pyrtfolio

Release 0.4

Nov 20, 2020

Contents:

1 Introduction 1

2 Installation 3

3 Usage 5

4 API Reference 7
- 4.1 pyrtfolio.Stock 7
- 4.2 pyrtfolio.StockPortfolio 8

5 Contribute 11

6 Indices and tables 13

Python Module Index 15

Index 17

CHAPTER 1

Introduction

pyrtfolio is a Python package created based on investpy data which aims to create custom stock portfolios. In investment, a portfolio is a grouping of financial assets as well as their fund counterparts; note that a portfolio can also consist of non-publicly tradable securities. So on, **investpy** data will be used to create custom portfolios from the data provided by the user such as the asset symbol, purchase date, number of bought shares, etc.

CHAPTER 2

Installation

In order to get the package working you will need to install it via pip by typing the following command in the terminal:

```
$ pip install pyrtfolio==0.4
```

Every package used is listed in requirements.txt file, which can also be installed via pip:

```
$ pip install -r requirements.txt
```

CHAPTER 3

Usage

pyrtfolio usage is pretty simple, since until the current release (0.4), just the StockPortfolios are available with the basic data. So on, the one and only usage that can be currently done with this package consists on generating portfolios for the introduced stocks.

An example will be presented below to show the user how to use the package to generate his/her custom stock portfolios.

```
from pyrtfolio.StockPortfolio import StockPortfolio

portfolio = StockPortfolio()

portfolio.add_stock(stock_name='bbva',
                    stock_country='spain',
                    purchase_date='04/01/2018',
                    num_of_shares=2,
                    cost_per_share=7.2)

portfolio.add_stock(stock_name='endesa',
                    stock_country='spain',
                    purchase_date='13/06/2019',
                    num_of_shares=15,
                    cost_per_share=23.8)

print(portfolio.data)
```

Which outputs the following stock portfolio:

```
  stock_name stock_country purchase_date  num_of_shares  cost_per_share  current_
↪price  gross_current_value
0       bbva          spain    04/01/2018              2             7.2          4.
↪597               9.194
1     endesa          spain    13/06/2019             15            23.8         23.
↪890             358.350
```

Further usage will be documented whenever the package is updated with new features. Make sure to watch and star the repo to be notified about all conversations!

CHAPTER 4

API Reference

4.1 pyrtfolio.Stock

class pyrtfolio.Stock.**Stock**(*stock_symbol*, *stock_country*, *purchase_date*, *num_of_shares*, *cost_per_share*)

Stock is a support class of *pyrtfolio* which validates the Stocks before adding them to StockPortfolio.

This class is a support class to validate the Stocks before finally adding them to the portfolio created in StockPortfolio since the historical data of the introduced Stock needs to be retrieved and, so on, it will error if the introduced Stock is not valid.

stock_symbol
symbol of the Stock that is going to be added to the StockPortfolio.

Type str

stock_country
country from where the specified stock_symbol is, so to validate it.

Type str

purchase_date
date when the shares of the introduced stock were bought, formatted as dd/mm/yyyy.

Type str

num_of_shares
amount of shares bought of the specified Stock in the specified date.

Type int

cost_per_share
price of every share of the Stock in the specified date.

Type float

valid
either True or False if the Stock is valid or not, respectively.

Type boolean

__init__(*stock_symbol*, *stock_country*, *purchase_date*, *num_of_shares*, *cost_per_share*)
This is the init method of Stock class which is launched every time the user instances it.

This method is the init method of this class, Stock, and its main function is to init all the attributes contained in it, as previously defined when the function *add_stock()* was called from StockPortfolio. The specified values in this class are going to be accessed through the self operator whenever the validation function is called, since it is the main function of this class, validating Stocks.

Parameters

- **stock_symbol** (str) – symbol of the Stock that is going to be added to the StockPortfolio.
- **stock_country** (str) – country from where the specified stock_symbol is, so to validate it.
- **purchase_date** (str) – date when the shares of the introduced stock were bought, formatted as dd/mm/yyyy.
- **num_of_shares** (int) – amount of shares bought of the specified Stock in the specified date.
- **cost_per_share** (float) – price of every share of the Stock in the specified date.

__weakref__
list of weak references to the object (if defined)

validate()
Method used to validate that the introduced Stock is valid before adding it to the StockPortfolio.

This method is the one in charge of the validation of the introduced Stock via checking the introduced data with the one indexed in investpy, so to check if the data match. Also, the introduced parameters are checked in order to determine if the type is correct of those values is correct, if not, an exception will be raised. The result of this function is just setting either True or False to the self.valid value if the Stock is valid or not, respectively.

4.2 pyrtfolio.StockPortfolio

class pyrtfolio.StockPortfolio.**StockPortfolio**
StockPortfolio is the main class of *pyrtfolio* which is going to manage all the introduced stocks.

This class is the one that contains the stocks information and the one that will be used by the user in order to generate a custom portfolio. So on, this function implements the methods to calculate all the values required in a basic portfolio and, as already mentioned, the method to add stocks.

stocks
this list contains all the introduced stocks, which will later be used to generate the portfolio.

Type list, protected

stock_objs
this list contains all the introduced Stock objects, in order to be able to refresh them.

Type list, protected

data
it is the generated portfolio, once the addition of every stock is validated.

Type pandas.DataFrame

__init__()
This is the init method of StockPortfolio class which is launched every time the user instances it.

This method is the init method of this class, StockPortfolio, and its main function is to init all the attributes contained in it. Every time this class is instanced, the attributes values are restored and, so on, the portfolio's data is lost if existing for that instance. This class does not take any parameters since they are filled once the class is instanced.

