SGS

Release 2.1.1

Aug 31, 2021

Contents

CHAPTER 1

Introduction

This library provides a pure Python interface for the Brazilian Central Bank's Time Series Management System (SGS) api. It works with Python 3.5 and above.

SGS is a service with more than 18,000 time series with economical and financial information. This library is intended to make it easier for Python programmers to use this data in projects of any kind, providing mechanisms to search for, extract and join series.

CHAPTER 2

The User Guide

In this section we discuss some of the basic ways in which the package can be used.

2.1 Installing SGS

2.1.1 pip install

To install **sgs**, simply run this command in your terminal of choice:

```
$ pip install sgs
```

2.1.2 Get the Source Code

Sgs is developed on GitHub, where the code is always available.

You can clone the public repository:

```
$ git clone git://github.com/rafpyprog/pySGS.git
```

Once you have a copy of the source, you can embed it in your own Python package, or install it into your site-packages easily:

```
$ cd pySGS
$ pip install .
```

2.2 Quickstart

Eager to get started? This page gives a good introduction in how to get started with **sgs**.

First, make sure that:

- Sgs is *installed*

Let's get started with some simple examples.

2.2.1 Time Serie

Access time series data with **sgs** is very simple

Begin by importing the `sgs` module:

```
>>> import sgs
```

Now, let's try to get a time serie. For this example, let's get the "Interest rate - CDI" time serie in 2018, wich has the code 12. Don't worry, on the next steps we will learn how to search for time series codes:

```
>>> CDI_CODE = 12
>>> ts = sgs.time_serie(CDI_CODE, start='02/01/2018', end='31/12/2018')
```

Now, we have a Pandas Series object called `ts`, with all the data and the index representing the dates.

```
>>> ts.head()
2018-01-02    0.026444
2018-01-03    0.026444
2018-01-04    0.026444
2018-01-05    0.026444
2018-01-08    0.026444
```

2.2.2 Dataframe

A common use case is building a dataframe with many time series. Once you have the desired time series codes, you can easily create a dataframe with a single line of code. PySGS will fetch the data and join the time series using the dates. Lets create a dataframe with two time series:

```
>>> CDI = 12
>>> INCC = 192  #  National Index of Building Costs
>>> df = sgs.dataframe([CDI, INCC], start='02/01/2018', end='31/12/2018')
```

Now, we have a Pandas DataFrame object called `df`, with all the data and the index representing the dates used to join the two time series.

```
>>> df.head()
                 12    192
2018-01-01       NaN  0.31
2018-01-02  0.026444   NaN
2018-01-03  0.026444   NaN
2018-01-04  0.026444   NaN
2018-01-05  0.026444   NaN
```

The `NaN` values are due to the fact that the INCC time serie frequency is monthly while CDI has a daily frequency.

2.2.3 Searching

The SGS service provides thousands of time series. It's possible to search for time series by code and also by name, with support to queries in English and Portuguese.

Search by name

Let's perform a search for time series with data about gold.

- English

```
>>> results = sgs.search_ts("gold", language="en")
>>> print(len(results)
29
>>> results[0]
{'code': 4, 'name': 'BM&F Gold - gramme', 'unit': 'c.m.u.',
 'frequency': 'D', 'first_value': Timestamp('1989-12-29 00:00:00'),
 'last_value': Timestamp('2019-06-27 00:00:00'), 'source': 'BM&FBOVESPA'}
```

- Portuguese

```
>>> results = sgs.search_ts("ouro", language="pt")
>>> print(len(results)
29
>>> results[0]
{'code': 4, 'name': 'Ouro BM&F - grama', 'unit': 'u.m.c.',
 'frequency': 'D', 'first_value': Timestamp('1989-12-29 00:00:00'),
 'last_value': Timestamp('2019-06-27 00:00:00'), 'source': 'BM&FBOVESPA'}
```

Search by code

If you already have the time serie's code, this may be usefull to get the metadata.

```
>>> GOLD_BMF = 4
>>> sgs.search_ts(GOLD_BMF, language="pt")
[{'code': 4, 'name': 'Ouro BM&F - grama', 'unit': 'u.m.c.', 'frequency': 'D',
  'first_value': Timestamp('1989-12-29 00:00:00'),
  'last_value': Timestamp('2019-06-27 00:00:00'),
  'source': 'BM&FBOVESPA'}]
```

