1. **Approach:** Based on the requirement of fetching exchange rate from an External API. Below is the approach which we I can think of to achieve the solution via python code.

* **Layered Architecture**: Developing the python code in such a way that it allows a layered architecture where each function represents a distinct layer of functionality. This separation of concerns enables better organization, easier testing, and maintenance.
* **Client-Server Interaction Approach**: The python code interacts with an external API (Exchange Rates API) to fetch exchange rate data. It follows a client-server communication model where the Python client requests data from the API server over HTTP.

1. **Coding Best Standards:** Below are the coding best standards incorporated in

the python program.

* **Modular Design:** The python code is structured into functions, each responsible for a specific task such as fetching exchange rates, finding best/worst rates, and calculating average rate. This modular approach enhances code readability, maintainability, and reusability.
* **Error Handling:** Comprehensive error handling is implemented using try-except blocks to handle potential exceptions that may occur during API requests, data processing, and calculations. This ensures the robustness and reliability of the code.
* **Parameterization:** The code utilizes parameters (base\_currency, target\_currency, days) to make it configurable and reusable for fetching exchange rates for different currency pairs and time periods.
* **Code Readability:** The code maintains readability by using descriptive function and variable names and providing inline comments where necessary. This improves code comprehension and maintainability. It also helps someone who has not written the code to anlayse the code.

1. **Performance**

* The python code demonstrates efficiency by minimizing unnecessary processing, avoiding redundant computations, and utilizing built-in Python libraries (requests, datetime) for common tasks. This ensures optimal performance and resource utilization.

1. **Scalability**

* The python framework build is relatively simple, the architectural approach allows for scalability and extensibility. For example, additional functionality, such as caching retrieved data, implementing user authentication, or integrating with other APIs, could be added without significantly modifying the existing structure.

In summary, the architectural approach for the provided code follows a client-server model with a layered structure, emphasizing modularity, separation of concerns, and scalability. This approach enables the code to effectively retrieve, process, and present exchange rate data while maintaining flexibility for future enhancements and modifications.