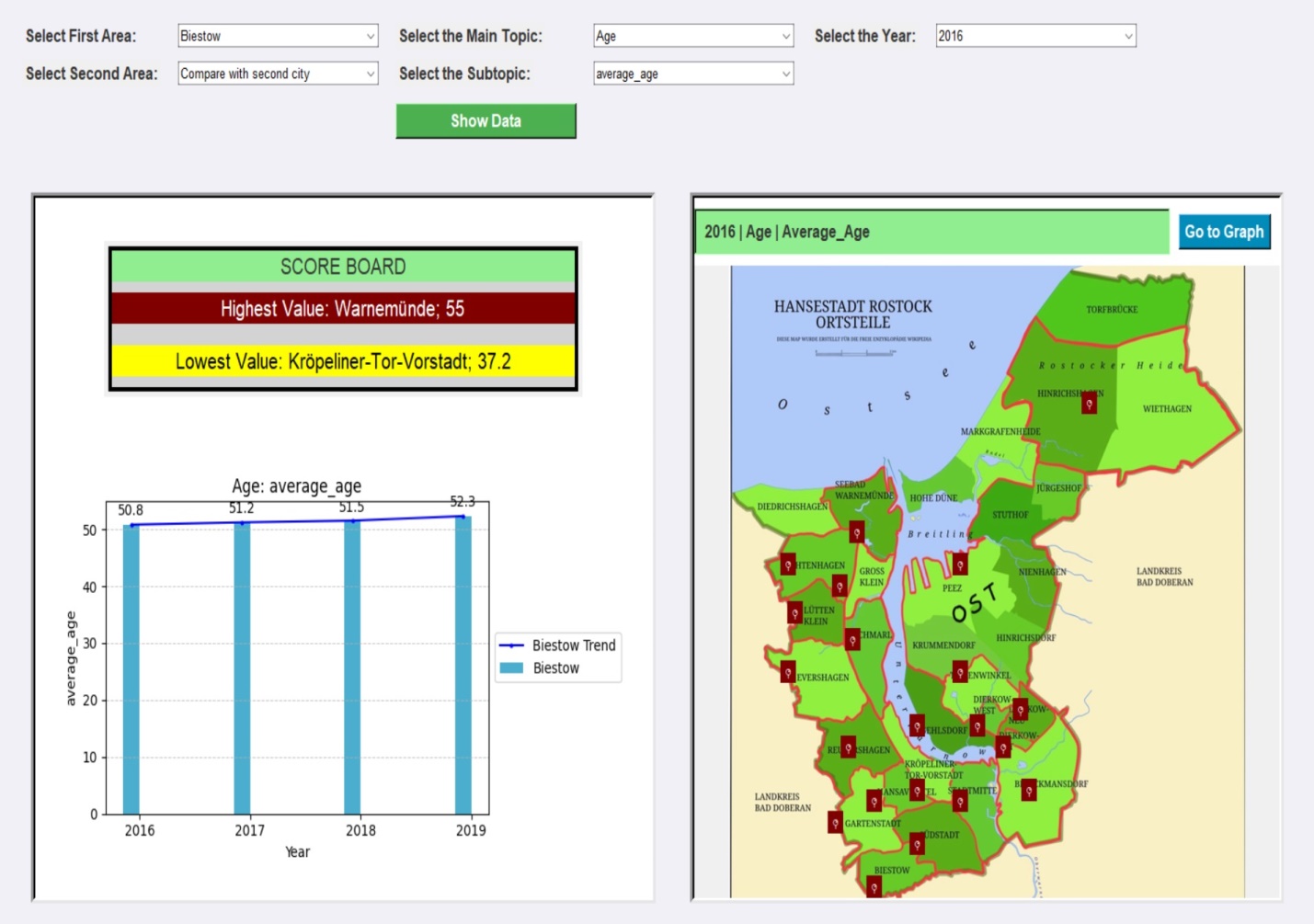
**“Select The Subtopic”** drowndown to choose Subtopic with respect to the main topic (needed).

**“Select the year”** drowndown to choose a year to display its data on the map (Optional).

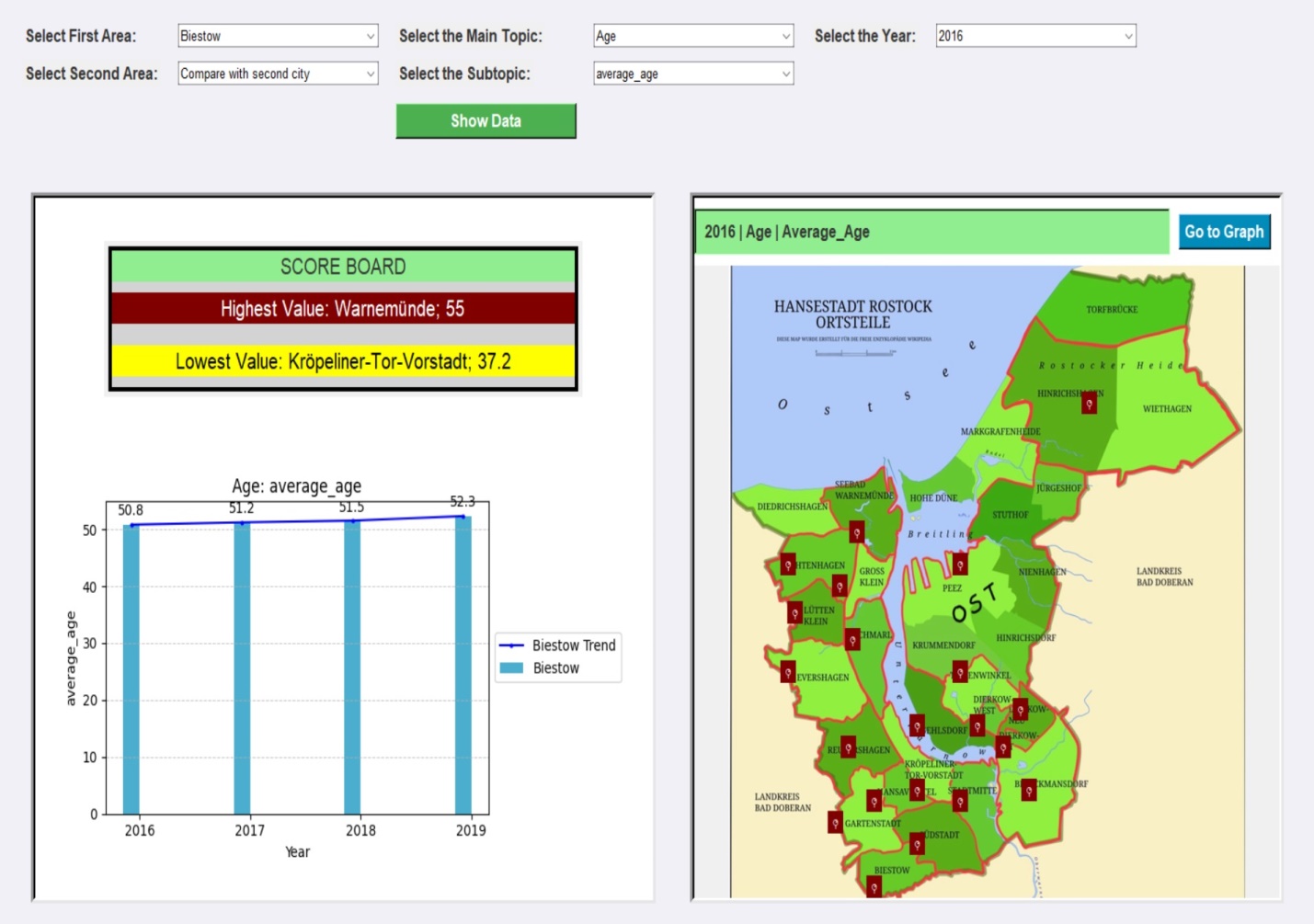
**2 Buttons:**

**“Show Data”** display data selected from three/four dropdowns to the graph and the text area.

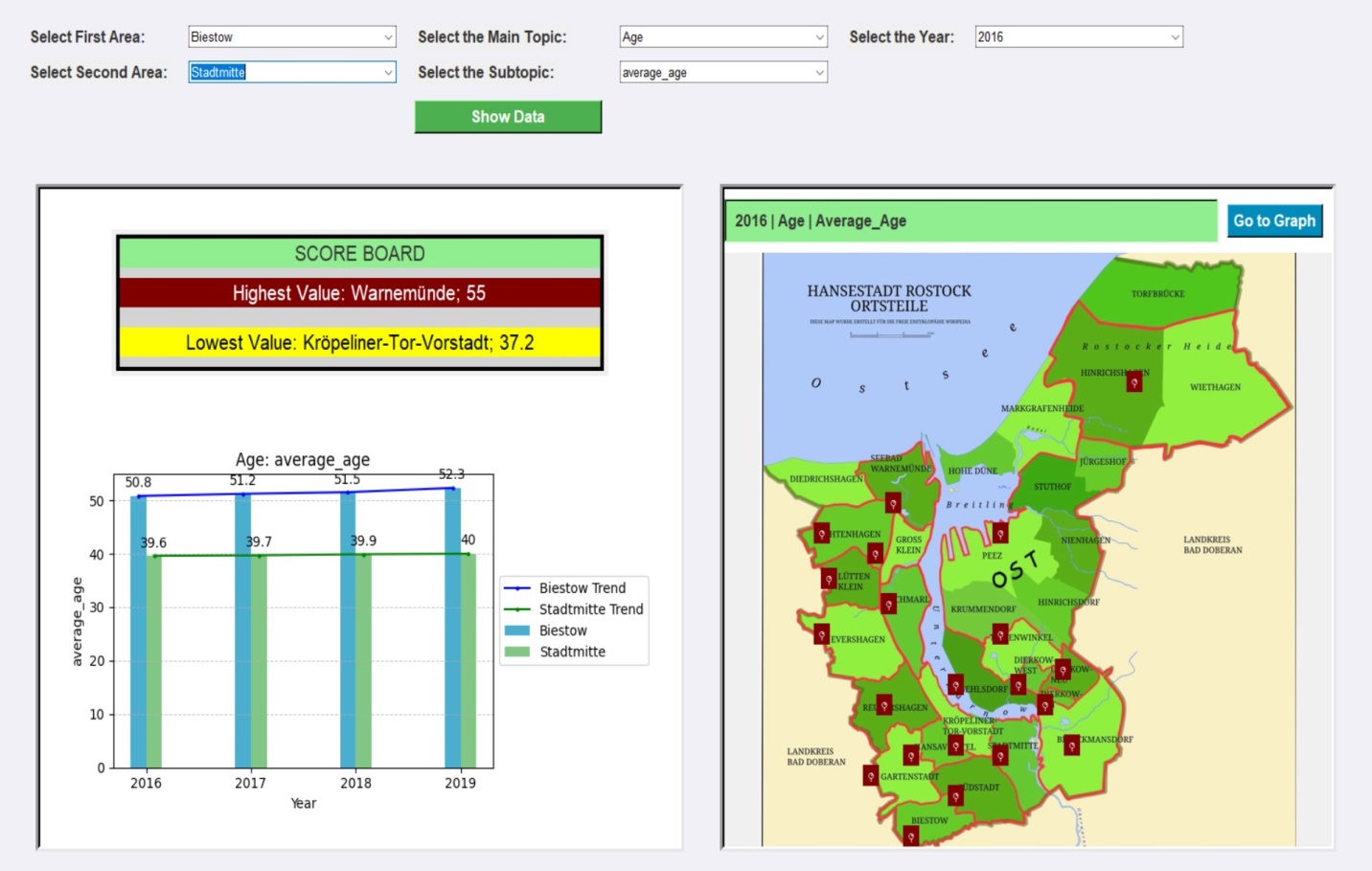
**“Go to Graph”** display data selected: subtopic for 21 areas at a selected year.

