

Stock Assistant

Marcin Krawiec Konrad Syrnik

Program:

W ramach demonstracji naszego projektu zostaną zrealizowane poszczególne punkty:

- PyQt5
- Yfinance
- MySQL with python
- Prezentacja aplikacji *Stock Assistant*
- Test
- Zadania praktyczne

PyQt5

PyQt5

=



+



Qt – co to to takiego?

Qt to zestaw wieloplatformowych bibliotek C++, które implementują interfejsy API wysokiego poziomu w celu uzyskania dostępu do wielu aspektów nowoczesnych systemów stacjonarnych i mobilnych.

Obejmują one usługi:

- lokalizacji i pozycjonowania,
- multimedia,
- łączność NFC i Bluetooth,
- przeglądarkę internetową opartą na Chromium,
- tradycyjne tworzenie interfejsu użytkownika

PyQt5

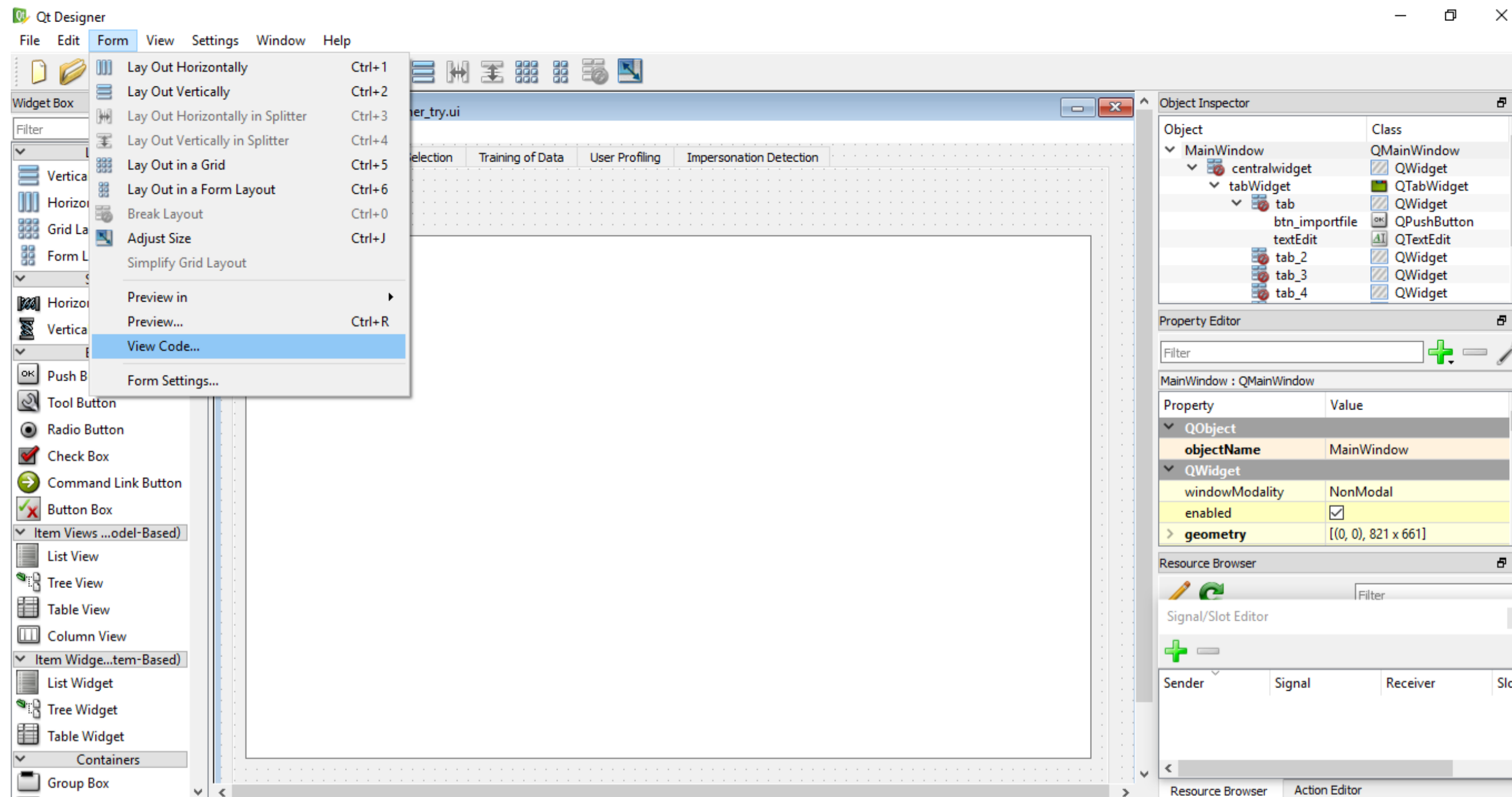
- PyQt5 to zestaw rozwiązań Pythona dla frameworka aplikacji Qt v5.
- Biblioteka Qt jest jedną z najpotężniejszych bibliotek GUI.
- Obecnie rozwijany przez *Riverbank Computing*.
- Zaimplementowany jako zestaw modułów Pythona.
- Posiada ponad 620 klas oraz 6000 funkcji i metod.
- Jest to wieloplatformowy zestaw narzędzi (Unix, Windows i Mac OS).
- Jest podwójnie licencjonowany (GPL oraz licencja komercyjna).

PyQt5 - lista często używanych modułów

- Qt
- QtCore
- QtWidgets
- QtGui
- QtNetwork
- QtMultimedia
- QtSql

QtDesigner

Aplikacja graficzna do definiowania graficznego interfejsu użytkownika



PyQt5 - instalacja

Dostarczona architektura 32-bitowa lub 64-bitowa jest zgodna z Pythonem w wersji 3.5 lub nowszej.

