Calculadora do Cidadão

Release 1.0.0

Eduardo Cuducos

07 ago. 2023

Índice:

1 Uso **3**
- 1.1 Adaptadores disponíveis . . . 3
- 1.2 Uso de um adaptador . . . 4
- 1.3 Exportando os dados . . . 5
- 1.4 Importando os dados . . . 6

2 Desenvolvendo novos adaptadores **7**
- 2.1 Método obrigatório . . . 7
- 2.2 Variáveis obrigatórias . . . 7
- 2.3 Métodos opcionais . . . 7
- 2.4 Variáveis opcionais . . . 8

3 API **9**
- 3.1 Adaptador Base . . . 9
- 3.2 Adaptadores . . . 11
- 3.3 Download . . . 14
- 3.4 Campos . . . 14

4 Referências **15**

Índice **17**

A Calculadora do Cidadão requer Python 3.7 ou 3.8. Como ela está disponível no PyPI, pode ser instalada com o *pip*:

```
$ pip install calculadora-do-cidadao
```

CAPÍTULO 1

Uso

Todos os adaptadores podem sem iniciados sem argumento algum. Nesse caso, os adaptadores fazem o download dos dados na hora que a classe é instanciada. Ou seja, criar uma instância demora e **é recomendado que sua aplicação faça isso na inicialização, e não a cada uso**.

Como alternativa, caso você já tenha salvo esses dados localmente (ver *Exportando os dados*), é possível iniciar qualquer adaptador passando um *pathlib.Path* de onde ele deve ler os dados.

```
from pathlib import Path

from calculadora_do_cidadao import Ipca


backup = Path("backup.csv")

ipca = Ipca()  # vai fazer o download nesse momento
ipca.to_csv(backup)

ipca = Ipca(backup)  # não fará o download, carregará do backup
```

1.1 Adaptadores disponíveis

Índice	Módulo
DIEESE Cesta Básica: média de todas as cidades disponíveis	*calculadora_do_cidadao.CestaBasica*
DIEESE Cesta Básica: média das capitais da Região Centro-Oeste	*calculadora_do_cidadao.CestaBasicaCentroOeste*
DIEESE Cesta Básica: média das capitais da Região Nordeste	*calculadora_do_cidadao.CestaBasicaNordeste*
DIEESE Cesta Básica: média das capitais da Região Norte	*calculadora_do_cidadao.CestaBasicaNorte*
DIEESE Cesta Básica: média das capitais da Região Sudeste	*calculadora_do_cidadao.CestaBasicaSudeste*
DIEESE Cesta Básica: média das capitais da Região Sul	*calculadora_do_cidadao.CestaBasicaSul*
DIEESE Cesta Básica: Aracaju	*calculadora_do_cidadao.CestaBasicaAracaju*