****

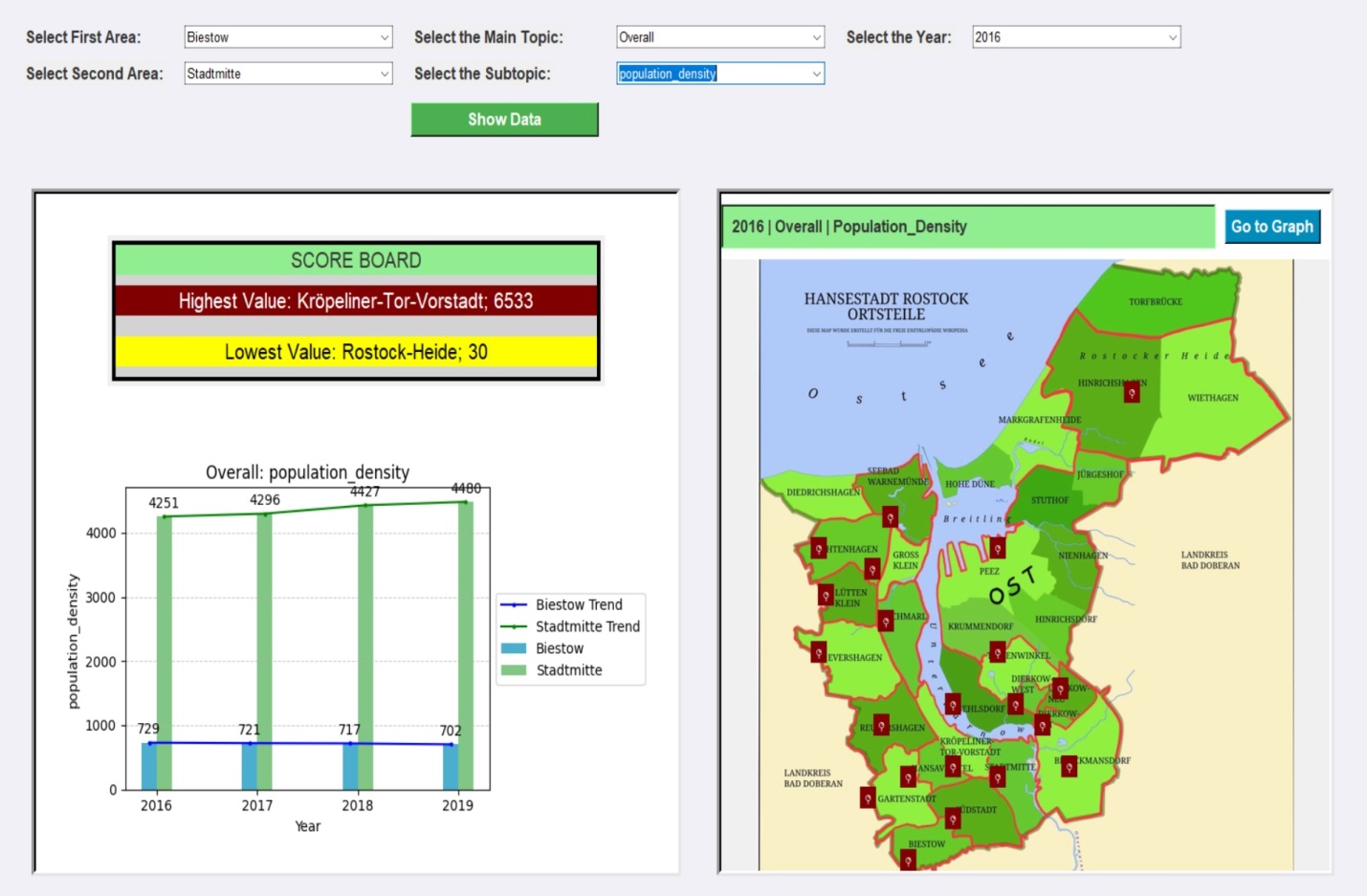
**Change First Area**

****

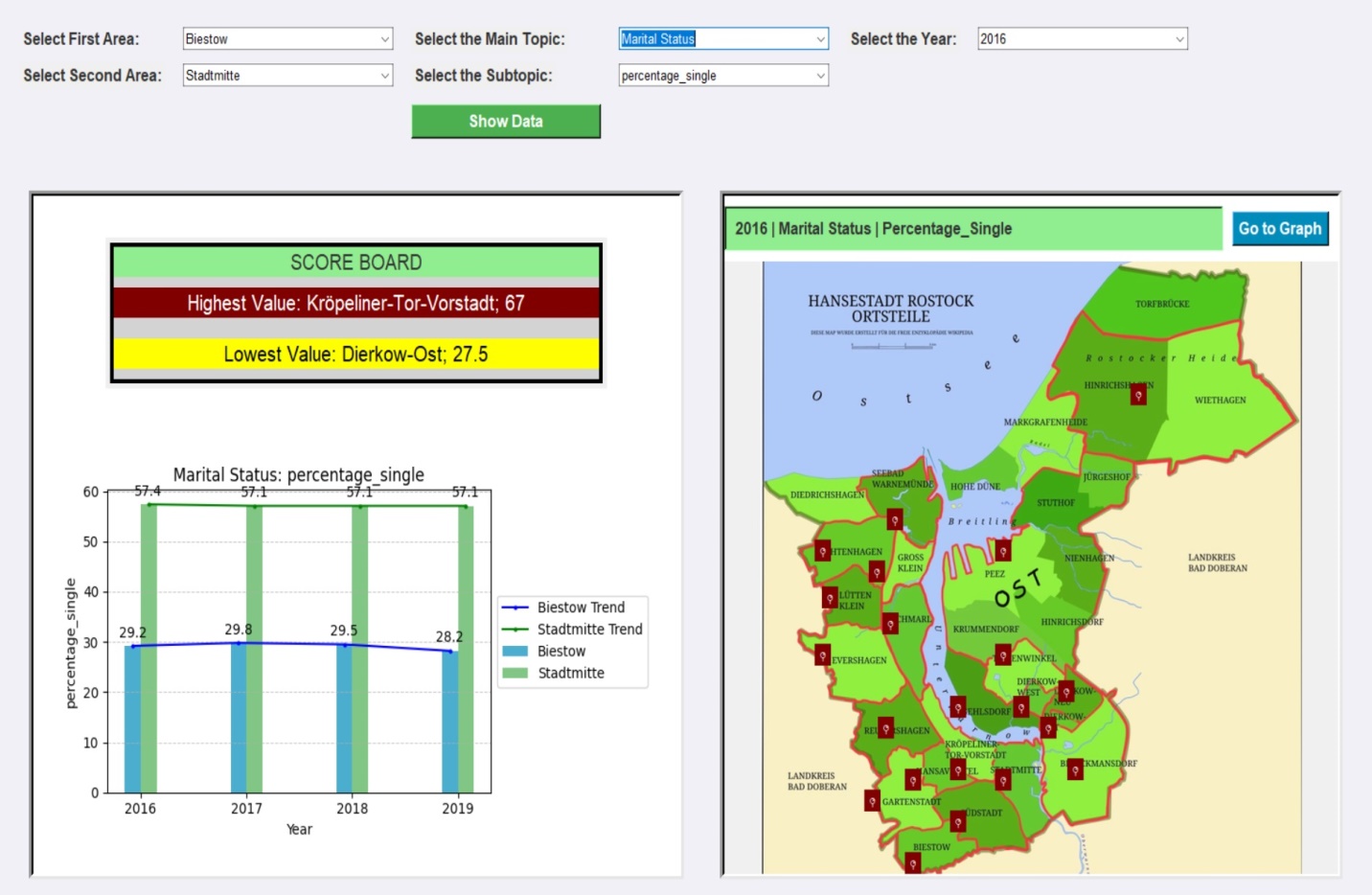
**Change Second Area**

****

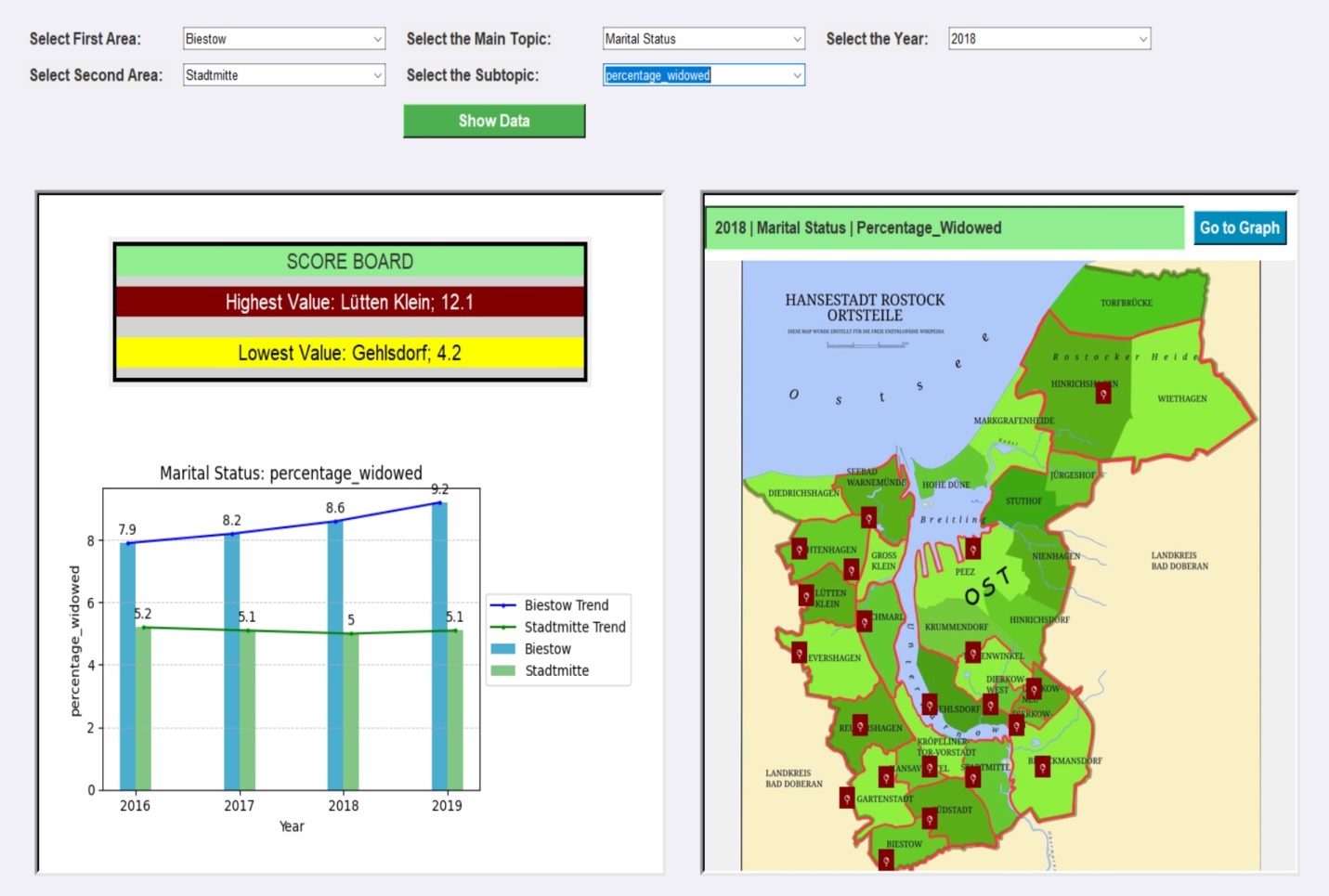
**Change the main topic**

****

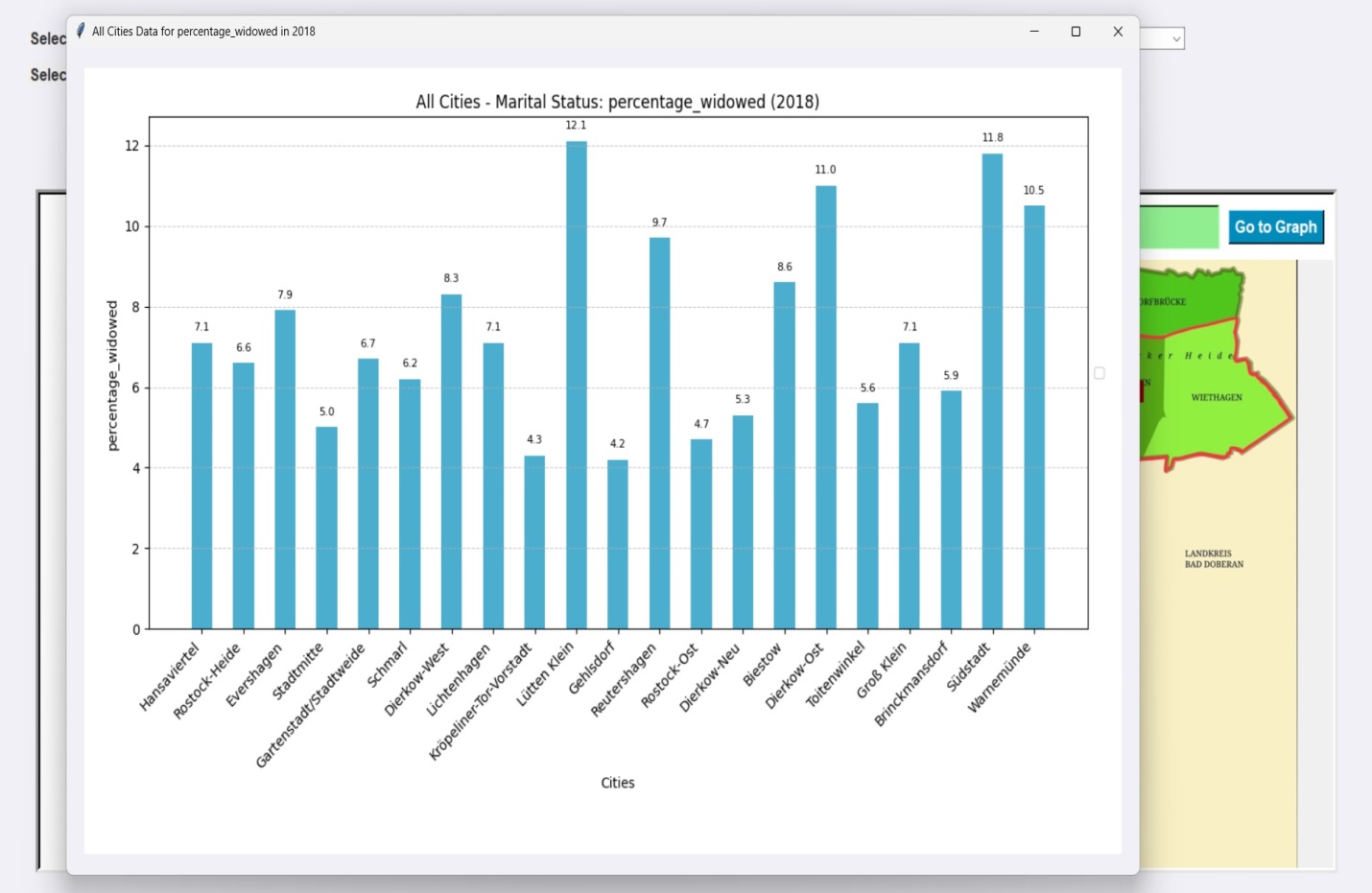
**Change the subtopic**

****

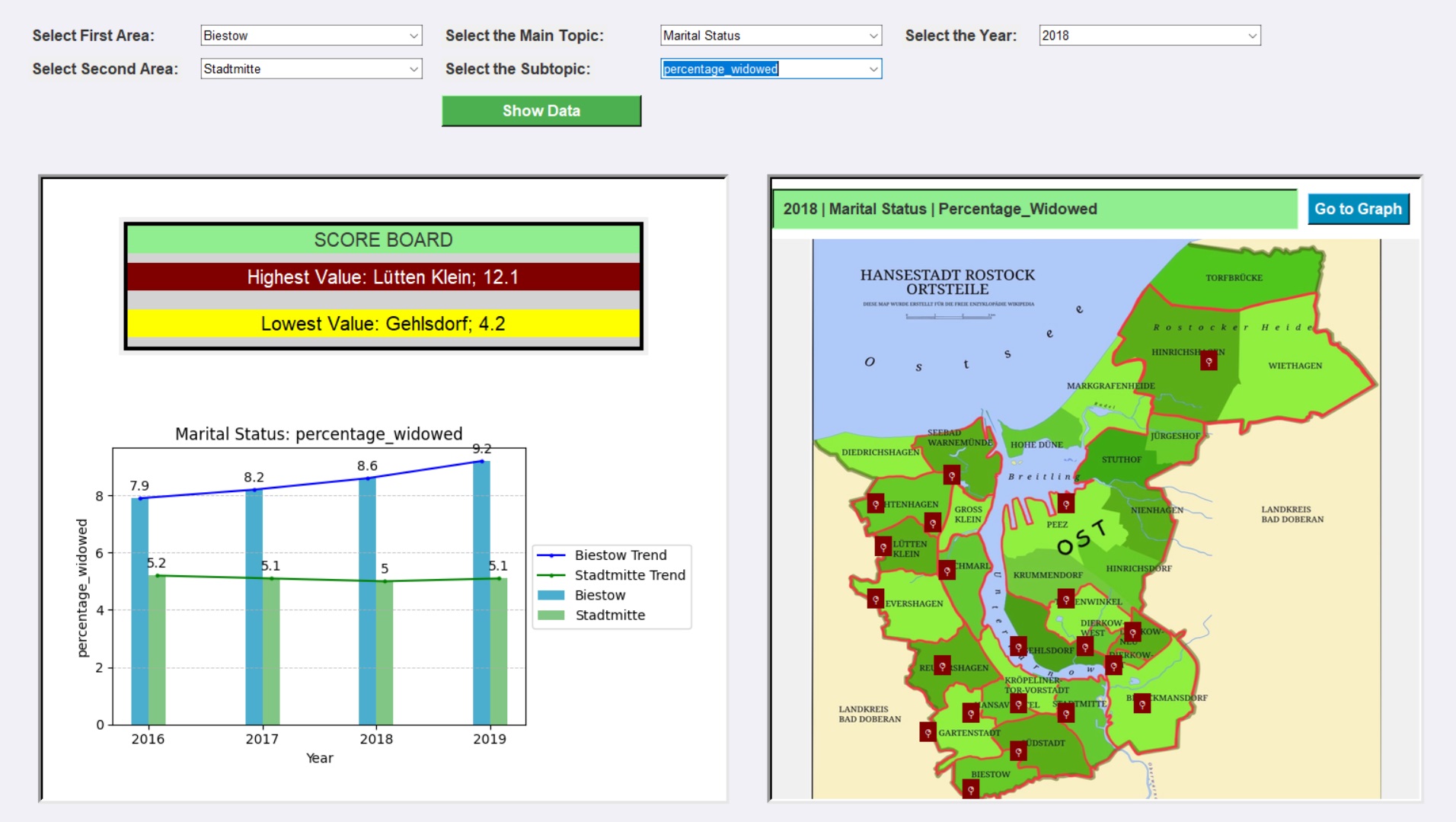
**Change the year**

****

**Click “Got to graph”**

****

**Score Board**

****

**Dataset integration process: (See Appendix A: Project Source Code Figure 1-4)**

**Purpose:**

The code reads data from multiple Excel files stored in different directories corresponding to different years (2016 to 2019), processes this data, combines it into a single dataset, renames columns according to a specified mapping, and finally saves the combined and processed data into a new Excel file named 'Rostock.xlsx'.

**Steps and Explanation:**

1. **Imports**:
   * import os: Used for interacting with the operating system (e.g., file paths).
   * import pandas as pd: Used for data manipulation and analysis.
2. **Directories and File Names**:
   * directories: List of directory paths where data files for each year are stored.
   * file\_names: List of specific Excel file names to be read from each directory.
3. **Function read\_and\_prepare\_data**:
   * Reads an Excel file from a given file\_path.
   * Removes any unnamed columns that may arise from merged cells.
   * Drops duplicate rows based on specified columns (stadtbereich\_code and stadtbereich\_bezeichnung).
4. **Processing Loop**:
   * Iterates through each directory (directories).
   * Reads each specified Excel file (file\_names) using read\_and\_prepare\_data.
   * Concatenates the cleaned data frames (all\_dfs) horizontally (axis=1).
   * Adds a year column based on the directory name to each concatenated dataset.
   * Inserts stadtbereich\_code and stadtbereich\_bezeichnung columns from the first sheet into desired positions.
5. **Combining Data Across Years**:
   * Stores each processed dataset (final\_data) in a dictionary (all\_years\_data) with the year as the key.
   * Concatenates all datasets from all\_years\_data into a single dataframe (combined\_data).
6. **Column Renaming**:
   * Renames columns in combined\_data according to a predefined mapping (column\_rename\_map).
7. **Saving Data**:
   * Saves the final processed dataset (combined\_data) to an Excel file 'Rostock.xlsx'

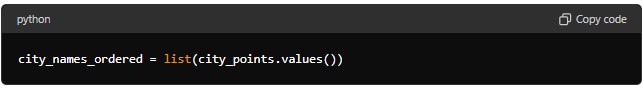
**Summary of integration:**

This script effectively gathers data from multiple Excel files across several years, processes and cleans the data, combines it into a unified dataset, standardizes column names, and exports the consolidated data to a new Excel file. It utilizes pandas for data manipulation and “os” for file handling, ensuring that the final output meets the specified format and content requirements.