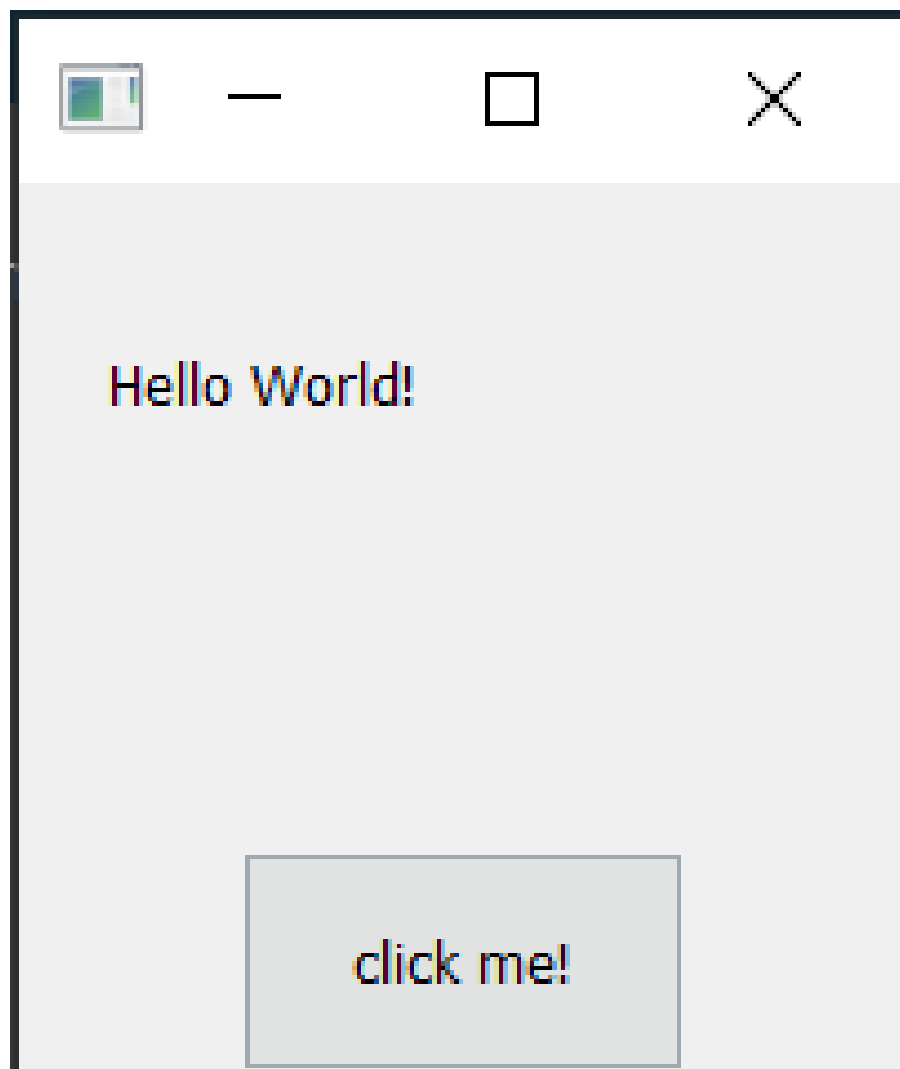
- Zalecanym sposobem instalacji jest użycie narzędzia PIP:

```
pip3 install PyQt5
```

- Aby zainstalować narzędzia programistyczne, takie jak QtDesigner, należy wykonać polecenie:

```
pip3 install pyqt5-tools
```

PyQt5 - hello_world



```
hello_world.py x
1  import sys
2      from PyQt5.QtCore import *
3      from PyQt5.QtGui import *
4      from PyQt5.QtWidgets import *
5
6
7  class window(QWidget):
8      def __init__(self):
9          super().__init__()
10         self.setGeometry(200, 200, 200, 200)
11         self.setWindowTitle("PyQt5")
12
13         self.button = QPushButton(self)
14         self.button.setText("click me!")
15         self.button.setGeometry(50, 150, 100, 50)
16
17         self.label = QLabel(self)
18         self.label.setGeometry(20, 20, 160, 50)
19
20         self.button.clicked.connect(lambda: self.label.setText("Hello World!"))
21
22
23  if __name__ == '__main__':
24      app = QApplication(sys.argv)
25      ex = window()
26      ex.show()
27      sys.exit(app.exec_())
28
```

PyQt5 - pomocne linki

Link do HelloWorld w PyQt5

https://www.tutorialspoint.com/pyqt5/pyqt5_hello_world.htm

Link do wszystkich Qt klas i komponentów

<https://doc.qt.io/qt-5/classes.html>

bardzo przydatny!

Link do dokumentacji

<https://www.riverbankcomputing.com/static/Docs/PyQt5/>

Yfinance

yfinance

II

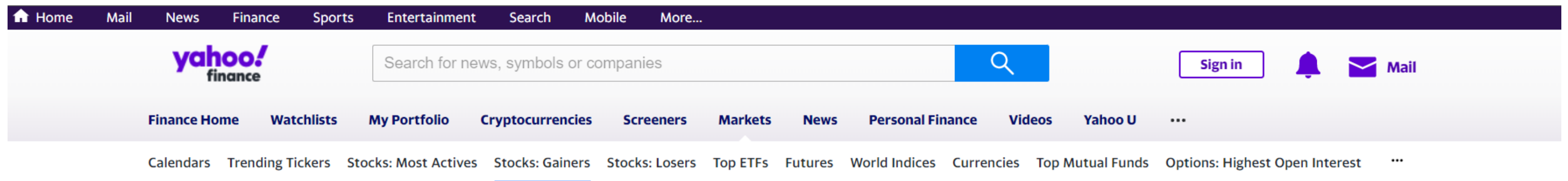


+

yahoo!
finance

Yahoo Finance

- Oferuje doskonały zakres danych rynkowych dotyczących **akcji**, **obligacji**, **walut** i **kryptowalut**.
- Oferuje również **wiadomości rynkowe**, **raporty** i **analizy**, a także różne opcje i fundamentalne dane.



yfinance

- pakiet Pythona,
- Umożliwia pobieranie historycznych danych rynkowych z Yahoo Finance API,
- Yfinance jest całkowicie otwarte i bezpłatne,
- Wysoka ziarnistość danych (dane 1min/2min/5min),
- Zwraca dane bezpośrednio w pandas dataframes/series.

yfinance - układ biblioteki

Yfinance zawiera tylko trzy moduły:

- `yf.Ticker(s)`
- `yf.download`
- `yf.pandas_datareader`

Moduł **download** służy do szybkiego pobierania danych historycznych wielu tickerów jednocześnie.

