تابع get_tick_size:

تابع `get_tick_size` یک تابع است که برای دریافت اندازه تیک (Tick Size) برای یک نماد مشخص استفاده می‌شود.

ورودی تابع، نماد (symbol) است که می‌خواهید اندازه تیک آن را دریافت کنید.

خروجی تابع، یک عدد اعشاری (float) است که مقدار اندازه تیک نماد را نشان می‌دهد.

در صورتی که نماد مورد نظر معتبر نباشد یا پیدا نشود، خطای ValueError پرتاب می‌شود.

مثال استفاده از تابع:

```
symbol = 'EURUSD'
try:
    tick_size = get_tick_size(symbol)
    print(f"اندازه تیک برای نماد '{symbol}' برابر است با {tick_size}.")
except ValueError as e:
    print(str(e))
```

در این مثال، مقدار اندازه تیک برای نماد 'EURUSD' با استفاده از تابع `get_tick_size` دریافت می‌شود و در خروجی نمایش داده می‌شود. در صورتی که نماد معتبر نباشد یا پیدا نشود، خطای مربوطه نمایش داده می‌شود.

تابع open_calc:

تابع `open_calc` یک تابع است که برای باز کردن یک فایل اکسل و خواندن داده‌های آن استفاده می‌شود.

ورودی‌های تابع عبارتند از:

- `path` (رشته): مسیر فایل اکسل که می‌خواهید باز کنید. پیش‌فرض آن "static/calc.xlsx" است.
- `sheetname` (رشته): نام برگه (شیت) فایل اکسل که می‌خواهید خوانده شود. پیش‌فرض آن "United States" است.

خروجی تابع، داده‌های موجود در فایل اکسل را به صورت یک DataFrame برمی‌گرداند.

مثال استفاده از تابع:

```
```

```
calc_data = open_calc("static/calc.xlsx", "United States")

print(calc_data)
```

در این مثال، فایل اکسل با مسیر "static/calc.xlsx" باز می‌شود و برگه "United States" آن خوانده می‌شود. سپس داده‌های موجود در فایل اکسل در یک DataFrame قرار داده می‌شوند و در خروجی نمایش داده می‌شوند.

تابع strtotimedate:

تابع `strtotimedate` یک لیست از رشته‌های تاریخ/زمان را به شیوه‌ی `DatetimeIndex` تبدیل می‌کند.

ورودی‌های تابع عبارتند از:

- `dates` (لیست رشته): لیستی از تاریخ‌ها/زمان‌ها که می‌خواهید به شیوه‌ی `DatetimeIndex` تبدیل شوند.
- `format_` (رشته): فرمت رشته‌های تاریخ/زمان ورودی. پیش‌فرض آن "d/%m/%Y_%H:%M%" است.

خروجی تابع، یک `DatetimeIndex` است که تاریخ‌ها/زمان‌ها را به شیوه‌ی `DatetimeIndex` نمایش می‌دهد.

تابع `price_calc`:

تابع `price_calc` برای محاسبه قیمت با استفاده از قیمت باز شمعه (_open), اندازه پیپ (pip) و ضریب ضرب (multiplier) استفاده می‌شود.

ورودی‌های تابع عبارتند از:

- `_open` (عدد): قیمت باز شمعه.
- `pip` (عدد): اندازه پیپ.
- `multiplier` (عدد): ضریب ضرب.

خروجی تابع، قیمت محاسبه شده است که با استفاده از فرمول `(open + (pip * multiplier)_` بدست می‌آید.

تابع `isfloat`:

تابع `isfloat` بررسی می‌کند که آیا یک رشته به عنوان ورودی قابل تبدیل به عدد اعشاری است یا خیر.

ورودی تابع یک رشته (`num`) است که قرار است بررسی شود.

خروجی تابع، یک مقدار بولین (True/False) است که در صورت قابل تبدیل بودن رشته به عدد اعشاری، True و در غیر این صورت False است.

تابع get_mean_var:

تابع `get_mean_var` یک رشته را به عنوان ورودی دریافت می‌کند و میانگین و واریانس را با استفاده از اعداد موجود در رشته محاسبه می‌کند.

ورودی‌های تابع عبارتند از:

- `string` (رشته): رشته‌ای که حاوی اعداد برای محاسبه میانگین و واریانس است.
- `sign` (عدد): یک ضریب عددی نمایانگر علامت محاسبه میانگین و واریانس است. پیش‌فرض آن 1 است.

خروجی تابع، میانگین و واریانس محاسبه شده است که با استفاده از اعداد موجود در رشته و فرمول‌های مربوطه به دست می‌آیند. مقادیر میانگین و واریانس در خروجی به تفکیک برگردانده می‌شوند.