**Code functionality:**

**main.py: (See Appendix A: Project Source Code Figure 5)**

**1. City\_names\_ordered (See Appendix A: Project Source Code Figure 6)**



This list contains city names ordered by city points. The points can be used to rank or sort cities based on specific criteria. This get its data from “dataload.py”

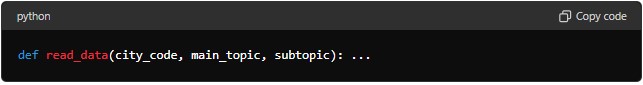
**2. Function: read\_subtopics(main\_topic) (See Appendix A: Project Source Code Figure 7)**

****

* **Purpose:** Reads and returns subtopics related to the selected main topic.
* **Parameters:** main\_topic (str) - The main topic for which subtopics are to be read.
* **Returns:** List of subtopics related to the main topic.

**3. Function: read\_data(city\_code, main\_topic, subtopic)**

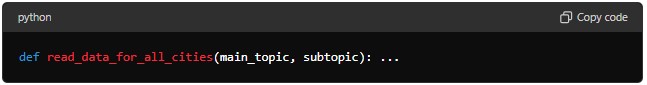
**(See Appendix A: Project Source Code Figure 8)**

****

* **Purpose:** Reads data for a specific city and subtopic.
* **Parameters:**
  + city\_code (str) - The code representing the specific city.
  + main\_topic (str) - The main topic of interest.
  + subtopic (str) - The subtopic under the main topic.
* **Returns:** Data related to the specified city, main topic, and subtopic.

**4. Function: read\_data\_for\_all\_cities(main\_topic, subtopic)**

**(See Appendix A: Project Source Code Figure 9)**

****

* **Purpose:** Reads data for all cities for a specific subtopic.
* **Parameters:**
  + main\_topic (str) - The main topic of interest.
  + subtopic (str) - The subtopic under the main topic.
* **Returns:** Data related to all cities for the specified main topic and subtopic.

**5. Function: get\_years()**

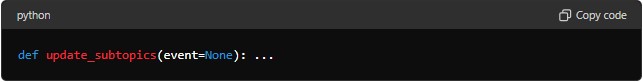
**(See Appendix A: Project Source Code Figure 10)**

****

* **Purpose:** Retrieves all unique years from the dataset.
* **Parameters:** None.
* **Returns:** List of unique years available in the dataset.

**6. Function: update\_subtopics(event=None)**

**(See Appendix A: Project Source Code Figure 11)**

****

* **Purpose:** Updates the subtopic dropdown based on the selected main topic.
* **Parameters:** event (optional) - The event that triggers the update.
* **Returns:** None.

**7. Function: update\_info\_text(event=None)**

**(See Appendix A: Project Source Code Figure 12)**

****

* **Purpose:** Updates the information text displayed based on user interaction.
* **Parameters:** event (optional) - The event that triggers the update.
* **Returns:** None.

**8. Function: on\_city\_hover(event)**

**(See Appendix A: Project Source Code Figure 13)**

****

* **Purpose:** Handles the city button hover event to provide interactive feedback.
* **Parameters:** event - The hover event.
* **Returns:** None.

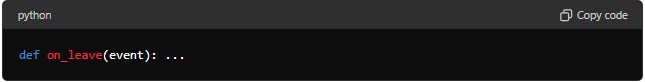
**9. Function: on\_hover(event, city)**

**(See Appendix A: Project Source Code Figure 14)**

****

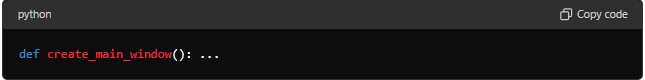
* **Purpose:** Handles the mouse hover event over a city button to highlight or provide additional information.
* **Parameters:**
  + event - The hover event.
  + city (str) - The city being hovered over.