A **pandas_datareader** jest dla wstecznej kompatybilności ze starszym kodem.

yfinance - pobieranie danych

Pobierzmy najnowsze dane dzienne z 30 dni dla Google.

```
import yfinance as yf

goog = yf.Ticker('goog')
data = goog.history()
```

	Open	High	Low	Close	Volume	Dividends	Stock Splits
Date							
2022-03-30	2857.399902	2869.610107	2843.360107	2852.889893	1052300	0	0
2022-03-31	2848.969971	2852.889893	2792.379883	2792.989990	1475800	0	0
2022-04-01	2800.199951	2819.000000	2775.939941	2814.000000	1173600	0	0
2022-04-04	2816.489990	2880.875000	2816.489990	2872.850098	953800	0	0
2022-04-05	2867.989990	2871.800049	2818.870117	2821.260010	962800	0	0
2022-04-06	2783.229980	2796.969971	2728.362061	2743.520020	1178700	0	0
2022-04-07	2732.360107	2754.030029	2697.145020	2729.300049	972400	0	0
2022-04-08	2725.000000	2725.000000	2675.050049	2680.209961	821000	0	0
2022-04-11	2658.000000	2658.783936	2592.350098	2595.929932	1209400	0	0
2022-04-12	2648.469971	2648.469971	2551.520020	2567.489990	1150200	0	0
2022-04-13	2572.530029	2613.114990	2568.771973	2605.719971	977100	0	0
2022-04-14	2612.989990	2614.205078	2542.229980	2545.060059	1171400	0	0
2022-04-18	2548.199951	2574.239990	2531.569092	2559.219971	745900	0	0
2022-04-19	2561.540039	2618.074951	2549.030029	2610.620117	1136000	0	0
2022-04-20	2625.679932	2638.469971	2557.881104	2564.909912	1130500	0	0
2022-04-21	2587.000000	2606.149902	2493.000000	2498.750000	1507900	0	0
2022-04-22	2500.000000	2509.040039	2382.810059	2392.280029	2317600	0	0
2022-04-25	2388.590088	2465.560059	2375.385010	2465.000000	1726100	0	0
2022-04-26	2455.000000	2455.000000	2383.237061	2390.120117	2469700	0	0
2022-04-27	2287.459961	2350.000000	2262.485107	2300.409912	3111900	0	0
2022-04-28	2342.300049	2408.770020	2302.877930	2388.229980	1839500	0	0
2022-04-29	2351.560059	2379.199951	2293.879883	2299.330078	1683500	0	0

yfinance - pobieranie danych

Teraz pobierzmy dane minutowe z ostatniego tygodnia; tylko tym razem użyjemy daty rozpoczęcia i zakończenia zamiast okresu.

- Okres musi przypadać w ciągu ostatnich 30 dni
- Na każde żądanie dozwolonych jest tylko siedem dni z dokładnością do 1 minuty

```
goog = yf.Ticker('goog')
data = goog.history(interval='1m', start='2022-04-23', end='2022-04-30')
```

		Open	High	Low	Close	Volume	Dividends	Stock Splits
Datetime								
2022-04-25	09:30:00-04:00	2388.590088	2388.590088	2375.385010	2381.459961	66760	0	0
2022-04-25	09:31:00-04:00	2382.530029	2392.280029	2380.995117	2387.524170	9926	0	0
2022-04-25	09:32:00-04:00	2387.080078	2397.449951	2385.739990	2392.270020	10321	0	0
2022-04-25	09:33:00-04:00	2390.965088	2395.370117	2389.000000	2392.030029	15981	0	0
2022-04-25	09:34:00-04:00	2394.205078	2396.469971	2392.159912	2396.469971	5293	0	0
...	

yfinance - pobieranie wielu tickerów

Pobierzmy najnowsze dane miesięczne dla Google i Facebooka (META).

```
data = yf.download(['GOOG', 'META'], period='1mo')
```

Poprawne wartości parametru **period**:

„1d”, „5d”, „1mo”, „3mo”, „6mo”, „1y”, „2y”, „5y”, „10y”, „ytd”, „max”.

	Adj Close		Close		High		Low		Open		Volume	
	GOOG	META	GOOG	META	GOOG	META	GOOG	META	GOOG	META	GOOG	META
Date												
2022-03-30	2852.889893	12.26	2852.889893	12.26	2869.610107	12.6100	2843.360107	12.1900	2857.399902	12.48	1052300	792705.0
2022-03-31	2792.989990	11.96	2792.989990	11.96	2852.889893	12.2700	2792.379883	11.9400	2848.969971	12.25	1475800	326389.0
2022-04-01	2814.000000	11.95	2814.000000	11.95	2819.000000	12.0600	2775.939941	11.8100	2800.199951	12.03	1173600	290255.0
2022-04-04	2872.850098	12.44	2872.850098	12.44	2880.875000	12.4700	2816.489990	12.0500	2816.489990	12.07	953800	382919.0
2022-04-05	2821.260010	12.05	2821.260010	12.05	2871.800049	12.4200	2818.870117	12.0050	2867.989990	12.40	962800	521028.0
2022-04-06	2743.520020	11.57	2743.520020	11.57	2796.969971	11.8200	2728.362061	11.4200	2783.229980	11.80	1178700	515777.0
2022-04-07	2729.300049	11.47	2729.300049	11.47	2754.030029	11.7000	2697.145020	11.2100	2732.360107	11.56	972400	356492.0
2022-04-08	2680.209961	11.27	2680.209961	11.27	2725.000000	11.4600	2675.050049	11.2300	2725.000000	11.46	821000	536573.0
2022-04-11	2595.929932	11.04	2595.929932	11.04	2658.783936	11.1950	2592.350098	10.9200	2658.000000	11.04	1209400	597443.0
2022-04-12	2567.489990	10.92	2567.489990	10.92	2648.469971	11.3700	2551.520020	10.8800	2648.469971	11.18	1150200	455259.0
2022-04-13	2605.719971	11.21	2605.719971	11.21	2613.114990	11.2500	2568.771973	10.8900	2572.530029	11.01	977100	362959.0
2022-04-14	2545.060059	10.85	2545.060059	10.85	2614.205078	11.2300	2542.229980	10.8400	2612.989990	11.20	1171400	401526.0
2022-04-18	2559.219971	10.76	2559.219971	10.76	2574.239990	10.8469	2531.569092	10.6000	2548.199951	10.73	745900	366416.0
2022-04-19	2610.620117	11.02	2610.620117	11.02	2618.074951	11.0650	2549.030029	10.6700	2561.540039	10.77	1136000	277844.0
2022-04-20	2564.909912	10.58	2564.909912	10.58	2638.469971	11.0110	2557.881104	10.5600	2625.679932	11.01	1130500	504993.0

