Trading with Interactive Brokers using Python

Speaker: Dr. Hui Liu

www.IBridgePy.com | IBridgePy@gmail.com

Nov 2016

Introduction

- *Author of IBridgePy, a flexible and easy-to-use python tool to trade with Interactive Brokers*
- *Founder of Running River Investment LLC,* a private hedge fund specialized in development of automated trading strategies using Python
- *Faculty at QuantInsti*, a pioneer institute in Algorithmic Trading since 2010

  www.IBridgePy.com; Email: IBridgePy@gmail.com
  www.quantinsti.com; Email: contact@quantinsti.com

Agenda

- Advantages of Interactive Brokers and Python
- IBridgePy
  - Installation
  - Connecting to IB
  - PythonXY and Spider IDE
  - Real time quotes, Request historical data, Place orders
  - An example code of moving average crossing
- Two special features
  - Manage multiple accounts
  - Backtest

Advantages of Interactive Brokers

- IB API to automate trading
- Low trading cost
- Global markets access
- Variety of products: stocks, options, futures, forex, bonds, ETFs and CFDs

Advantages of Python for trading

- Easy to learn
- Availability of variety of modules
- Open source

IBridgePy

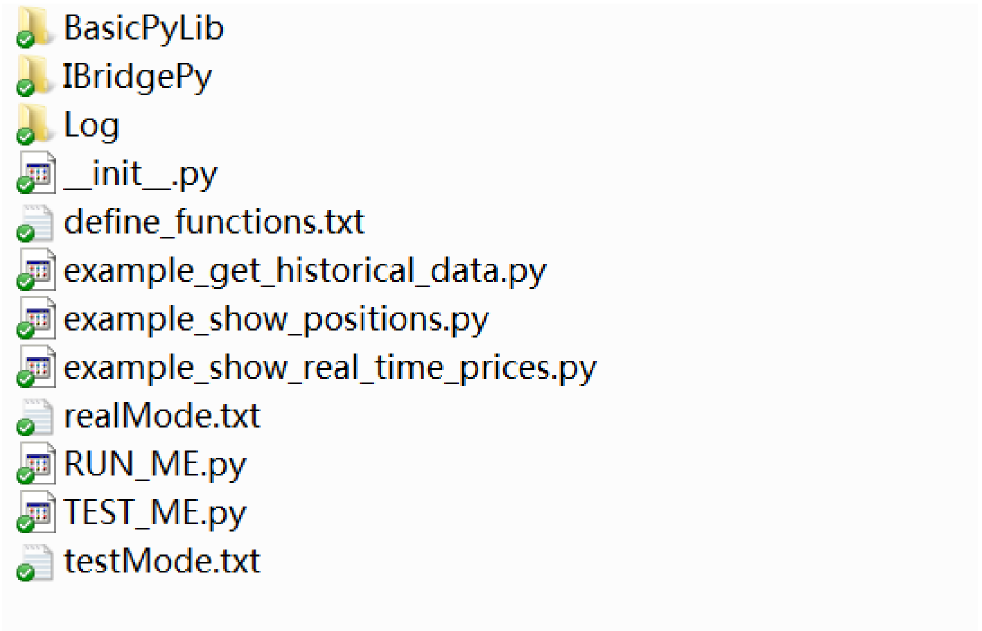
Are you looking for a simple tool to trade with Interactive Brokers API using Python

- **Python tool to trade with Interactive Brokers**
  - Flexible
  - Easy to use
  - Privacy

  Main features:
  - Trade any securities or commodities offered by IB
  - Manage multiple accounts at same time
  - Execute multiple trading strategies at same time

Installation

- Follow the link to download IBridgePy to your local folder and unzip it. Then you will see file structures like that



```
BasicPyLib
IBridgePy
Log
__init__.py
define_functions.txt
example_get_historical_data.py
example_show_positions.py
example_show_real_time_prices.py
realMode.txt
RUN_ME.py
TEST_ME.py
testMode.txt
```

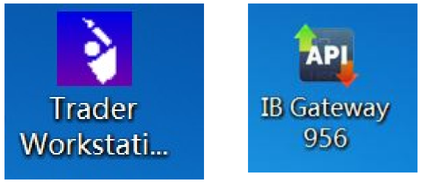
- You are good to test your trading algo!

Connecting to IB

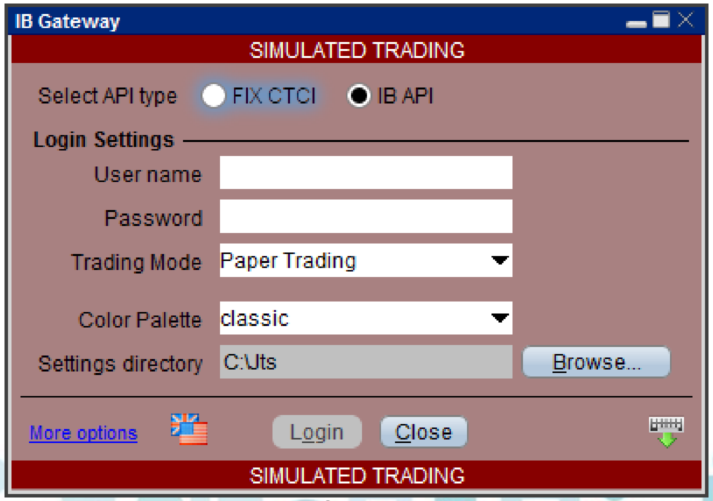
- You need have at least a demo account at Interactive Brokers to try your algo.
- Download either TWS or IB Gateway to feed your needs

  TWS will give you much more information about markets and interactive ways to communicate with IB

  IB Gateway is the better choice if TWS automatic log off is a concern for you



- Check 'IB API'
- Input your credentials and click 'Login'



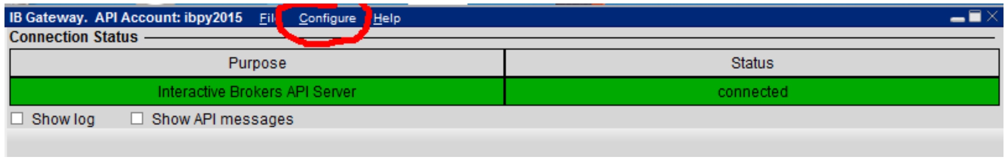
Select API type ○ FIX CTCI [● IB API]

Login Settings

User name	
Password	
Trading Mode	Paper Trading
Color Palette	classic
Settings directory	C:\Jts

Connecting to IB (2)

- In IB Gateway, → Configure → Settings

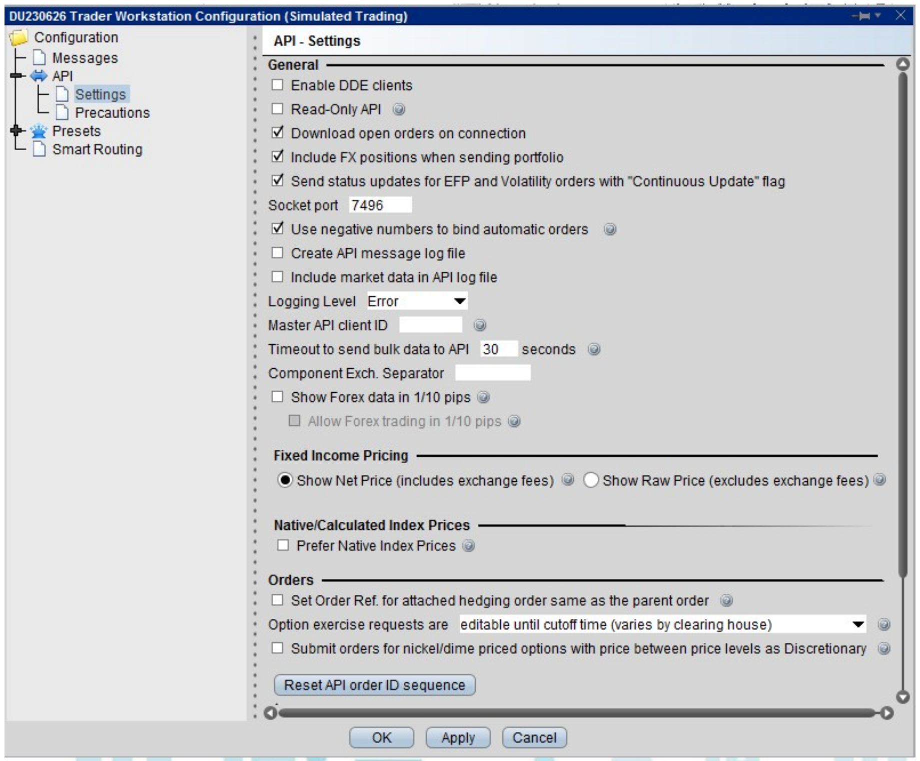


IB Gateway. API Account: ibpy2015 File Configure Help

Connection Status

Purpose	Status
Interactive Brokers API Server	connected

☐ Show log ☐ Show API messages



DU230626 Trader Workstation Configuration (Simulated Trading)