continua na próxima página

Tabela 1 – continuação da página anterior

Índice	Módulo
DIEESE Cesta Básica: Belém	`calculadora_do_cidadao.CestaBasicaBelem`
DIEESE Cesta Básica: Belo Horizonte	`calculadora_do_cidadao.CestaBasicaBeloHorizonte`
DIEESE Cesta Básica: Boa Vista	`calculadora_do_cidadao.CestaBasicaBoaVista`
DIEESE Cesta Básica: Brasília	`calculadora_do_cidadao.CestaBasicaBrasilia`
DIEESE Cesta Básica: Campo Grande	`calculadora_do_cidadao.CestaBasicaCampoGrande`
DIEESE Cesta Básica: Cuiaba	`calculadora_do_cidadao.CestaBasicaCuiaba`
DIEESE Cesta Básica: Curitiba	`calculadora_do_cidadao.CestaBasicaCuritiba`
DIEESE Cesta Básica: Florianópolis	`calculadora_do_cidadao.CestaBasicaFlorianopolis`
DIEESE Cesta Básica: Fortaleza	`calculadora_do_cidadao.CestaBasicaFortaleza`
DIEESE Cesta Básica: Goiânia	`calculadora_do_cidadao.CestaBasicaGoiania`
DIEESE Cesta Básica: João Pessoa	`calculadora_do_cidadao.CestaBasicaJoaoPessoa`
DIEESE Cesta Básica: Macaé	`calculadora_do_cidadao.CestaBasicaMacae`
DIEESE Cesta Básica: Macapá	`calculadora_do_cidadao.CestaBasicaMacapa`
DIEESE Cesta Básica: Maceió	`calculadora_do_cidadao.CestaBasicaMaceio`
DIEESE Cesta Básica: Manaus	`calculadora_do_cidadao.CestaBasicaManaus`
DIEESE Cesta Básica: Natal	`calculadora_do_cidadao.CestaBasicaNatal`
DIEESE Cesta Básica: Palmas	`calculadora_do_cidadao.CestaBasicaPalmas`
DIEESE Cesta Básica: Porto Alegre	`calculadora_do_cidadao.CestaBasicaPortoAlegre`
DIEESE Cesta Básica: Porto Velho	`calculadora_do_cidadao.CestaBasicaPortoVelho`
DIEESE Cesta Básica: Recife	`calculadora_do_cidadao.CestaBasicaRecife`
DIEESE Cesta Básica: Rio Branco	`calculadora_do_cidadao.CestaBasicaRioBranco`
DIEESE Cesta Básica: Rio de Janeiro	`calculadora_do_cidadao.CestaBasicaRioDeJaneiro`
DIEESE Cesta Básica: Salvador	`calculadora_do_cidadao.CestaBasicaSalvador`
DIEESE Cesta Básica: São Luís	`calculadora_do_cidadao.CestaBasicaSaoLuis`
DIEESE Cesta Básica: São Paulo	`calculadora_do_cidadao.CestaBasicaSaoPaulo`
DIEESE Cesta Básica: Teresina	`calculadora_do_cidadao.CestaBasicaTeresina`
DIEESE Cesta Básica: Vitória	`calculadora_do_cidadao.CestaBasicaVitoria`
FED's Consumer Price Index for All Urban Consumers: All Items	`calculadora_do_cidadao.AllUrbanCityAverage`
IGP-M	`calculadora_do_cidadao.Igpm`
INPC	`calculadora_do_cidadao.Inpc`
IPCA	`calculadora_do_cidadao.Ipca`
IPCA-15	`calculadora_do_cidadao.Ipca15`
IPCA-E	`calculadora_do_cidadao.IpcaE`

1.2 Uso de um adaptador

Todos os adaptadores tem o método *adjust* (`calculadora_do_cidadao.adapters.Adapter.adjust()`) que recebe três argumentos:

Argumento	Obrigatório	Tipo	Descrição	Valor padrão
original_date		*datetime.date*, *datetime.datetime*, *str*, *int* ou *float*	Data original do valor a ser corrigido.	
value		*decimal.Decimal*, *float* ou *int*	Valor a ser corrigido.	*decimal.Decimal('1')*
target_date		*datetime.date*, *datetime.datetime*, *str*, *int* ou *float*	Data para quando o valor tem que ser corrigido.	*datetime.date.today()*

1.2.1 Exemplo

```
In [1]: from datetime import date
   ...: from decimal import Decimal
   ...: from calculadora_do_cidadao import Ipca

In [2]: ipca = Ipca()

In [3]: ipca.adjust(date(2018, 7, 6))
Out[3]: Decimal('1.0512022066305612800035407253')

In [4]: ipca.adjust(date(2014, 7, 8), 7)
Out[4]: Decimal('9.407523138792336916983267321')

In [5]: ipca.adjust(date(1998, 7, 12), 3, date(2006, 7, 1))
Out[5]: Decimal('5.279855889296777979447848574')
```

1.2.2 Formatos dos campos de data

Os adaptadores aceitam diversos formatos de data, como descrevem os exemplos a seguir:

Entrada	Tipo	Saída
datetime.date(2018, 7, 6)	*datetime.date*	*datetime.date(2018, 7, 6)*
datetime.datetime(2018, 7, 6, 21, 0, 0)	*datetime.datetime*	*datetime.date(2018, 7, 6)*
"2018-07-06T21:00:00"	*str*	*datetime.date(2018, 7, 6)*
"2018-07-06 21:00:00"	*str*	*datetime.date(2018, 7, 6)*
"2018-07-06"	*str*	*datetime.date(2018, 7, 6)*
"06/07/2018"	*str*	*datetime.date(2018, 7, 6)*
"2018-07"	*str*	*datetime.date(2018, 7, 1)*
"Jul/2018"	*str*	*datetime.date(2018, 7, 1)*
"Jul-2018"	*str*	*datetime.date(2018, 7, 1)*
"Jul 2018"	*str*	*datetime.date(2018, 7, 1)*
"07/2018" `	*str*	*datetime.date(2018, 7, 1)*
"2018"	*str*	*datetime.date(2018, 1, 1)*
1530925200	*int* (timestamp)	*datetime.date(2018, 7, 6)*
1530925200.0	*float* (timestamp)	*datetime.date(2018, 7, 6)*

