Exness MetaTrader5 trading with Python.

MetaTrader module for integration with Python

Python is a modern high-level programming language for developing scripts and applications. It contains multiple libraries for machine learning, process automation, as well as data analysis and visualization.

MetaTrader package for Python is designed for convenient and fast obtaining of exchange data via interprocessor communication directly from the MetaTrader 5 terminal. The data received this way can be further used for statistical calculations and machine learning.

For full documentation, see https://www.mql5.com/en/docs/integration/python_metatrader5

Use MetaTrader5 Library to easily build trading system in a given trading strategy on Python

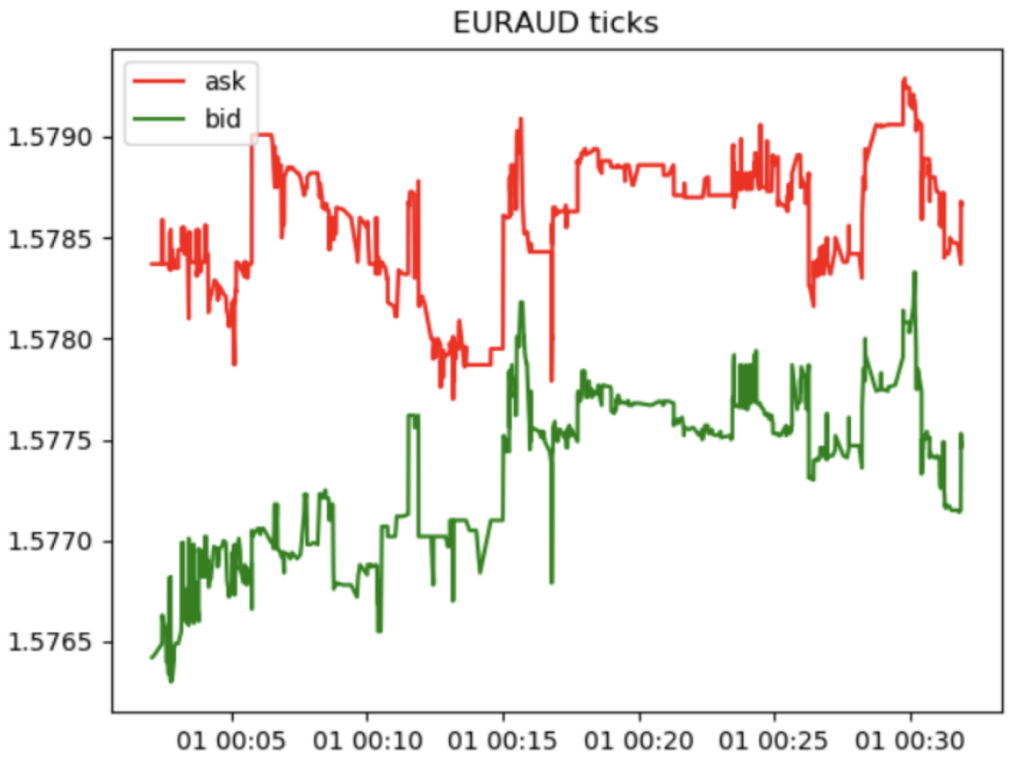
- What is MetaTrader5 Library
- Install Anaconda the package manipulated
- Coding with Jupyter notebook IDE
- Install MetaTrader5 library via pip command
- Import necessary library for build trading strategy
- Set up Account
    - Calling data
    - Cleaning data
    - Set up Strategy from Quantpedia
    - Test Sending order
- Result

MetaTrader 5 Integration with Python and Strategy Tester Improvements

MT5's API which enables requests of MetaTrader 5 terminal data through applications using Python language.