مثال استفاده از تابع:

```
data_string = "[10, 20, 30, 40, 50]"
try:
    mean, var = get_mean_var(data_string, sign=-1)
    print(f"میانگین: {mean}, واریانس: {var}")
except ValueError as e:
    print(str(e))
```

در این مثال، رشته `"[10, 20, 30, 40, 50]"` حاوی اعداد برای محاسبه میانگین و واریانس است. با استفاده از تابع `get_mean_var`، میانگین و واریانس با علامت منفی محاسبه می‌شوند و در خروجی نمایش داده می‌شوند. در صورت بروز خطا، پیغام خطا نمایش داده می‌شود.

تابع calc_position_size:

تابع `calc_position_size` یک سری ورودی شامل نماد (`symbol`)، نقطه ورود (`entry`)، سطح توقف ضرر (`sl`) و ریسک (`risk`) را دریافت می‌کند و اندازه موقعیت معامله را محاسبه می‌کند.

ورودی‌های تابع عبارتند از:

- `symbol` (رشته): نماد مورد نظر.
- `entry` (عدد اعشاری): نقطه ورود به معامله.
- `sl` (عدد اعشاری): سطح توقف ضرر.
- `risk` (عدد اعشاری): میزان ریسک درصدی.

خروجی تابع، اندازه موقعیت معامله به صورت عدد اعشاری است.

تابع در ابتدا با استفاده از `mt5.symbol_select` نماد مورد نظر را انتخاب می‌کند و سپس اطلاعات مربوط به نماد را با استفاده از `mt5.symbol_info` دریافت می‌کند.

سپس، اندازه تیک (`tick_size`) و ارزش تیک (`tick_value`) را از اطلاعات نماد محاسبه می‌کند.

مقدار `pips_at_risk` را با محاسبه فاصله نقاط بین نقطه ورود و سطح توقف ضرر و تقسیم بر اندازه تیک محاسبه می‌کند.

سپس، اندازه موقعیت معامله (`lot`) را با تقسیم ریسک بر (مقدار `pips_at_risk` ضربدر `tick_value`) محاسبه می‌کند.

در صورتی که نماد `symbol` برابر با 'XAUUSD' باشد، اندازه موقعیت معامله را بر 10 تقسیم می‌کند.

در نهایت، اندازه موقعیت معامله را به صورت عدد اعشاری (در دو رقم اعشار) با استفاده از `np.round` برمی‌گرداند.

مثال استفاده از تابع:

```
import MetaTrader5 as mt5

import numpy as np

mt5.initialize()

symbol = "EURUSD"

entry = 1.1234

sl = 1.1200

risk = 2.5

result = calc_position_size(symbol, entry, sl, risk)
```

```
print(result)

mt5.shutdown()
```

در این مثال، تابع `calc_position_size` با ورودی‌های مورد نیاز فراخوانی می‌شود. سپس نماد `EURUSD`، نقطه ورود 1.1234، سطح توقف ضرر 1.1200 و ریسک 2.5 به عنوان ورودی‌ها در نظر گرفته می‌شود. خروجی تابع در متغیر `result` ذخیره شده و در نهایت چاپ می‌شود. توجه داشته باشید که قبل از استفاده از تابع `calc_position_size`، باید کتابخانه `MetaTrader5` را مقداردهی اولیه کنید و پس از استفاده از تابع، آن را ببندید.

تابع strategy:

تابع `strategy` دارای پارامترهای زیر است:

- `df`: یک DataFrame از داده‌های معاملاتی.
- `symbol`: نماد یا نام ارز معاملاتی.
- `news`: اسم خبر مرتبط با معامله.
- `_open`: قیمت باز کردن معامله.
- `time_open`: زمان باز کردن معامله.
- `multiplier`: ضریبی که در محاسبه نقاط ورود اضافی استفاده می‌شود.
- `timeframe` (اختیاری): بازه زمانی مورد استفاده برای محاسبه نقاط ورود اضافی.
- `risk` (اختیاری): سطح ریسک مورد نظر برای معامله.

این تابع از تابع `get_extra_points` استفاده می‌کند تا نقاط ورود اضافی برای معامله را محاسبه کند. سپس اطلاعات معامله را به صورت یک لیست از دیکشنری‌ها برمی‌گرداند.

هر دیکشنری در لیست نمایانگر یک معامله است و شامل اطلاعات زیر است:

- `"News"`: اسم خبر مرتبط با معامله.
- `"Action"`: عملیات معامله (خرید یا فروش).
- `"Currency"`: نماد یا نام ارز معاملاتی.
- `"EntryPoint"`: قیمت ورود به معامله.
- `"TakeProfit"`: قیمت تقاضای سود.

- `"StepLoss"`: قیمت تقاضای تلفات.

- `"EntryTime"`: زمان ورود به معامله.

- `"PendingTime"`: زمان معلق بودن معامله به ثانیه.

- `"RR"`: نسبت سود به تلفات معامله.

- `"WinRate"`: نرخ پیروزی معامله.

- `"PositionSize"`: اندازه موقعیت معاملاتی محاسبه شده براساس ریسک مورد نظر.

