Starting Now

‹packt›

From Idea to Execution: A Practical Workshop on Algorithmic Trading with Python

Learn to design, backtest, and automate trading strategies using Python, VectorBT, and Interactive Brokers.

Jason Strimpel

Founder, PyQuant News

Section 00: Your instructor

Jason Strimpel

Founder, PyQuant News

About Jason Strimpel

- Founder of PyQuant News, a leading newsletter on Python and quantitative finance with over 35,000 subscribers.
- Over 20 years of professional trading and risk management experience across hedge funds, banks, and trading firms.
- Former hedge fund trader specializing in systematic and algorithmic strategies.
- Built and led quant development and risk technology teams in global trading businesses across the US, Europe, and Asia.
- Deep expertise in algorithmic trading systems, backtesting frameworks, and factor-based strategies using Python.
- Instructor of *Getting Started With Python for Quant Finance*, helping thousands of retail traders and professionals systematize their trading.
- Actively builds and publishes practical tutorials, research, and tools for quantitative finance practitioners.
- Combines real-world trading experience with hands-on coding to bridge the gap between institutional and retail quant finance.

PyQuant News

About PyQuant News

- Leading newsletter on Python and quantitative finance with over 37,000 subscribers worldwide.
- Publishes a leading course on quant finance, Getting Started With Python for Quant Finance
- Provides practical, code-driven tutorials on algorithmic trading, backtesting, and quantitative research.
- Bridges the gap between academic finance theory and institutional-grade trading practices.
- Covers Python tools, modern data science libraries, factor models, and risk management workflows.
- Trusted by both retail traders and finance professionals to systematize and improve their strategies.
- Founded and authored by Jason Strimpel, former hedge fund trader and risk manager with 20+ years of global experience.
- Mission: democratize quant finance by giving anyone the skills to build, test, and deploy strategies using Python.

Agenda

01 Introduction to Algorithmic Trading

02 What is Edge and How to Find it

03 The Python Quant Stack

04 The Algorithmic Trading Workflow

05 Backtesting Strategies the Right Way

06 Break

Agenda

07 Prototyping the Crack v. Refiner Trade

08 Backtesting the Strategy with VectorBT

09 Building an Algo Trading App with IBKR

10 Where to Improve the Strategy

11 Questions and Resources

Section 01: Introduction to Algorithmic Trading

What is Algorithmic Trading?

A computer algorithm that follows a **set of instructions** to place trades to buy or sell **when an asset's price is not what (we think) it should be**.

Some key points:

- Not the same as high frequency trading
- Can be passive portfolio management
- Different from discretionary trading
- Does not require dozens of strategies

Type 1: Systems Based on Data Mining

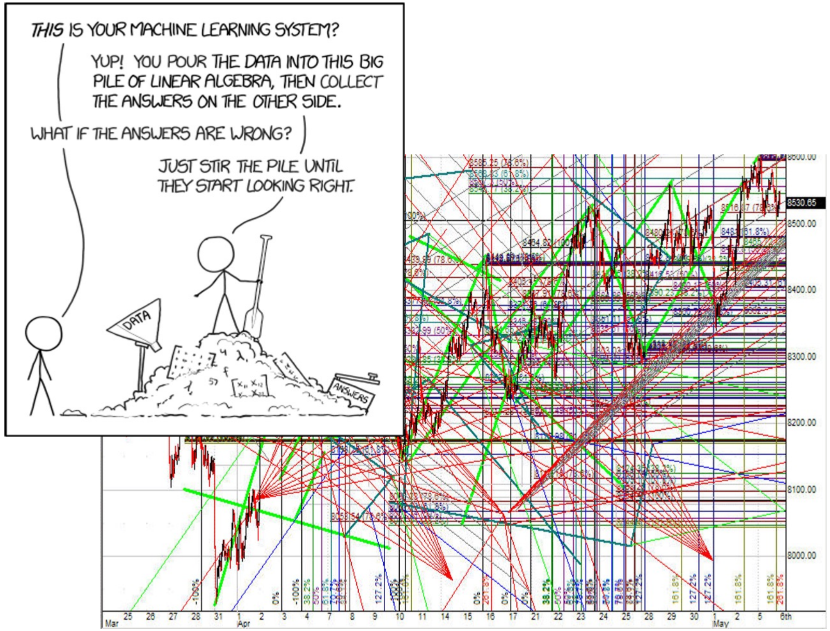
Advantages:

- Market data is abundant
- Computing is relatively cheap
- Avoid the hard work of hypothesis formation

Disadvantages:

- Prone to overfitting
- Vanishing edge
- No economic rationale

This **is not** what we cover in this course!