Configuration
- Messages
- API
  - Settings
  - Precautions
- Presets
- Smart Routing

API - Settings

General
- ☐ Enable DDE clients
- ☐ Read-Only API
- ☑ Download open orders on connection
- ☑ Include FX positions when sending portfolio
- ☑ Send status updates for EFP and Volatility orders with "Continuous Update" flag
- Socket port 7496
- ☑ Use negative numbers to bind automatic orders
- ☐ Create API message log file
- ☐ Include market data in API log file
- Logging Level Error
- Master API client ID
- Timeout to send bulk data to API 30 seconds
- Component Exch. Separator
- ☐ Show Forex data in 1/10 pips
- ☐ Allow Forex trading in 1/10 pips

Fixed Income Pricing
- ● Show Net Price (includes exchange fees) ○ Show Raw Price (excludes exchange fees)

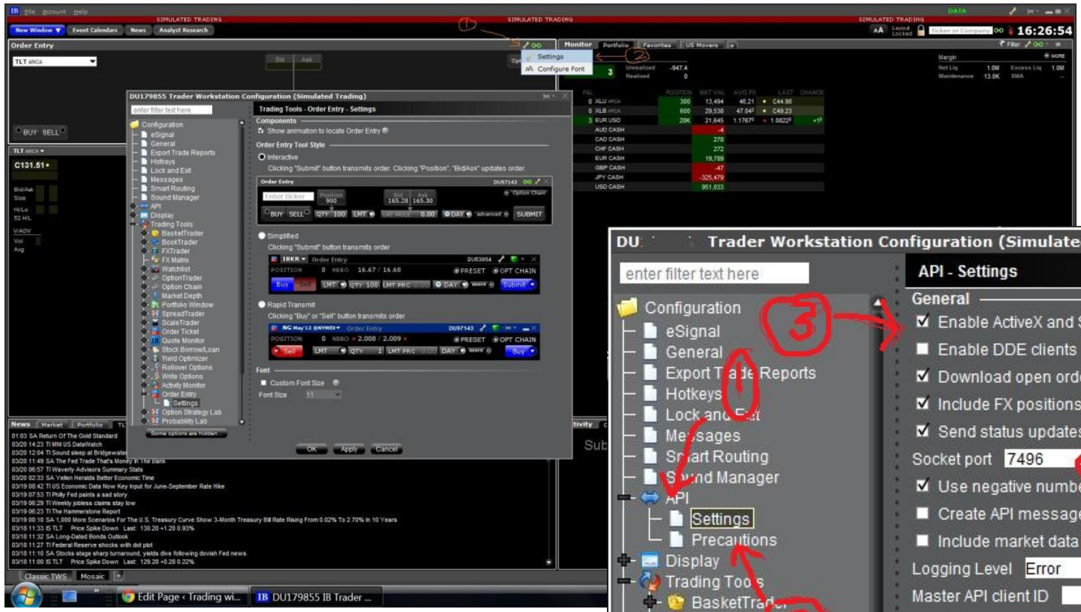
Native/Calculated Index Prices
- ☐ Prefer Native Index Prices

Orders
- ☐ Set Order Ref. for attached hedging order same as the parent order
- Option exercise requests are editable until cutoff time (varies by clearing house)
- ☐ Submit orders for nickel/dime priced options with price between price levels as Discretionary

Reset API order ID sequence

OK Apply Cancel

Connecting to IB (3)



DU179855 Trader Workstation Configuration (Simulated Trading)

Trading Tools - Order Entry - Settings

Components

Show animation to locate Order Entry

Order Entry Tool Style

Interactive

Clicking "Submit" button transmits order. Clicking "Position", "Bid/Ask" updates order.

Simplified

Clicking "Submit" button transmits order

Rapid Transmit

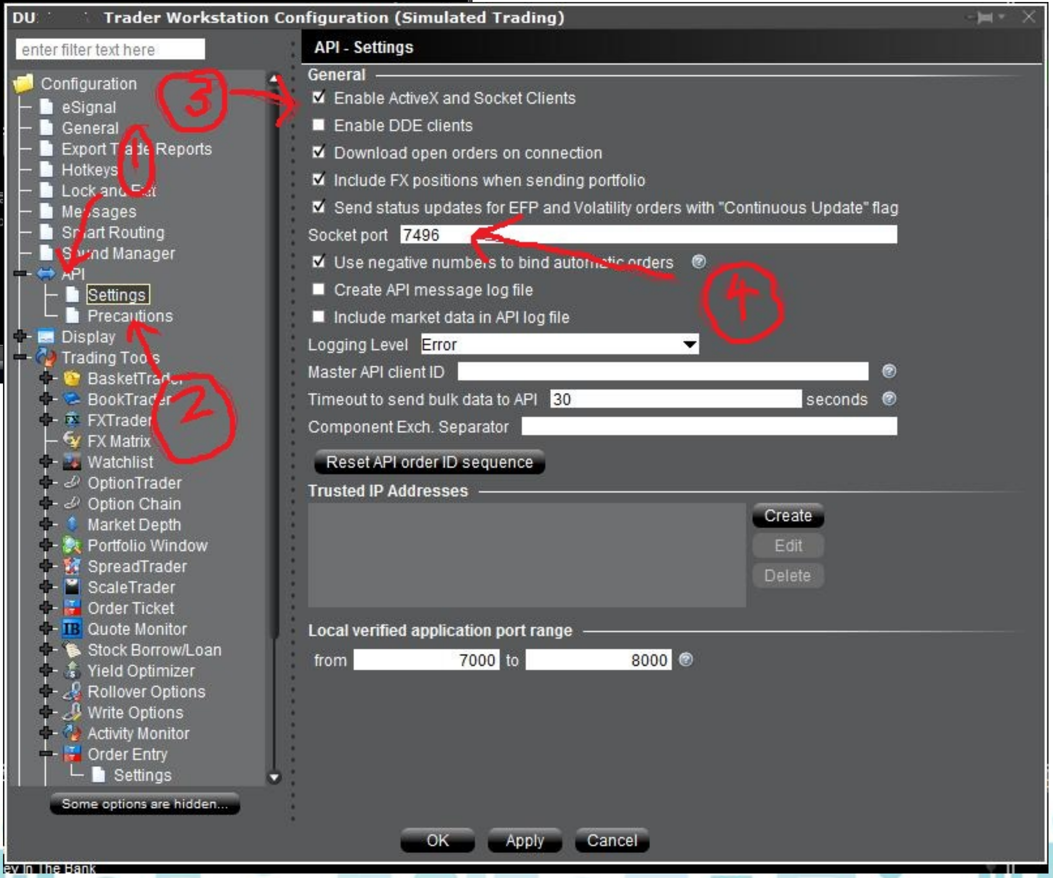
Clicking "Buy" or "Sell" button transmits order

Font

Custom Font Size

Font Size 11

OK Apply Cancel



DU Trader Workstation Configuration (Simulated Trading)

enter filter text here

Configuration
- eSignal
- General
- Export Trade Reports
- Hotkeys
- Lock and Exit
- Messages
- Smart Routing
- Sound Manager
- API
  - Settings
  - Precautions
- Display
- Trading Tools
  - BasketTrader
  - BookTrader
  - FXTrader
  - FX Matrix
  - Watchlist
  - OptionTrader
  - Option Chain
  - Market Depth
  - Portfolio Window
  - SpreadTrader
  - ScaleTrader
  - Order Ticket
  - Quote Monitor
  - Stock Borrow/Loan
  - Yield Optimizer
  - Rollover Options
  - Write Options
  - Activity Monitor
  - Order Entry
    - Settings

Some options are hidden...

API - Settings

General

- [✓] Enable ActiveX and Socket Clients
- [ ] Enable DDE clients
- [✓] Download open orders on connection
- [✓] Include FX positions when sending portfolio
- [✓] Send status updates for EFP and Volatility orders with "Continuous Update" flag

Socket port 7496

- [✓] Use negative numbers to bind automatic orders
- [ ] Create API message log file
- [ ] Include market data in API log file