2.2.4 Metadata

To get the metadata about all the series present in a dataframe use the `metadata` function:

```
>>> CDI = 12
>>> INCC = 192  #  National Index of Building Costs
>>> df = sgs.dataframe([CDI, INCC], start='02/01/2018', end='31/12/2018')
>>> sgs.metadata(df)
[{'code': 12, 'name': 'Interest rate - CDI', 'unit': '% p.d.', 'frequency': 'D',
'first_value': Timestamp('1986-03-06 00:00:00'), 'last_value': Timestamp('2019-06-27
↪00:00:00'),
'source': 'Cetip'}, {'code': 192, 'name': 'National Index of Building Costs (INCC)',
'unit': 'Monthly % var.', 'frequency': 'M', 'first_value': Timestamp('1944-02-29
↪00:00:00'),
'last_value': Timestamp('2019-05-01 00:00:00'), 'source': 'FGV'}]
```

CHAPTER 3

The API Documentation / Guide

If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

3.1 Developer Interface

3.1.1 Main Interface

All of PySGS functionality can be accessed by these 4 methods.

`sgs.`**`time_serie`**(*ts_code: int*, *start: str*, *end: str*, *strict: bool = False*) → pandas.core.series.Series
Request a time serie data.

Parameters

- **`ts_code`** – time serie code.
- **`start`** – start date (DD/MM/YYYY).
- **`end`** – end date (DD/MM/YYYY).
- **`strict`** – boolean to enforce a strict date range.

Returns Time serie values as pandas Series indexed by date.

Return type pandas.Series

Usage:

```
>>> CDI = 12
>>> ts = sgs.time_serie(CDI, start='02/01/2018', end='31/12/2018')
>>> ts.head()
2018-01-02    0.026444
2018-01-03    0.026444
2018-01-04    0.026444
2018-01-05    0.026444
2018-01-08    0.026444
```

sgs.**dataframe**(*ts_codes: Union[int, List[T], Tuple], start: str, end: str, strict: bool = False*) → pandas.core.frame.DataFrame

Creates a dataframe from a list of time serie codes.

Parameters

- **ts_codes** – single code or list/tuple of time series codes.
- **start** – start date (DD/MM/YYYY).
- **end** – end date (DD/MM/YYYY).
- **strict** – boolean to enforce a strict date range.

Returns Pandas dataframe.

Return type pandas.DataFrame

Usage:

```
>>> CDI = 12
>>> INCC = 192  #  National Index of Building Costs
>>> df = sgs.dataframe([CDI, INCC], start='02/01/2018', end='31/12/2018')
>>> df.head()
                 12    192
2018-01-01       NaN  0.31
2018-01-02  0.026444   NaN
2018-01-03  0.026444   NaN
2018-01-04  0.026444   NaN
2018-01-05  0.026444   NaN
```

sgs.**search_ts**(*query: Union[int, str], language: str*) → Optional[list]

Search for time series and return metadata about it.

Parameters

- **query** – code(int) or name(str) used to search for a time serie.
- **language** – string (en or pt) used in query and return results.

Returns List of results matching the search query.

Return type list

Usage:

```
>>> results = sgs.search_ts("gold", language="en")
>>> len(results)
29
>>> results[0]
{'code': 4, 'name': 'BM&F Gold - gramme', 'unit': 'c.m.u.',
'frequency': 'D', 'first_value': Timestamp('1989-12-29 00:00:00'),
'last_value': Timestamp('2019-06-27 00:00:00'), 'source': 'BM&FBOVESPA'}
```

sgs.**metadata**(*ts_code: Union[int, pandas.core.frame.DataFrame], language: str = 'en'*) → Optional[List[T]]

Request metadata about a time serie or all time series in a pandas dataframe.

Parameters

- **ts_code** – time serie code or pandas dataframe with time series as columns.
- **language** – language of the returned metadata.

Returns List of dicts containing time series metadata.

Return type list

Usage:

```
>>> CDI = 12
>>> INCC = 192  #  National Index of Building Costs
>>> df = sgs.dataframe([CDI, INCC], start='02/01/2018', end='31/12/2018')
>>> sgs.metadata(df)
[{'code': 12, 'name': 'Interest rate - CDI', 'unit': '% p.d.', 'frequency': 'D',
'first_value': Timestamp('1986-03-06 00:00:00'), 'last_value': Timestamp('2019-06-
↪27 00:00:00'),
'source': 'Cetip'}, {'code': 192, 'name': 'National Index of Building Costs (INCC)
↪',
'unit': 'Monthly % var.', 'frequency': 'M', 'first_value': Timestamp('1944-02-29␣
↪00:00:00'),
'last_value': Timestamp('2019-05-01 00:00:00'), 'source': 'FGV'}]
```

Python Module Index

S

Index