imfdatapy

Irina Klein, Sou-Cheng Choi

Jan 23, 2024

CONTENTS

CHAPTER

ONE

IMFDATAPY

A package for data discovery and extraction from the International Monetary Fund (IMF)! This repository contains Python source code and Jupyter notebooks with examples on how to extract data from the IMF.

1.1 Installation

```
$ pip install imfdatapy
```

1.2 Usage

`IMFDataPy` can be used to search through and extract data as follows. The examples below show how to search through the IFS (International Financial Statistics) and BOP (Balance of Payments) using `serach_terms` and download all the data with matching economic indicator names.

```
from imfdatapy.imf import *
ifs = IFS(search_terms=["gross domestic product, real"], countries=["US"], period='Q',
start_date="2000", end_date="2022")
df = ifs.download_data()

bop = BOP(search_terms=["current account, total, credit"], countries=["US"], period='Q',
start_date="2000", end_date="2022")
df = bop.download_data()
```

1.3 Contributing

Interested in contributing? Check out the contributing guidelines. Please note that this project is released with a *Code of Conduct*. By contributing to this project, you agree to abide by its terms.

1.4 License

`imfdatapy` was created by Sou-Cheng T. Choi and Irina Klein, Illinois Institute of Technology. It is licensed under the terms of the Apache License, v2.0.

With regard to the terms for using IMF data, please refer to IMF's Copyright and Usage and pay special attention to the section *SPECIAL TERMS AND CONDITIONS PERTAINING TO THE USE OF DATA*.

1.5 Credits

`imfdatapy` was created with `cookiecutter` and the `py-pkgs-cookiecutter` template.

CHAPTER

TWO

CHANGELOG

2.1 v0.1.5 (26/11/2022)

- Resolve dependecies issue and mkdir before pandas.to_csv.

2.2 v0.1.1 (25/11/2022)

- Change source code to OO version for search and data extraction.

2.3 v0.1.0 (24/11/2022)

- First release of `IMFDataPy`! Reserve name on PyPi.

CHAPTER

THREE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.1 Types of Contributions

3.1.1 Report Bugs

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

3.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

3.1.4 Write Documentation

You can never have enough documentation! Please feel free to contribute to any part of the documentation, such as the official docs, docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.2 Get Started!

Ready to contribute? Here's how to set up `IMFDataPy` for local development.

1. Download a copy of `IMFDataPy` locally.

```
$ git clone https://github.com/Economic-and-Financial-Data-Discovery/imfdatapy.git
$        cd imfdatapy
$        git checkout develop
```

2. Install `IMFDataPy` using `conda`:

```
$ conda env create --file environment.yml
$ conda activate imfdatapy
$ jupyter nbextensions_configurator enable --user
$ python -m ipykernel install --user --name=imfdatapy
$ pip install -e .
```

3. Use `git` (or similar) to create a branch for local development and make your changes:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

4. When you're done making changes, check that your changes conform to any code formatting requirements and pass any tests.
5. Commit your changes and open a pull request.

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include additional tests if appropriate.
2. If the pull request adds functionality, the docs should be updated.
3. The pull request should work for all currently supported operating systems and versions of Python.

3.4 Code of Conduct

Please note that the `IMFDataPy` project is released with a Code of Conduct. By contributing to this project you agree to abide by its terms.

CHAPTER

FOUR

CODE OF CONDUCT

4.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

4.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

4.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

4.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

4.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

4.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant homepage, version 1.4.

CHAPTER

FIVE

DEMOS

5.1 IMFDataPy Architecture

```
from IPython.display import IFrame
```

```
IFrame("mermaid/imfdatapy_architecture_state_diagram.html", width=500, height=400)
```

```
IFrame("mermaid/imfdatapy_architecture_sequence_diagram.html", width=500, height=500)
```

```
IFrame("mermaid/imfdatapy_architecture_class_diagram.html", width=500, height=400)
```

5.2 Extraction of International Financial Statistics data from the IMF

The example below retrieves quarterly (period: Q) Seasonally Adjusted Real Gross Domestic Product (indicator: NGDP_R_SA_XDC) for the USA (country code: US), from the International Financial Statistics (IFS) series using the **IMFDataPy** package. The function call returns the observation values, and the time period for each value (in the format YYYY-MM-DD).

First, we begin with loading the **IMFDataPy** library for data extraction and from the IMF and **pandas** for data manipulation.