Logging Level Error

Master API client ID

Timeout to send bulk data to API 30 seconds

Component Exch. Separator

Reset API order ID sequence

Trusted IP Addresses

Create Edit Delete

Local verified application port range

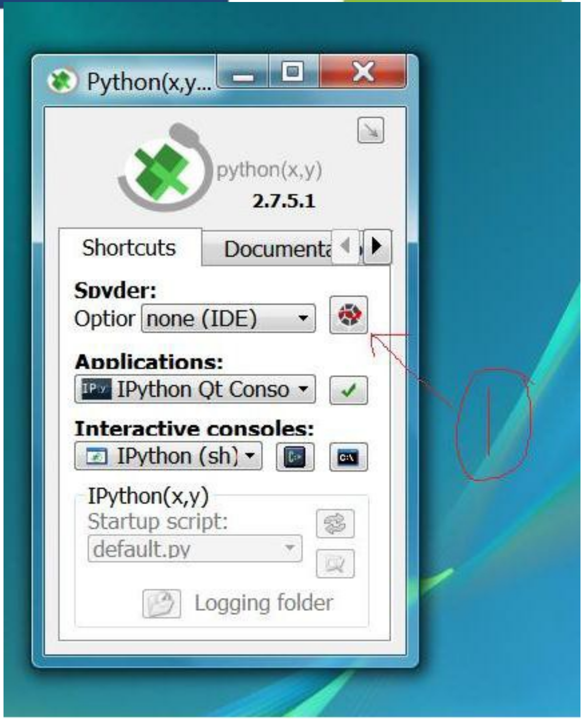
from 7000 to 8000

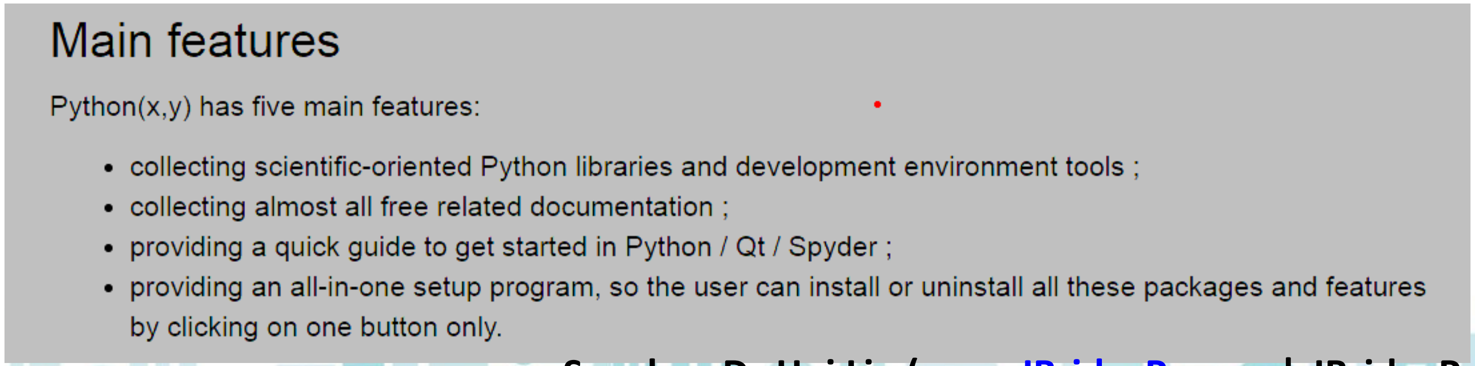
OK Apply Cancel

Speaker: Dr. Hui Liu (www.IBridgePy.com | IBridgePy@gmail.com)

PythonXY

- Python(x,y) is a free scientific and engineering development software for numerical computations, data analysis and data visualization based on Python programming language, Qt graphical user interfaces and Spyder interactive scientific development environment.





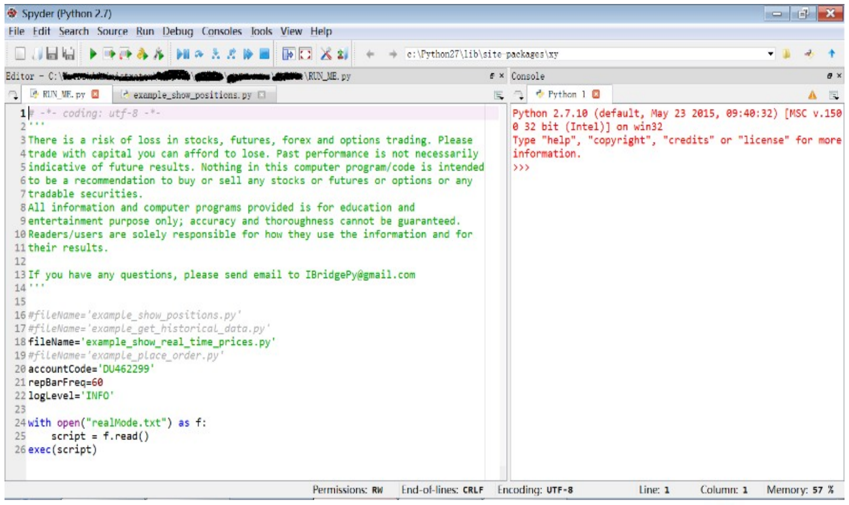
Main features

Python(x,y) has five main features:

- collecting scientific-oriented Python libraries and development environment tools ;
- collecting almost all free related documentation ;
- providing a quick guide to get started in Python / Qt / Spyder ;
- providing an all-in-one setup program, so the user can install or uninstall all these packages and features by clicking on one button only.

Prepare IDE

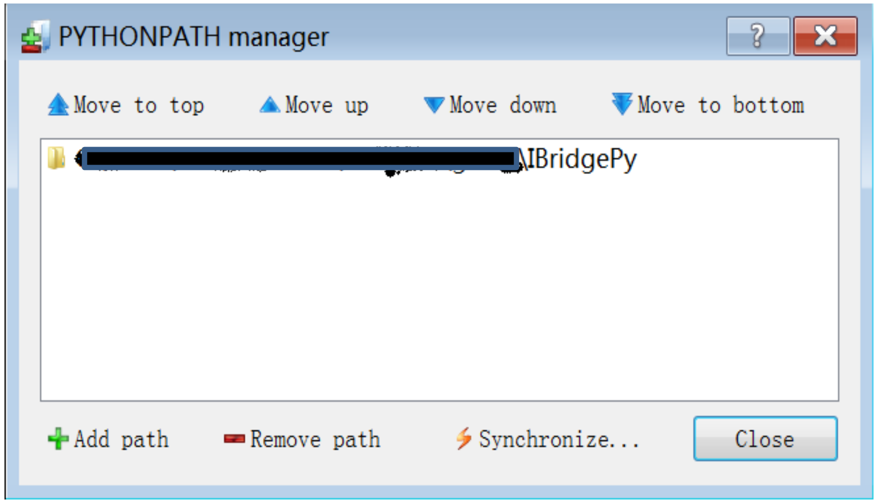
- 'View'-> 'Panes'-> Check 'Editor' and 'Console'
- Add the folder to the Python Path by clicking 'Tools' -> 'PYTHONPATH manager' -> 'Add path' -> choose the folder where you unzip IBridgePy.
- Then, click 'Close' to accept it.
- It is preferred to restart the Spider IDE to make sure the Python path is effective.


Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Tools View Help
c:\Python27\lib\site-packages\xy
Editor - C:\[redacted]\RUN_ME.py
RUN_ME.py
example_show_positions.py
Console
Python 1

```
# -*- coding: utf-8 -*-
'''
There is a risk of loss in stocks, futures, forex and options trading. Please
trade with capital you can afford to lose. Past performance is not necessarily
indicative of future results. Nothing in this computer program/code is intended
to be a recommendation to buy or sell any stocks or futures or options or any
tradable securities.
All information and computer programs provided is for education and
entertainment purpose only; accuracy and thoroughness cannot be guaranteed.
Readers/users are solely responsible for how they use the information and for
their results.

If you have any questions, please send email to IBridgePy@gmail.com
'''

#fileName='example_show_positions.py'
#fileName='example_get_historical_data.py'
fileName='example_show_real_time_prices.py'
#fileName='example_place_order.py'
accountCode='DU462299'
repBarFreq=60
logLevel='INFO'

with open("realMode.txt") as f:
    script = f.read()
exec(script)
```

```
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.150
0 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 1 Memory: 57 %

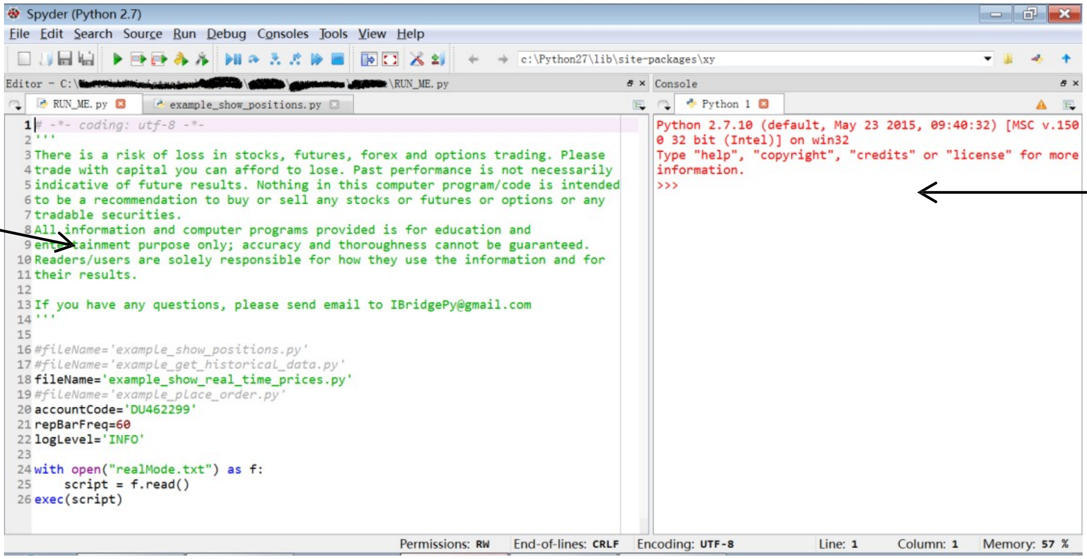


[redacted]\IBridgePy

Run a sample code

- Open RUN_ME.py in Spider IDE by clicking 'File'-> 'Open'
- You may see Spider show up like the following

You may edit your code here

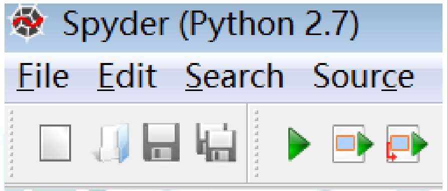

Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Tools View Help
c:\Python27\lib\site-packages\xy
Editor - C:\[redacted]\RUN_ME.py
RUN_ME.py
example_show_positions.py
Console
Python 1
Permissions: RW
End-of-lines: CRLF
Encoding: UTF-8
Line: 1
Column: 1
Memory: 57 %