yfinance - pobieranie wielu tickerów

Jeszcze jeden przykład z użyciem listy i parametru '*group_by*'

```
stocks = ['AMZN', 'AAPL', 'GOOG']
data = yf.download(stocks, start="2017-01-01",
                   end="2017-04-30", group_by='tickers')
```

	GOOG						AMZN				AAPL						
	Open	High	Low	Close	Adj Close	Volume	Open	High	...	Adj Close	Volume	Open	High	Low	Close	Adj Close	Volume
Date									...								
2017-01-03	778.809998	789.630005	775.799988	786.140015	786.140015	1657300	757.919983	758.760010	...	753.669983	3521100	28.950001	29.082500	28.690001	29.037500	27.297693	115127600
2017-01-04	788.359985	791.340027	783.159973	786.900024	786.900024	1073000	758.390015	759.679993	...	757.179993	2510500	28.962500	29.127501	28.937500	29.004999	27.267139	84472400
2017-01-05	786.080017	794.479980	785.020020	794.020020	794.020020	1335200	761.549988	782.400024	...	780.450012	5830100	28.980000	29.215000	28.952499	29.152500	27.405804	88774400
2017-01-06	795.260010	807.900024	792.203979	806.150024	806.150024	1640200	782.359985	799.440002	...	795.989990	5986200	29.195000	29.540001	29.117500	29.477501	27.711332	127007600
2017-01-09	806.400024	809.966003	802.830017	806.650024	806.650024	1274600	798.000000	801.770020	...	796.919983	3446100	29.487499	29.857500	29.485001	29.747499	27.965149	134247600
...
2017-04-24	851.200012	863.450012	849.859985	862.760010	862.760010	1372500	908.679993	909.989990	...	907.409973	3122900	35.875000	35.987499	35.794998	35.910000	33.904789	68537200
2017-04-25	865.000000	875.000000	862.809998	872.299988	872.299988	1672000	907.039978	909.479980	...	907.619995	3380600	35.977501	36.224998	35.967499	36.132500	34.114868	75486000
2017-04-26	874.229980	876.049988	867.747986	871.729980	871.729980	1237200	910.299988	915.750000	...	909.289978	2608900	36.117500	36.150002	35.845001	35.919998	33.914227	80164800
2017-04-27	873.599976	875.400024	870.380005	874.250000	874.250000	2026800	914.390015	921.859985	...	918.380005	5305500	35.980000	36.040001	35.827499	35.947498	33.940178	56985200
2017-04-28	910.659973	916.849976	905.770020	905.960022	905.960022	3276300	948.830017	949.590027	...	924.989990	7364700	36.022499	36.075001	35.817501	35.912498	33.907139	83441600


```
>>> import yfinance as yf
>>> obj = yf.Ticker('goog')
>>> obj.
```

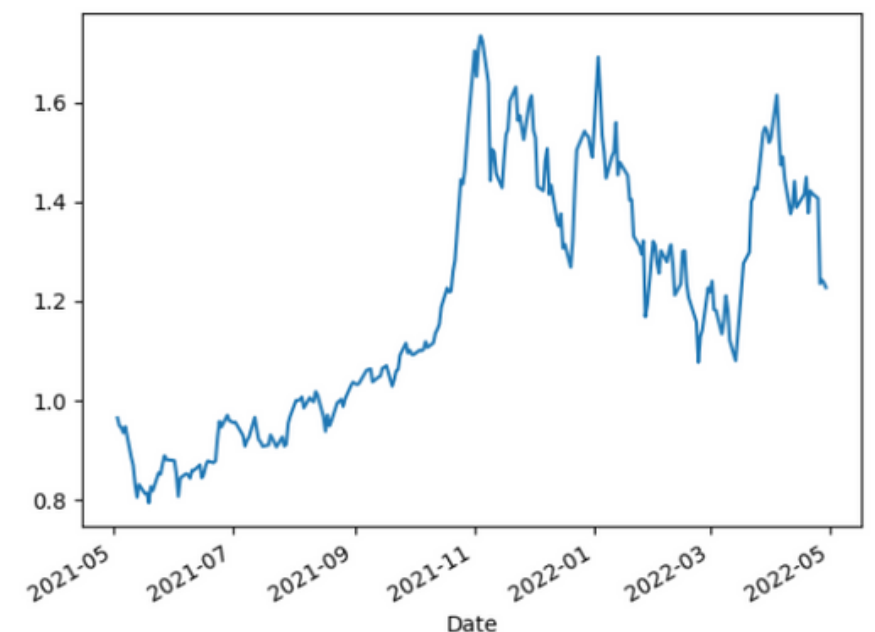
```
actions
balance_sheet
calendar
dividends
financials
get_analysis
get_balancesheet
get_cashflow
get_earnings
get_info
get_isin
get_mutualfund_holders
get_recommendations
get_splits
history
institutional_holders
major_holders
news
options
quarterly_balancesheet
quarterly_earnings
recommendations
shares
stats
analysis
balancesheet
cashflow
earnings
get_actions
get_balance_sheet
get_calendar
get_dividends
get_financials
get_institutional_holders
get_major_holders
get_news
get_shares
get_sustainability
info
isin
mutualfund_holders
option_chain
quarterly_balance_sheet
quarterly_cashflow
quarterly_financials
session
splits
sustainability
```