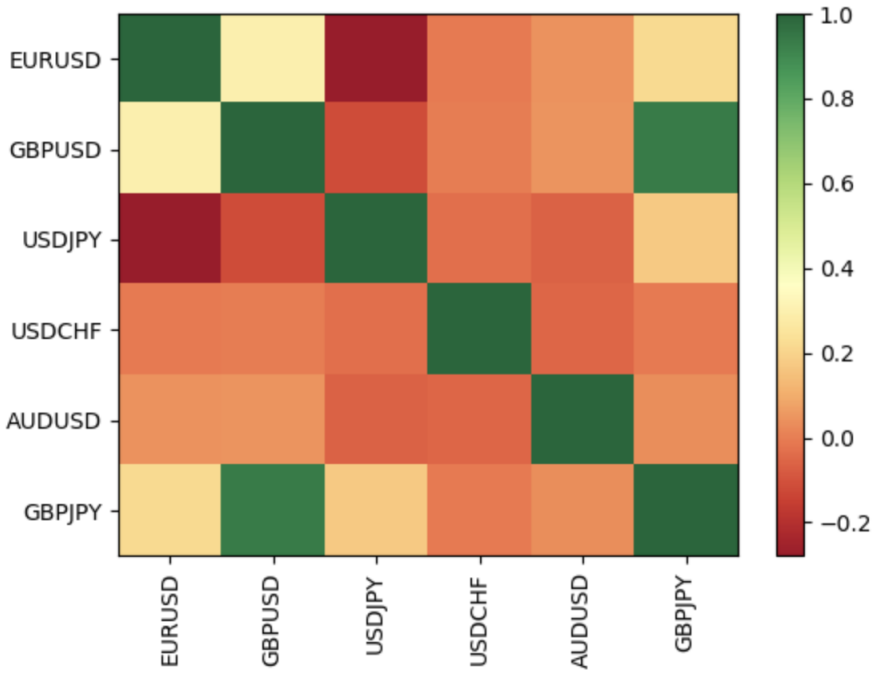
Python is a modern high-level programming language for developing scripts and applications. It contains multiple libraries for machine learning, process automation, as well as data analysis and visualization.

MetaTrader package for Python is designed for efficient and fast obtaining of exchange data via interprocessor communication, directly from the MetaTrader 5 terminal. The data received via this pathway can be further used for statistical calculations and machine learning.



Why integrate MQL5 and Python?

Comprehensive data processing requires extensive tools and is often beyond the sandbox of one single application. Specialized programming languages are used for processing and analyzing data, statistics and machine learning. One of the leading programming languages for data processing is Python. A very effective solution is to use the power of the language and included libraries for the development of trading systems.



FOREX Correlations Heat Map

Functions for integrating MetaTrader 5 and Python

Function	Action
initialize	Establish a connection with the MetaTrader 5 terminal
login	Connect to a trading account using specified parameters
shutdown	Close the previously established connection to the MetaTrader 5 terminal
version	Return the MetaTrader 5 terminal version
last_error	Return data on the last error

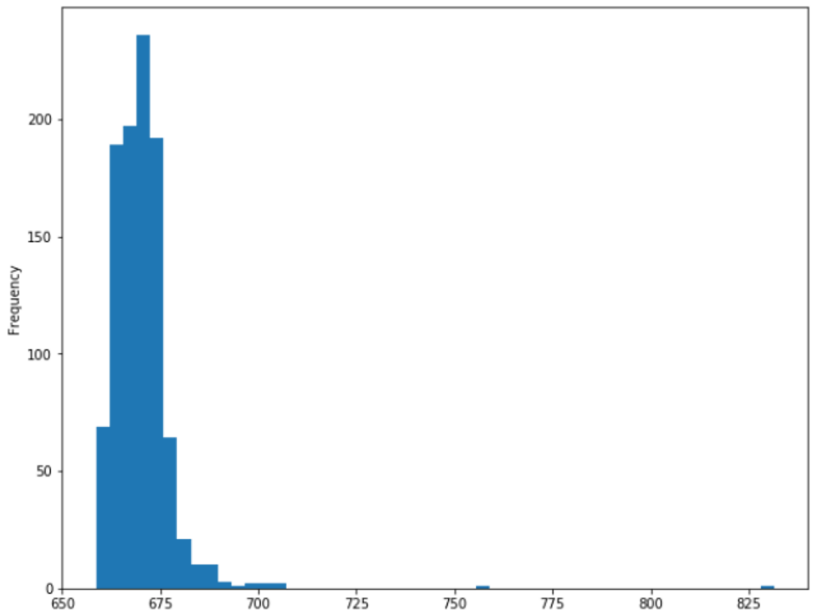
account_info	Get info on the current trading account
terminal_Info	Get status and parameters of the connected MetaTrader 5 terminal
symbols_total	Get the number of all financial instruments in the MetaTrader 5 terminal
symbols_get	Get all financial instruments from the MetaTrader 5 terminal
symbol_info	Get data on the specified financial instrument
symbol_info_tick	Get the last tick for the specified financial instrument
symbol_select	Get the last tick for the specified financial instrument
copy_rates_from	Get bars from the MetaTrader 5 terminal starting from the specified date
copy_rates_from_pos	Get bars from the MetaTrader 5 terminal starting from the specified index
copyrates_range	Get bars in the specified date range from the MetaTrader 5 terminal
copy_ticks_from	Get ticks from the MetaTrader 5 terminal starting from the specified date
copy_ticks_range	Get ticks for the specified date range from the MetaTrader 5 terminal
orders_total	Get the number of active orders.
orders_get	Get active orders with the ability to filter by symbol or ticket
order_calc_margin	Return margin in the account currency to perform a specified trading operation
order_calc_profit	Return profit in the account currency for a specified trading operation
order_check	Check funds sufficiency for performing a required trading operation
order_send	Send a request to perform a trading operation.
positions_total	Get the number of open positions
positions_get	Get open positions with the ability to filter by symbol or ticket

history_orders_total	Get the number of orders in trading history within the specified interval
history_orders_get	Get orders from trading history with the ability to filter by ticket or position
history_deals_total	Get the number of deals in trading history within the specified interval
history_deals_get	Get deals from trading history with the ability to filter by ticket or position

Low Latency for connection.