* **Returns:** None.

**10. Function: on\_leave(event)(See Appendix A: Project Source Code Figure 15)**

****

* **Purpose:** Handles the mouse leave event from a city button to remove any highlights or additional information.
* **Parameters:** event - The leave event.
* **Returns:** None.

**11. Function: create\_main\_window()(See Appendix A: Project Source Code Figure 16 - 23)**

****

**Description:** Creates the main window and UI components of the application, including dropdowns, buttons, labels, and frames for displaying data and graphs.

**Details:**

* Sets up the main window with a full-screen attribute.
* Creates control, content, display, and image frames for organizing UI components.
* Configures dropdowns for selecting cities, main topics, subtopics, and years.
* Displays data and graphs based on user selections.
* Loads an image of Rostock and places markers for different city points.
* Sets up event bindings for hover and click events.

**Usage:** This function is called when the script is executed as the main program.

**Main Execution**

****

**Description:** Checks if the script is run directly (not imported as a module) and creates the main window.

**dataload.py:**

**1. Reading the Excel File (See Appendix A: Project Source Code Figure 24)**

**file\_path:** Specifies the path to the Excel file.

**wb:** Loads the workbook from the specified file path.

**sheet:** Sets the active sheet in the workbook for data extraction.

**2. Defining Column Ranges (See Appendix A: Project Source Code Figure 25)**

**column\_ranges:** Adictionary that maps main topics to their corresponding column ranges in the Excel sheet. Each topic is associated with a range of columns.

**3. Defining map Points (See Appendix A: Project Source Code Figure 26)**

**map\_points:** A list of tuples representing geographic coordinates (x, y) for different districts in the city.

**4. City Points Dictionary (See Appendix A: Project Source Code Figure 27)**

**city\_points:** A dictionary mapping city codes to their corresponding city names.

**5. Reading Cities from Excel (See Appendix A: Project Source Code Figure 28)**

**read\_cities\_from\_excel():** A function that reads city codes and names from the Excel sheet starting from the second row.

* **cities:** A dictionary to store city codes and names.
* **sorted\_cities:** A dictionary of cities sorted by their names.

**6. Storing City Names Globally (See Appendix A: Project Source Code Figure 29)**

* **city\_names:** A global variable to store the sorted dictionary of city names.

**Conclusion:**

The Data Viewer for Structure and Characteristics of Urban Areas in Rostock City supports evidence-based decision-making and strategic planning for urban development. By harnessing demographic insights, the tool empowers stakeholders to allocate resources effectively, address community needs, and foster inclusive growth across Rostock's urban landscape.