yfinance - pct_change, cumprod

- Funkcja `pct_change()` oblicza procentową zmianę między bieżącym elementem a poprzednim.
- `cumprod()` oblicza skumulowane zwroty

```
import yfinance as yf
import matplotlib.pyplot as plt
import numpy as np

data = yf.Ticker('TSLA')
price = data.history(period='1y')
x = price['Close'].pct_change()
returns = (x + 1).cumprod()
returns.plot()
plt.show()
```



yfinance - describe

Statystyki opisowe obejmują:

- średnią,
- odchylenie standardowe,
- wartość minimalną
- wartość maksymalną,
- percentyle 25/50/75%.

```
data = yf.download('AAPL MSFT TSLA', start='2021-01-01')
x = data['Close'].pct_change()
print(x.describe())
```

	AAPL	MSFT	TSLA
count	334.000000	334.000000	334.000000
mean	0.000658	0.000788	0.001302
std	0.016855	0.015855	0.036899
min	-0.041674	-0.042323	-0.121841
25%	-0.009244	-0.007584	-0.020435
50%	0.000309	0.000200	0.001573
75%	0.012270	0.010923	0.020186
max	0.069778	0.051094	0.196412

yfinance - corr

```
data = yf.download('AAPL MSFT TSLA', start='2021-01-01')
x = data['Close'].pct_change()
corr = x.corr()
print(corr)
```

	AAPL	MSFT	TSLA
AAPL	1.000000	0.715340	0.519194
MSFT	0.715340	1.000000	0.466133
TSLA	0.519194	0.466133	1.000000

- W finansach korelacja to statystyka mierząca stopień, w jakim dwa papiery wartościowe poruszają się względem siebie.
- Funkcja `corr()` daje w wyniku macierz, która zawiera wartości dla każdej pary giełdowej.
- Wartości mieszczą się w zakresie od -1 do 1.

yfinance - weights

```
stocks = ['GOOG', 'TSLA', 'MSFT', 'LMT']  
weights = [0.7035, 0.185, 0.0858, 0.0257]  
  
data = yf.download(stocks, start='2021-01-01')  
x = data['Close'].pct_change()  
  
ret = (x * weights).sum(axis=1)  
cumulative = (ret + 1).cumprod()
```

yfinance - volatility

```
print(np.std(ret))  
annual_std = np.std(ret) * np.sqrt(252)
```

- Wykorzystywana do pomiaru ryzyka, niestabilności cen akcji.
- Zmienność jest obliczana przy użyciu odchylenia standardowego zwrotu portfela.
- Możemy również obliczyć roczną zmienność, wyciągając pierwiastek kwadratowy z liczby dni handlowych w roku (252) i pomnożyć ją przez dzienną zmienność.

yfinance - sharpe ratio

```
sharpe = (np.mean(ret)/np.std(ret)) * np.sqrt(252)
print('Sharpe: %f' % sharpe)
```

- Wskaźnik Sharpe'a jest miarą zwrotu z portfela skorygowanego o ryzyko.
- Portfel o wyższym współczynniku Sharpe'a jest uważany za lepszy.
- Aby obliczyć współczynnik Sharpe'a, musimy wziąć średni zwrot i podzielić go przez zmienność.
- Wskaźniki Sharpe'a większe niż 1 są uważane za optymalne.

yfinance - instalacja



Search projects



[Help](#)

[Sponsors](#)

[Log in](#)

[Register](#)

yfinance 0.1.70

`pip install yfinance`



[Latest version](#)

Released: Jan 30, 2022

Download market data from Yahoo! Finance API

yfinance – pomocne linki

Link for official site

<https://pypi.org/project/yfinance/>

Link for Quick Tutorial

<https://analyzingalpha.com/yfinance-python>

Link for complete Guide

<https://algotrading101.com/learn/yfinance-guide/>

MySQL

MySQL

- System baz danych używany do tworzenia aplikacji internetowych.
- Używany zarówno w małych, jak i dużych aplikacjach.
- System zarządzania relacyjnymi bazami danych (RDBMS).
- Szybki, niezawodny, elastyczny i łatwy w użyciu.
- Obsługuje standardowy język SQL (Structured Query Language).
- Jest darmowy.
- Został opracowany przez Michaela Wideniusa i Davida Axmarka w 1994 roku.
- Jest obecnie rozwijany, dystrybuowany i wspierany przez Oracle Corporation.
- Napisany w C, C++.

MySQL – główne cechy

- Konstrukcja serwera jest wielowarstwowa z niezależnymi modułami.
- Jest w pełni wielowątkowy dzięki wykorzystaniu wątków jądra.
- Zapewnia transakcyjne i nietransakcyjne silniki pamięci masowej.
- Posiada szybki, oparty na wątkach system alokacji pamięci.
- Obsługuje tablicę sterty w pamięci.
- Obsługuje duże bazy danych.
- Server działa w systemach klient/serwer lub systemach wbudowanych.
- Działa na wielu różnych platformach.

MySQL - kto używa?