__weakref__
list of weak references to the object (if defined)

add_stock(*stock_symbol*, *stock_country*, *purchase_date*, *num_of_shares*, *cost_per_share*)
Method to add a stock to the portfolio.

This method adds a stock to the custom portfolio data. Some parameters need to be specified for the introduced stock such as the purchase date of the shares, the number of shares bought and the price payed (cost) per every share. From this data, the portfolio will be created and the specified calculations will be done, so to give the user an overview of his/her own portfolio.

Parameters

- **stock_symbol** (str) – symbol of the Stock that is going to be added to the StockPortfolio.
- **stock_country** (str) – country from where the specified stock_symbol is, so to validate it.
- **purchase_date** (str) – date when the shares of the introduced stock were bought, formatted as dd/mm/yyyy.
- **num_of_shares** (int) – amount of shares bought of the specified Stock in the specified date.
- **cost_per_share** (float) – price of every share of the Stock in the specified date.

static calculate_gross_current_value(*current_price*, *num_of_shares*)
This method calculates the gross current value which is the total current value of the shares bought, which is the result of the multiplication of the current price with the number of bought shares.

static calculate_net_current_value(*gross_current_value*, *total_dividends*)
This method calculates the net current value which is the total of the gross current value and the value of the dividends, i.e., the total of returns from both capital gains and dividends.

static calculate_purchase_cost(*cost_per_share*, *num_of_shares*)
This method calculates the purchase cost, which is the paid value for every stock share hold by the user at the time that stock shares were bought.

static calculate_total_dividends(*dividends*, *num_of_shares*)
This method calculates the total dividend's value which is the sum of all the dividends paid in the time range that the user owned/owns the stock shares. Each dividend is paid per the number of stock shares hold by the user.

static calculate_total_gain_loss(*net_current_value*, *purchase_cost*)
This method calculates the difference between the current value of the stock shares, including the dividends received, with the value that the user paid of them.

static calculate_total_gain_loss_percentage(*total_gain_loss*, *purchase_cost*)
This method compares the stock total gain loss to what the user paid for the stock shares owned and the result is a percentage that can show the total stock performance.

static get_current_price(*data*)

This method gets the current price value of the introduced stock, which is the last close value indexed in the `pandas.DataFrame`.

refresh()

Method to refresh/reload the StockPortfolio information.

This method is used to refresh or reload the StockPortfolio information since the values may have changed since the last time the portfolio was generated. So on, this function will return to get the information from every Stock listed in the StockPortfolio.

CHAPTER 5

Contribute

As this is an open source project it is open to contributions, bug reports, bug fixes, documentation improvements, enhancements and ideas.

Also there is an open tab of issues where anyone can contribute opening new issues if needed or navigate through them in order to solve them or contribute to its solving. Remember that issues are not threads to describe multiple issues, this does not mean that issues can't be discussed, but if new issues are reported, a new issue should be open so to keep a structured project management.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

pyrtfolio.Stock, 7
pyrtfolio.StockPortfolio, 8

Index

Symbols

__init__() (*pyrtfolio.Stock.Stock method*), 8
__init__() (*pyrtfolio.StockPortfolio.StockPortfolio method*), 8
__weakref__ (*pyrtfolio.Stock.Stock attribute*), 8
__weakref__ (*pyrtfolio.StockPortfolio.StockPortfolio attribute*), 9

A

add_stock() (*pyrtfolio.StockPortfolio.StockPortfolio method*), 9

C

calculate_gross_current_value() (*pyrtfolio.StockPortfolio.StockPortfolio static method*), 9
calculate_net_current_value() (*pyrtfolio.StockPortfolio.StockPortfolio static method*), 9
calculate_purchase_cost() (*pyrtfolio.StockPortfolio.StockPortfolio static method*), 9
calculate_total_dividends() (*pyrtfolio.StockPortfolio.StockPortfolio static method*), 9
calculate_total_gain_loss() (*pyrtfolio.StockPortfolio.StockPortfolio static method*), 9
calculate_total_gain_loss_percentage() (*pyrtfolio.StockPortfolio.StockPortfolio static method*), 9
cost_per_share (*pyrtfolio.Stock.Stock attribute*), 7

D

data (*pyrtfolio.StockPortfolio.StockPortfolio attribute*), 8

G

get_current_price() (*pyrtfolio.StockPortfolio.StockPortfolio static method*), 9

N

num_of_shares (*pyrtfolio.Stock.Stock attribute*), 7

P

purchase_date (*pyrtfolio.Stock.Stock attribute*), 7
pyrtfolio.Stock (*module*), 7
pyrtfolio.StockPortfolio (*module*), 8

R

refresh() (*pyrtfolio.StockPortfolio.StockPortfolio method*), 10

S

Stock (*class in pyrtfolio.Stock*), 7
stock_country (*pyrtfolio.Stock.Stock attribute*), 7
stock_objs (*pyrtfolio.StockPortfolio.StockPortfolio attribute*), 8
stock_symbol (*pyrtfolio.Stock.Stock attribute*), 7
StockPortfolio (*class in pyrtfolio.StockPortfolio*), 8
stocks (*pyrtfolio.StockPortfolio.StockPortfolio attribute*), 8

V

valid (*pyrtfolio.Stock.Stock attribute*), 7
validate() (*pyrtfolio.Stock.Stock method*), 8