mstarpy

Release 1.0.1

Maël Jourdain

Aug 25, 2023

DOCUMENTATION

CHAPTER

ONE

INTRODUCTION

MStarpy is a Python Package to extract data from morningstar.com.

MStarpy provides stock and fund public data to retail and professional investors for **free**. The main goal is to give investors access to the same information and help them in their investment process.

The project is **open-source** and anyone can contribute on github.

CHAPTER

TWO

GETTING STARTED

2.1 Installation

You can install it **via pip** on the terminal by typing:

```
pip install mstarpy
```

You can also install it **via git** on the terminal bu using :

```
pip install git+https://github.com/Mael-J/mstarpy.git@master
```

2.2 First commands

2.2.1 Look for funds with *search_funds*

You can look for funds by using the method *search_funds*. In the following example, we will look for 40 funds in the US market with the term "technology" in their name. We want to get the name, the ID and the 12 months return. We transform the result in a pandas DataFrame to make it more clear.

```
import mstarpy
import pandas as pd

response = mstarpy.search_funds(term="technology", field=["Name", "fundShareClassId",
↪"GBRReturnM12"], country="us", pageSize=40, currency ="USD")

df = pd.DataFrame(response)
print(df.head())
```

```
                          Name fundShareClassId  GBRReturnM12
0       Baron Technology Instituitional       F00001CUJ3        -21.64
1                    Baron Technology R6       F00001CUJ1        -21.88
2                Baron Technology Retail       F00001CUJ2        -21.91
3          Black Oak Emerging Technology       FOUSA00LIX         -8.33
4  BlackRock Technology Opportunities K       F000014AX6        -21.09
```

2.2.2 Look for fields with *search_field*

You can find the field you need for the *search_funds* and *search_stock* methods using *search_field*. In the following example, we get all fields.

```
from mstarpy import search_field

response = search_field(pattern='')

print(response)
```

```
['AdministratorCompanyId', 'AlphaM36', 'AnalystRatingScale', 'AverageCreditQualityCode',
↪'AverageMarketCapital', 'BetaM36', 'BondStyleBox', 'brandingCompanyId', 'categoryId',
↪'CategoryName', 'ClosePrice', 'currency', 'DebtEquityRatio', 'distribution',
↪'DividendYield', 'EBTMarginYear1', 'EffectiveDuration', 'EPSGrowth3YYear1',
↪'equityStyle', 'EquityStyleBox', 'exchangeCode', 'ExchangeId', 'ExpertiseAdvanced',
↪'ExpertiseBasic', 'ExpertiseInformed', 'FeeLevel', 'fundShareClassId', 'fundSize',
↪'fundStyle', 'FundTNAV', 'GBRReturnD1', 'GBRReturnM0', 'GBRReturnM1', 'GBRReturnM12',
↪'GBRReturnM120', 'GBRReturnM3', 'GBRReturnM36', 'GBRReturnM6', 'GBRReturnM60',
↪'GBRReturnW1', 'geoRegion', 'globalAssetClassId', 'globalCategoryId', 'iMASectorId',
↪'IndustryName', 'InitialPurchase', 'instrumentName', 'investment', 'investmentExpertise
↪', 'investmentObjective', 'investmentType', 'investorType',
↪'InvestorTypeEligibleCounterparty', 'InvestorTypeProfessional', 'InvestorTypeRetail',
↪'LargestSector', 'LegalName', 'managementStyle', 'ManagerTenure', 'MarketCap',
↪'MarketCountryName', 'MaxDeferredLoad', 'MaxFrontEndLoad', 'MaximumExitCostAcquired',
↪'MorningstarRiskM255', 'Name', 'NetMargin', 'ongoingCharge', 'OngoingCostActual',
↪'PEGRatio', 'PERatio', 'PerformanceFeeActual', 'PriceCurrency', 'QuantitativeRating',
↪'R2M36', 'ReturnD1', 'ReturnM0', 'ReturnM1', 'ReturnM12', 'ReturnM120', 'ReturnM3',
↪'ReturnM36', 'ReturnM6', 'ReturnM60', 'ReturnProfileGrowth', 'ReturnProfileHedging',
↪'ReturnProfileIncome', 'ReturnProfileOther', 'ReturnProfilePreservation', 'ReturnW1',
↪'RevenueGrowth3Y', 'riskSrri', 'ROATTM', 'ROETTM', 'ROEYear1', 'ROICYear1', 'SecId',
↪'SectorName', 'shareClassType', 'SharpeM36', 'StandardDeviationM36', 'starRating',
↪'StarRatingM255', 'SustainabilityRank', 'sustainabilityRating', 'TenforeId', 'Ticker',
↪'totalReturn', 'totalReturnTimeFrame', 'TrackRecordExtension', 'TransactionFeeActual',
↪'umbrellaCompanyId', 'Universe', 'Yield_M12', 'yieldPercent']
```

