Alpaca

BUSI 722: Data-Driven Finance II

Kerry Back, Rice University

Outline

1. Create connection
2. View account information
3. View market information
4. Make trades

Note: log on to alpaca at https://alpaca.markets/ to get key and secret key

In [1]:

```
import pandas as pd

from alpaca.trading.client import TradingClient
from alpaca.trading.requests import MarketOrderRequest
from alpaca.trading.enums import OrderSide, TimeInForce
from alpaca.trading.requests import GetAssetsRequest
from alpaca.data import StockHistoricalDataClient
from alpaca.data.requests import StockLatestQuoteRequest
from alpaca.trading.enums import AssetClass
```

1. Create Connection

- Substitute your key and your secret key.

In [2]:

```
with open("keys.txt", "r") as f:
    keys = f.readlines()
    key, secret_key = [x.strip() for x in keys]

trading_client = TradingClient(key, secret_key, paper=True)
```

2. Account Information

In [3]:
```
account = trading_client.get_account()
account
```

Out[3]:
```
{   'account_blocked': False,
    'account_number': 'PA3J0LUAM1VV',
    'accrued_fees': '0',
    'buying_power': '194450.95',
    'cash': '95929.6',
    'created_at': datetime.datetime(2024, 2, 11, 16, 46, 29, 12993, t
zinfo=datetime.timezone.utc),
    'crypto_status': <AccountStatus.ACTIVE: 'ACTIVE'>,
    'currency': 'USD',
    'daytrade_count': 0,
    'daytrading_buying_power': '0',
    'equity': '99994.55',
    'id': UUID('8955151d-0998-4e09-bae8-6f8ed3979f47'),
    'initial_margin': '2953.23',
    'last_equity': '99945.2',
    'last_maintenance_margin': '1759.8',
    'long_market_value': '4985.7',
    'maintenance_margin': '1771.94',
    'multiplier': '2',
    'non_marginable_buying_power': '95041.32',
    'pattern_day_trader': False,
    'pending_transfer_in': '0',
    'pending_transfer_out': None,
    'portfolio_value': '99994.55',
    'regt_buying_power': '194450.95',
    'short_market_value': '-920.75',
```

Examples of retrieving account details

In [4]:

```
print("Account balance is", account.equity)
print("Yesterday's account balance was", account.last_equity)
print("Buying power is", account.buying_power)
print("Cash balance is", account.cash)
```

```
Account balance is 99994.55
Yesterday's account balance was 99945.2
Buying power is 194450.95
Cash balance is 95929.6
```

Open positions

In [5]:

```
positions = trading_client.get_all_positions()
quantity = {x.symbol: float(x.qty) for x in positions}
mkt_value = {x.symbol: float(x.market_value) for x in positions}
cost_basis = {x.symbol: float(x.cost_basis) for x in positions}

positions = pd.DataFrame(
    [quantity, mkt_value, cost_basis], index=["quantity", "mkt_value", "cost_
)
positions.T
```

Out[5]:

	quantity	mkt_value	cost_basis
AAPL	-5.0	-920.75	-942.39
SPY	10.0	4985.70	5012.77

3. Market Information

Information about a particular asset

In [6]:
```
symbol = 'SPY'
trading_client.get_asset(symbol)
```

Out[6]:
```
{   'asset_class': <AssetClass.US_EQUITY: 'us_equity'>,
    'easy_to_borrow': True,
    'exchange': <AssetExchange.ARCA: 'ARCA'>,
    'fractionable': True,
    'id': UUID('b28f4066-5c6d-479b-a2af-85dc1a8f16fb'),
    'maintenance_margin_requirement': 30.0,
    'marginable': True,
    'min_order_size': None,
    'min_trade_increment': None,
    'name': 'SPDR S&P 500 ETF Trust',
    'price_increment': None,
    'shortable': True,
    'status': <AssetStatus.ACTIVE: 'active'>,
    'symbol': 'SPY',
    'tradable': True}
```

Quotes

In [7]:
```
lst = ['SPY','AAPL','IBM']

data_client = StockHistoricalDataClient(key, secret_key)
params = StockLatestQuoteRequest(symbol_or_symbols=lst)
quotes = data_client.get_stock_latest_quote(params)
quotes
```

Out[7]:
```
{'AAPL': {   'ask_exchange': 'V',
     'ask_price': 185.52,
     'ask_size': 1.0,
     'bid_exchange': 'V',
     'bid_price': 180.5,
     'bid_size': 1.0,
     'conditions': ['R'],
     'symbol': 'AAPL',
     'tape': 'C',
     'timestamp': datetime.datetime(2024, 2, 14, 20, 59, 59, 996783,
tzinfo=datetime.timezone.utc)},
 'SPY': {   'ask_exchange': 'V',
     'ask_price': 502.09,
     'ask_size': 10.0,
     'bid_exchange': 'V',
     'bid_price': 498.57,
     'bid_size': 1.0,
     'conditions': ['R'],
     'symbol': 'SPY',
     'tape': 'B',
     'timestamp': datetime.datetime(2024, 2, 14, 21, 59, 55, 17822, t
zinfo=datetime.timezone.utc)}
```

List of all US equities (including OTC)

In [8]:

```
search_params = GetAssetsRequest(asset_class=AssetClass.US_EQUITY)
assets = trading_client.get_all_assets(search_params)

print(f"There are {len(assets):,} US equities.")
print("First stock in list is")
assets[0]
```

```
There are 31,618 US equities.
First stock in list is
```

Out[8]:

```
{   'asset_class': <AssetClass.US_EQUITY: 'us_equity'>,
    'easy_to_borrow': False,
    'exchange': <AssetExchange.NASDAQ: 'NASDAQ'>,
    'fractionable': False,
    'id': UUID('345a9d38-1c90-4b33-806a-c5271ca795ea'),
    'maintenance_margin_requirement': 100.0,
    'marginable': False,
    'min_order_size': None,
    'min_trade_increment': None,
    'name': 'LG Display Co. Rights (expiration 03/01/24)',
    'price_increment': None,
    'shortable': False,
    'status': <AssetStatus.INACTIVE: 'inactive'>,
    'symbol': '501RGT013',
    'tradable': False}
```

Tradable and shortable stocks

In [9]:

```
tradable = [x.symbol for x in assets if x.tradable]
shortable = [x.symbol for x in assets if x.shortable]

print(f"Number of tradable stocks is {len(tradable):,}")
print(f"Number of shortable stocks is {len(shortable):,}")
```

```
Number of tradable stocks is 11,336
Number of shortable stocks is 5,093
```

Tradable exchanges

In [10]:
```
set([x.exchange[:] for x in assets])
```

Out[10]:
```
{'AMEX', 'ARCA', 'BATS', 'NASDAQ', 'NYSE', 'OTC'}
```

4. Trading

In [11]:

```
order = MarketOrderRequest(
    symbol="SPY",
    qty=10,
    side=OrderSide.BUY,
    time_in_force=TimeInForce.DAY
    )
_ = trading_client.submit_order(order)
```

In [12]:

```
order = MarketOrderRequest(
    symbol="AAPL",
    qty=5,
    side=OrderSide.SELL,
    time_in_force=TimeInForce.DAY
    )
_ = trading_client.submit_order(order)
```