5.2.1 **IMFDataPy** package

Source code foris available on Github.

The package can be installed using `pip`. !pip install imfdatapy

```
from imfdatapy.imf import *
```

```
import pandas as pd # for QoQ change calculation
```

To download the data from the International Financial Statistics, we use the `IFS` class, provding the search terms for the index we are looking for, the country code, the period frequency (Q) and the period. Use the `download_data` method

to download the data and the metadata to '../out' folder and create a pandas dataframe. The log messages specify which files are created in the '../out' directory.

```
pd.options.display.max_colwidth = 90
```

```
ifs = IFS(search_terms=["gross domestic product"], countries=["US"], period='Q', start_
↪date="2010",
                  end_date="2023")
df = ifs.download_data()
df
```

```
2022-12-01 00:10:26,551 imf.py:117 - INFO - Output all IMF series table to ../out/series_
↪imf.csv
2022-12-01 00:10:26,569 imf.py:119 - INFO - Output series containing 'IFS' table to ../
↪out/series_ifs.csv
2022-12-01 00:10:39,190 imf.py:249 - WARNING - Failed to download CL_FREQ.
2022-12-01 00:10:39,770 imf.py:132 - INFO - Output dimension CL_AREA_IFS table to ../out/
↪dim_cl_area_ifs.csv
2022-12-01 00:10:39,784 imf.py:132 - INFO - Output dimension CL_INDICATOR_IFS table to ..
↪/out/dim_cl_indicator_ifs.csv
2022-12-01 00:10:40,153 imf.py:149 - INFO - Output meta data of IFS table to ../out/meta_
↪ifs.csv
2022-12-01 00:10:40,172 imf.py:151 - INFO - Output meta data of IFS containing 'gross␣
↪domestic product' table to ../out/meta_gross domestic product_US_Q_2010_2023.csv
2022-12-01 00:10:40,482 imf.py:171 - INFO - Output data of IFS containing 'gross␣
↪domestic product' table to ../out/data_gross domestic product_US_Q_2010_2023.csv
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

<svg xmlns="http://www.w3.org/2000/svg" height="24px"viewBox="0 0 24 24" width="24px">

```
  <script>
    const buttonEl =
      document.querySelector('#df-beed8f22-19d1-48ce-94f1-44112fd49b9f button.colab-df-
↪convert');
    buttonEl.style.display =
      google.colab.kernel.accessAllowed ? 'block' : 'none';

    async function convertToInteractive(key) {
      const element = document.querySelector('#df-beed8f22-19d1-48ce-94f1-44112fd49b9f');
      const dataTable =
        await google.colab.kernel.invokeFunction('convertToInteractive',
                                                 [key], {});
      if (!dataTable) return;

      const docLinkHtml = 'Like what you see? Visit the ' +
        '<a target="_blank" href=https://colab.research.google.com/notebooks/data_table.
```

(continues on next page)

(continued from previous page)

```
↪ipynb>data table notebook</a>'
        + ' to learn more about interactive tables.';
      element.innerHTML = '';
      dataTable['output_type'] = 'display_data';
      await google.colab.output.renderOutput(dataTable, element);
      const docLink = document.createElement('div');
      docLink.innerHTML = docLinkHtml;
      element.appendChild(docLink);
    }
  </script>
</div>
```

Here, all the data that matched the search term is loaded. To view the index names, use the meta data file as shown below.

```
meta = pd.read_csv('../out/meta_gross domestic product_US_Q_2010_2023.csv')
```

```
meta
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

<svg xmlns="http://www.w3.org/2000/svg" height="24px"viewBox="0 0 24 24" width="24px">

```
  <script>
    const buttonEl =
      document.querySelector('#df-6e4d3215-86e4-42cf-ae20-dc99625e5931 button.colab-df-
↪convert');
    buttonEl.style.display =
      google.colab.kernel.accessAllowed ? 'block' : 'none';

    async function convertToInteractive(key) {
      const element = document.querySelector('#df-6e4d3215-86e4-42cf-ae20-dc99625e5931');
      const dataTable =
        await google.colab.kernel.invokeFunction('convertToInteractive',
                                                 [key], {});
      if (!dataTable) return;

      const docLinkHtml = 'Like what you see? Visit the ' +
        '<a target="_blank" href=https://colab.research.google.com/notebooks/data_table.
↪ipynb>data table notebook</a>'
        + ' to learn more about interactive tables.';
      element.innerHTML = '';
      dataTable['output_type'] = 'display_data';
      await google.colab.output.renderOutput(dataTable, element);
      const docLink = document.createElement('div');
      docLink.innerHTML = docLinkHtml;
```