```
# -*- coding: utf-8 -*-
'''
There is a risk of loss in stocks, futures, forex and options trading. Please
trade with capital you can afford to lose. Past performance is not necessarily
indicative of future results. Nothing in this computer program/code is intended
to be a recommendation to buy or sell any stocks or futures or options or any
tradable securities.
All information and computer programs provided is for education and
entertainment purpose only; accuracy and thoroughness cannot be guaranteed.
Readers/users are solely responsible for how they use the information and for
their results.

If you have any questions, please send email to IBridgePy@gmail.com
'''

#fileName='example_show_positions.py'
#fileName='example_get_historical_data.py'
fileName='example_show_real_time_prices.py'
#fileName='example_place_order.py'
accountCode='DU462299'
repBarFreq=60
logLevel='INFO'

with open("realMode.txt") as f:
    script = f.read()
exec(script)
```

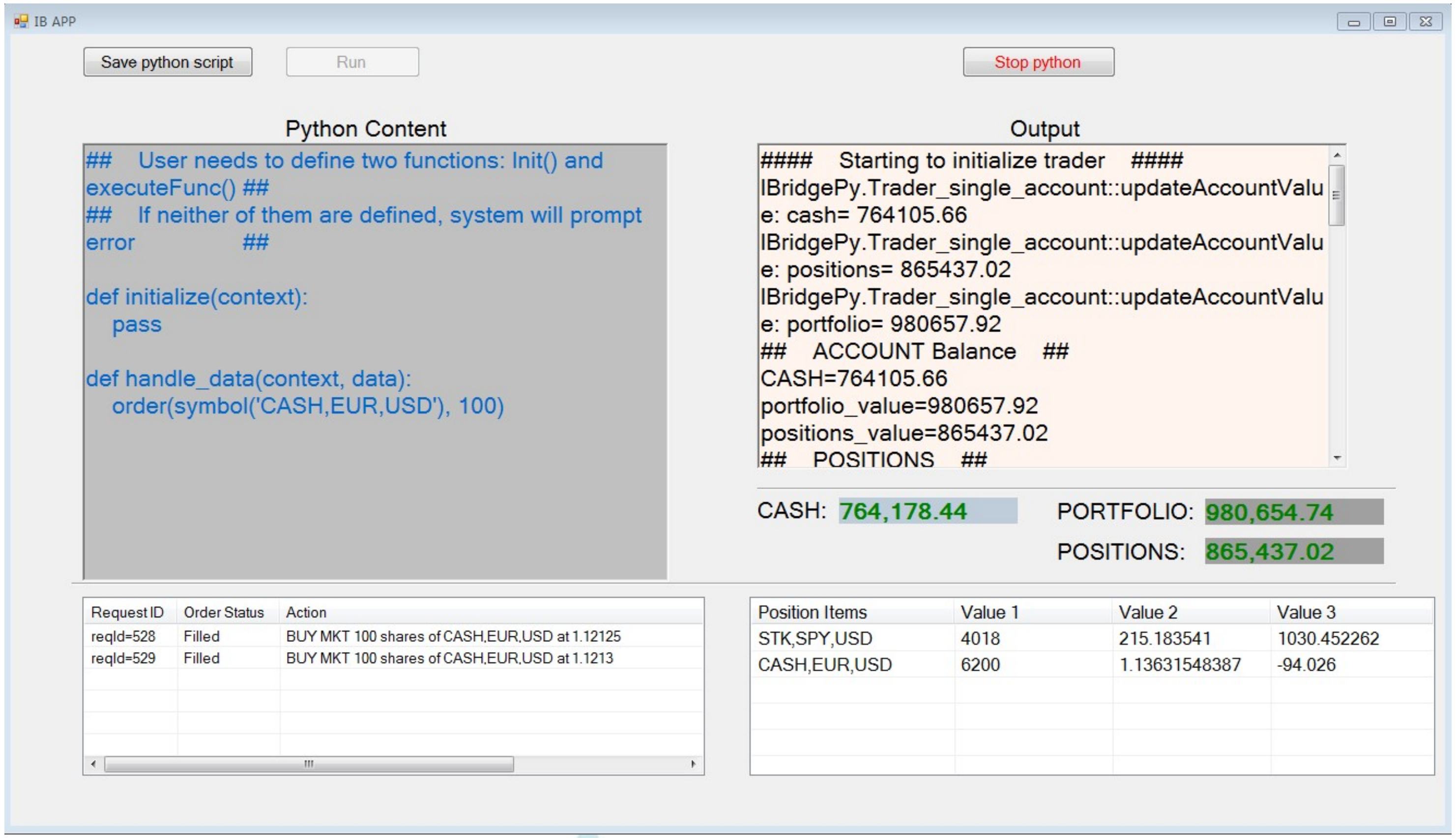
```
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.150
0 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

You will see results here

- To run the python code, click the green triangle or 'F5'



IBridgePy next generation



```
##    User needs to define two functions: Init() and
executeFunc() ##
##    If neither of them are defined, system will prompt
error              ##

def initialize(context):
    pass

def handle_data(context, data):
    order(symbol('CASH,EUR,USD'), 100)
```

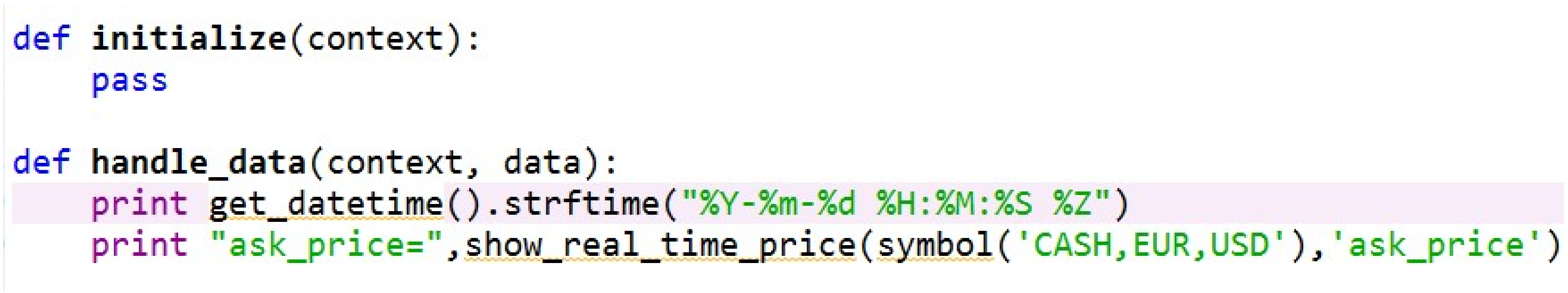
```
####    Starting to initialize trader    ####
IBridgePy.Trader_single_account::updateAccountValu
e: cash= 764105.66
IBridgePy.Trader_single_account::updateAccountValu
e: positions= 865437.02
IBridgePy.Trader_single_account::updateAccountValu
e: portfolio= 980657.92
##    ACCOUNT Balance    ##
CASH=764105.66
portfolio_value=980657.92
positions_value=865437.02
##    POSITIONS    ##
```

CASH: 764,178.44   PORTFOLIO: 980,654.74
POSITIONS: 865,437.02

Request ID	Order Status	Action
reqId=528	Filled	BUY MKT 100 shares of CASH,EUR,USD at 1.12125
reqId=529	Filled	BUY MKT 100 shares of CASH,EUR,USD at 1.1213

Position Items	Value 1	Value 2	Value 3
STK,SPY,USD	4018	215.183541	1030.452262
CASH,EUR,USD	6200	1.13631548387	-94.026

Code structure



```
def initialize(context):
    pass

def handle_data(context, data):
    print get_datetime().strftime("%Y-%m-%d %H:%M:%S %Z")
    print "ask_price=",show_real_time_price(symbol('CASH,EUR,USD'),'ask_price')
```