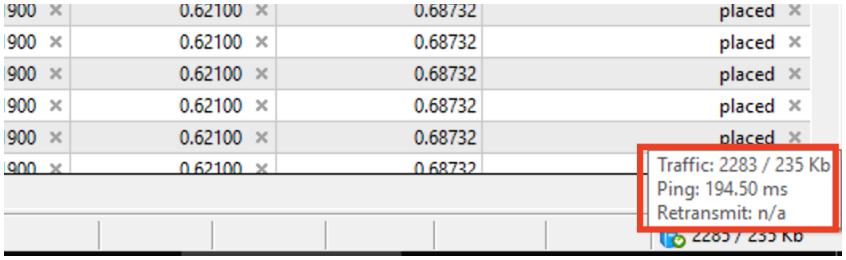
The latency that is calculated from subtract between time_request and time_placed.

	order_ticket	time_placed	time_request	price_diviate	latency	latency_2
count	1.000000e+03	1.000000e+03	1.000000e+03	1000.0	1.000000e+03	1000.000000
mean	3.564219e+07	1.592460e+12	1.592311e+12	0.0	1.486807e+08	670.120014
std	4.022692e+02	1.124684e+05	1.124665e+05	0.0	8.391302e+00	8.391302
min	3.564146e+07	1.592459e+12	1.592311e+12	0.0	1.486807e+08	658.595703
25%	3.564186e+07	1.592459e+12	1.592311e+12	0.0	1.486807e+08	665.415039
50%	3.564219e+07	1.592460e+12	1.592311e+12	0.0	1.486807e+08	669.564453
75%	3.564253e+07	1.592460e+12	1.592311e+12	0.0	1.486807e+08	673.211182
max	3.564290e+07	1.592460e+12	1.592311e+12	0.0	1.486808e+08	831.457275



Distribution of Latency (1000 order request)

- The latency between MT5 Terminal and MT5 Broker Server is around 194msc Which is two step (from terminal to broker then from broker to terminal)
- Half-way latency is 194/2 = 97
- The mean of latency between Python API To MT5 Broker Server is 670ms
- So the latency between Python API and MT5 Terminal is 670-97= **573ms**



Latency for each function.

	initialize	shutdown	version	last_error	account_info
count	100.000000	100.000000	100.000000	100.000000	100.000000
mean	670.155000	0.446000	35.752000	0.360000	21.547000
std	1291.658573	1.329192	14.017453	0.271918	16.653137
min	251.400000	0.300000	16.500000	0.300000	13.900000
25%	312.700000	0.300000	28.800000	0.300000	14.400000
50%	361.250000	0.300000	32.250000	0.300000	15.200000
75%	492.225000	0.300000	40.625000	0.400000	18.750000
max	10514.000000	13.600000	130.000000	3.000000	106.400000

	terminal_info	symbols_total	symbols_get	symbol_info	symbol_info_tick
count	100.000000	100.000000	100.000000	100.000000	100.000000
mean	11389.863000	34.285000	7668.923000	60.143000	32.281000
std	1926.344455	13.260509	5042.022591	42.703931	28.232779
min	9491.400000	13.000000	4666.400000	30.200000	16.400000
25%	10093.650000	26.875000	5603.075000	32.875000	22.000000
50%	10670.850000	30.250000	6219.950000	36.750000	23.250000
75%	12492.800000	39.975000	7020.225000	74.725000	29.350000
max	20112.800000	92.500000	40323.000000	207.600000	232.900000

	symbol_select	copy_rates_from	copy_rates_from_pos	copy_rates_range	copy_ticks_from
count	100.000000	100.000000	100.000000	100.00000	100.000000
mean	32.678000	37.602000	86.129000	106.02800	6365.171000
std	9.279799	9.239649	181.827932	213.69906	1007.050413
min	12.600000	21.300000	34.500000	34.10000	4989.200000
25%	25.400000	31.050000	46.075000	51.55000	5849.500000
50%	30.800000	36.750000	61.050000	62.05000	6177.400000
75%	34.325000	42.175000	71.225000	84.17500	6614.100000
max	58.700000	78.900000	1772.800000	2096.40000	11306.800000