2.2.3 Analysis of funds

Once, you know what fund you want to analyse, you can load it with the class *Funds* and then access all the methods to get data.

```
import mstarpy

fund = mstarpy.Funds(term="FOUSA00LIX", country="us")
```

You can access to his property name.

```
print(fund.name)
```

```
'Black Oak Emerging Technology Fund'
```

You can show the equity holdings of the fund.

```
df_equity_holdings = fund.holdings(holdingType="equity")
print(df_equity_holdings[["securityName", "weighting", "susEsgRiskScore"]].head())
```

```
                    securityName  weighting  susEsgRiskScore
0                     Apple Inc    5.03336          16.6849
1                      KLA Corp    4.90005          16.6870
2  Kulicke & Soffa Industries Inc    4.23065          17.2155
3      SolarEdge Technologies Inc    4.13637          24.6126
4                    Ambarella Inc    4.10950          33.1408
```

You can find the historical Nav and total return of the fund.

```
import datetime
import pandas as pd
start_date = datetime.datetime(2023,1,1)
end_date = datetime.datetime(2023,3,2)
#get historical data
history = fund.nav(start_date=start_date,end_date=end_date, frequency="daily")
#convert it in pandas DataFrame
df_history = pd.DataFrame(history)

print(df_history.head())
```

```
    nav  totalReturn        date
0  6.28     10.21504  2022-12-30
1  6.23     10.13371  2023-01-03
2  6.31     10.26383  2023-01-04
3  6.18     10.05238  2023-01-05
4  6.37     10.36143  2023-01-06
```

2.2.4 Look for stock with *search_stock*

You can look for stocks by using the method *search_stock*. In the following example, we will look for 20 stocks on the Paris Stock Exchange with the term "AB" in their name. We want to get the name, the ID and the Sector. We transform the result in a pandas DataFrame to make it more clear.

```
import mstarpy
import pandas as pd

response = mstarpy.search_stock(term="AB",field=["Name", "fundShareClassId", "SectorName
↪"], exchange='PARIS',pageSize=20)

df = pd.DataFrame(response)
print(df.head())
```

```
                          Name fundShareClassId          SectorName
0                    AB Science       0P0000NQNE          Healthcare
1               ABC arbitrage SA       0P00009W9I  Financial Services
2                       Abeo SA       0P00018PIU   Consumer Cyclical
3  Abionyx Pharma Ordinary Shares       0P00015JGM          Healthcare
4                      Abivax SA       0P00016673          Healthcare
```

Tips : You can get different exchange by looking at the variable EXCHANGE in mstarpy.utils. 'WORLDWIDE_EQUITY' allows you to search in all exchanges.

```
from mstarpy.utils import EXCHANGE

print(list(EXCHANGE))
```

```
['NYSE', 'NASDAQ', 'LSE', 'AMSTERDAM', 'ATHENS', 'BOLSA_DE_VALORES', 'BOMBAY', 'BORSA_
↪ITALIANA', 'BRUSSELS', 'COPENHAGEN', 'HELSINKI', 'HONG-KONG', 'ICELAND', 'INDIA', 'IPSX
↪', 'IRELAND', 'ISTANBUL', 'LISBON', 'LUXEMBOURG', 'OSLO_BORS', 'PARIS', 'RIGA',
↪'SHANGAI', 'SHENZHEN', 'SINGAPORE', 'STOCKHOLM', 'SWISS', 'TAIWAN', 'TALLIN', 'THAILAND
↪', 'TOKYO', 'VILNIUS', 'WARSAW', 'WIENER_BOERSE', 'WORLDWIDE_EQUITY']
```

2.2.5 Analysis of stocks

Once, you know what stock you want to analyse, you can load it with the class *Stock* and then access all the methods to get data.

```
import mstarpy

stock = stock = mstarpy.Stock(term="0P00018PIU", exchange="PARIS")
```

You can access to his property name.

```
print(stock.name)
```

```
'Abeo SA'
```