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS
8530.65
8500.00
8400.00
8300.00
8200.00
8100.00
8000.00
Mar
Apr
May

Munroe, Randall. *"Machine Learning."* xkcd, no. 1838, 2017. https://xkcd.com/1838/

Type 2: Systems that Model Market Inefficiencies

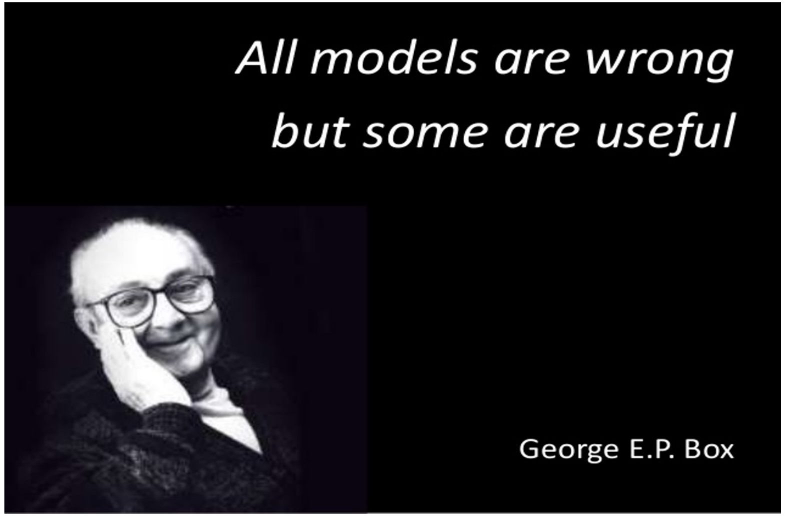
Advantages:

- Edge is more persistent
- Robust to parameter changes
- Strategies less sensitive to overfitting

Disadvantages:

- Very little signal
- More difficult to model and build
- Impossible to prove (or disprove)

This **is** what we cover in this course.



All models are wrong
but some are useful

George E.P. Box

Section 02: What is Edge and How to Find It

What are Market Inefficiencies?

An **asset mispricing** that periodically and consistently appears in the market.

This is where we find edge:

- Occasionally present themselves against the randomness of the market
- Based on trader psychology, economics, microstructure, or company events
- Our job to detect, predict, and trade these patterns

*Our **edge** comes from the ability to consistently buy “cheap” and sell “expensive” over and over again.*

What is Edge?

Positive expected value derived from exploiting a market inefficiency.

Said a different way:

- Probability-weighted value of all payoffs summed over all possible outcomes
- The return you expect to realize if you execute an infinite number of trades
- A single trade can win or lose but over the long run you make money

The best edges are explainable, which helps with risk management and portfolio construction.

How much would you pay to play?

I propose a game:

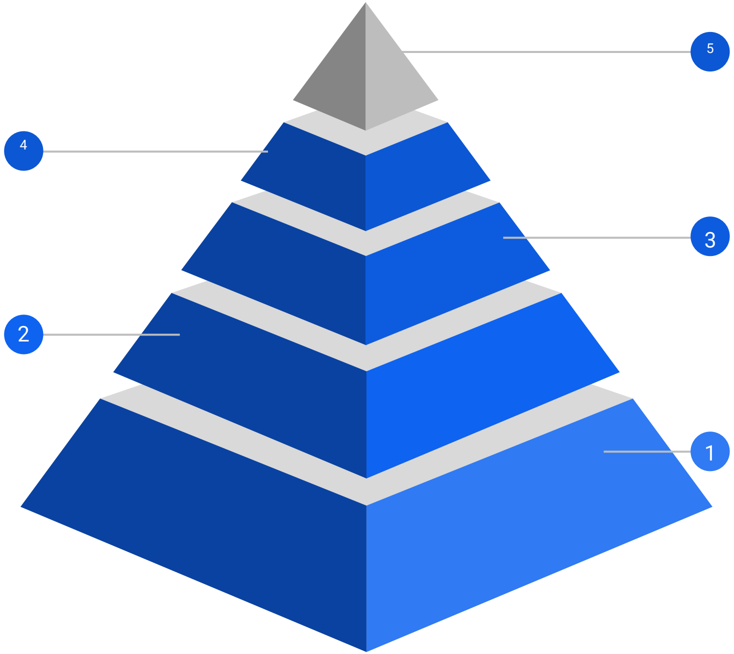
1. Roll a six-sided die
2. You get paid $1,000 if it lands on 5 or 6
3. You get paid $0 if it lands on anything else