	copy_ticks_range	orders_total	orders_get	order_calc_margin	order_calc_profit
count	100.000000	100.000000	100.000000	100.000000	100.00000
mean	5984.174000	40.166000	74.892000	25.488000	22.10500
std	983.595447	154.308546	379.182762	13.219241	3.69136
min	5080.300000	16.900000	19.100000	20.400000	19.00000
25%	5452.450000	17.875000	21.175000	21.275000	20.07500
50%	5711.550000	20.350000	25.200000	21.850000	20.70000
75%	6260.850000	24.100000	27.425000	25.350000	22.62500
max	12039.000000	1562.100000	3779.400000	142.000000	43.90000

	orders_check	orders_send	positions_total	positions_get	history_orders_total
count	100.000000	1.000000e+02	100.000000	100.000000	100.000000
mean	31.503000	3.231281e+05	20.543000	24.410000	2093.200000
std	9.951483	1.240702e+05	4.859485	33.654863	1434.960135
min	21.700000	2.779933e+05	17.300000	16.800000	1121.000000
25%	26.800000	2.806549e+05	17.600000	19.700000	1214.425000
50%	27.900000	2.892965e+05	18.050000	20.100000	1533.300000
75%	31.325000	3.199420e+05	22.000000	20.300000	2290.375000
max	98.700000	1.257569e+06	55.300000	350.300000	8549.200000

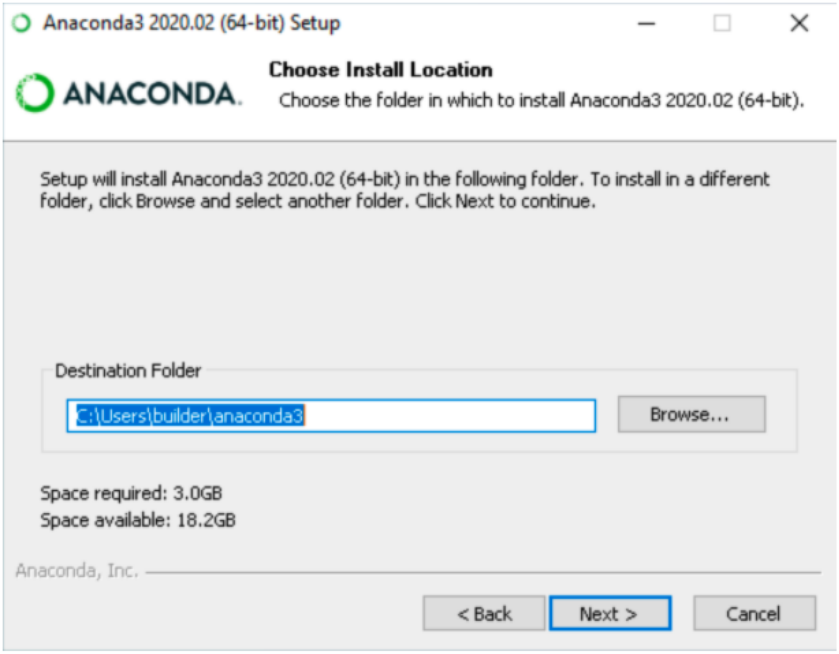
	history_orders_get	history_deals_total	history_deals_get
count	100.00000	100.000000	100.000000
mean	4936.35400	29.966000	56.731000
std	3382.78715	13.216643	198.026501
min	3757.90000	15.300000	16.100000
25%	3904.20000	17.700000	29.225000
50%	4040.80000	26.950000	39.250000
75%	4322.72500	39.325000	43.425000
max	24705.00000	67.300000	2013.200000