- "initialize()" is an built-in method to claim variables. It will only be run once
- "handle_data()" is also an built-in method where trading decisions are made. Two inputs are given here (context, data). "Context" contains the variables claimed in initialize(). "data" contains account information and near real-time quotes received from IB

Run a sample code

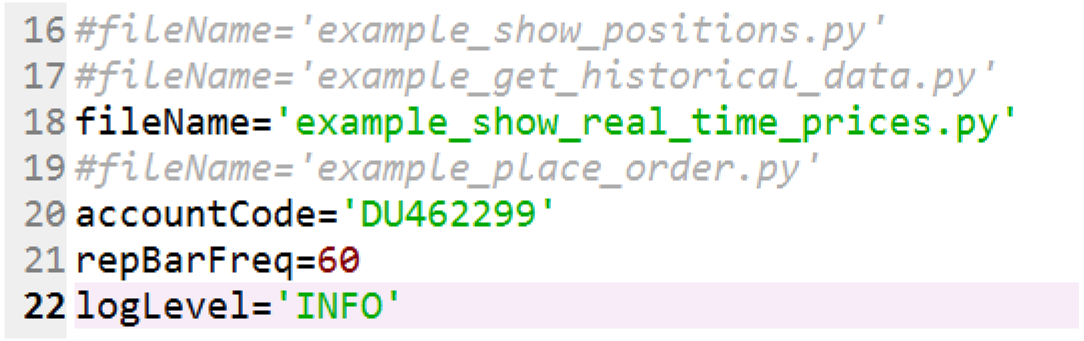
```
####    Starting to initialize trader    ####
##    ACCOUNT Balance    ##
CASH=769185.34
portfolio_value=987851.33
positions_value=872629.21
##    POSITIONS    ##
Symbol Amount Cost_basis Latest_profit
STK,SPY,USD 4018 215.183541 NA
CASH,EUR,USD 5600 1.13600375 -90.741
##    END    ##
##    NO OPEN orders    ##
####    Initialize trader COMPLETED    ####
2016-09-23 00:57:50 EDT
ask_price= 1.11985
2016-09-23 00:57:51 EDT
ask_price= 1.11985
2016-09-23 00:57:52 EDT
ask_price= 1.11985
2016-09-23 00:57:53 EDT
ask_price= 1.11985
2016-09-23 00:57:54 EDT
ask_price= 1.11985
2016-09-23 00:57:55 EDT
ask_price= 1.11985
```

Summary of the account

handle_data function is running every second

Results:
Print the ask price every second

Closer look at RUN_ME.py



```
16 #fileName='example_show_positions.py'
17 #fileName='example_get_historical_data.py'
18 fileName='example_show_real_time_prices.py'
19 #fileName='example_place_order.py'
20 accountCode='DU462299'
21 repBarFreq=60
22 logLevel='INFO'
```

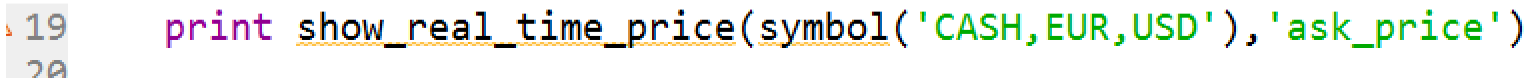
- Line 16~ line 19: You may choose one algo that you want to execute by commenting out others
- Line 20: You account code at IB
- Line 21: How often the function of handle_data(context, data), 60 means to run it every minute and 1 means to run it every second.
- Line 22: There are 4 levels to show results
  - ERROR: only show error messages
  - INFO: typical users will use it to know the results of your algo
  - DEBUG: you may know more info when you debug your algo
  - NOTSET: You will see tremendous info if you really want to know what is going on

Three corner stones

- Real time price
- Historical data
- Place order

Ask for real time quotes

- When you need a real time price, simply call a build-in function of show_real_time_price

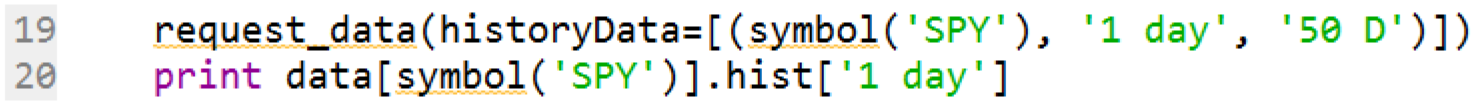


```
    print show_real_time_price(symbol('CASH,EUR,USD'),'ask_price')
```

- For most of US stocks and ETF, you can simply put 'AAPL' for apple instead of 'STK, AAPL, USD'
- For Forex, Future and Options, more info is needed to specify. For example,
  - 'FUT,ES,USD,201503'
  - 'OPT,AAPL,USD,20150702,133,P,100'
  - 'CASH, EUR, USD'

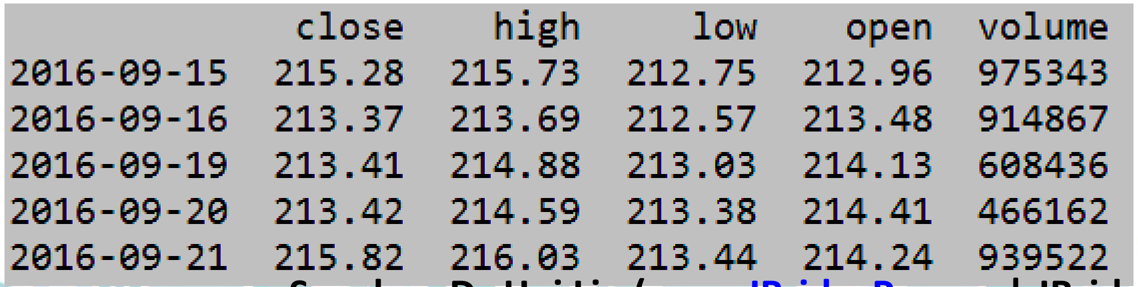
Ask for historical data from IB

- Request_data is used to ask all kinds of data from IB server
- To ask for historical data from IB, a parameter "historyData" need to be specified.



```
    request_data(historyData=[(symbol('SPY'), '1 day', '50 D')])
    print data[symbol('SPY')].hist['1 day']
```

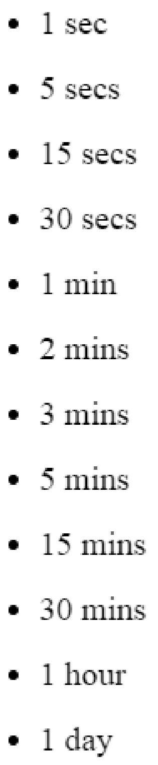
  – Selecting the instrument for which the historical data needs to be obtained, in the above example, SPY, an ETF tracking S&P500 index
  – Fixing the granularity (time gap), '1 day'
  – The period of go-back, '50 D' means go back 50 days from today.
  – The retrieved historical data are saved in a pandas dataframe that are saved at hist, an attribute of the DataClass, saved in a dictionary called data



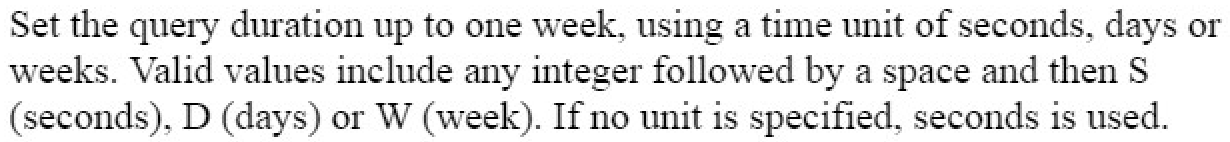
```
             close    high     low    open  volume
2016-09-15  215.28  215.73  212.75  212.96  975343
2016-09-16  213.37  213.69  212.57  213.48  914867
2016-09-19  213.41  214.88  213.03  214.13  608436
2016-09-20  213.42  214.59  213.38  214.41  466162
2016-09-21  215.82  216.03  213.44  214.24  939522
```

Ask for historical data from IB (2)

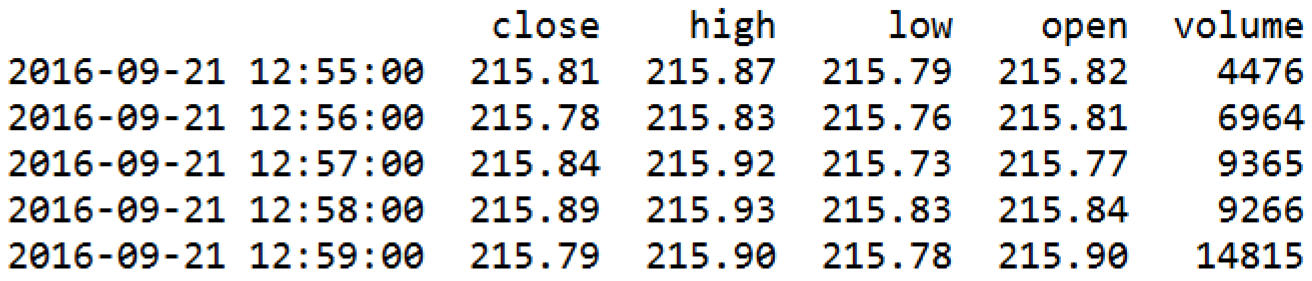
- Of course you may request historical data from IB from other time period, at other frequencies
- The request format is defined in the IB reference guide at: https://www.interactivebrokers.com/en/software/api/api.htm