How much would you pay to play?

The fair value is $333.34

Roll	Probability	Payoff	Weighted Payoff
1	16.7%	$0	$0
2	16.7%	$0	$0
3	16.7%	$0	$0
4	16.7%	$0	$0
5	16.7%	$1,000	$166.67
6	16.7%	$1,000	$166.67

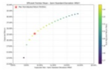
Section 03: The Python Quant Stack



Pricing & Optimization

4

QuantLib, Riskfolio-Lib



Numerical Computing & Statistics

2

NumPy, SciPy, Statsmodels, Scikit-Learn

Backtesting & Trading

5

VectorBT, Zipline Reloaded Ecosystem

Data Acquisition & Manipulation

3

OpenBB Platform, Pandas

Python

1

The Language, Anaconda Distribution,
Package Managers, Modules, IDEs

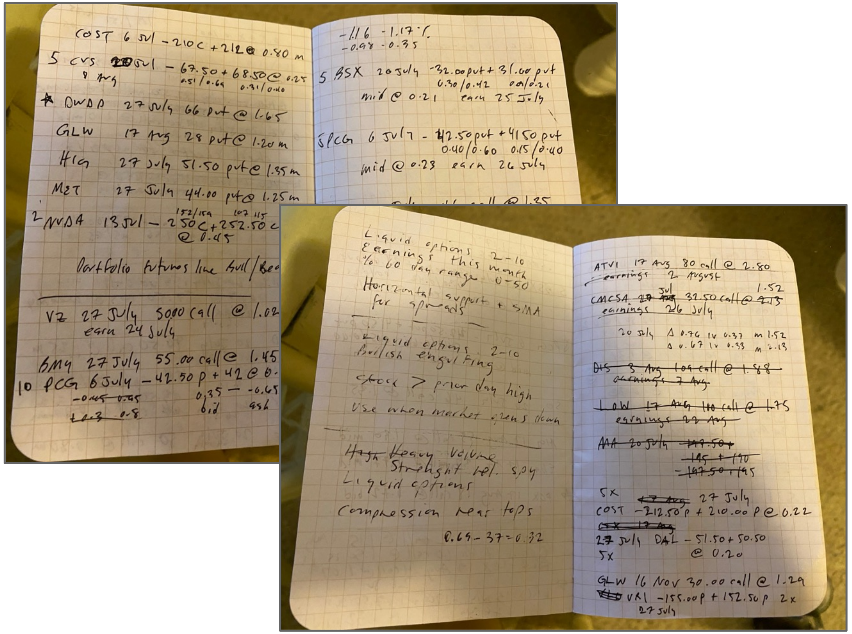
Section 04: The Algorithmic Trading Workflow

Step 1: Ask a Question (Define the Trading Problem)

Observe the market and economy and generate ideas for market inefficiencies.

Ideas come from many places:

1. Different expressions of the same hypothesis
2. Illiquid or mispriced exchange traded funds
3. Temporary dislocations of normal relationships
4. Geopolitical shocks
5. Supply and demand imbalances
6. Company announcements



COST 6 Jul -210c +212 @ 0.80 m
5 CVS ~~20~~ Jul - 67.50 + 68.50 @ 0.25
8 Aug 0.51/0.69 0.31/0.40
★ DWDP 27 July 66 put @ 1.65
GLW 17 Aug 28 put @ 1.20 m
HIG 27 July 51.50 put @ 1.35 m
MET 27 July 44.00 put @ 1.25 m
2 NVDA 13 Jul - 250 c + 252.50 c
152/159 107 115
@ 0.45
Portfolio futures line Bull/Bea