**References:**

*GitHub: Let’s build from here*. (n.d.). GitHub. https://github.com/

*3.12.2 Documentation*. (n.d.). https://docs.python.org/

*Stack Overflow - Where Developers Learn, Share, & Build Careers*. (n.d.). Stack Overflow. https://stackoverflow.com/

*PyPI · The Python Package Index*. (n.d.). PyPI. https://pypi.org/

*data.europa.eu*. (n.d.). https://data.europa.eu/data/datasets/821ea501-d5a9-46b8-aa21-e01f4f9dbd22?locale=en

*data.europa.eu*. (n.d.-b). https://data.europa.eu/data/datasets/b879e3cf-7671-4989-aa3a-5c8cbc7442bd?locale=en

*data.europa.eu*. (n.d.-c). https://data.europa.eu/data/datasets/e0f296a7-adbe-4a0f-a0f6-6a464b3e6dc7?locale=en

**https://data.europa.eu/data/datasets/38cc66ec-be3c-45e3-bef3-c781ed9c1e2d?locale=en**

*tkinter — Python interface to Tcl/Tk*. (n.d.). Python Documentation. https://docs.python.org/3/library/tkinter.html#threading-model

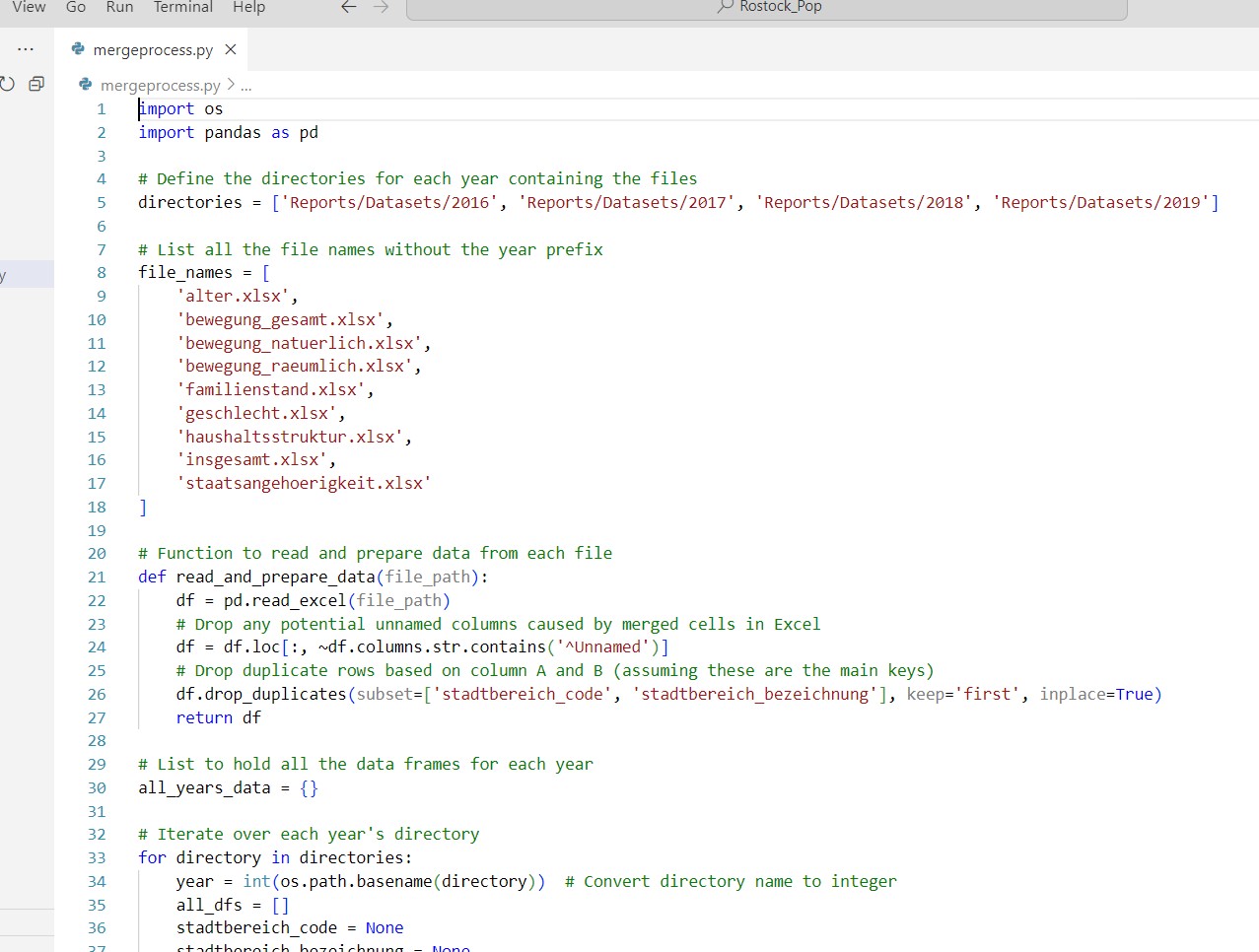
*Pillow*. (n.d.). Pillow (PIL Fork). https://pillow.readthedocs.io/en/stable/

*Plot types — Matplotlib 3.9.0 documentation*. (n.d.). https://matplotlib.org/stable/plot\_types/index.html

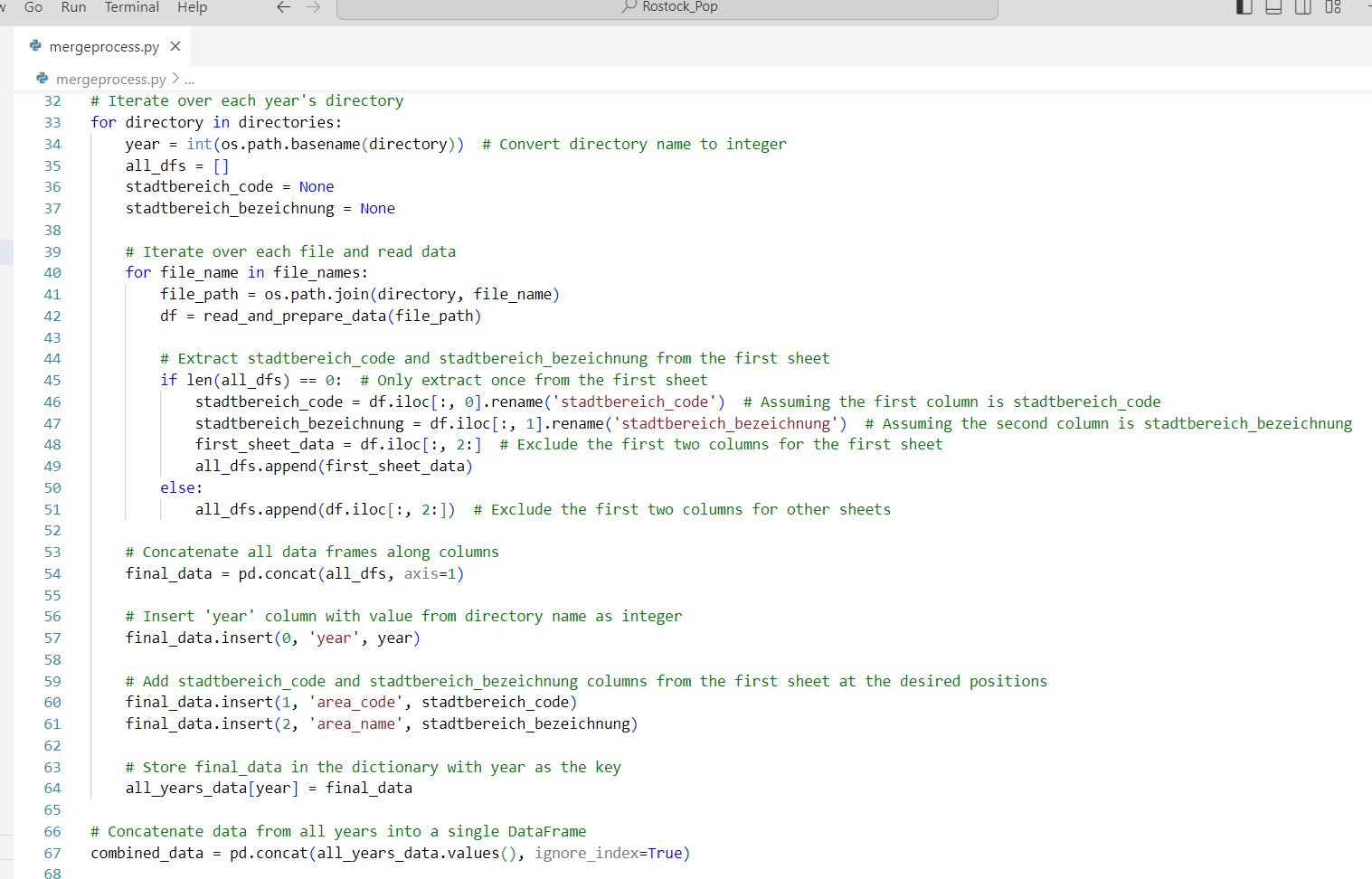
*NumPy Documentation*. (n.d.). https://numpy.org/doc/