1.3 Exportando os dados

Todos os adaptadores tem o método *to_csv* (`calculadora_do_cidadao.adapters.Adapter.to_csv()`) para exportar os dados no formato CSV. O único argumento que esse método recebe é um *pathlib.Path* que é o caminho do arquivo para onde os dados serão exportados.

Para exportar os dados de todos os índices (adaptadores) de uma vez só é só chamar o pacote pela linha de comando (será criado o arquivo *calculadora-do-cidadao.csv* com os dados):

```
$ python -m calculadora_do_cidadao
```

1.4 Importando os dados

Todos os adaptadores tem o método *from_csv* (`calculadora_do_cidadao.adapters.Adapter.from_csv()`) para importar os dados de um arquivo CSV. O único argumento que esse método recebe é um *pathlib.Path* que é o caminho do arquivo onde os dados estão. O arquivo deve ter duas colunas, *date* no formato *YYYY-MM-DD*, e *value* utilizando um ponto como separador das casas decimais.

CAPÍTULO 2

Desenvolvendo novos adaptadores

Todos os adaptadores herdam de `calculadora_do_cidadao.adapters.Adapter`.

2.1 Método obrigatório

Todo adaptador precisa de um método *serialize*. Esse método sempre recebe uma linha da tabela (*NamedTuple* instanciada pela rows e é um **gerador** que devolve:

- ou *None* (caso seja uma linha inválida)
- ou uma tupla contendo um *datetime.date* e um *decimal.Decimal*

2.2 Variáveis obrigatórias

Variável	Descrição
url	URL da fonte para baixar os dados.
file_type	*"html"* ou *"xls"*, indicando o formato dos dados na fonte.

2.3 Métodos opcionais

2.3.1 *post_processing*

Um método estático (`staticmethod`) ou função que recebe *bytes* como seu único argumento e também retorna *bytes*. Utilizado quando o documento a ser baixado está corrimpido na fonte, por exemplo. Essa função é executada antes de salvar o arquivo, dando a chance de corrigi-lo caso necessário.

2.4 Variáveis opcionais

2.4.1 *HEADERS*

No caso de a URL usar o protocolo HTTP, essa variável pode ser um dicionário que será incluído como _headers_ em cada requisição HTTP.

2.4.2 *COOKIES*

No caso de a URL usar o protocolo HTTP, essa variável pode ser um dicionário que será incluído como _cookies_ da sessão na requisição HTTP.

2.4.3 *SHOULD_UNZIP*

Um booleano informando se o arquivo baixado da URL precisa ser descompactado ou não (apenas *.zip* é suportado por enquanto).

2.4.4 *SHOULD_AGGREGATE*

Um booleano informando se os dados estão desagregados (por exemplo, 0,42%) ou se eles já representam o acumulado desde o início da série (1,0042, por exemplo).

2.4.5 *IMPORT_KWARGS*

Argumentos nomeados que serem passados passados para a função de leitura dos dados (*rows.import_from_html*, por exemplo).

Essa variável pode ser um dicionário e, nesse caso, a função de leitura será chamada apenas uma vez, desempacotando o dicionário como argumentos nomeados.

Ainda, essa variável pode ser uma sequência de dicionários e, nesse caso, a função de leitura será chamada várias vezes, uma vez para cada dicionário da sequência.

2.4.6 *POST_DATA*

Dicionário com valores que serão passados via HTTP POST para a URL especificada nesse adaptdor. A requisição HTTP será do tipo GET caso essa variável não seja criada.

Ainda, essa variável pode ser uma sequência de dicionários e, nesse caso, serão feitas várias requisições, uma com cada conunto de dados dessa sequência.