Install Anaconda : https://docs.anaconda.com/anaconda/install/windows/

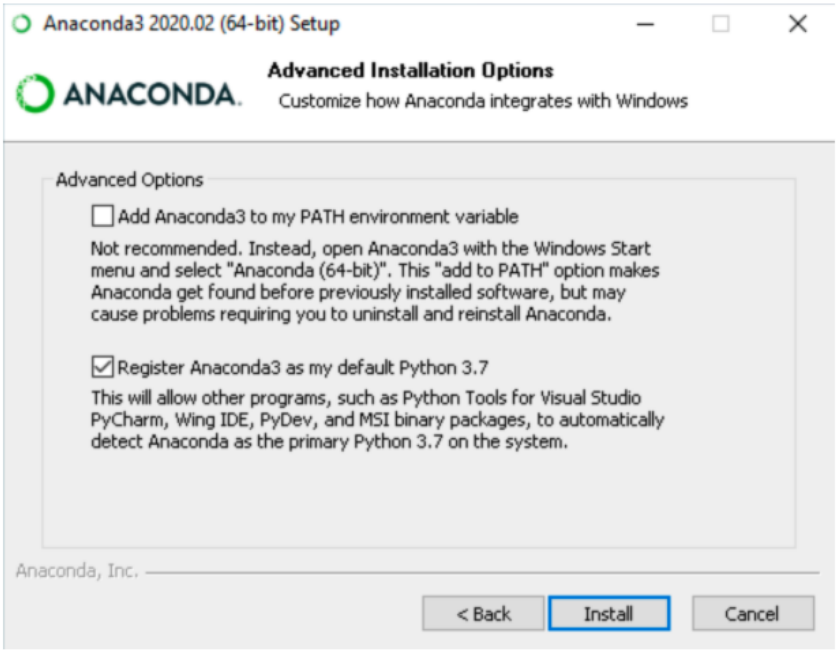
1. Download the Anaconda installer :

   https://www.anaconda.com/products/individual#windows

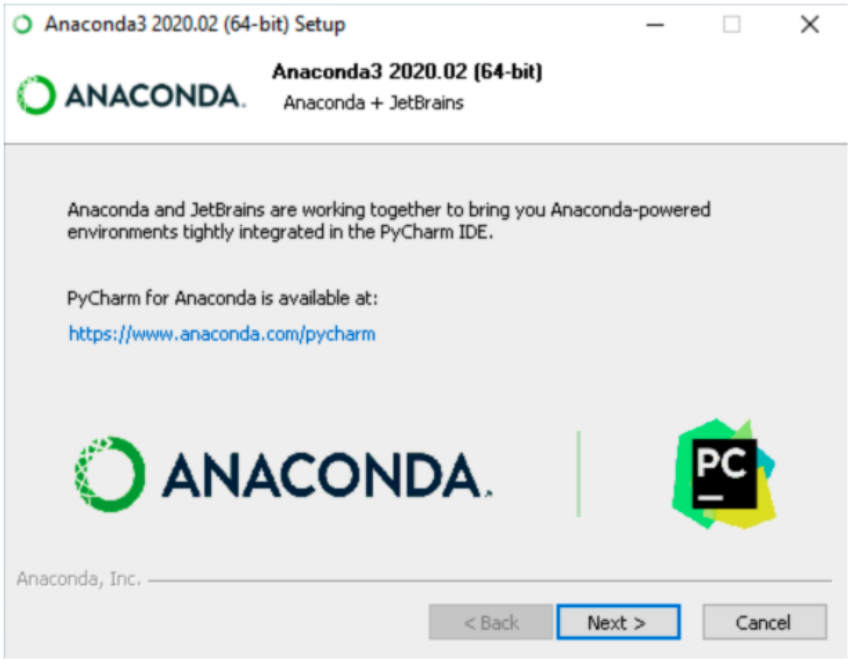
2. Double click the installer to launch.
3. Click Next
4. Read the licensing terms and click “I Agree”.
5. Select an install for “Just me” unless you’re installing for all users and click Next.



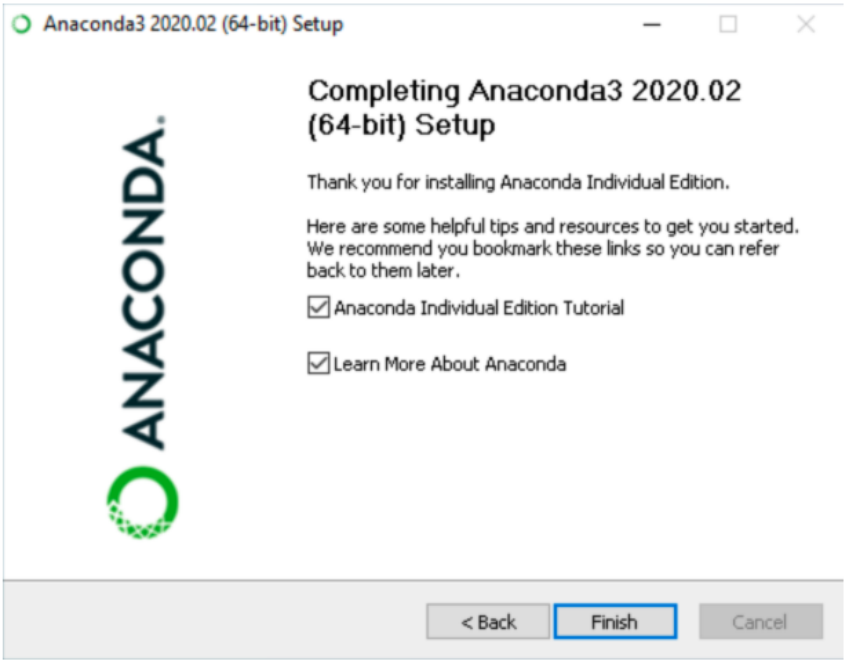
Select a destination folder to install Anaconda and click the Next button.