**Appendix A: Project Source Code**

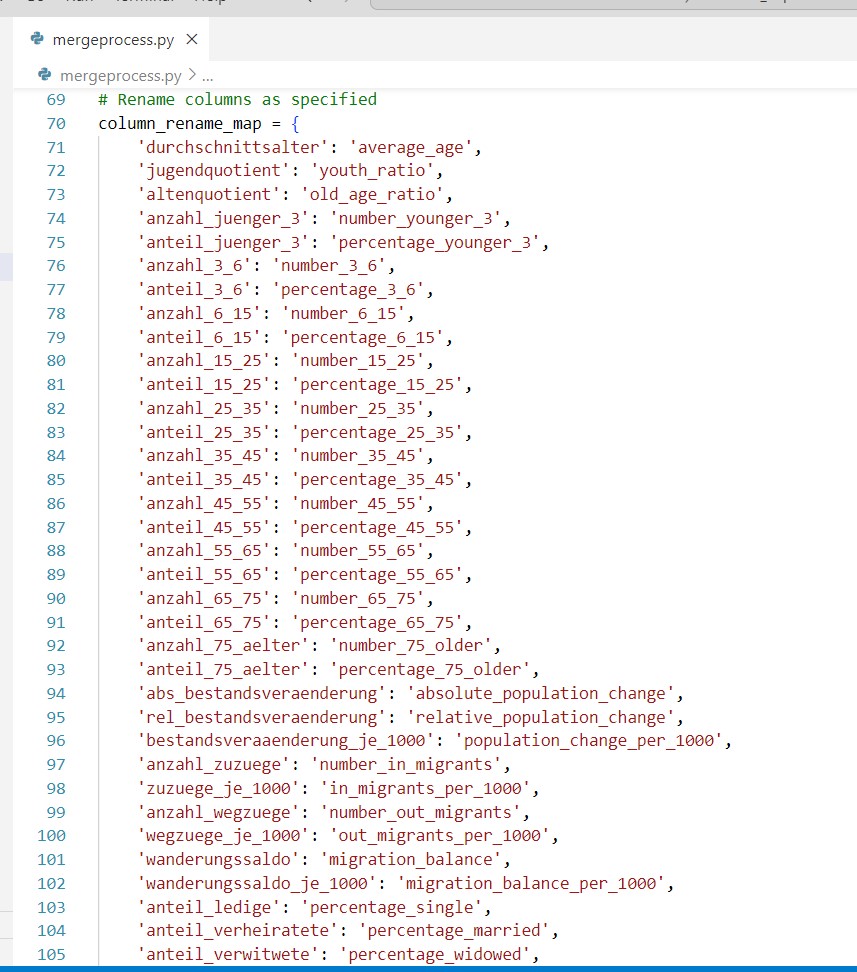
**Figure 1**

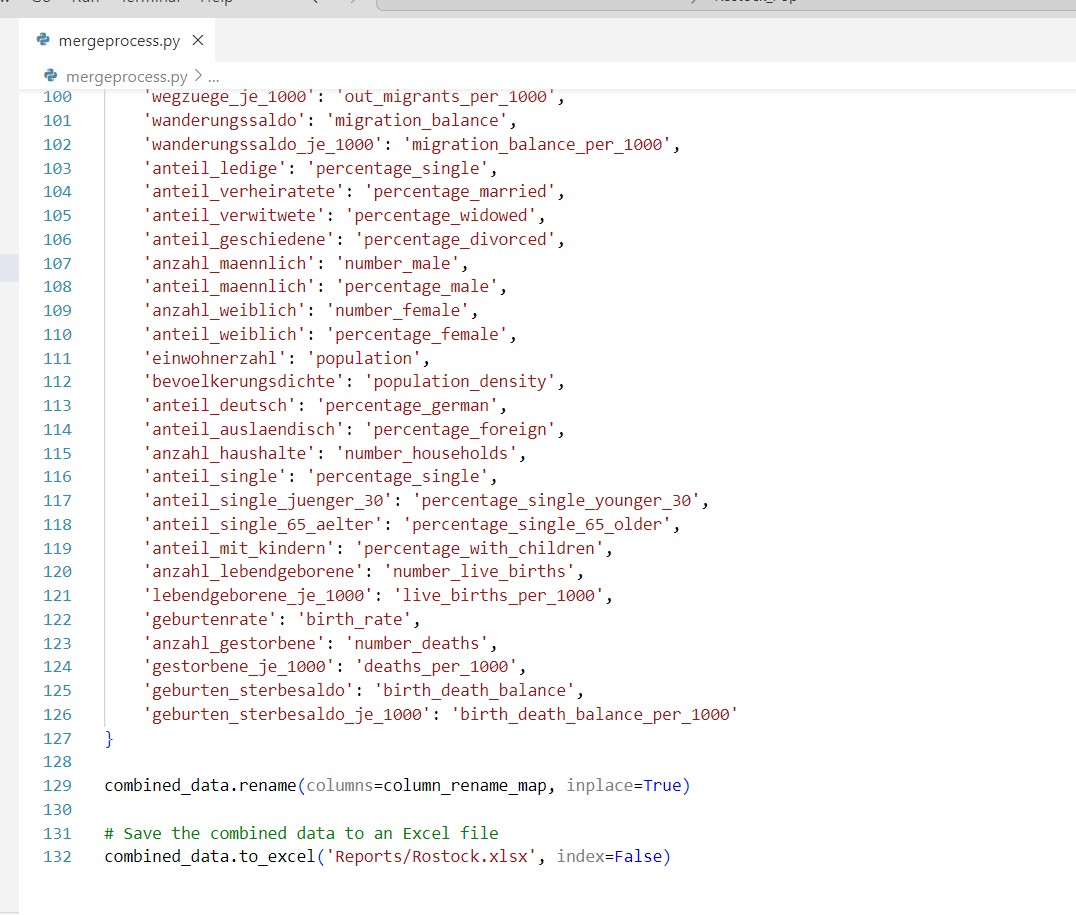
****

**Figure 2**

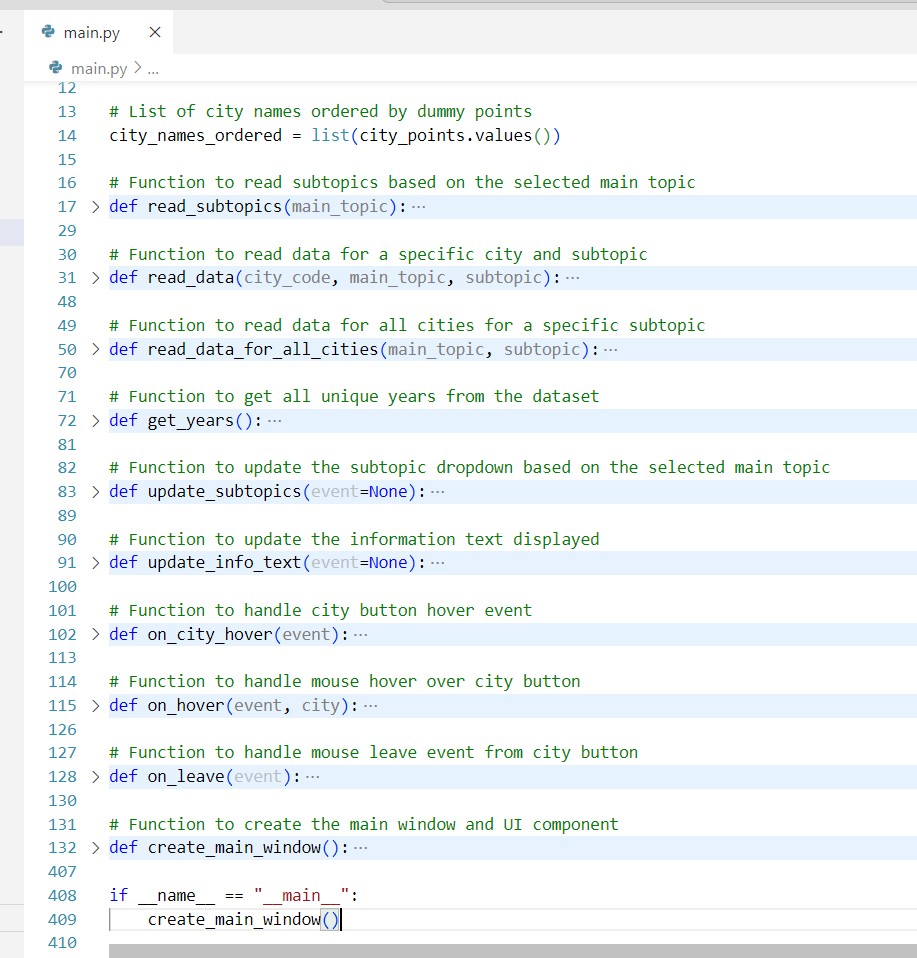
****

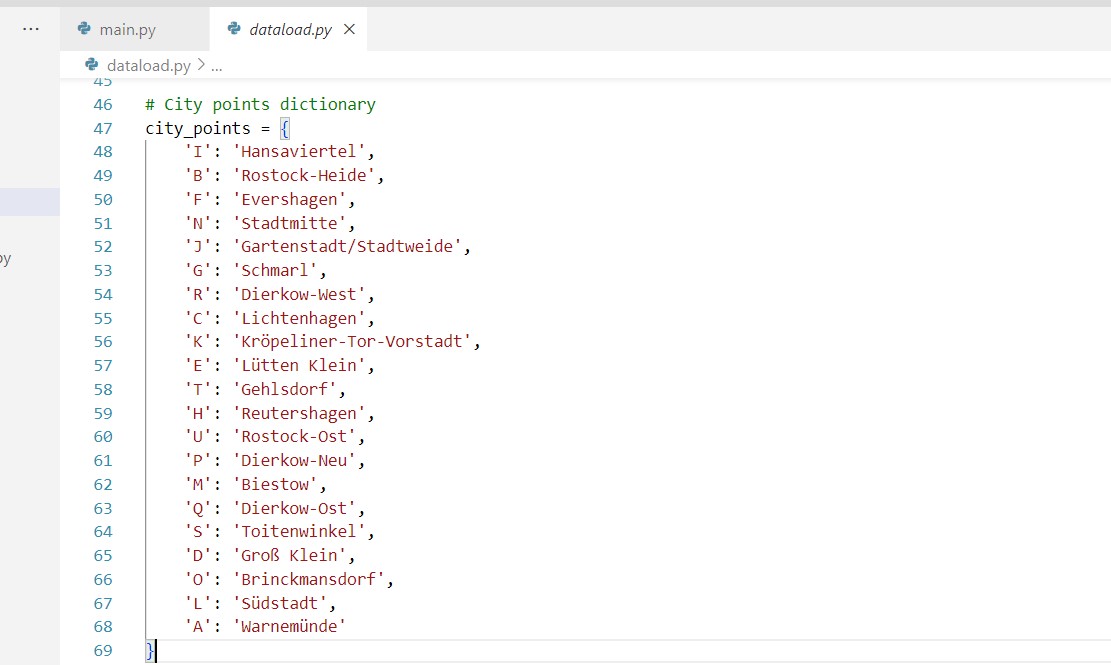
**Figure 3**

****

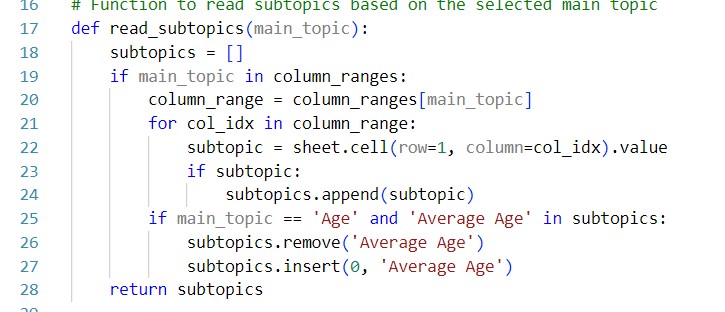
**Figure 4**

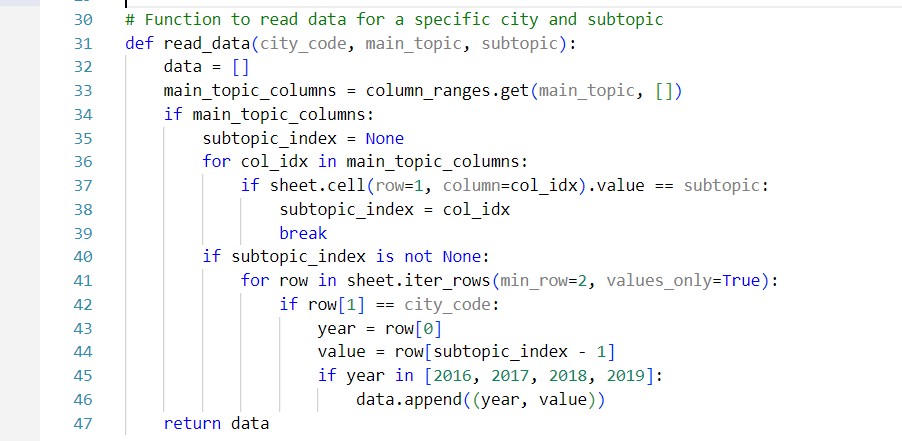
**Figure 5**

****

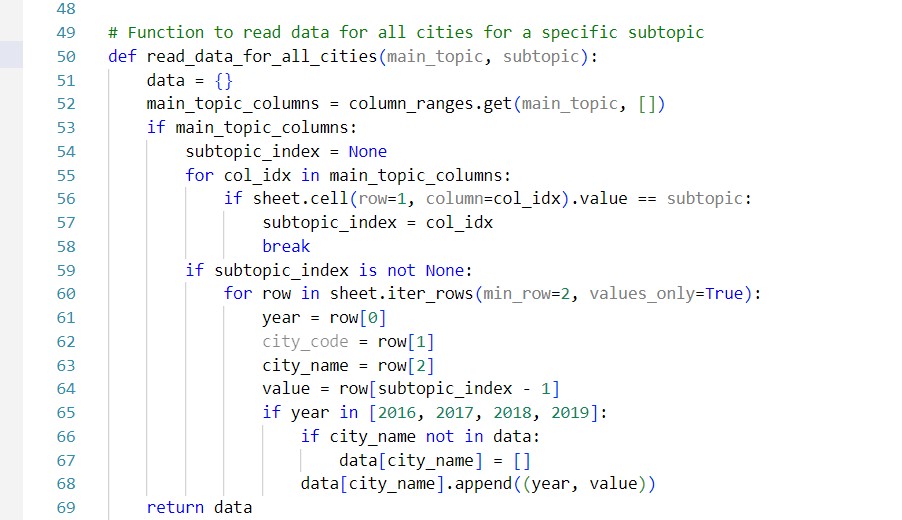
**Figure 6**

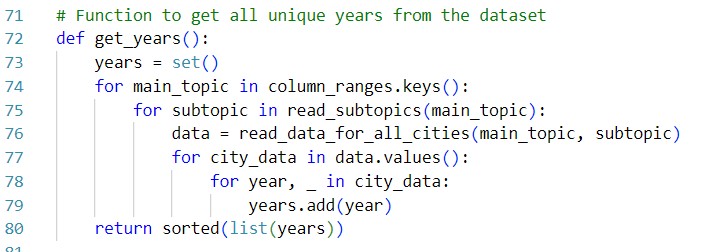