- `"Risk"`: سطح ریسک مورد نظر برای معامله.

نمونه استفاده از این تابع به شکل زیر است:

```
df = pd.DataFrame(...)  # داده‌های معاملاتی
symbol = "ABC"  # نماد یا نام ارز معاملاتی
news = اسم خبر
open_ = 100.0
time_open = pd.Timestamp("2023-09-15 10:00:00")
multiplier = 1.5
info = strategy(df, symbol, news, open_, time_open, multiplier)
print(info)
```

این کد یک DataFrame تصادفی `df` را ایجاد کرده و با استفاده از تابع `strategy`، اطلاعات معامله را برای نماد "ABC" و باز کردن معامله در قیمت 100.0، در زمان "10:00:00 15-09-2023" و با ضریب نقاط ورود اضافی 1.5 محاسبه متابع `strategy` که توضیح داده شد به صورت زیر تعریف شده است:

تابع `strategy` از یک DataFrame به نام `df`، که شامل داده‌های معاملاتی است، و سایر پارامترهای ورودی مانند `symbol` (نماد ارز)، `news` (اسم خبر)، `open_` (قیمت باز کردن معامله)، `time_open` (زمان باز کردن معامله)، `multiplier` (ضریب)، `timeframe` (بازه زمانی) و `risk` (سطح ریسک) استفاده می‌کند.

ابتدا تابع `get_extra_points` را با استفاده از این پارامترها فراخوانی می‌کند تا نقاط ورود اضافی برای معامله را دریافت کند و در متغیر `positions` ذخیره می‌کند.

سپس، اطلاعات معامله را به صورت یک لیست از دیکشنری‌ها در متغیر `info` قرار می‌دهد. هر دیکشنری شامل اطلاعات یک معامله است و دارای کلیدهای `"News"` (اسم خبر)، `"Action"` (عملیات معامله)، `"Currency"` (نماد ارز)، `"EntryPoint"` (قیمت ورود)، `"TakeProfit"` (قیمت تقاضای سود)، `"StepLoss"` (قیمت تقاضای تلفات)، `"EntryTime"` (زمان ورود)، `"PendingTime"` (زمان معلق)

تابع trade_on_news:

تابع `trade_on_news` یک تابع معاملاتی است که با دریافت پارامترهای مختلف، اطلاعات معاملات مرتبط با یک خبر را محاسبه می‌کند.

پارامتر های ورودی:

- `initialize`: یک پارامتر بولین (True یا False) که نشان می‌دهد آیا برنامه در حال اجرا در حالت اولیه است یا خیر.
- `news`: متن خبر مرتبط با معامله.
- `country`: نام کشور مرتبط با خبر و معامله.
- `risk`: سطح ریسک مورد نظر برای معامله.
- `time_open`: زمان باز شدن معامله.
- `symbol`: نماد یا نام ارز معاملاتی. (پارامتر اختیاری)
- `timeframe`: بازه زمانی معاملاتی. (پارامتر اختیاری)

تابع ابتدا یک DataFrame محاسباتی را با استفاده از تابع `open_calc` محاسبه می‌کند. سپس در صورتی که پارامترهای `timeframe` و `symbol` مقدار نداشته باشند، سطرهای جدول محاسباتی را که متن خبر آنها شامل `news` است، انتخاب می‌کند و بر اساس نرخ پیروزی، به صورت نزولی مرتب می‌کند. سپس نماد و بازه زمانی بهترین معامله بر اساس نرخ پیروزی را استخراج می‌کند و آنها را به عنوان مقادیر `symbol` و `timeframe` در نظر می‌گیرد.

در ادامه، تابع داده معاملاتی را با استفاده از تابع `get_data_from_mt5` برای نماد و بازه زمانی مشخص شده دریافت می‌کند. سپس قیمت باز شدن معامله را از آخرین ردیف داده‌های معاملاتی استخراج می‌کند. در نهایت، با استفاده از تابع `strategy` و پارامترهای محاسبه شده، اطلاعات معامله را محاسبه می‌کند و در قالب یک لیست از دیکشنری‌ها برمی‌گرداند. این دیکشنری‌ها شامل

اطلاعات معاملات، مانند نماد ارز، نوع عملیات، قیمت ورود، زمان ورود، زمان معلق بودن معامله، سطح ریسک و سایر اطلاعات مرتبط با معامله هستند.

نمونه استفاده از این تابع به صورت زیر است:

```
trade_on_news(initialize=True, news="خبر معاملاتی", country="کشور", risk=0.5, time_open="12:00", symbol="EUR/USD", timeframe="2h")
```

در این مثال، تابع به عنوان ورودی دریافت می‌کند که برنامه در حالت اولیه قرار دارتابع `trade_on_news` یک تابع معاملاتی است که با دریافت پارامترهای مختلف، اطلاعات معاملات مرتبط با یک خبر را محاسبه می‌کند.