VZ 27 July 50.00 call @ 1.02
earn 24 July
BMY 27 July 55.00 call @ 1.45
10 PCG 6 July - 42.50 p + 42 @ 0.
~~-0.45 0.45~~ 0.35 - -0.65
~~+0.3 0.8~~ bid ask

-1.16 -1.17 %
-0.98 -0.35
5 BSX 20 July -32.00 put + 31.00 put
0.30/0.42 0.0/0.21
mid @ 0.21 earn 25 July
5 PCG 6 July - 42.50 put + 41.50 put
0.40/0.60 0.15/0.40
mid @ 0.23 earn 26 July

Liquid options 2-10
Earnings this month
% 60 day range 0-50
Horizontal support + SMA
for spreads

Liquid options 2-10
Bullish engulfing
Stock > prior day high
Use when market opens down

~~High~~ Heavy Volume
Strenght rel. spy
Liquid options
Compression near tops
0.69 - 37 = 0.32

ATVI 17 Aug 80 call @ 2.80
~~earnings~~ 2 August
CMCSA ~~27 Aug~~ Jul 32.50 call @ ~~2.13~~ 1.52
earnings 26 July
20 July Δ 0.76 IV 0.37 m 1.52
Δ 0.67 IV 0.33 m 2.13
~~DIS 3 Aug 109 call @ 1.88~~
~~earnings 7 Aug~~
~~LOW 17 Aug 110 call @ 1.75~~
~~earnings 22 Aug~~
~~AAA 20 July 147.50 +~~
~~145 + 190~~
~~-147.50 + 145~~
5x ~~17 Aug~~ 27 July
COST -212.50 p + 210.00 p @ 0.22
~~5x 17 Aug~~
27 July DAL - 51.50 + 50.50
5x @ 0.20
GLW 16 Nov 30.00 call @ 1.29
VRL -155.00 p + 152.50 p 2x
27 July

Step 2: Conduct Background Research

Use Minimum Viable Python to collect data and work to disprove your evidence.

Get intuition behind your idea:

Study the economic fundamentals of the strategy

Analyze how the relationships behaved in the past

NumPy

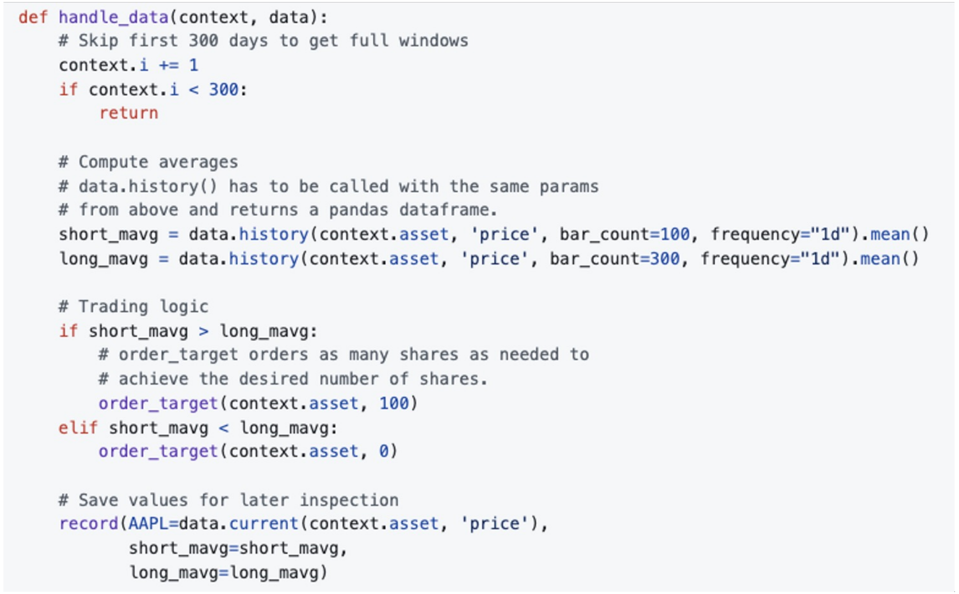
SciPy

pandas

Step 3: Formulate a Hypothesis (Define Strategy Rules)

Design the strategy to generate trade signals based on the economic rationale.