CAPÍTULO 3

API

3.1 Adaptador Base

calculadora_do_cidadao.adapters.**import_from_json**(*path: Path*, *json_path: List[str]*) → Iterable[NamedTuple]

Imports data form a JSON file *path* creating an iterable of named tuples similar to the Rows's import functions.

json_path is a sequence of keys or array indexes to get to the array with the desired data.

class calculadora_do_cidadao.adapters.**Adapter**(*exported_csv: Optional[Path] = None*)

This is the base adapter, all adapters should inherit from it. Its children require at least a *url* and *file_type* class variables.

adjust(*original_date: Union[date, datetime, int, float, str]*, *value: Optional[Union[Decimal, float, int]] = 0*, *target_date: Optional[Union[date, datetime, int, float, str]] = None*) → Decimal

Main method of an adapter API, the one that actually makes the monetary correction using adapter's data. It requires a *datetime.date* used as the reference for the operation.

If no *value* if given, it returns considering the value is *decimal.Decimal('1')*.

If no *target_date* is given, it returns considering the target date is *datetime.date.today()*.

aggregate()

Being disaggregated here means the index for each month is a percentage relative to the previous month. However the *adjust* method gets way simpler if the indexes as stored as the percentage of the month before the first month of the series. For example, if a given index starts at January 1994, and all values should be a percentage referring to December 1993.

property cookies: dict

Wrapper to get COOKIES if set, avoiding error if not set.

download() → Iterable[Tuple[date, Decimal]]

Wrapper to use the *Download* class and pipe the result to *rows* imported method, yielding a series of rows parsed by *rows*.

export(*include_name: bool = False*) → Iterable[dict]

Wraps adapter's data in a sequence of dictionaries to be used with *rows.import_from_dicts*.

export_index(*key*, *include_name: bool = False*) → dict

Export a given index as a dictionary to be used with *rows.import_from_dicts*.

abstract property file_type: str

File type of the response from the *url*, usually html or xls.

from_csv(*path: Path*) → Iterable[Tuple[date, Decimal]]

Load adapter's data from a CSV file. If the CSV file has two columns it is assumed it was generated with the *to_csv* method. If it has 3 columns, it is assumed it is a export of all adapters' data generated by the CLI.

property headers: Optional[dict]

Wrapper to get HEADERS if set, avoiding error if not set.

property import_kwargs: Iterable[dict]

Wrapper to get IMPORT_KWARGS if set, avoiding error if not set.

invalid_date_error_message(*wanted: date*) → str

Helper to generate an error message usually used together with *AdapterDateNotAvailableError*.

property post_data: Optional[dict]

Wrapper to get POST_DATA if set, avoiding error if not set.

round_date(*obj: Union[date, datetime, int, float, str]*, *validate: bool = False*) → date

Method to round *Date* objects to hold *day = 1*, as indexes usually refers to monthly periods, not daily periods. It also validates if the intended date is valid and in the adapter data range.

abstract serialize(*row: NamedTuple*) → Iterable[Optional[Tuple[date, Decimal]]]

This method should be a generator that receives a row from *rows* (which is a *NamedTuple*) and yields *None* if the row does not hold any valid index data, or yields *calculadora_do_cidadao.typing.Index* type if the row has valid data. A row can yield more than one *calculadora_do_cidadao.typing.Index*.

property should_aggregate: bool

Wrapper to get SHOULD_AGGREGATE if set, avoiding error if not set.

property should_unzip: bool

Wrapper to get SHOULD_UNZIP if set, avoiding error if not set.

to_csv(*path: Path*) → Path

Export the adapter's data to a CSV file.

abstract property url: str

The URL where to get data from.

exception calculadora_do_cidadao.adapters.**AdapterNoImportMethod**

To be used when the adapter has no *rows* import method set.

exception calculadora_do_cidadao.adapters.**AdapterDateNotAvailableError**

To be used when using a date outside of the range available.

3.2 Adaptadores

3.2.1 DIEESE Cesta Básica

class calculadora_do_cidadao.**CestaBasica**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index. If no *cities* variable is created, it averages the value of all available cities in any given date (this is used in subclasses).

static post_processing(*body: bytes*) → bytes

Fixes broken HTML syntax in DIEESE's the source file.

serialize(*row: NamedTuple*) → Iterable[Optional[Tuple[date, Decimal]]]

Serialize method to unpack rows's row into a tuple. Calculates the mean for adapters including different cities if needed.

class calculadora_do_cidadao.**CestaBasicaCentroOeste**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index including Brasília, Cuiabá, Campo Grande and Goiânia.