- 1 sec
- 5 secs
- 15 secs
- 30 secs
- 1 min
- 2 mins
- 3 mins
- 5 mins
- 15 mins
- 30 mins
- 1 hour
- 1 day



Set the query duration up to one week, using a time unit of seconds, days or weeks. Valid values include any integer followed by a space and then S (seconds), D (days) or W (week). If no unit is specified, seconds is used.



```
                      close    high     low    open  volume
2016-09-21 12:55:00  215.81  215.87  215.79  215.82    4476
2016-09-21 12:56:00  215.78  215.83  215.76  215.81    6964
2016-09-21 12:57:00  215.84  215.92  215.73  215.77    9365
2016-09-21 12:58:00  215.89  215.93  215.83  215.84    9266
2016-09-21 12:59:00  215.79  215.90  215.78  215.90   14815
```

Ask for historical data from IB (3)

- Request multiple historical data at once

```
request_data(historyData=[(symbol('SPY'), '1 min', '1 D'),
                          (symbol('AAPL'), '1 day', '10 D') ])
print data[symbol('SPY')].hist['1 min'].tail()
print data[symbol('AAPL')].hist['1 day'].tail()
```

SPY

AAPL

```
                       close    high     low    open  volume
2016-09-21 12:55:00  215.81  215.87  215.79  215.82    4476
2016-09-21 12:56:00  215.78  215.83  215.76  215.81    6964
2016-09-21 12:57:00  215.84  215.92  215.73  215.77    9365
2016-09-21 12:58:00  215.89  215.93  215.83  215.84    9266
2016-09-21 12:59:00  215.79  215.90  215.78  215.90   14815
              close    high     low    open  volume
2016-09-15  115.57  115.73  113.49  113.86  792590
2016-09-16  114.92  116.13  114.04  115.17  601513
2016-09-19  113.58  116.18  113.25  115.25  411859
2016-09-20  113.57  114.12  112.51  112.96  298369
2016-09-21  113.55  113.99  112.44  113.85  295489
```

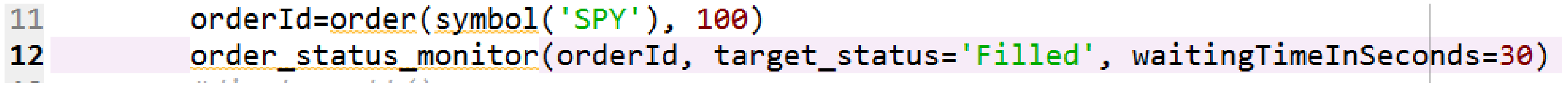
Place orders

- Place market orders: order(symbol('SPY'), 100)
  - Place an market
  - The target security is SPY
  - The action is to BUY 100 shares when n > 0
  - Negative number means SELL, -100 = SELL 100 shares
  - order_target(symbol('SPY'), 100) will adjust positions based on your holding positions by either BUY or SELL until you hold 100 shares
- Place Limit/Stop orders
  - order(symbol('SPY'), 100, LimitOrder(213.42)) place a limit order to BUY 100 shares of SPY at price = $213.42 per share*
  - order(symbol('SPY'), -100, StopOrder(213.42)) place a stop order to SELL 100 shares of SPY at price = $213.42 per share*

*When the limit price or stop price is reached, the orders will be filled at the best available price

Place orders (2)

- It is highly recommended to follow up on the status of the order you placed by order_status_monitor()

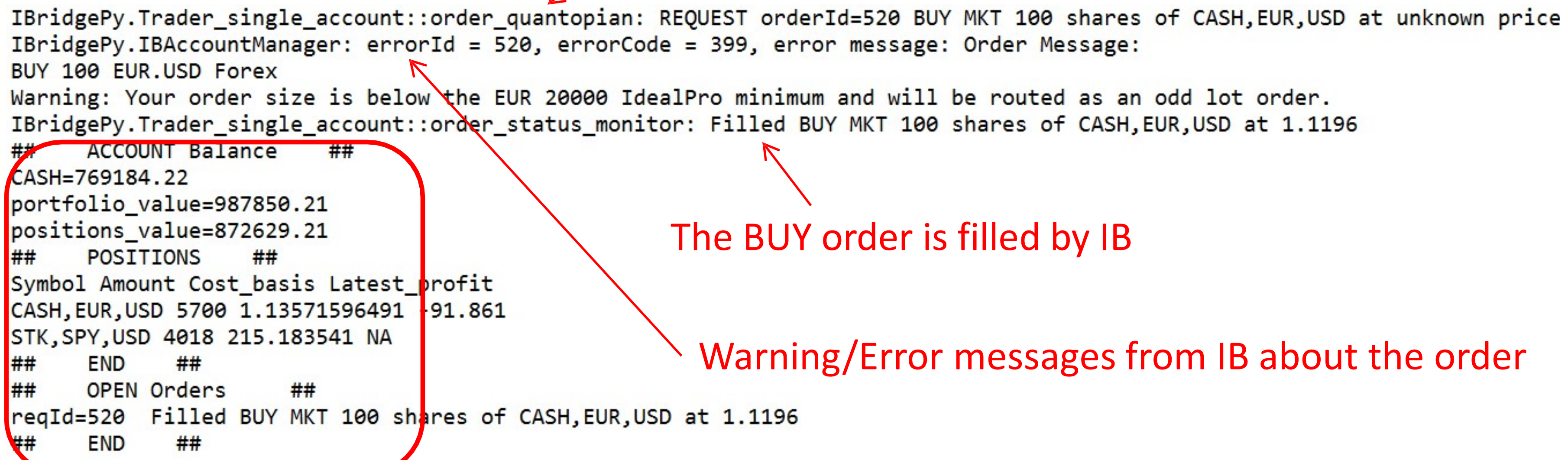


```
        orderId=order(symbol('SPY'), 100)
        order_status_monitor(orderId, target_status='Filled', waitingTimeInSeconds=30)
```

  - orderId is the unique identity of your order requests
  - For market orders, you should expect ‘Filled’ as the ending point of your order request, which means the orders have been executed.
  - For limit and stop orders, the expected status is ‘Submitted’, which means the orders have been accepted by IB and waiting for executions
  - For highly liquidity securities, it won’t take too long (a few micro seconds) to complete the transactions.

Place orders (3)