**Figure 7**

****

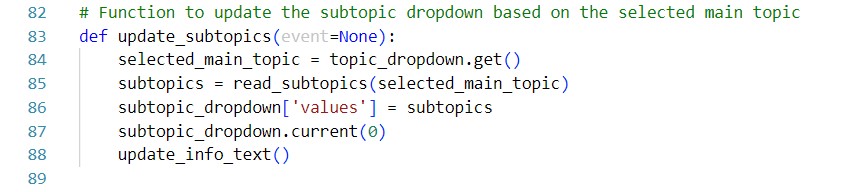
**Figure 8**

**Figure 9**

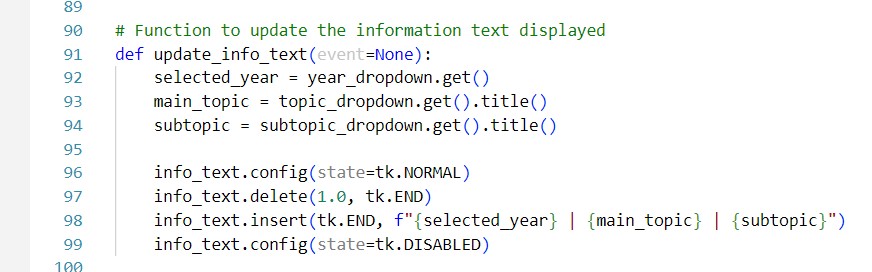
****

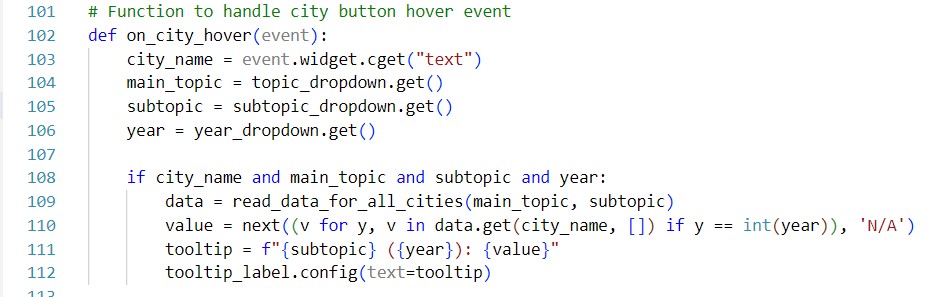
**Figure 10**

**Figure 11**

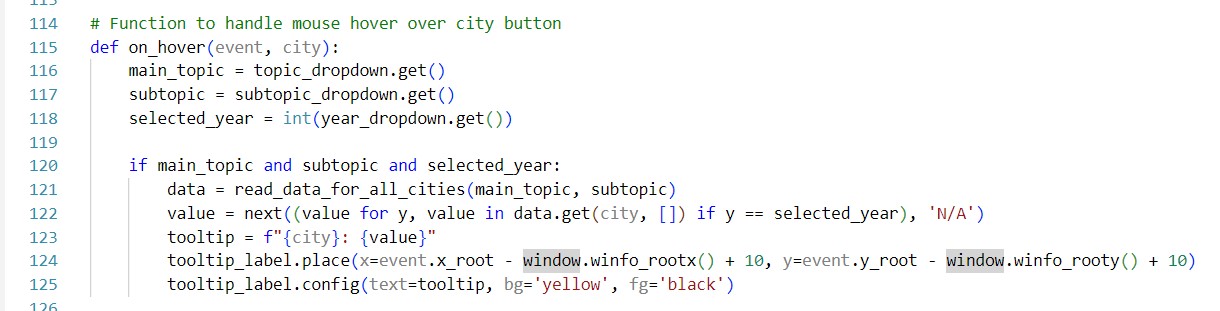
****

**Figure 12**

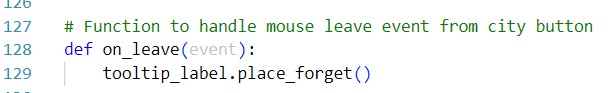
****

**Figure 13**

**Figure 14**

****

**Figure 15**

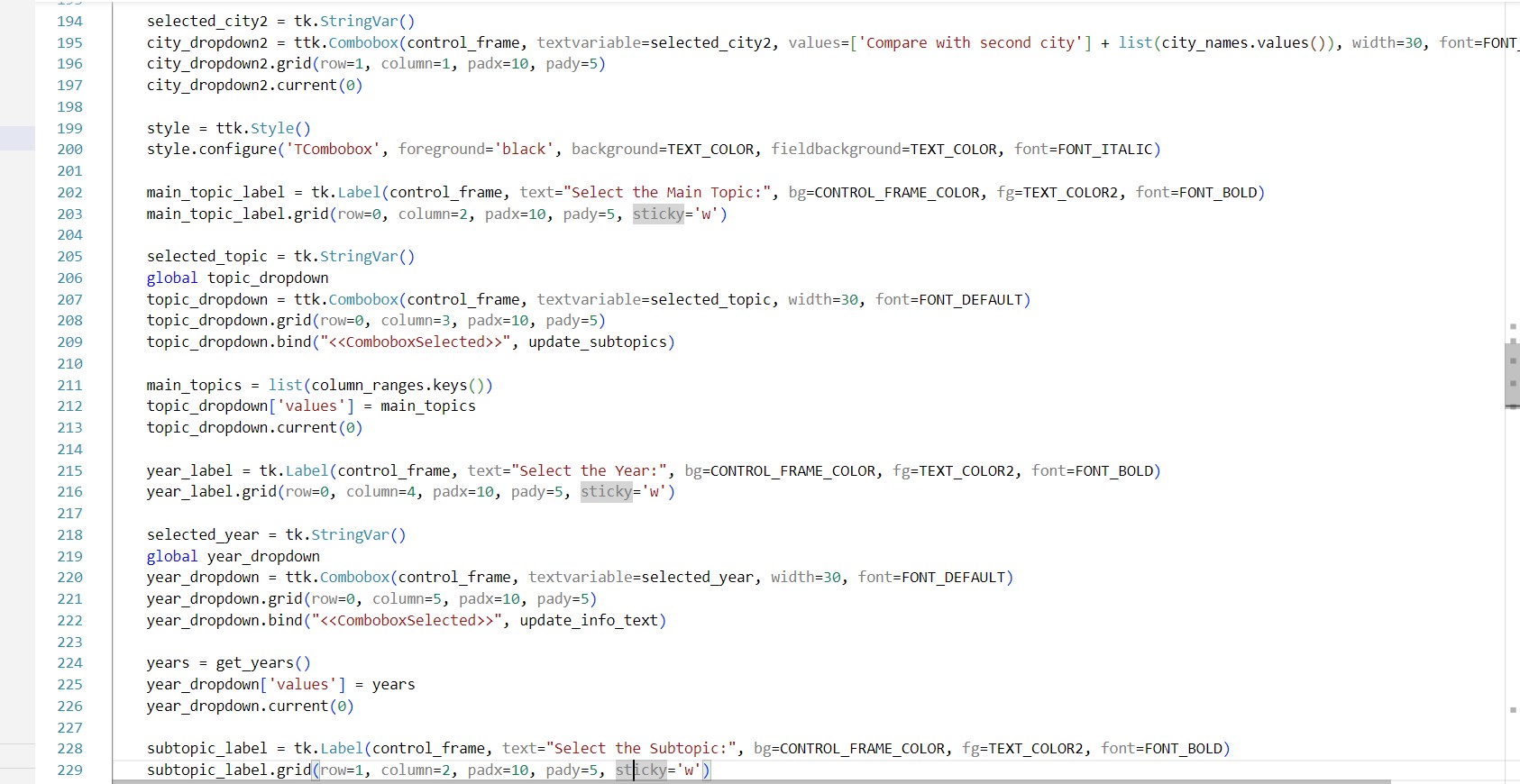