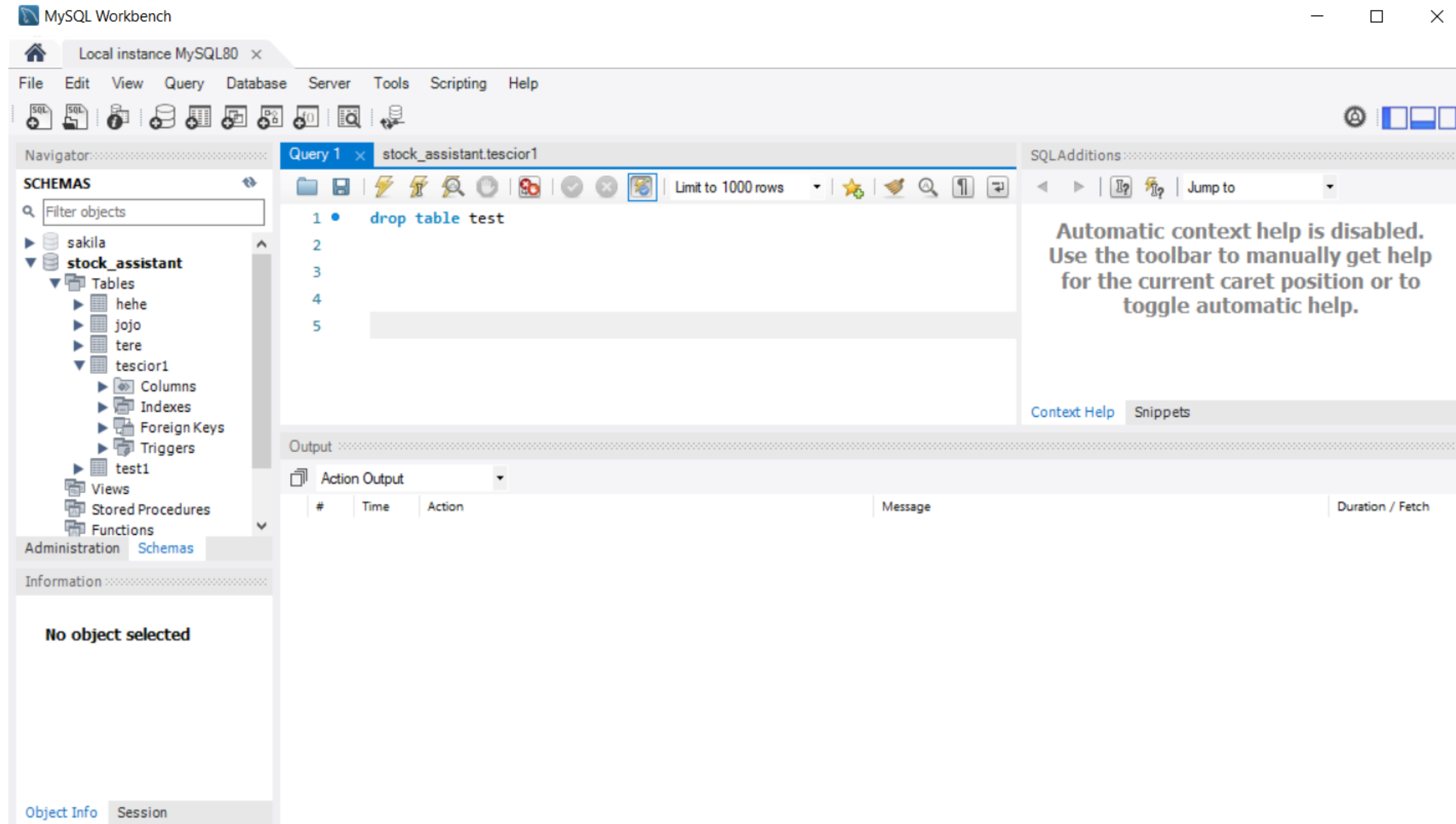


WIKIPEDIA

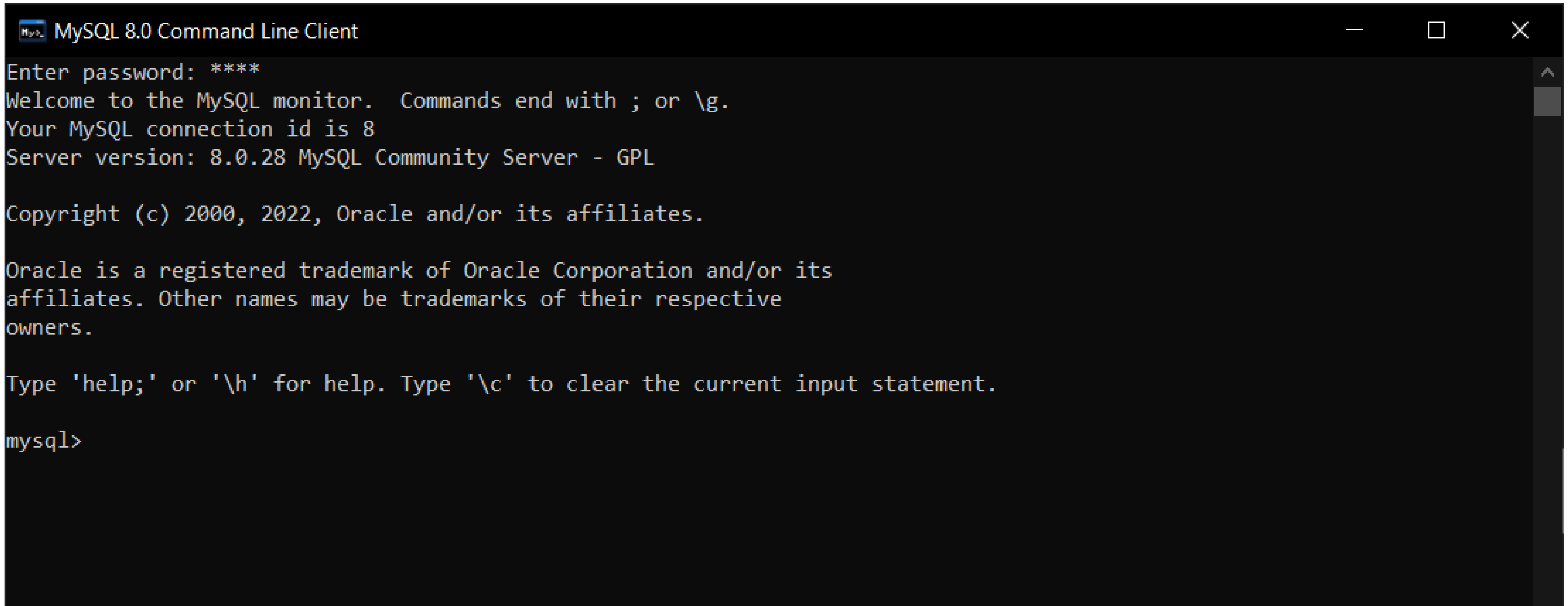


Joomla!®

MySQL Workbench



MySQL Command Line Client

A screenshot of the MySQL 8.0 Command Line Client window. The window has a dark background and a title bar that reads "MySQL 8.0 Command Line Client". The text inside the window is as follows:

```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.28 MySQL Community Server - GPL

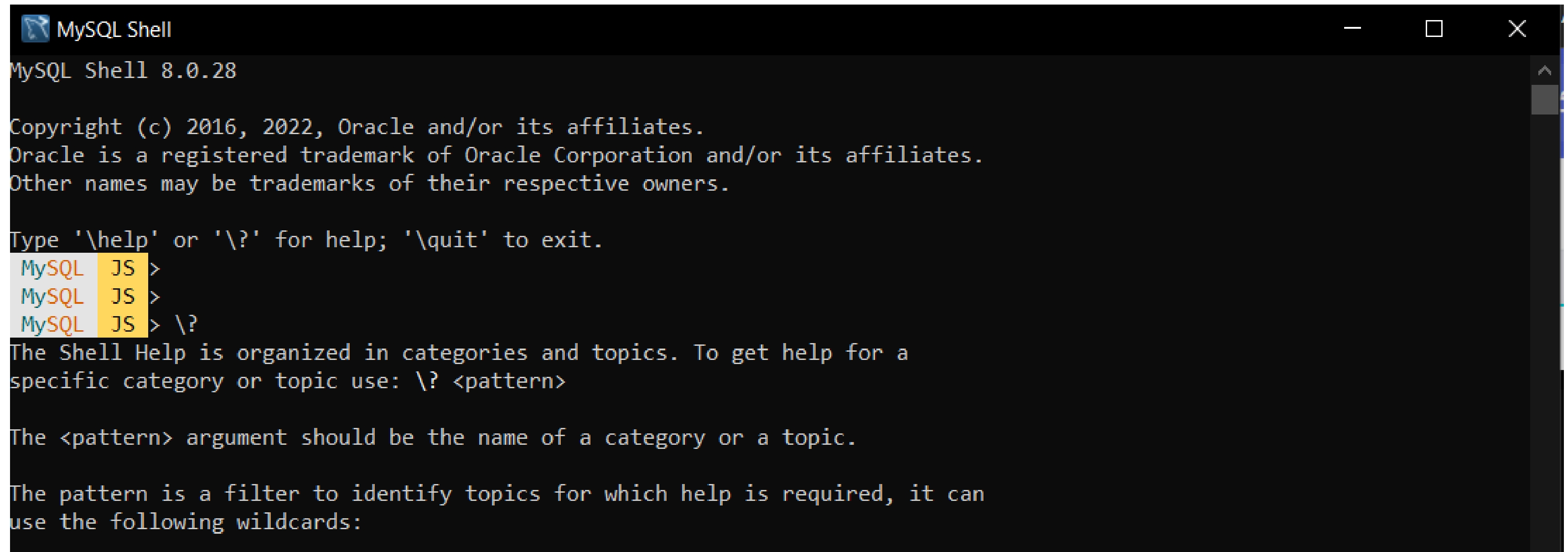
Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

MySQL Shell



```
MySQL Shell 8.0.28

Copyright (c) 2016, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS >
MySQL JS >
MySQL JS > \?
The Shell Help is organized in categories and topics. To get help for a
specific category or topic use: \? <pattern>

The <pattern> argument should be the name of a category or a topic.

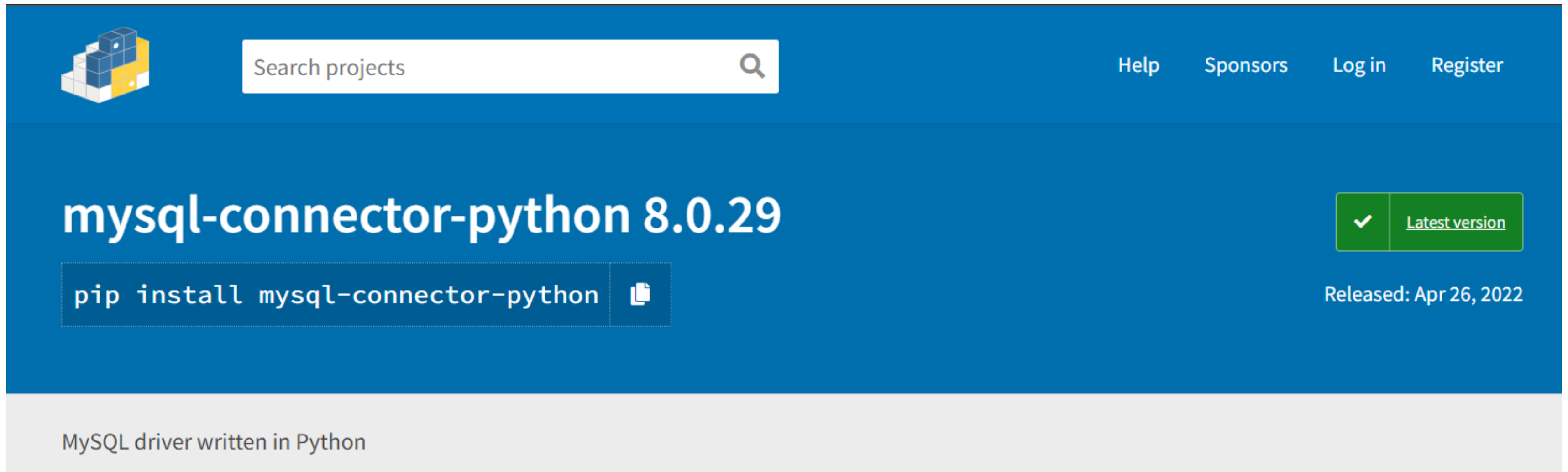
The pattern is a filter to identify topics for which help is required, it can
use the following wildcards:
```

MySQL w pythonie


Python potrzebuje sterownika MySQL, aby uzyskać dostęp do bazy danych MySQL.

W naszym projekcie wykorzystaliśmy *MySQL Connector*.

Instalacja sterownika odbywa się poprzez wykonanie poniższego polecenia w terminalu.





The screenshot shows the PyPI project page for `mysql-connector-python` version `8.0.29`. The page has a blue header with the project logo, a search bar, and links for Help, Sponsors, Log in, and Register. The main content area is also blue and features the package name and version in large white text. Below this, there is a command box containing `pip install mysql-connector-python` and a copy icon. To the right, a green badge indicates it is the 'Latest version' with a checkmark. Below the badge, it says 'Released: Apr 26, 2022'. At the bottom of the page, on a light gray background, it says 'MySQL driver written in Python'.



HelpSponsorsLog inRegister

mysql-connector-python 8.0.29

`pip install mysql-connector-python`

 [Latest version](#)

Released: Apr 26, 2022

MySQL driver written in Python

MySQL - tworzenie bazy danych