You request to place a BUY order



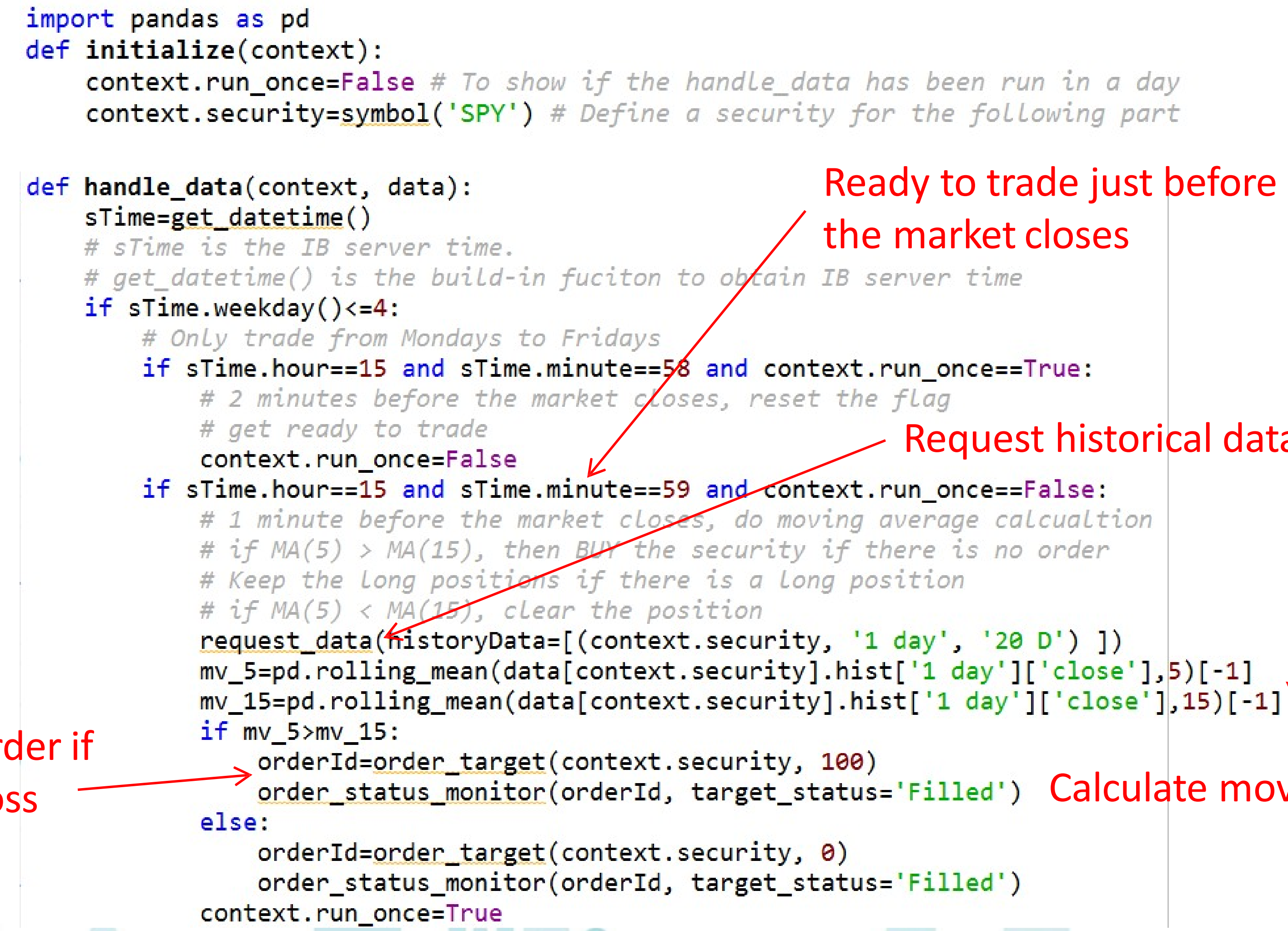
```
IBridgePy.Trader_single_account::order_quantopian: REQUEST orderId=520 BUY MKT 100 shares of CASH,EUR,USD at unknown price
IBridgePy.IBAccountManager: errorId = 520, errorCode = 399, error message: Order Message:
BUY 100 EUR.USD Forex
Warning: Your order size is below the EUR 20000 IdealPro minimum and will be routed as an odd lot order.
IBridgePy.Trader_single_account::order_status_monitor: Filled BUY MKT 100 shares of CASH,EUR,USD at 1.1196
##     ACCOUNT Balance     ##
CASH=769184.22
portfolio_value=987850.21
positions_value=872629.21
##     POSITIONS     ##
Symbol Amount Cost_basis Latest_profit
CASH,EUR,USD 5700 1.13571596491 -91.861
STK,SPY,USD 4018 215.183541 NA
##     END     ##
##     OPEN Orders      ##
reqId=520  Filled BUY MKT 100 shares of CASH,EUR,USD at 1.1196
##     END     ##
```

The BUY order is filled by IB

Warning/Error messages from IB about the order

Account summary after the order is filled

Moving average crossing



```
import pandas as pd
def initialize(context):
    context.run_once=False # To show if the handle_data has been run in a day
    context.security=symbol('SPY') # Define a security for the following part

def handle_data(context, data):
    sTime=get_datetime()
    # sTime is the IB server time.
    # get_datetime() is the build-in fuciton to obtain IB server time
    if sTime.weekday()<=4:
        # Only trade from Mondays to Fridays
        if sTime.hour==15 and sTime.minute==58 and context.run_once==True:
            # 2 minutes before the market closes, reset the flag
            # get ready to trade
            context.run_once=False
        if sTime.hour==15 and sTime.minute==59 and context.run_once==False:
            # 1 minute before the market closes, do moving average calcualtion
            # if MA(5) > MA(15), then BUY the security if there is no order
            # Keep the long positions if there is a long position
            # if MA(5) < MA(15), clear the position
            request_data(historyData=[(context.security, '1 day', '20 D') ])
            mv_5=pd.rolling_mean(data[context.security].hist['1 day']['close'],5)[-1]
            mv_15=pd.rolling_mean(data[context.security].hist['1 day']['close'],15)[-1]
            if mv_5>mv_15:
                orderId=order_target(context.security, 100)
                order_status_monitor(orderId, target_status='Filled')
            else:
                orderId=order_target(context.security, 0)
                order_status_monitor(orderId, target_status='Filled')
            context.run_once=True
```

Ready to trade just before the market closes

Request historical data

Calculate moving average

Place order if MAs cross

Manage multiple accounts

- **Able to handle multiple accounts**

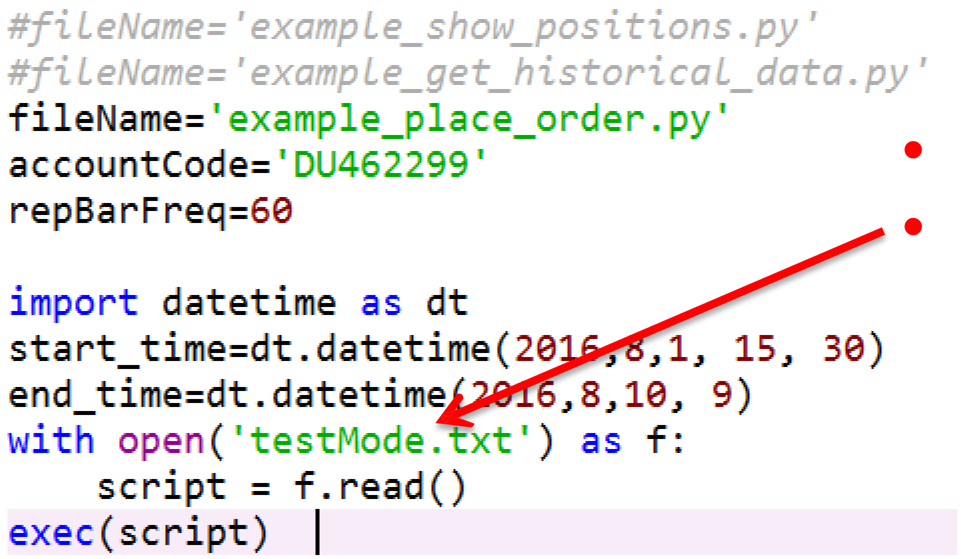
A very useful feature for fund managers

In the following example, a signal triggers BUY 100 shares in account 1 and BUY 500 shares in account 2

```
if mv_5>mv_15:
    orderId1=order_target(context.security, 100, ACCOUNT1)
    orderId2=order_target(context.security, 500, ACCOUNT2)
    order_status_monitor(orderId1, target_status='Filled')
    order_status_monitor(orderId2, target_status='Filled')
else:
    orderId1=order_target(context.security, 0, ACCOUNT1)
    orderId2=order_target(context.security, 0, ACCOUNT2)
    order_status_monitor(orderId1, target_status='Filled')
    order_status_monitor(orderId1, target_status='Filled')
context.run_once=True
```

Backtest strategies

- Run IBridePy in test mode
  - Download daily data from yahoo finance
  - Simulate IB server to process market orders



```
#fileName='example_show_positions.py'
#fileName='example_get_historical_data.py'
fileName='example_place_order.py'
accountCode='DU462299'
repBarFreq=60

import datetime as dt
start_time=dt.datetime(2016,8,1, 15, 30)
end_time=dt.datetime(2016,8,10, 9)
with open('testMode.txt') as f:
    script = f.read()
exec(script)
```

- Debug your python code
- Simple Backtester

Thank You!

If you need help, WE are here !

IBridgePy@gmail.com; www.IBridgePy.com

contact@quantinsti.com; www.quantinsti.com

We have highly experienced in implementing various trading algorithms quickly

Take the Next Step with EPAT™

Professionals from 30+ countries have benefited from EPAT™.
If you want to learn Python to trade through IB's API, then enroll for **EPAT™** now!

Executive Programme in Algorithmic Trading™

Get Trained by Industry Experts and Build a Successful Career in Algo!

Register here for the next batch **on Algorithmic Trading**: www.quantinsti.com/epat
Starting Date: November 12, 2016
Duration: 16 Weekends, 100+ hours of live sessions

Follow us on

Quantinsti Quantinsti Quantinsti Quantinsti

An introduction to Interactive Brokers

Presented by Mr. Ankit Shah, Director of Sales – Interactive Brokers India Pvt Ltd

Interactive Brokers

IB Group was founded over 39 years ago, currently headquartered in Greenwich, Connecticut, USA

IB Group affiliates have offices in the USA, Switzerland, Canada, Hong Kong, UK, Australia, Hungary, Russia, Japan, India, China and Estonia

IB Group affiliates are regulated by the SEC, FINRA, NYSE, FCA and other regulatory agencies around the world

IB Group affiliates execute nearly 1 million trades per day
and have more than 900 employees

Our mission remains **unchanged**, "Create technology to provide liquidity on better terms. Compete on price, speed, size, diversity of global products and advanced trading tools."