You can find the historical price and volume of the stock.

```
import datetime
import pandas as pd
start_date = datetime.datetime(2023,1,1)
end_date = datetime.datetime(2023,3,2)
#get historical data
history = stock.historical(start_date=start_date,end_date=end_date, frequency="daily")
#convert it in pandas DataFrame
df_history = pd.DataFrame(history)

print(df_history.head())
```

```
open   high    low  close  volume  previousClose        date
0  18.60  18.60  18.55  18.55     194          18.55  2022-12-30
1  18.70  18.70  18.70  18.70       9          18.55  2023-01-02
2  18.65  18.70  18.55  18.60     275          18.70  2023-01-03
3  18.65  18.65  18.50  18.60     994          18.60  2023-01-04
4  18.65  18.95  18.50  18.60     999          18.60  2023-01-05
```

You can show the financial statements such as the balance sheet.

```
bs = stock.balanceSheet(period='annual', reportType='original')
```

2.3 More commands

You can find all the methods of the classes *Funds* and *Stocks* in the part Indices and tables of this documentation.

CHAPTER

THREE

SEARCH WITH FILTERS

You can use filters to search funds and stocks more precisely with methods *search_funds* and *search_stock*.

3.1 Choose filters

You can find the possible filters with the methods *search_filter*

for funds:

```
from mstarpy import search_filter

filter_fund = search_filter(pattern = '', asset_type ='fund')

print(filter_fund)
```

```
['AdministratorCompanyId', 'AnalystRatingScale', 'BondStyleBox', 'BrandingCompanyId',
↪'CategoryId', 'CollectedSRRI', 'distribution', 'EquityStyleBox', 'ExpertiseInformed',
↪'FeeLevel', 'FundTNAV', 'GBRReturnM0', 'GBRReturnM12', 'GBRReturnM120', 'GBRReturnM36',
↪ 'GBRReturnM60', 'GlobalAssetClassId', 'GlobalCategoryId', 'IMASectorID', 'IndexFund',
↪'InvestorTypeProfessional', 'LargestRegion', 'LargestSector', 'OngoingCharge',
↪'QuantitativeRating', 'ReturnProfilePreservation', 'ShareClassTypeId',
↪'SustainabilityRank', 'UmbrellaCompanyId', 'Yield_M12']
```

for stocks:

```
from mstarpy import search_filter

filter_stock = search_filter(pattern = '', asset_type ='stock')

print(filter_stock)
```

```
['debtEquityRatio', 'DividendYield', 'epsGrowth3YYear1', 'EquityStyleBox', 'GBRReturnM0',
↪ 'GBRReturnM12', 'GBRReturnM36', 'GBRReturnM60', 'GBRReturnM120', 'IndustryId',
↪'MarketCap', 'netMargin', 'PBRatio', 'PEGRatio', 'PERatio', 'PSRatio', 'revenueGrowth3Y
↪', 'roattm', 'roettm', 'SectorId']
```

3.2 Find filters values

Once, you know what filters you want you use the method *filter_universe* to show the possible values of each filter.

```
from mstarpy import filter_universe

filter_value = filter_universe(["GBRReturnM12", "PERatio", "LargestSector"])

print(filter_value)
```

You have two types of filters values, either qualitative or quantitative. By example, the filter LargestSector has qualitative values such as SB_Healthcare or SB_Utilities. The filter PERatio works with quantitative values between 0 and 100000.

3.3 Filter funds

Let say we want to find funds that invest mainly in the consumer defensive sector. We can use filters like in this example:

```
from mstarpy import search_funds

response = search_funds(term='',field=["Name", "fundShareClassId", "GBRReturnM12"],␣
↪country='fr', filters = {"LargestSector" : "SB_ConsumerDefensive"})

df = pd.DataFrame(response)

print(df.head())
```

```
                                Name fundShareClassId  GBRReturnM12
0            AB US High Yield A2 EUR H       F00000O4X9         -9.71
1              AB US High Yield A2 USD       F00000O4XA         -6.88
2            AB US High Yield I2 EUR H       F00000O4X6         -9.18
3              AB US High Yield I2 USD       F00000O4XB         -6.36
4  abrdn China A Share Sus Eq A Acc EUR       F000015MAW         -8.41
```

If we want to search for funds which invest mainly in consumer defensive or healthcare sectors, we can add filters values to a list.

```
from mstarpy import search_funds

response = search_funds(term='',field=["Name", "fundShareClassId", "GBRReturnM12"],␣
↪country='fr', filters = {"LargestSector" : ["SB_ConsumerDefensive", "SB_Healthcare"]})

df = pd.DataFrame(response)

print(df.head())
```

```
                            Name fundShareClassId  GBRReturnM12
0  AB Concentrated Global Eq A EUR H       F00000SJ2P        -10.46
1  AB Concentrated Global Eq I EUR H       F00000SJ2J         -9.77
2    AB Concentrated Global Eq I USD       F00000SE91         -5.77
```

(continues on next page)

(continued from previous page)

```
3    AB Concentrated Global Eq S USD       F00000SE93          1.16
4   AB Concentrated Global Eq S1 EUR       F00001CYZS         -1.89
```

In the previous examples, we saw how to search for securities with a qualitative filter, now let see how to use quantitativer filters.