"A widening crack spread indicates higher refining profitability, potentially leading to increased crude oil demand and upward pressure on crude oil prices."



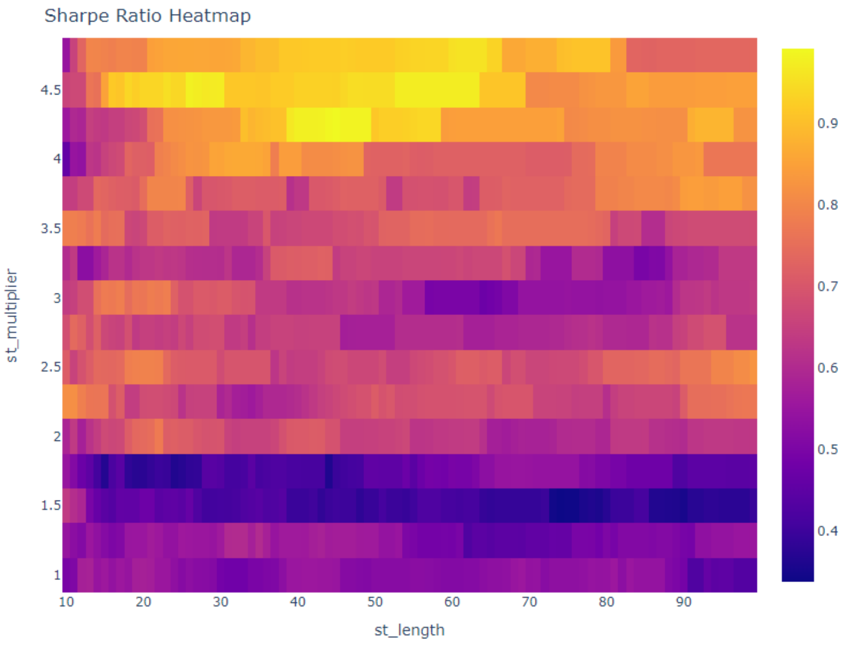
```
def handle_data(context, data):
    # Skip first 300 days to get full windows
    context.i += 1
    if context.i < 300:
        return

    # Compute averages
    # data.history() has to be called with the same params
    # from above and returns a pandas dataframe.
    short_mavg = data.history(context.asset, 'price', bar_count=100, frequency="1d").mean()
    long_mavg = data.history(context.asset, 'price', bar_count=300, frequency="1d").mean()

    # Trading logic
    if short_mavg > long_mavg:
        # order_target orders as many shares as needed to
        # achieve the desired number of shares.
        order_target(context.asset, 100)
    elif short_mavg < long_mavg:
        order_target(context.asset, 0)

    # Save values for later inspection
    record(AAPL=data.current(context.asset, 'price'),
           short_mavg=short_mavg,
           long_mavg=long_mavg)
```

Step 4: Design and Conduct an Experiment

Backtest the strategy which is a **simulation of past market dynamics** to assess how a trading strategy might perform in the future.

More than just PnL:

Significance of the strategy
Sensitivity to parameter changes
Impact of commissions and slippage