class calculadora_do_cidadao.**CestaBasicaNordeste**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index including Aracajú, Fortaleza, João Pessoa, Maceió, Natal, Recife, Salvador, São Luís and Teresina.

class calculadora_do_cidadao.**CestaBasicaNorte**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index including Belém, Boa Vista, Macapá, Manaus, Palmas, Porto Velho and Rio Branco.

class calculadora_do_cidadao.**CestaBasicaSudeste**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index including Belo Horizonte, Rio de Janeiro, São Paulo and Vitória.

class calculadora_do_cidadao.**CestaBasicaSul**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index including Curitiba, Florianópolis and Porto Alegre.

class calculadora_do_cidadao.**CestaBasicaAracaju**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Aracajú.

class calculadora_do_cidadao.**CestaBasicaBelem**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Belém.

class calculadora_do_cidadao.**CestaBasicaBeloHorizonte**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Belo Horizonte.

class calculadora_do_cidadao.**CestaBasicaBoaVista**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Boa Vista.

class calculadora_do_cidadao.**CestaBasicaBrasilia**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Brasília.

class calculadora_do_cidadao.**CestaBasicaCampoGrande**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Campo Grande.

class calculadora_do_cidadao.**CestaBasicaCuiaba**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Cuiabá.

class calculadora_do_cidadao.**CestaBasicaCuritiba**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Curitiba.

class calculadora_do_cidadao.**CestaBasicaFlorianopolis**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Florianópolis.

class calculadora_do_cidadao.**CestaBasicaFortaleza**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Fortaleza.

class calculadora_do_cidadao.**CestaBasicaGoiania**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Goiânia.

class calculadora_do_cidadao.**CestaBasicaJoaoPessoa**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for João Pessoa.

class calculadora_do_cidadao.**CestaBasicaMacae**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Macaé.

class calculadora_do_cidadao.**CestaBasicaMacapa**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Macapá.

class calculadora_do_cidadao.**CestaBasicaMaceio**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Maceió.

class calculadora_do_cidadao.**CestaBasicaManaus**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Manaus.

class calculadora_do_cidadao.**CestaBasicaNatal**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Natal.

class calculadora_do_cidadao.**CestaBasicaPalmas**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Palmas.

class calculadora_do_cidadao.**CestaBasicaPortoAlegre**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Porto Alegre.

class calculadora_do_cidadao.**CestaBasicaPortoVelho**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Porto Velho.

class calculadora_do_cidadao.**CestaBasicaRecife**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Recife.

class calculadora_do_cidadao.**CestaBasicaRioBranco**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Rio Branco.

class calculadora_do_cidadao.**CestaBasicaRioDeJaneiro**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Rio de Janeiro.

class calculadora_do_cidadao.**CestaBasicaSalvador**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for Salvador.

class calculadora_do_cidadao.**CestaBasicaSaoLuis**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for São Luís.

class calculadora_do_cidadao.**CestaBasicaSaoPaulo**(*exported_csv: Optional[Path] = None*)
Adapter for DIEESE's basic shopping basket (cesta básica) price index for São Paulo.

class calculadora_do_cidadao.**CestaBasicaTeresina**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Teresina.

class calculadora_do_cidadao.**CestaBasicaVitoria**(*exported_csv: Optional[Path] = None*)

Adapter for DIEESE's basic shopping basket (cesta básica) price index for Vitória.

3.2.2 FED's Consumer Price Index for All Urban Consumers: All Items

class calculadora_do_cidadao.**AllUrbanCityAverage**(*exported_csv: Optional[Path] = None*)

Adapter for FED's Consumer Price Index for All Urban Consumers: All Items.

serialize(*row: NamedTuple*) → Iterable[Optional[Tuple[date, Decimal]]]

Serialize method to unpack Rows's row into a tuple.

3.2.3 IGP-M

class calculadora_do_cidadao.**Igpm**(*exported_csv: Optional[Path] = None*)

Adapter for FGV's IGPM series.

serialize(*row: NamedTuple*) → Iterable[Optional[Tuple[date, Decimal]]]

Serialize method to discard the rows that are not valid data.