3.4 Filter stocks

We want to find stocks with a 12 months return superior to 20%. The value of filter is a 2 length tuple. the first element is the sign superior ">", the second element the 12 months return of 20.

```
from mstarpy import search_stock

response = search_stock(term='',field=["Name", "fundShareClassId", "GBRReturnM12",
↪"PERatio"], exchange='PARIS', filters={"GBRReturnM12" : (">", 20)})

df = pd.DataFrame(response)

print(df.head())
```

```
0    1000Mercis SA       0P0000DKX2         24.89    95.24
1          Abeo SA       0P00018PIU         21.73    14.84
2  ABL Diagnostics       0P00009WGF        279.01      NaN
3           Acteos       0P00009W9O         27.01      NaN
4      Actia group       0P00009W9P         44.36      NaN
```

It will work similar if we are looking for stocks with a PERatio inferior to 10. The value of filter is a 2 length tuple. the first element is the sign inferior "<", the second element is the PERatio 10.

```
from mstarpy import search_stock

response = search_stock(term='',field=["Name", "fundShareClassId", "GBRReturnM12",
↪"PERatio"], exchange='PARIS', filters={"PERatio" : ("<", 10)})

df = pd.DataFrame(response)

print(df.head())
```

```
                   Name fundShareClassId  GBRReturnM12  PERatio
0  Acanthe Developpement SA       0P00009W9K        -23.27     5.78
1                   ALD SA       0P0001AM22         31.89     5.07
2              Altarea SCA       0P00009WAG         -2.20     8.18
3  Altur Investissement SCA       0P0000DKYA         33.38     1.98
4                   Archos       0P00009WAT        -97.02     0.00
```

We can also look like stocks with a PERatio between 10 and 20. The value of filter is a 2 length tuple. the first element is the lower bound PERatio of 10, the second element is the upper bound PERatio of 20.

```
from mstarpy import search_stock

```

(continues on next page)

(continued from previous page)

```
response = search_stock(term='',field=["Name", "fundShareClassId", "GBRReturnM12",
↪"PERatio"], exchange='PARIS', filters={"PERatio" : (10, 20)})

df = pd.DataFrame(response)

print(df.head())
```

```
            Name fundShareClassId  GBRReturnM12  PERatio
0  ABC arbitrage SA       0P00009W9I         -5.73    14.10
1           Abeo SA       0P00018PIU         21.73    14.84
2           AdUX SA       0P00009WIO        -32.05    11.49
3       Altareit SA       0P00009WHA        -11.03    12.69
4             Alten       0P00009WAH         14.25    19.96
```

Now we know how to use filters, we can combine them to find a precise securities universe. The world is your oyster.

```
from mstarpy import search_stock

response = search_stock(term='',field=["Name", "fundShareClassId", "GBRReturnM12",
↪"PERatio"],
                        exchange='PARIS', filters={"PERatio" : ("<", '10'), "GBRReturnM12
↪" : (">", 20),
                                                   "debtEquityRatio" : (0, 5), "SectorId
↪" : ["IG000BA008", "IG000BA006"] })

df = pd.DataFrame(response)

print(df.head())
```

```
                      Name fundShareClassId  GBRReturnM12  PERatio
0                  ALD SA       0P0001AM22         31.89     5.07
1                 Coheris       0P00009WDN         72.68     5.27
2  Ediliziacrobatica SpA       0P0001GZM9         24.07     6.85
3                Rexel SA       0P00009WO9         32.27     7.96
4             Soditech SA       0P00009WQ2         97.45     4.49
```

CHAPTER

FOUR

MSTARPY IN THE WORLD

4.1 Albertine.io

The site albertine.io uses MStarpy to compare funds. You can create PDF reports and extract data in Excel format.

4.2 Contribution

The project is **open-source** and you can contribute on github.

CHAPTER

FIVE

DISCLAIMER

MStarpy is not affiliated to morningstar.com or any other companies.

The package aims to share public information about funds and stocks to automatize analysis. It is the result of a free, free and independent work.

MStarpy does not give any investment recommendations.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search