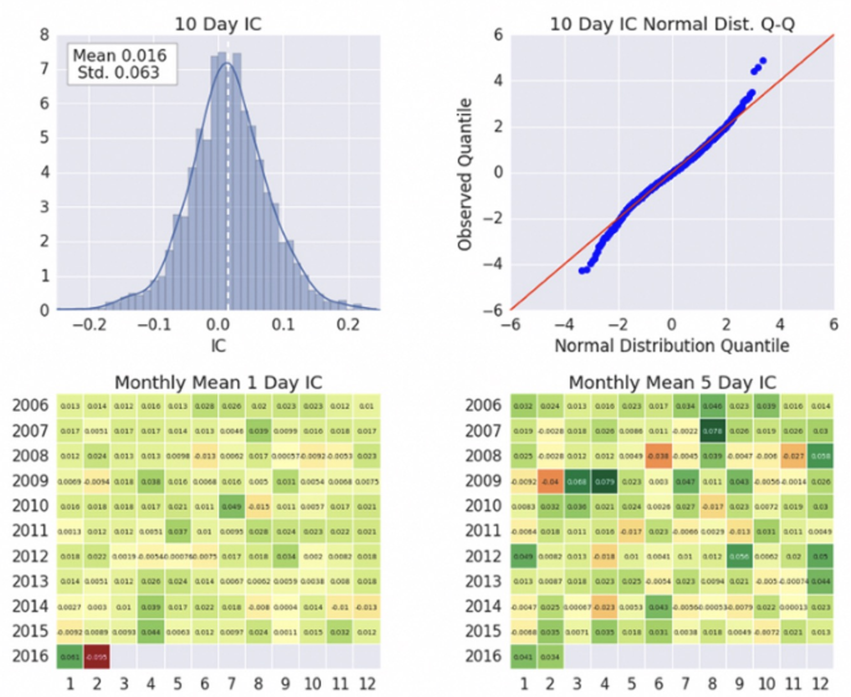


Step 5: Collect and Analyze Data

Backtests generate data on a strategy performance which is used to assess the statistical significance of a strategy and its dynamics.

Risk and performance analytics:

- Transactional-based metrics
- Returns-based metrics
- Factor-based metrics

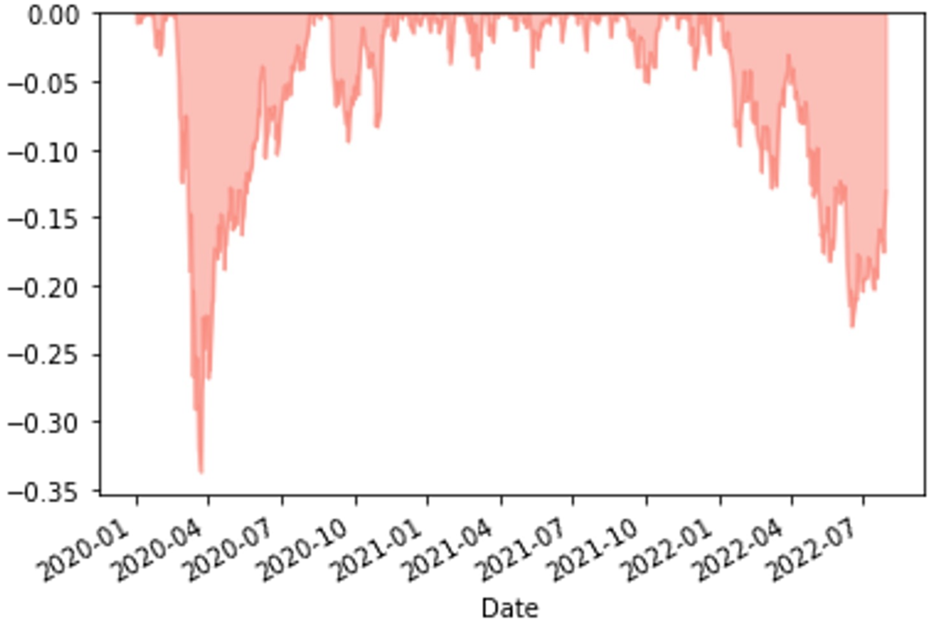


Step 6: Draw Conclusions (Validate the Strategy)

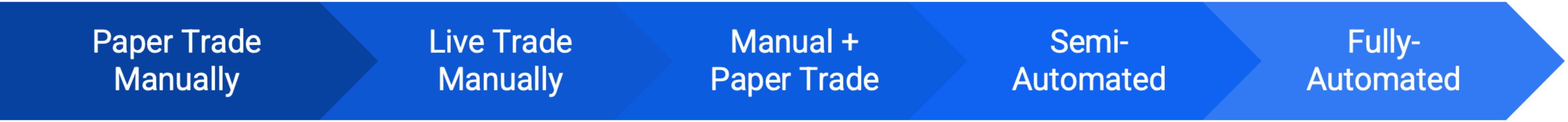
Your strategy should be statistically significant and should meet risk parameters you're comfortable with.

Does the strategy work for your:

- Risk-adjusted returns
- Drawdown duration
- Volatility and win rate



Step 7: Execution



Paper Trade Manually	Live Trade Manually	Manual + Paper Trade	Semi-Automated	Fully-Automated
1 month	3-6 months	1 month	3-6 months	∞

Section 05: Backtesting Strategies the Right Way

What is a Backtest?