Interactive Brokers

OVER US$ 5 BILLION IN EQUITY CAPITAL
STRONG CAPITAL POSITION

BBB+ RATING FROM S&P WITH A
STABLE OUTLOOK

REAL TIME MARGINING SYSTEM
CONSERVATIVE RISK APPROACH

NO CDO'S MBS OR CDS
CONSERVATIVE BALANCE SHEET

IBG STRENGTH & SECURITY

84.3% OF THE COMPANY STILL OWNED BY OUR EMPLOYEES AND AFFILIATES
SOLID VESTED INTEREST

SOLID POSITIVE EARNINGS FOR 20 CONSECUTIVE YEARS
DEDICATION

For more information see ibkr.com/strength

Interactive Brokers

TRADE ON OVER 100 MARKET CENTRES ACROSS 24 COUNTRIES!

GLOBAL ACCESS

STOCKS OPTIONS FUTURES FX BONDS ETFs **WORLDWIDE**

Interactive Brokers

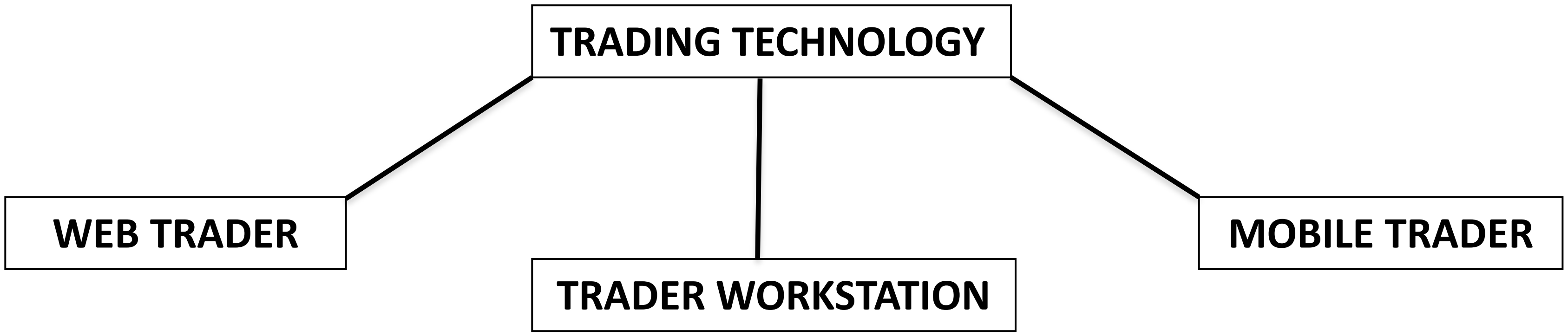
Optimize your trading speed and efficiency with our Trading Platforms

Advanced Trading Tools

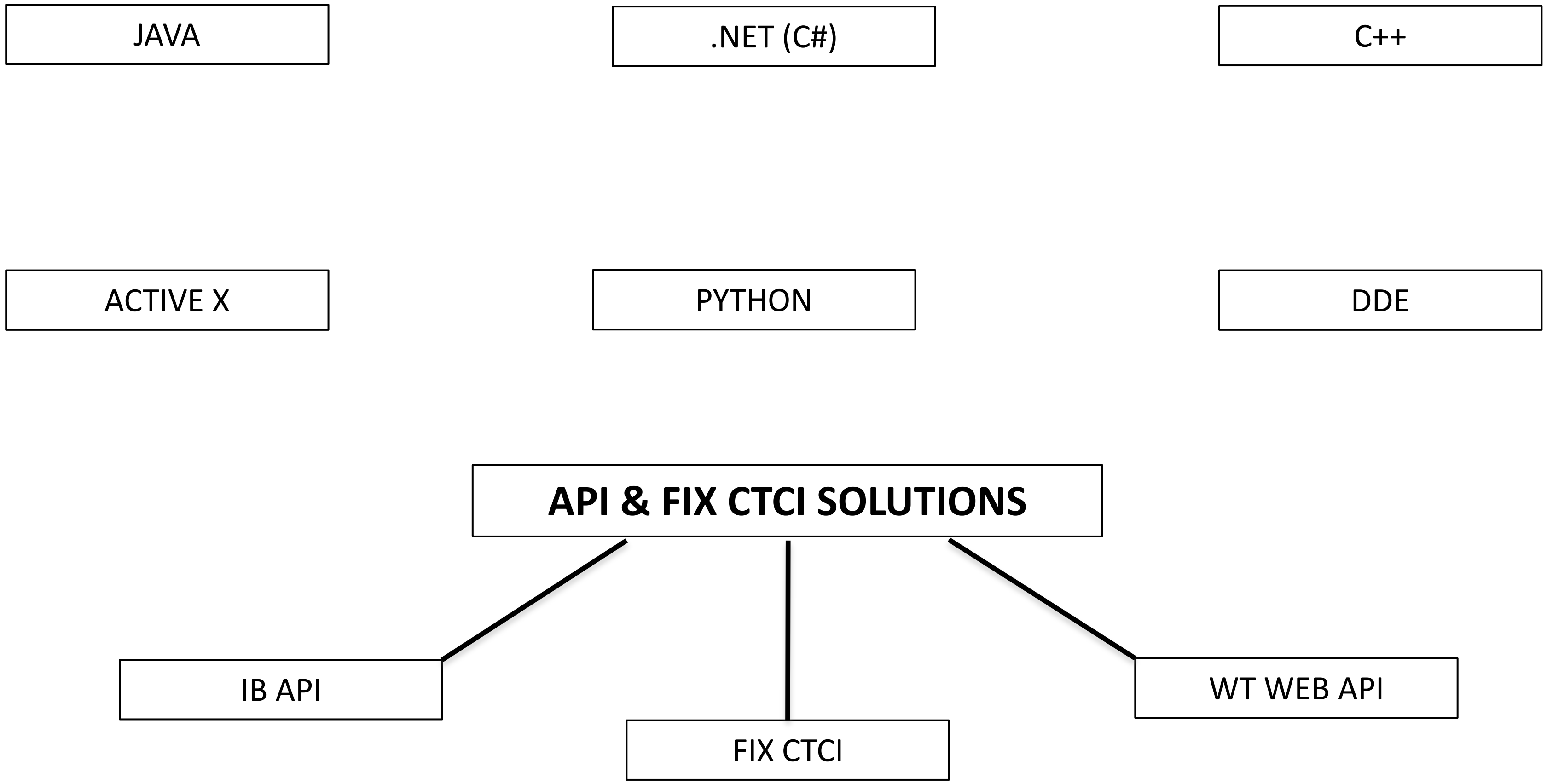
Sophisticated Risk Management

Over 60 order Types and Algos

Comprehensive Reporting



Interactive Brokers



Interactive Brokers

FLAT **US$ 0.005** PER SHARE – US EQUITIES

US OPTIONS BROKERAGE STARTING AT **US$ 0.7** PER CONTRACT

LOW COST & BEST EXECUTION*

FLAT BROKERAGE OF **0.08%** OF TRADE VALUE FOR EQUITIES IN JAPAN, HK, AUSTRALIA AND SINGAPORE

FLAT **US$ 0.85** PER CONTRACT FOR US FUTURES AND FOP'S (Plus Exchange and Regulatory Fees)**

*Lower rates available for high volume traders under IB's tiered commission structure.

** Commissions for Globex E-Mini FX Futures is USD 0.50 per contract and Globex E-Micro FX futures is USD 0.15 per contract

Interactive Brokers

AWARDS

2016 WSL Awards
INSTITUTIONAL TRADING
WINNER

2016 Winner
"Best Options Trading Platform"

BARRON'S

Rated a Top Online Broker for the Sixth Consecutive Year by Barron's 2016

HFMWEEK
US Hedge Fund Services Awards 2016
Winner

2016 Winner
"Best prime broker – startups"

For more information see ibkr.com/awards

Interactive Brokers

THANK YOU!

Mr. Ankit Shah
Director of Sales
Interactive Brokers – India
Tel: +91 22 61289836
E-mail: ashah@interactivebrokers.com

Interactive Brokers

Interactive Brokers and Python – an Innovative Approach to Algo Trading and Financial Data Mining

IB API

Build a complete trading application that connects to our advanced order routing and trading system using our IB Application Programming Interface (API).

Our dedicated API support team is ready to help you with your IB API and FIX CTCI questions.

IB Account Offerings

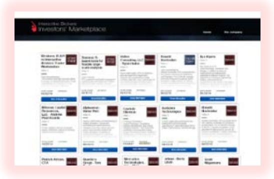
- **Lowest Margin Rates and Commissions**[1]
- **Security Financing** real-time depth of availability and indicative rates help protect against buy-ins and recalls.
- **Safety of Assets** strong balance sheet, large relative equity capital, SIPC and excess SIPC protection[2].
- **Innovative trading technology** trade in over 100 market centers in 24 countries, direct market access to stocks, options, futures, forex, bonds, ETFs and CFDs from a single account
- **Speed of Execution and Risk Management**

Python

Use Python's robust algo trading programing capability in conjunction with your IB API trading application.

Investors' Marketplace

Connect with an established **Python** provider for custom solutions.



IB API Reference Guide
www.ibtweet.com/apidoc

Follow *IB Quant* on LinkedIn:
www.ibtweet.com/quant