Choose whether to add Anaconda to your PATH environment variable. We recommend not adding Anaconda to the PATH environment variable.

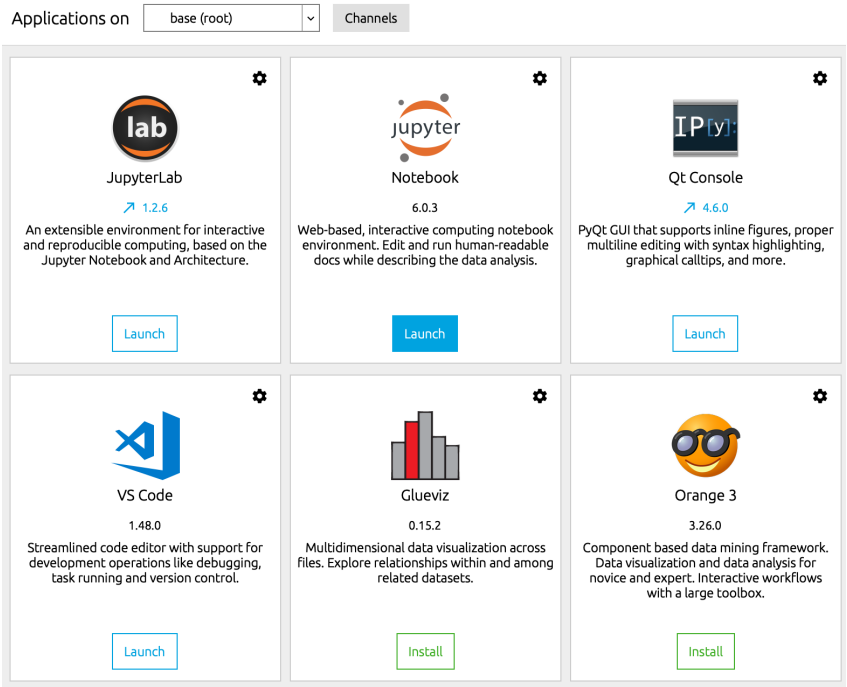


Optional: To install PyCharm for Anaconda, click on the link to https://www.anaconda.com/pycharm.



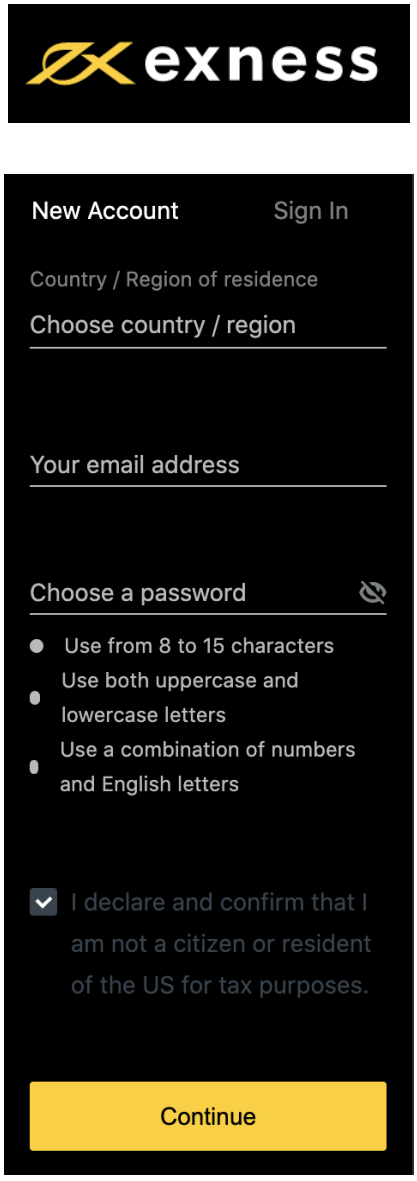
After a successful installation you will see the "Thanks for installing Anaconda" dialog box.