3.2.4 Família IPCA & INPC

class calculadora_do_cidadao.adapters.ibge.**IbgeAdapter**(*exported_csv: Optional[Path] = None*)

This base class is incomplete and should not be used directly. It missed the *url* class variable to be set in its children. In spite of that, it implements the serialize and settings that work with most price adjustment indexes done by IBGE.

serialize(*row: NamedTuple*) → Iterable[Optional[Tuple[date, Decimal]]]

Serialize used for different IBGE price adjustment indexes.

class calculadora_do_cidadao.**Inpc**(*exported_csv: Optional[Path] = None*)

Adapter for IBGE's INPC series.

class calculadora_do_cidadao.**Ipca**(*exported_csv: Optional[Path] = None*)

Adapter for IBGE's IPCA series.

class calculadora_do_cidadao.**Ipca15**(*exported_csv: Optional[Path] = None*)

Adapter for IBGE's IPCA-15 series.

class calculadora_do_cidadao.**IpcaE**(*exported_csv: Optional[Path] = None*)

Adapter for IBGE's IPCA-E series.

3.3 Download

class calculadora_do_cidadao.download.**Download**(*url: str*, *should_unzip: bool = False*, *headers: Optional[dict] = None*, *cookies: Optional[dict] = None*, *post_data: Optional[Union[dict, Iterable[dict]]] = None*, *post_processing: Optional[Callable[[bytes], bytes]] = None*)

Abstraction for the download of data from the source.

It can be initialized informing that the resulting file is a Zip archive that should be unarchived.

Cookies and headers are just relevant if the URL uses HTTP (and, surely both are optional).

The *post_data* dictionary is used to send an HTTP POST request (instead of the default GET). If this field is a sequence of dictionaries, it will result in one request per dictionary.

The *post_processing* as a bytes to bytes function that is able to edit the contents before saving it locally, allowing adapter to fix malformed documents.

http() → Iterable[bytes]

Download the source file(s) using HTTP.

static unzip(*path: Path*, *target: Path*) → Path

Unzips the first file of an archive and returns its path.

exception calculadora_do_cidadao.download.**DownloadMethodNotImplementedError**

To be used when the *Download* class does not have a method implemented to download a file using the protocol specified in the *url* argument.

3.4 Campos

class calculadora_do_cidadao.fields.**DateField**

DateField which supports different date formats, including Brazilian

classmethod deserialize(*value: Union[date, datetime, int, float, str]*, **args*, ***kwargs*) → date

Deserialize a value just after importing it

cls.deserialize should always return a value of type *cls.TYPE* or *None*.

class calculadora_do_cidadao.fields.**PercentField**

Field for reading percentage in Brazilian Portuguese format.

classmethod deserialize(*value: str*) → Decimal

Deserialize decimals using a comma as a decimal separator.

CAPÍTULO 4

Referências

- genindex
- modindex
- search

Índice

A

Adapter (*classe em calculadora_do_cidadao.adapters*), 9
AdapterDateNotAvailableError, 10
AdapterNoImportMethod, 10
adjust() (*método calculadora_do_cidadao.adapters.Adapter*), 9
aggregate() (*método calculadora_do_cidadao.adapters.Adapter*), 9
AllUrbanCityAverage (*classe em calculadora_do_cidadao*), 13

