**Currency Code Table Generator**

**Developer Documentation**

# Overview

This module fetches global country and currency information from the public REST Countries API and builds a Currency Code Table. It retrieves country names, ISO country codes, currency names, ISO currency codes, regions, and subregions, with an option to exclude specific regions entirely (e.g., 'antarctic'). The resulting dataset is returned as a pandas DataFrame for further integration or storage.

# Table of Contents

* Overview
* Table of Contents
* Dependencies
* Setup and Configuration
* Data Flow
* Module Components
  1. 4.1 Currency Code Table Generation Function
  2. 4.2 Main Usage Pattern
* Key Design Decisions
* Data Structure
* Error Handling
* Testing Considerations
* Future Enhancements
* Developer Checklist

# Dependencies

The module relies on the following external libraries:  
• pandas  
• requests  
• Python standard library: set, dict  
  
Ensure these dependencies are installed:  
pip install pandas

pip install requests

# Setup and Configuration

No additional runtime configuration is required. An optional parameter exclude\_regions can be provided as a set of lowercase region names to skip (e.g., {'antarctic'}).

# 3. Data Flow

1) Accept optional exclude\_regions set; default to {'antarctic'} if none provided.

2) Call the REST Countries API endpoint with required fields (name, currencies, cca2, region, subregion).

3) Parse JSON response into a structured Python list.

4) Skip countries belonging to excluded regions.

5) For each country:

- Extract country name and 2-letter code.

- Iterate over all available currencies.

- Extract currency name and ISO code.

- Collect region and subregion details.

6) Convert list of records to a pandas DataFrame.

7) Return the DataFrame to the caller.

## 4.1 Currency Code Table Generation Function

**Function:** get\_country\_currency\_df(exclude\_regions=None)  
  
**Purpose:**  
Fetches country and currency details from the REST Countries API, filters out excluded regions, and returns a DataFrame with structured currency reference data.  
  
**Process:**  
• Define default excluded regions if none are provided.  
• Request country/currency metadata from the REST Countries API.  
• Raise exceptions on failed requests.  
• Parse JSON results.  
• Exclude matching regions based on lowercase region name.  
• Extract and validate required fields for each country/currency combination.  
• Store records in a list and return as a pandas DataFrame.  
  
**Data Sources:**  
• REST Countries API — https://restcountries.com/v3.1/all (fields: name, currencies, cca2, region, subregion)  
  
**Output Structure:**  
• One row per country–currency combination containing country name, code, currency name, code, region, subregion.

## 4.2 Main Usage Pattern

There is no dedicated main() function.  
  
The module is typically imported, and get\_country\_currency\_df() is called directly.  
Example:  
from currencycodetable import get\_country\_currency\_df  
df = get\_country\_currency\_df(exclude\_regions={'antarctic'})  
print(df.head())

# 5. Key Design Decisions

• Use of a public API for up-to-date country/currency metadata.

• Default exclusion of the 'antarctic' region to avoid irrelevant data.

• Flexible filtering via exclude\_regions parameter.

• Minimal dependencies for portability.

• Return results as a pandas DataFrame for easy downstream use.

# 6. Data Structure

• country\_name (str) — Common country name.

• country\_code (str) — ISO alpha-2 country code.

• currency\_name (str) — Full currency name.

• currency\_code (str) — ISO 4217 currency code.

• region (str) — Geopolitical region.

• subregion (str) — Geopolitical subregion.

# 7. Error Handling

• Network or API errors raise exceptions via requests.raise\_for\_status().

• Missing or incomplete fields are skipped — only valid records are appended.

• Region filtering is case-insensitive (converted to lowercase).

# 8. Testing Considerations

**Unit Testing**

✓ Verify DataFrame schema matches specification.

✓ Confirm region exclusion works as intended.

✓ Ensure no records contain missing required fields.

✓ Check DataFrame is non-empty for valid API responses.

**Integration Testing**

✓ Validate correct API endpoint usage and response parsing.

✓ Mock API responses to test edge cases.

✓ Confirm output correctness for custom exclude\_regions inputs.

# 9. Future Enhancements

• Add caching to reduce API calls.

• Support offline mode with a saved dataset.

• Allow filtering by specific regions or subregions.

• Include currency symbols if available in API.

• Add CLI interface for quick CSV export.

# 10. Developer Checklist

✓ All dependencies installed.

✓ API endpoint reachable.

✓ Function call returns DataFrame with correct columns.

✓ Region filtering tested.

✓ Code reviewed for clarity and maintainability.