Open Jupyter IDE from Anaconda:



Select Jupyter Notebook

Open MetaTrader5 Account with Exness

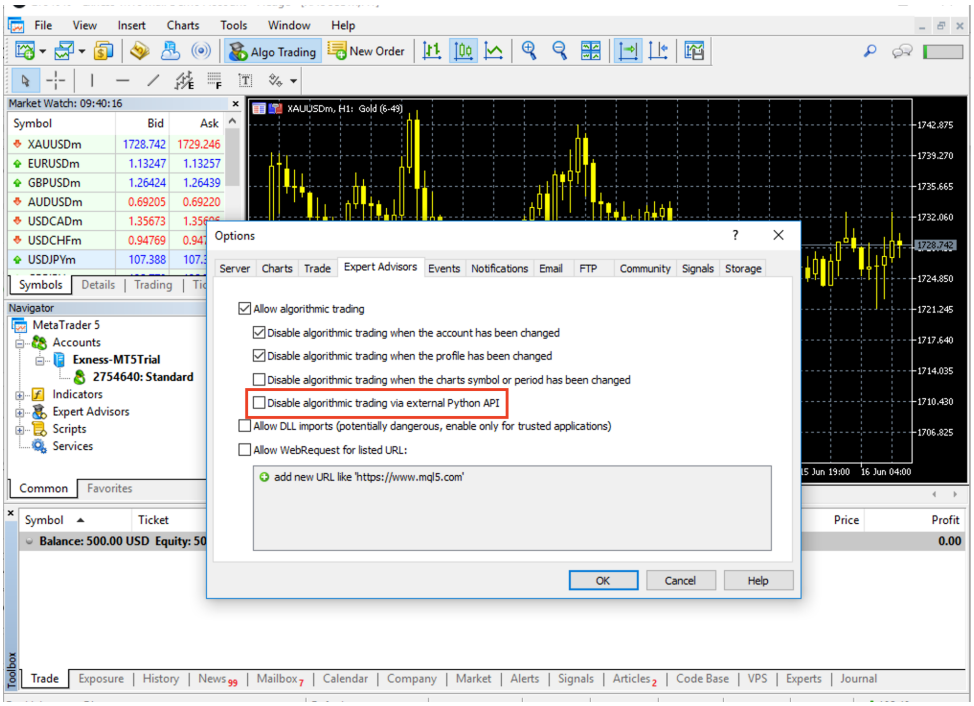


www.exness.com/metatrader_5/

Download MetaTrader 5 from www.exness.com

Download MetaTrader 5

Select MeteTrader5 for your device



Enable Python trading API

Install MetaTrader5 library.

Installing the package from the command line:

`pip install MetaTrader5`

Updating the package from the command line:

`pip install --upgrade MetaTrader5`

Import necessary libraries.

`import pandas  # for data manipulation and analysis.`

`import numpy  # for working with arrays.`

`Import metatrader5  # for working between MQL5 and Python`

`import matplotlib.pyplot as plt  # plotting library for the Python`

Set up Account connection

initialize

Establish a connection with the MetaTrader 5 terminal. There are three call options.

Call without parameters. The terminal for connection is found automatically.

```
initialize(
  path,                // path to the MetaTrader 5 terminal EXE file
  login=LOGIN,         // account number
  password="PASSWORD",     // password
  server="SERVER",       // server name as it is specified in the terminal
  timeout=TIMEOUT,       // timeout
  portable=False         // portable mode
  )
```

Coding:

```
import MetaTrader5 as mt5

# establish MetaTrader 5 connection to a specified trading account

if not mt5.initialize(login=12345678, server="Exness-real5",password="exness1234"):
   print("initialize() failed, error code =",mt5.last_error())
   quit()

# display data on connection status, server name and trading account

print(mt5.terminal_info())
```

login

Connect to a trading account using specified parameters.

```
# connect to the trade account
account=12345678
authorized=mt5.login(account)  # the terminal database password is applied if connection data is set to be remembered
# connect to another trading account specifying the password
account=12345678
authorized=mt5.login(account, password="exness1234")
```

Calling the data

```
# Create a currency watchlist for which correlation matrix is to be plotted.
symbol = ["EURUSDm", "GBPUSDm", "USDJPYm", "USDCHFm", "AUDUSDm", "GBPJPYm"]
```

copy_rates_from_pos

Get bars from the MetaTrader 5 terminal starting from the specified index.

```
copy_rates_from_pos(
  symbol,      // symbol name
  timeframe,   // timeframe
  start_pos,   // initial bar index
  count        // number of bars
  )
```

Example:

```
# get 10 GBPUSD D1 bars from the current day
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)
```

```
# Copying data to dataframe

data = pandas.DataFrame()


for i in symbol:

    rates = mt5.copy_rates_from_pos(i, mt5.TIMEFRAME_D1, 0, 500)

    data[i] = [y["close"] for y in rates]
```