****

**Figure 16**

**Figure 17**

****

**Figure 18**

****

**Figure 19**

****

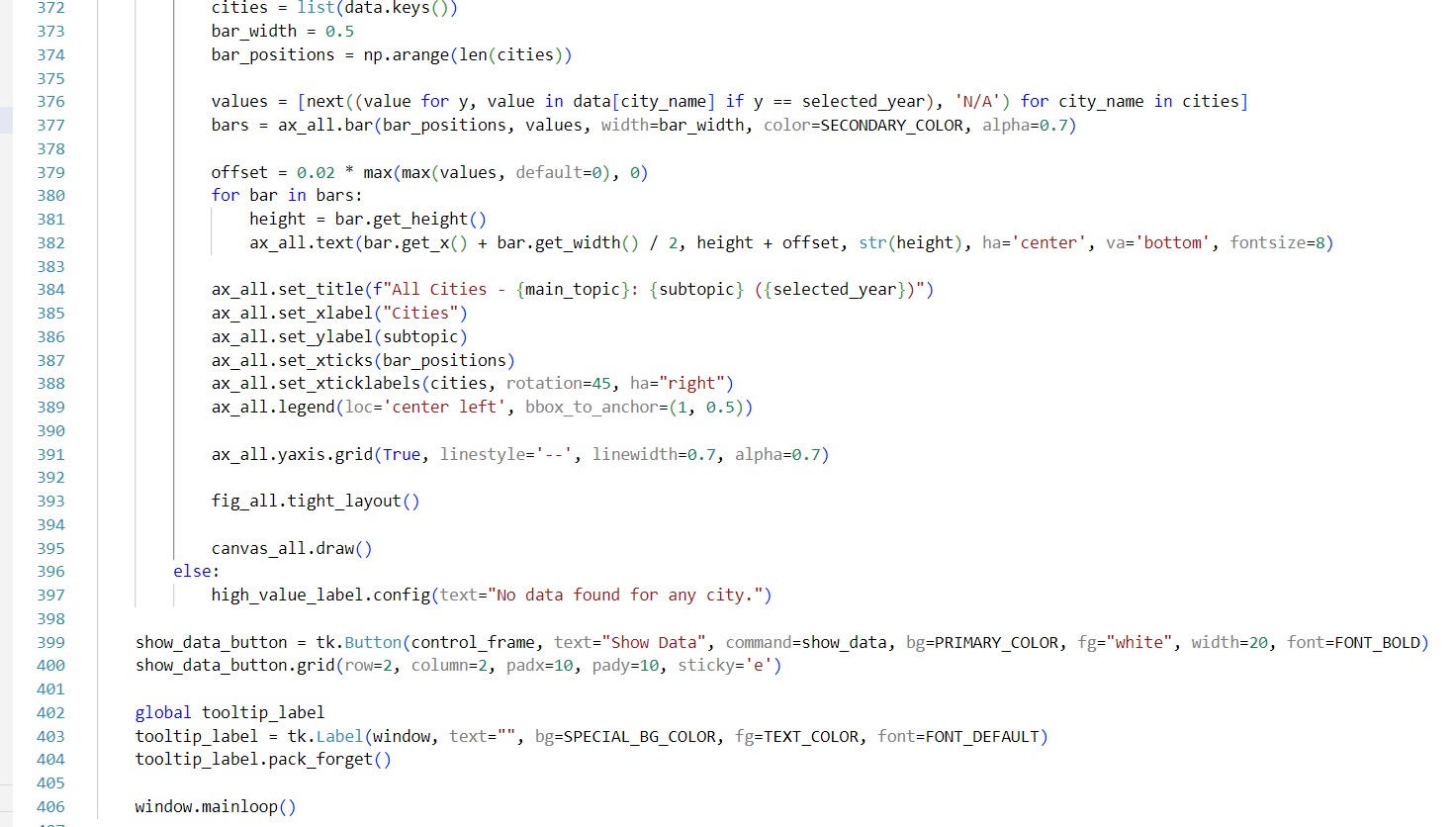
**Figure 20**

**Figure 21**

****

**Figure 22**

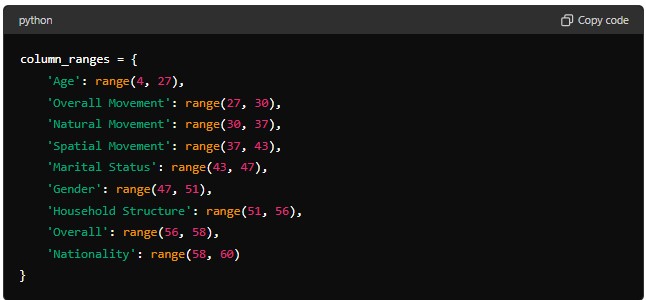
**Figure 23**

****

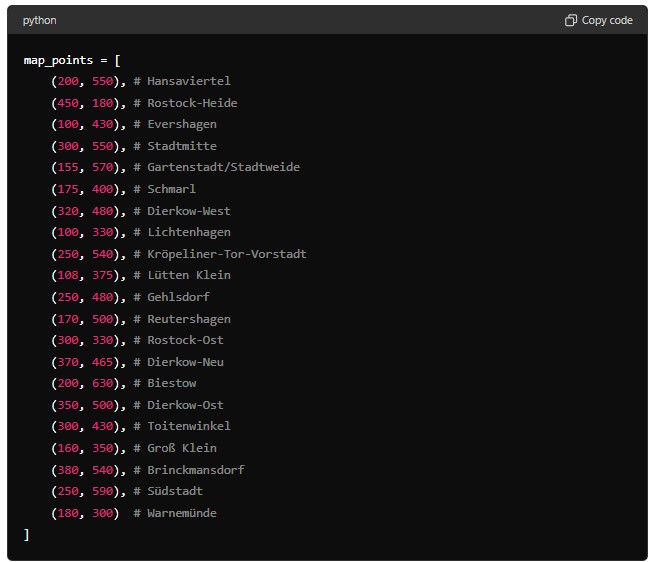
**Figure 24**

****

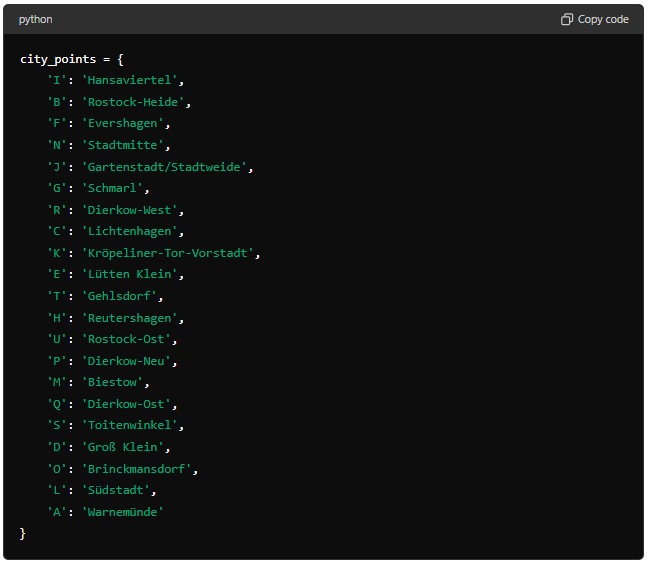
**Figure 25**

****

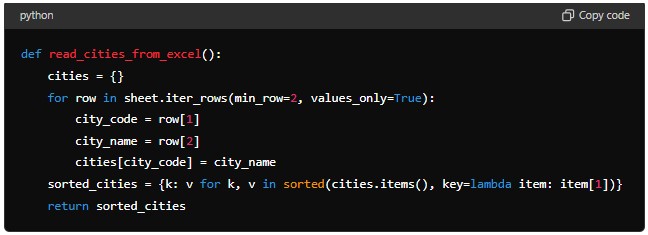
**Figure 26**

****

**Figure 27**

****

**Figure 28**

****

**Figure 29**

****