Backend Code Documentation

Source code documentation of NBP Crasher application focusing on backend functions, implementation and usages

Table of contents

Project Information	3
Backend Purpose	
Used Technologies	
Code documentation	
API	4
Data analysis	4
json_to_data_frame	
calculate_statistical_measures	
count_session	
calculate_distribution	
create_dynamic_ranges	
calculate_statistics	

Project Information

Project Name: NBP Crasher

• Project Manager: Przemysław Kowalski

• **Report Date:** 07.01.2025

• **Project Duration:** 08.12.2024 – 31.01.2025

Backend Purpose

The goal of the backend for the NBP Crasher project is to provide a functional and complete API for obtaining statistical measures based on currency exchange rates retrieved from an external API (NBP API).

Used Technologies

This part of system was developed using easily available technologies.

The language used is <u>Python</u> – version 3.12

API was developed with framework <u>FastAPI</u> in version 0.115.6

Data analysis was perfored with help of following external packages:

- <u>numpy</u> version 2.2.1
- pandas version 2.2.3

Web server hosting our API implementation is <u>uvicorn</u> – version 0.34.0

For final deployment we used contenerization using **Docker**

Code documentation

API

API documentation is put in other file in this file directory under name "NBP Crasher.pdf"

Data analysis

```
json_to_data_frame
```

Converts a JSON object into a pandas DataFrame.

Parameters:

json_content (dict): Input JSON data

Returns:

pd.DataFrame: DataFrame containing the JSON data

calculate statistical measures

Calculates basic statistical measures for numerical data.

Parameters:

data (set): Input numerical values set

Returns: Dictionary containing:

'mode': Dictionary of mode values and frequencies

`standard_deviation`: Standard deviation (4 decimal places)

`variation_coefficient`: Variation coefficient as percentage

(4 decimal places)

'median': Median value (4 decimal places)

Raises:

ValueError: If data is None

count_session

Counts increasing, decreasing and unchanged sessions.

Parameters:

data (pd.Series): Input numerical values series

Returns: Dictionary containing:

`increasing_sessions`: Count of value increases `decreasing_sessions`: Count of value decreases

`no_change_sessions`: Count of unchanged values

Raises:

ValueError: If data is None

calculate_distribution

Calculates distribution of absolute currency rate changes.

Parameters:

```
currency_rate (pd.Series): Currency rate values
```

Returns: List of dictionaries containing:

`rangeBegin`: Range interval start `rangeEnd`: Range interval end

`value`: Count in range

Raises:

ValueError: If currency_rate is None

create_dynamic_ranges

Creates dynamic range boundaries and labels.

Parameters:

data (pd.Series): Input data

n_ranges (int, optional): Number of ranges (default: 14)

Returns: Tuple containing:

boundaries: Range boundary values list

labels: Formatted range labels list

Raises:

ValueError: If data is None

calculate_statistics

Calculates comprehensive statistics for one or two currencies.

Parameters:

first_currency (dict): First currency data

second_currency (dict, optional): Second currency data

Returns: Dictionary containing:

`statistics`: Statistical measures

`sessions`: Session counts

`changes_distribution`: Rate changes distribution

Raises:

ValueError: If first_currency is None KeyError: If data structure invalid