```
mysql> create database test;  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> show databases;
```

Database
information_schema
mysql
performance_schema
sakila
stock_assistant
sys
test
world

```
8 rows in set (0.00 sec)
```

```
mysql> drop database test;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> show databases;
```

Database
information_schema
mysql
performance_schema
sakila
stock_assistant
sys
world

```
7 rows in set (0.00 sec)
```

MySQL - połączenie z bazą

```
import mysql.connector
import logging
import sys

class DatabaseConnector:

    # establishing connection to database
    try:
        database = mysql.connector.connect(host='127.0.0.1', user='root', password='root', database='stock_assistant',
                                           auth_plugin='mysql_native_password')
    except mysql.connector.Error as e:
        logging.critical('Connection to database has not been established: ' + str(e))
        sys.exit()
```


MySQL - CREATE

CREATE TABLE IF NOT EXISTS *nazwa_tabeli* (*nazwa_kolumny_1* typ *inne_parametry*,
nazwa_kolumny_2 typ *inne_parametry*, ...)

```
@staticmethod
def create_table(name):

    query = 'CREATE TABLE IF NOT EXISTS %s (id INT AUTO_INCREMENT PRIMARY KEY, stock VARCHAR(250) NOT NULL, ' \
           ' amount FLOAT NOT NULL, value FLOAT NOT NULL, date VARCHAR(250) NOT NULL )' % name

    try:
        cursor = DatabaseConnector.database.cursor()
        cursor.execute(query)
    except (mysql.connector.Error, AttributeError) as e:
        logging.error('Query has not been executed: ' + str(e))
```

MySQL - INSERT

INSERT INTO *nazwa_tabeli* VALUES (*wartość_1*, *wartość_2*, ...)

```
@staticmethod
def insert_into(name, stock, amount, value, date):

    query = 'INSERT INTO %s (stock, amount, value, date) VALUES (%s, %s, %s, %s)' % (name, stock, amount, value, date)

    try:
        cursor = DatabaseConnector.database.cursor()
        cursor.execute(query)
        DatabaseConnector.database.commit()
    except (mysql.connector.Error, AttributeError) as e:
        logging.error('Query has not been executed: ' + str(e))
```

MySQL – SELECT

`SELECT wartość_1, wartość_2, ... FROM nazwa_tabeli`

`SELECT * FROM nazwa_tabeli`

```
@staticmethod
def select_from(name):

    query = 'SELECT stock, amount, value, date FROM %s' % name

    try:
        cursor = DatabaseConnector.database.cursor()
        cursor.execute(query)
        data = []
        for element in cursor:
            # tuple unpacking
            (stock, amount, value, date) = element
            line = [stock, amount, value, date]
            data.append(line)
        return data
    except (mysql.connector.Error, AttributeError) as e:
        logging.error('Query has not been executed: ' + str(e))
```

MySQL - DELETE

DELETE FROM *nazwa_tabeli* WHERE *wartość_kolumny* = *x*

```
@staticmethod
def delete_from(name, stock, date):

    query = 'DELETE FROM %s WHERE stock=%s and date=%s' % (name, stock, date)

    try:
        cursor = DatabaseConnector.database.cursor()
        cursor.execute(query)
        DatabaseConnector.database.commit()
    except (mysql.connector.Error, AttributeError) as e:
        logging.error('Query has not been executed: ' + str(e))
```

MySQL - DROP

DROP TABLE *nazwa_tabeli*

```
@staticmethod
def drop_table(name):

    query = 'DROP TABLE %s' % name

    try:
        cursor = DatabaseConnector.database.cursor()
        cursor.execute(query)
        DatabaseConnector.database.commit()
    except (mysql.connector.Error, AttributeError) as e:
        logging.error('Query has not been executed: ' + str(e))
```

MySQL - SHOW

SHOW TABLES

```
@staticmethod
def show_tables():

    query = 'SHOW TABLES'

    try:
        cursor = DatabaseConnector.database.cursor()
        cursor.execute(query)
        names = []
        for name in cursor:
            # tuple unpacking
            (n,) = name
            names.append(n)
        return names
    except (mysql.connector.Error, AttributeError) as e:
        logging.error('Query has not been executed: ' + str(e))
```

Demonstracja aplikacji

Stock Assistant

Test

Zadania praktyczne

Dziękujemy

za uwagę