Functionalities: -

1 – Scane file system

2 - analyzes file size

3 – visualize the distribution of file sizes in different directories

Features: -

1 – Recursive file system scanning

2 – File size analysis

3 – insights of file size distribution

4 – Data representation

5 – Visualization

6 – Handle various file types

7 – Eror handling

8 – User-friendly interface

In and Out requirements:

IN

1 – accepted file system (User-provided)

2 – default path is home directory

3 – Windows path format

4 – check for a valid path (provides a feed-back in case of no valid or non-existing path)

Out

1 – Display our analysis in a visualization

2 – manage large out puts (Pagination – summary statistics – user-defined depth – filtering and sorting – Memory efficiency – streaming out-put – progress indicator)

3 – User-friendly messages

4 – Eror-handling out-put

Project Tasks

1. **Understand the Project Requirements:**

Read and understand the project requirements thoroughly. Identify the key functionalities needed for the File System Analyzer.

1. **Setup Your Project:**

Create a new Python project or directory for your File System Analyzer. Set up the necessary project structure.

1. **Explore the os Module:**

Familiarize yourself with the os module in Python. This module will be essential for interacting with the file system.

1. **Create a Function for Listing Files and Directories:**

Write a function that uses the os module to list all files and directories in a given path. This function can be a starting point for your recursive exploration.

1. **Implement Recursive File System Scan:**

Create a recursive function that navigates through directories, calling itself for each subdirectory. This is where you'll specifically implement recursion.

1. **Collect File Size Information:**

Modify your recursive function to collect information about each file's size. You might want to store this information in a data structure like a dictionary or a list.

1. **Data Analysis:**

Write a function to analyze the collected data. For example, calculate the total size of files in each directory, find the average file size, identify the largest and smallest files, etc. This is where the data analysis part comes into play.

1. **Visualization (Optional):**

If you want to enhance your project, consider visualizing the data. You can use libraries like matplotlib to create plots representing the distribution of file sizes in different directories.

1. **Refine and Optimize:**

Review your code, make improvements, and optimize where possible. This could involve improving the efficiency of your recursive function or enhancing the data analysis process.

1. **Test Your File System Analyzer:**

Test your tool on different file system structures to ensure it handles various scenarios correctly.

1. **Documentation:**

Write clear and concise documentation for your project. Explain how to use your File System Analyzer, the expected input/output, and any other relevant information.

1. **Reflect on the Experience:**

Reflect on what you've learned from the project. Consider any challenges you faced and how you overcame them. This reflection can deepen your understanding of recursion and data analysis.

Tasks 4, 5, and 6 primarily touch on recursion. Task 7 specifically focuses on data analysis, and task 8 (optional) involves visualization, which is also a form of data representation and analysis. The other tasks involve a mix of both recursion and data analysis. Good luck, and enjoy the project!