(continues on next page)

(continued from previous page)

```
      element.appendChild(docLink);
    }
  </script>
</div>
```

We are interested in Gross Domestic Product, Real, Seasonally Adjusted, Domestic Currency. We will filter the dataframe to contain only this index.

```
df = df[df['ID']=='NGDP_SA_XDC'].reset_index()
```

```
df.tail(n=5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

<svg xmlns="http://www.w3.org/2000/svg" height="24px"viewBox="0 0 24 24" width="24px">

```
  <script>
    const buttonEl =
      document.querySelector('#df-8591373e-e28e-4bab-bf8e-8ccb702676ff button.colab-df-
↪convert');
    buttonEl.style.display =
      google.colab.kernel.accessAllowed ? 'block' : 'none';

    async function convertToInteractive(key) {
      const element = document.querySelector('#df-8591373e-e28e-4bab-bf8e-8ccb702676ff');
      const dataTable =
        await google.colab.kernel.invokeFunction('convertToInteractive',
                                                 [key], {});
      if (!dataTable) return;

      const docLinkHtml = 'Like what you see? Visit the ' +
        '<a target="_blank" href=https://colab.research.google.com/notebooks/data_table.
↪ipynb>data table notebook</a>'
        + ' to learn more about interactive tables.';
      element.innerHTML = '';
      dataTable['output_type'] = 'display_data';
      await google.colab.output.renderOutput(dataTable, element);
      const docLink = document.createElement('div');
      docLink.innerHTML = docLinkHtml;
      element.appendChild(docLink);
    }
  </script>
</div>
```

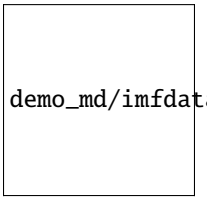
```
df['Value'] = pd.to_numeric(df['Value'])
```

```
df['QoQ'] = df['Value'].pct_change()
```

Now, we may plot the results using **matplotlib**.

```
import matplotlib.pyplot as plt
```

```
plt.rcParams.update({'font.size': 15})

t = df['Period']
data1 = df['Value'] * 10**6
data2 = df['QoQ']

labels = [f'Q{int(ts.month/3)+1}\n{ts.year}' if ts.month == 1
          else f'Q{int(ts.month/3)+1}' for ts in t]

fig, ax1 = plt.subplots()

ax1.set_xlabel('Period')
ax1.set_ylabel('Real GDP', color='blue')
ax1.set_xticks(t)
ax1.set_xticklabels(labels);
ax1.plot(t, data1, color='blue')
ax1.tick_params(axis='y', labelcolor='black')

ax2 = ax1.twinx()

ax2.set_ylabel('QoQ change', color='red')
ax2.set_xticks(t)
ax2.set_xticklabels(labels);
ax2.plot(t, data2, '--', color='red')
ax2.tick_params(axis='y', labelcolor='black')

fig.set_size_inches(25.5, 5.5)
plt.title('Real GDP by Quarter in the US')
fig.tight_layout()
plt.show()
```



demo_md/imfdatapy_IFS_GDP_example_files/imfdatapy_IFS_GDP_example_19_0.png

```
```

5.3 imfdatapy_demo

```
from imfdatapy.imf import *
```

```
ifs = IFS(search_terms=["gross domestic product, real"], countries=["US"], period='Q',␣
↪start_date="2000",
                  end_date="2022")
df = ifs.download_data()
df
```