C

CestaBasica (*classe em calculadora_do_cidadao*), 11
CestaBasicaAracaju (*classe em calculadora_do_cidadao*), 11
CestaBasicaBelem (*classe em calculadora_do_cidadao*), 11
CestaBasicaBeloHorizonte (*classe em calculadora_do_cidadao*), 11
CestaBasicaBoaVista (*classe em calculadora_do_cidadao*), 11
CestaBasicaBrasilia (*classe em calculadora_do_cidadao*), 11
CestaBasicaCampoGrande (*classe em calculadora_do_cidadao*), 11
CestaBasicaCentroOeste (*classe em calculadora_do_cidadao*), 11
CestaBasicaCuiaba (*classe em calculadora_do_cidadao*), 11
CestaBasicaCuritiba (*classe em calculadora_do_cidadao*), 11
CestaBasicaFlorianopolis (*classe em calculadora_do_cidadao*), 12
CestaBasicaFortaleza (*classe em calculadora_do_cidadao*), 12
CestaBasicaGoiania (*classe em calculadora_do_cidadao*), 12
CestaBasicaJoaoPessoa (*classe em calculadora_do_cidadao*), 12
CestaBasicaMacae (*classe em calculadora_do_cidadao*), 12
CestaBasicaMacapa (*classe em calculadora_do_cidadao*), 12
CestaBasicaMaceio (*classe em calculadora_do_cidadao*), 12
CestaBasicaManaus (*classe em calculadora_do_cidadao*), 12
CestaBasicaNatal (*classe em calculadora_do_cidadao*), 12
CestaBasicaNordeste (*classe em calculadora_do_cidadao*), 11
CestaBasicaNorte (*classe em calculadora_do_cidadao*), 11
CestaBasicaPalmas (*classe em calculadora_do_cidadao*), 12
CestaBasicaPortoAlegre (*classe em calculadora_do_cidadao*), 12
CestaBasicaPortoVelho (*classe em calculadora_do_cidadao*), 12
CestaBasicaRecife (*classe em calculadora_do_cidadao*), 12
CestaBasicaRioBranco (*classe em calculadora_do_cidadao*), 12
CestaBasicaRioDeJaneiro (*classe em calculadora_do_cidadao*), 12
CestaBasicaSalvador (*classe em calculadora_do_cidadao*), 12
CestaBasicaSaoLuis (*classe em calculadora_do_cidadao*), 12
CestaBasicaSaoPaulo (*classe em calculadora_do_cidadao*), 12
CestaBasicaSudeste (*classe em calculadora_do_cidadao*), 11
CestaBasicaSul (*classe em calculadora_do_cidadao*), 11
CestaBasicaTeresina (*classe em calculadora_do_cidadao*), 12
CestaBasicaVitoria (*classe em calcula-*

dora_do_cidadao), 13
cookies (*propriedade calculadora_do_cidadao.adapters.Adapter*), 9

D

DateField (*classe em calculadora_do_cidadao.fields*), 14
deserialize() (*método de classe calculadora_do_cidadao.fields.DateField*), 14
deserialize() (*método de classe calculadora_do_cidadao.fields.PercentField*), 14
Download (*classe em calculadora_do_cidadao.download*), 14
download() (*método calculadora_do_cidadao.adapters.Adapter*), 9
DownloadMethodNotImplementedError, 14

E

export() (*método calculadora_do_cidadao.adapters.Adapter*), 9
export_index() (*método calculadora_do_cidadao.adapters.Adapter*), 10

F

file_type (*propriedade calculadora_do_cidadao.adapters.Adapter*), 10
from_csv() (*método calculadora_do_cidadao.adapters.Adapter*), 10

H

headers (*propriedade calculadora_do_cidadao.adapters.Adapter*), 10
http() (*método calculadora_do_cidadao.download.Download*), 14

I

IbgeAdapter (*classe em calculadora_do_cidadao.adapters.ibge*), 13
Igpm (*classe em calculadora_do_cidadao*), 13
import_from_json() (*no módulo calculadora_do_cidadao.adapters*), 9
import_kwargs (*propriedade calculadora_do_cidadao.adapters.Adapter*), 10
Inpc (*classe em calculadora_do_cidadao*), 13
invalid_date_error_message() (*método calculadora_do_cidadao.adapters.Adapter*), 10
Ipca (*classe em calculadora_do_cidadao*), 13
Ipca15 (*classe em calculadora_do_cidadao*), 13
IpcaE (*classe em calculadora_do_cidadao*), 13

P

PercentField (*classe em calculadora_do_cidadao.fields*), 14
post_data (*propriedade calculadora_do_cidadao.adapters.Adapter*), 10
post_processing() (*método estático calculadora_do_cidadao.CestaBasica*), 11

R

round_date() (*método calculadora_do_cidadao.adapters.Adapter*), 10

S

serialize() (*método calculadora_do_cidadao.adapters.Adapter*), 10
serialize() (*método calculadora_do_cidadao.adapters.ibge.IbgeAdapter*), 13
serialize() (*método calculadora_do_cidadao.AllUrbanCityAverage*), 13
serialize() (*método calculadora_do_cidadao.CestaBasica*), 11
serialize() (*método calculadora_do_cidadao.Igpm*), 13
should_aggregate (*propriedade calculadora_do_cidadao.adapters.Adapter*), 10
should_unzip (*propriedade calculadora_do_cidadao.adapters.Adapter*), 10

T

to_csv() (*método calculadora_do_cidadao.adapters.Adapter*), 10

U

unzip() (*método estático calculadora_do_cidadao.download.Download*), 14
url (*propriedade calculadora_do_cidadao.adapters.Adapter*), 10