Describe:

for i in symbol = for all index in variables name "symbol"

rates = set up variable name "rate" to collect the price from method "copy_rates_from_pos"

data[i] = data that collect the index rates

Output:

	EURUSDm	GBPUSDm	USDJPYm	USDCHFm	AUDUSDm	GBPJPYm
1	0.001469	0.004299	-0.003412	-0.004223	-0.004705	0.000925
2	-0.002318	-0.004161	-0.004877	-0.002004	-0.004020	-0.008801
3	-0.000054	-0.000056	0.001014	0.000495	0.000710	0.000768
4	-0.002242	-0.002951	-0.000474	-0.001624	0.000694	-0.003464
5	0.001746	0.004175	-0.003997	-0.003597	0.003544	0.000142
...	...	...	...	...	...	...
247	0.000338	-0.000168	0.001173	0.000243	0.001384	0.000283
248	0.005851	0.001643	-0.002091	-0.003334	0.003829	-0.000440
249	-0.003359	0.003706	0.001984	0.000658	-0.004689	0.005696
250	0.005570	0.005725	-0.002696	-0.004414	0.006281	0.003017
251	0.001649	0.002054	-0.000317	-0.001620	0.000172	0.001722

Select Strategy for trade:

The Currency Momentum Factor

Initially discovered in equity markets, momentum strategy is a well-known, well-researched, and, most importantly, a robust anomaly. Moreover, it has been documented in various asset classes across the world and by many academics and is considered as a powerful anomaly. Momentum is a trend-following strategy, where the strategy buys the assets which have performed well in the past and sells the assets which have performed badly. No surprise, the momentum anomaly is present also in the currencies. In the currency market, momentum is a widely observed feature that many exchange rates trend on a multi-year basis. Therefore, a strategy that follows the trend typically makes positive returns over time.

The financial literature about the anomaly is consistent, and the academics agree that the momentum anomaly has its place in the FX market, for example, Menkhoff, Sarno, Schmeling, and Schrimpf in the Currency Momentum Strategies, tested the momentum strategy during the period from 1976 to 2010, with an investment universe consisting of more than 40 currencies. They have found a large and significant cross-sectional spread in excess returns of up to 10% p.a. between a past winner and past loser currencies, i.e., currencies with recent high returns continue to outperform currencies with recent low returns by a significant margin.

Markets Traded : Currencies

Period of Rebalancing : Weekly

Complexity Evaluation : Simply strategy

Strategy:

```
def Rebalance(self):

        sorted_by_performance = sorted([x for x in self.data.items() if

        x[1].IsReady], key = lambda x: x[1].Current.Value, reverse = True)

        long = [x[0] for x in sorted_by_performance[:3]]

        short = [x[0] for x in sorted_by_performance[-3:]]
```

Describe:

Sort_by_performance = Ranking returns for all currencies.

Long = Long position on Top 3 currencies

Short = Short position on Last 3 currencies

Select Strategy for trade:

FOREX Value investment : Purchasing Power Parity (PPP)

One of these factors is an FX value, deeply connected with a Purchasing power parity (PPP), a theory concerning the long-term equilibrium exchange rates based on a relative price level of two countries. The concept mentioned above was founded on the law of one price – the idea that in the absence of transaction costs, identical goods in different markets would be priced the same. No surprise, different countries consume different baskets of goods; however, it is partially possible to assess a relative price level. More precisely, it is possible to find out which country is "cheaper" and on the other hand, which country is "more expensive" for living.

Concentrating on the FX market in the long-run, currencies tend to move towards their "fair value". Consequently, systematically buying "undervalued" currencies and selling "overvalued" currencies leads to a profitable trading strategy in the medium-term. Last but not least, the world of academics recognizes the value in the foreign exchange market as one of the critical factors, as well as FX momentum or FX carry.

Markets Traded : Currencies

Period of Rebalancing : Quarterly

Complexity Evaluation : Simply strategy

We will create a strategy using "Real Effective Exchange Rate" (REER) valuation.

Where : REER = CERn x CERn x CERn x 100

CER = Country Exchange Rate

- Fetch the monthly REER values of the 8 countries namely: Singapore, Australia, Canada, Switzerland, United Kingdom, Japan, New Zealand and Euro area
- Compute the past 6 months rolling mean of the REER value

- Generate Trading Signal
    - If REER value >= 6 months mean of the REER then sell the currency pair
    - If REER value < 6 months mean of the REER then buy the currency pair

Coding :

```
reer = pd.read_csv('REER.csv',index_col=0)

reer.index = pd.to_datetime(reer.index)



# Calculate the rolling mean of reer

reer_mean = reer.rolling(6).mean()

signal = reer < reer_mean
```