```
2022-11-25 02:01:42,761 - root - INFO - countries_str = 'US', self.start_date = '2000',␣
↪self.end_date = '2022'
2022-11-25 02:01:43,334 - root - INFO - Output all IMF series table to ../out/series_imf.
↪csv
2022-11-25 02:01:43,365 - root - INFO - Output series containing 'IFS' table to ../out/
↪series_ifs.csv
2022-11-25 02:02:07,999 - root - WARNING - Failed to download CL_FREQ.
2022-11-25 02:02:08,736 - root - INFO - Output dimension CL_AREA_IFS table to ../out/dim_
↪cl_area_ifs.csv
2022-11-25 02:02:08,750 - root - INFO - Output dimension CL_INDICATOR_IFS table to ../
↪out/dim_cl_indicator_ifs.csv
2022-11-25 02:02:09,235 - root - INFO - Output meta data of IFS table to ../out/meta_ifs.
↪csv
2022-11-25 02:02:09,249 - root - INFO - self.meta_df.shape = (3, 3)
2022-11-25 02:02:09,250 - root - INFO - Output meta data of IFS containing 'gross␣
↪domestic product, real' table to ../out/meta_gross domestic product, real_US_Q_2000_
↪2022.csv
2022-11-25 02:03:00,718 - root - INFO - self.data_df.shape = (174, 5)
2022-11-25 02:03:00,722 - root - INFO - Output data of IFS containing 'gross domestic␣
↪product, real' table to ../out/data_gross domestic product, real_US_Q_2000_2022.csv
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

```
ifs.describe_data()
```

```
ifs = IFS(search_terms=["gross Domestic Product, Real"], countries=["CA", "RU"],
                  period='Q',  start_date="1970", end_date="2022")
df = ifs.download_data()
df
```

```
2022-11-25 02:03:15,465 - root - INFO - countries_str = 'CA, RU', self.start_date = '1970
↪', self.end_date = '2022'
2022-11-25 02:03:16,065 - root - INFO - Output all IMF series table to ../out/series_imf.
↪csv
2022-11-25 02:03:16,097 - root - INFO - Output series containing 'IFS' table to ../out/
```

(continues on next page)

(continued from previous page)

```
↪series_ifs.csv
2022-11-25 02:03:32,074 - root - WARNING - Failed to download CL_FREQ.
2022-11-25 02:03:32,756 - root - INFO - Output dimension CL_AREA_IFS table to ../out/dim_
↪cl_area_ifs.csv
2022-11-25 02:03:32,769 - root - INFO - Output dimension CL_INDICATOR_IFS table to ../
↪out/dim_cl_indicator_ifs.csv
2022-11-25 02:03:33,126 - root - INFO - Output meta data of IFS table to ../out/meta_ifs.
↪csv
2022-11-25 02:03:33,141 - root - INFO - self.meta_df.shape = (3, 3)
2022-11-25 02:03:33,143 - root - INFO - Output meta data of IFS containing 'gross␣
↪Domestic Product, Real' table to ../out/meta_gross Domestic Product, Real_CA_RU_Q_1970_
↪2022.csv
2022-11-25 02:04:33,893 - root - INFO - self.data_df.shape = (361, 5)
2022-11-25 02:04:33,904 - root - INFO - Output data of IFS containing 'gross Domestic␣
↪Product, Real' table to ../out/data_gross Domestic Product, Real_CA_RU_Q_1970_2022.csv
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

```
bop = BOP(search_terms=["current account, total, credit"], countries=["US"], period='Q',
                  start_date="2000", end_date="2022")
df = bop.download_data()
```

```
2022-11-25 02:07:43,545 - root - INFO - countries_str = 'US', self.start_date = '2000',␣
↪self.end_date = '2022'
2022-11-25 02:07:44,079 - root - INFO - Output all IMF series table to ../out/series_imf.
↪csv
2022-11-25 02:07:44,108 - root - INFO - Output series containing 'BOP' table to ../out/
↪series_bop.csv
2022-11-25 02:08:10,200 - root - WARNING - Failed to download CL_FREQ.
2022-11-25 02:08:12,137 - root - INFO - Output dimension CL_AREA_BOP table to ../out/dim_
↪cl_area_bop.csv
2022-11-25 02:08:12,169 - root - INFO - Output dimension CL_INDICATOR_BOP table to ../
↪out/dim_cl_indicator_bop.csv
2022-11-25 02:08:13,664 - root - INFO - Output meta data of BOP table to ../out/meta_bop.
↪csv
2022-11-25 02:08:13,680 - root - INFO - self.meta_df.shape = (6, 3)
2022-11-25 02:08:13,682 - root - INFO - Output meta data of BOP containing 'current␣
↪account, total, credit' table to ../out/meta_current account, total, credit_US_Q_2000_
↪2022.csv
2022-11-25 02:08:43,021 - root - INFO - self.data_df.shape = (90, 5)
2022-11-25 02:08:43,026 - root - INFO - Output data of BOP containing 'current account,␣
↪total, credit' table to ../out/data_current account, total, credit_US_Q_2000_2022.csv
```

