**TESTING SCENARIO’S**

1. **Valid Data Retrieval**:

**Objective:** Verify that the code successfully retrieves exchange rate data from the Exchange Rates API.

**Steps:**

1. Execute the code with valid parameters (base currency, target currency, number of days).
2. Check whether the code successfully retrieves exchange rate data from the API.
3. Validate the correctness of the retrieved data.

**Expected Outcome:** The code successfully fetches exchange rate data without any errors and returns the expected exchange rates.

**2. Invalid API Key Handling:**

**Objective:** Test how the code handles errors when an invalid API key is provided.

**Steps:**

1. Modify the code to use an invalid API key or simulate an invalid API key scenario.
2. Execute the code with the modified parameters.
3. Check whether the code correctly handles the invalid API key error and provides appropriate feedback.

**Expected Outcome:** The code should detect the invalid API key and handle the error gracefully, providing a clear error message to the user.

**3. Network Error Handling:**

**Objective:** Test how the code handles network errors (e.g., connection timeout, server unreachable).

**Steps:**

1. Simulate a network error scenario, such as disconnecting from the internet or blocking outgoing requests.
2. Execute the code with the simulated network error.
3. Check whether the code handles the network error gracefully and provides informative feedback.

**Expected Outcome:** The code should detect the network error and handle it without crashing, providing a meaningful error message to the user.

**4. Invalid Input Parameters:**

**Objective:** Test how the code handles errors when invalid input parameters are provided (e.g., incorrect currency codes, negative number of days).

**Steps:**

1. Modify the input parameters to include invalid values, such as non-existent currency codes or negative numbers of days.
2. Execute the code with the modified parameters.
3. Check whether the code properly validates the input parameters and handles the errors.

**Expected Outcome:** The code should validate the input parameters and reject invalid values, providing appropriate error messages to the user.

**5. Empty Response Handling:**

**Objective:** Test how the code handles scenarios where the API returns an empty response.

**Steps:**

1. Modify the code to simulate an empty response scenario from the API.
2. Execute the code with the modified parameters.
3. Check whether the code detects the empty response and handles it appropriately.

**Expected Outcome:** The code should detect the empty response from the API and provide a clear message indicating the absence of exchange rate data.

**6. Data Processing:**

**Objective:** Verify that the code correctly preprocesses the retrieved data and extracts relevant information.

**Steps:**

1. Execute the code with valid parameters to retrieve exchange rate data.
2. Check whether the code preprocesses the data correctly, extracting the relevant exchange rates for the specified currencies.
3. Validate the correctness of the preprocessing step.

**Expected Outcome:** The code should preprocess the data accurately, extracting the exchange rates for the specified currencies without errors.

**7. Calculation Accuracy:**

**Objective:** Test whether the code accurately calculates the best and worst exchange rates and computes the average exchange rate.

Steps:

1. Execute the code with valid parameters to retrieve exchange rate data.
2. Check whether the code correctly calculates the best and worst exchange rates and computes the average exchange rate.
3. Validate the correctness of the calculated values.

**Expected Outcome:** The code should accurately calculate the best and worst exchange rates and compute the average exchange rate based on the retrieved data.

These testing scenarios cover various aspects of the code functionality, including data retrieval, error handling, data processing, and calculation accuracy, ensuring the robustness and reliability of the code.