```
df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

```
ifs = IFS(series="DOT", search_terms=["trade"], countries=["US"], period='Q', start_date=
↪"2000",
          end_date="2022")
df = ifs.download_data()
df
```

```
ifs = IFS(series="BOP", search_terms=["current account, total, credit"], countries=["US
↪"], period='Q',
      start_date="2000", end_date="2022")
df = ifs.download_data()
df
```

```
ifs = IFS(series="GFSR", search_terms=["central government"], countries=["US"], period='A
↪', start_date="2000", end_date="2022")
df = ifs.download_data()
df
```

```
ifs = IFS(series="FSI", search_terms=["Value of large exposures"], countries=["US"],␣
↪period='A', start_date="2000", end_date="2022")
df = ifs.download_data()
df
```

5.4 Bulk download using imf_data_py

```
from imfdatapy.imf import *
```

```
2023-02-03 17:37:55,360 imf_log.py:39 - INFO - Current directory /Users/terrya/Documents/
↪ProgramData/imfdatapy/demo
2023-02-03 17:37:55,360 imf_log.py:40 - INFO - Started log ../log/imfdatapy_2023-02-03-
↪17-37-55.log
```

```
ifs = IFS(search_terms=["NGDP_R_SA_XDC"], countries=None, period='Q')
df = ifs.download_data()
print(df.head)
print(df.shape)
```

```
2023-02-03 17:37:55,363 imf.py:59 - INFO - Inputs: series = 'IFS', search_terms = ['NGDP_
↪R_SA_XDC']
2023-02-03 17:38:01,740 imf.py:139 - INFO - Output all IMF series in a (259, 11) table␣
↪to ../out/series_imf.csv
```

(continues on next page)

(continued from previous page)

```
2023-02-03 17:38:01,748 imf.py:141 - INFO - Output series containing 'IFS' in a (49, 11)␣
↪table to ../out/series_ifs.csv
```

```
ifs = IFS(search_terms=None, countries=["US"], period='Q')
df = ifs.download_data()
print(df.head)
print(df.shape)
```

CHAPTER

SIX

API REFERENCE

This page contains auto-generated API reference documentation[1].

6.1 imf_log

6.1.1 Module Contents

Classes

LogFile

class imf_log.**LogFile**(*logdir='log'*, *is_log_to_screen=True*)

start_log()

6.2 imf

Authors: Sou-Cheng T. Choi and Irina Klein, Illinois Institute of Technology Updated Date: Dec 7, 2022 Creation Date: Jun 15, 2022

6.2.1 Module Contents

[1] Created with sphinx-autoapi

Classes

Series	Helper class that provides a standard way to create an ABC using
IMF	Helper class that provides a standard way to create an ABC using
AFRREO	Sub-Saharan Africa Regional Economic Outlook (AFRREO). A child class of IMF.
IFS	International Financial Statistics (IFS). A child class of IMF.
DOT	Direction of Trade Statistics (DOT). A child class of IMF.
BOP	Balance of Payments (BOP). A child class of IMF.
FSI	Financial Soundness Indicators (FSIs). A child class of IMF.
GFSR	Government Finance Statistics (GFS), Revenue. A child class of IMF.
COFOG	Government Finance Statistics (GFS), Expenditure by Function of Government (COFOG). A child class of IMF.
HPDD	Historical Public Debt Database (HPDD). A child class of IMF.

Attributes

MAX_FILENAME_LEN

imf.**MAX_FILENAME_LEN = 260**

class imf.**Series**(*series='IFS'*, *search_terms=None*, *countries=None*, *period='Q'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

get_series_names()

abstract download_meta()

abstract download_data()

abstract get_meta()

abstract get_data()

abstract describe_meta()

abstract describe_data()

class imf.**IMF**(*series='IFS'*, *search_terms=None*, *countries=None*, *period='Q'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *Series*

Helper class that provides a standard way to create an ABC using inheritance.

output_series(*series=None*)

This function outputs all or some IMF series dataframe to a csv file, and logs its file path.

Parameters
series – Series code as a string. If *series* is *None*, then the function outputs all series to a csv file. If *series* is not *None*, then the function outputs only the series that contain the string *series* to a csv file.

output_dim(*dim_name=None*)

This function outputs all or some dimension tables to a .csv file in 'out'

Parameters
dim_name – Dimension name as a string

output_meta(*indicator=None*)

Method to output all or some indicators in a tables to a .csv file in 'out'

Parameters
indicator – Indicator code as a string

output_data(*is_gen_filename=False*)

This function outputs the data to a csv file.

Parameters
is_gen_filename – generate the csv file name from user inputs if *True*, defaults to *False*

gen_data_filename(*is_meta=False*)

It takes the search terms, countries, period, start time, and end time, and creates a filename for the data up to 250 characters

Parameters
is_meta – whether to generate a filename for the meta data or the actual data, defaults to False

(optional). Defaults to False

Returns
The filename and the search terms

get_series_names()

It takes a list of series names, and returns a dataframe with the series names and their corresponding IDs

Returns
A dataframe with the series names and their corresponding IDs.

get_dimensions()

It downloads the dimensions of the series, and then downloads the details of each dimension

download_meta()

The function downloads the meta data of the time series from the IMF API

Returns
The meta data of the time series.

`read_meta_df()`

The function reads a csv file into a Pandas dataframe, renames the columns, and then cleans the column names.

`read_dim_df(dim_name='CL_FREQ')`

The function reads a csv file with dimension data into a Pandas dataframe.

Parameters
dim_name – name of dimension

`download_data()`

It downloads data and its meta data from the IMF web server, and saves it to a csv file.

Returns
The data is being returned as a pandas dataframe.

`validate_inputs()`

The function checks if the user inputs are valid. It will change an invalid input to a valid value with a warning.

`clean_column_names(df)`

It replaces special characters in all column names and makes them upper case

Parameters
df – The Pandas dataframe to be cleaned

Returns
The Pandas dataframe with the cleaned column names.

`repeat_request(url)`

It will try to get a response from the IMF data server for a given url, and if it doesn't get a response, it will wait a few seconds and try again

Parameters
url – the url to request

Returns
The json object is being returned. It will returns *None* if it does not get a valid response after making a few requests to the IMF data server.

`get_meta()`

This function returns the meta data of the IMF economic indicator(s).

Returns
The meta data of the time series data

`get_data()`

This function returns the time series data.

Returns
The dataframe of the time series data.

`describe_data()`

The function returns a summary of data.

Returns
A Pandas dataframe that contains the summary statistics of the data.

describe_meta()

This function takes the meta data dataframe and returns a summary of the meta data.

Returns
Summary of meta data

class imf.**AFRREO**(*series='AFRREO'*, *search_terms=None*, *countries=None*, *period='Q'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *IMF*

Sub-Saharan Africa Regional Economic Outlook (AFRREO). A child class of IMF.

class imf.**IFS**(*series='IFS'*, *search_terms=None*, *countries=None*, *period='Q'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *IMF*

International Financial Statistics (IFS). A child class of IMF.

class imf.**DOT**(*series='DOT'*, *search_terms=None*, *countries=None*, *period='Q'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *IMF*

Direction of Trade Statistics (DOT). A child class of IMF.

class imf.**BOP**(*series='BOP'*, *search_terms=None*, *countries=None*, *period='Q'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *IMF*

Balance of Payments (BOP). A child class of IMF.

class imf.**FSI**(*series='FSI'*, *search_terms=None*, *countries=None*, *period='M'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *IMF*

Financial Soundness Indicators (FSIs). A child class of IMF.

class imf.**GFSR**(*series='GFSR'*, *search_terms=None*, *countries=None*, *period='A'*, *start_date=None*, *end_date=None*, *sector=''*, *unit=''*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *IMF*

Government Finance Statistics (GFS), Revenue. A child class of IMF.

download_data()

It downloads data and its meta data from the IMF web server, and saves it to a csv file.

Returns
The data is being returned as a pandas dataframe.

class imf.**COFOG**(*series='COFOG'*, *search_terms=None*, *countries=None*, *period='A'*, *start_date=None*, *end_date=None*, *outdir=None*)

Bases: *IMF*

Government Finance Statistics (GFS), Expenditure by Function of Government (COFOG). A child class of IMF.

class imf.**HPDD**(*series='HPDD'*, *search_terms=None*, *countries=None*, *period='A'*, *start_date=None*, *end_date=None*, *outdir='out'*, *logdir='log'*, *is_log_to_screen=True*)

Bases: *IMF*

Historical Public Debt Database (HPDD). A child class of IMF.

6.3 __inti__

PYTHON MODULE INDEX

INDEX