A **simulation of past market dynamics** to assess how a **trading strategy might perform in the future**.

More than just PnL:

- Significance of the strategy
- Sensitivity to parameter changes
- Impact of commissions and slippage

A backtest is a framework to rigorously test the features of strategy.

A thought exercise…

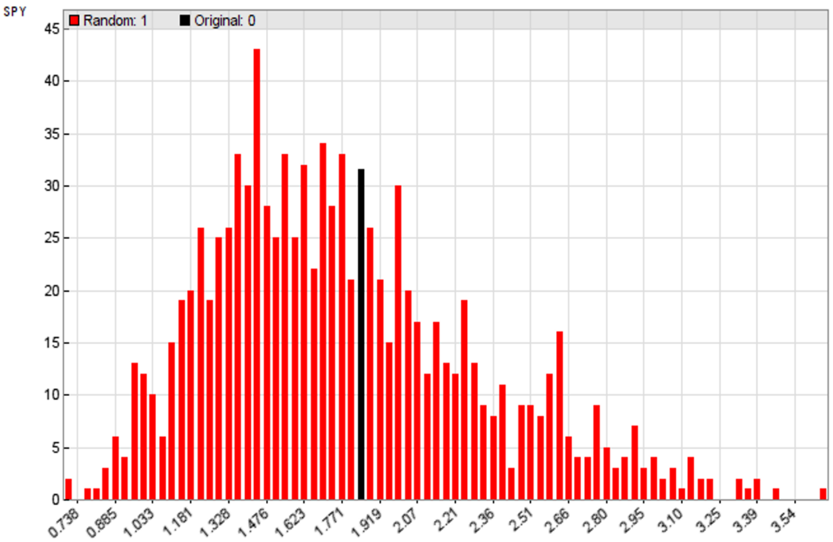
Imagine you have a strategy…

The gods of trading know your strategy randomly produces a positive PnL 50% of the time…

You run a backtest with negative results…

Optimize…

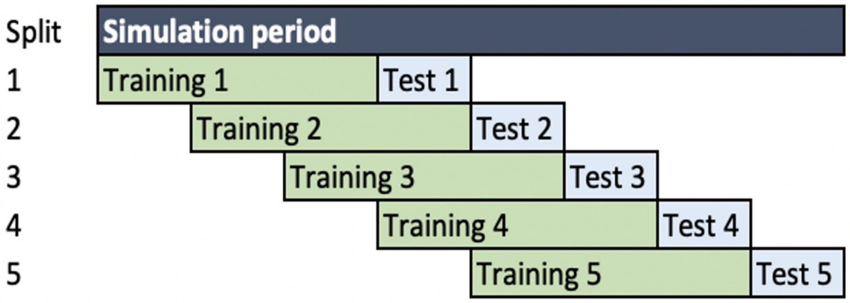
Until you get positive results…



Walk Forward Optimization (WFO)

A method of testing and optimizing a trading strategy by dividing historical data into sequential in-sample and out-of-sample periods.

WFO is the same as cross-fold validation and is the state-of-the-art method for reducing the risk of overfitting.



Section 06: Prototyping the Crack v. Refiner Spread Trade

Spread Trades

Goal: Take a position in one or more assets that deviate from a historically consistent relationship.

Why is this important?

- Considered market neutral
- Natural hedge
- Linked to economics

Refiner Crack Spread

What is it: Captures the price differential between crude oil and its refined products which is used as a proxy for refiner margins.

The strategy: Buy the refiner stock when it sufficiently deviates from the crack spread.

Refiners purchase crude oil and sell its refined products. When the spread between the two widens, the refiner's profit margin inherently expands. Consequently, refiner stocks should move in tandem with crack spreads

pandas

pandas is a Python library for data manipulation and analysis, providing data structures like Series and DataFrame.

Why is pandas important?

- Enables efficient data manipulation
- Essential for cleaning messy datasets
- Supports various data formats
- Integrates well with data science tools
- Widely used for data analysis tasks

Let's code!

Section 07: Backtesting the Strategy with VectorBT

What is a Vector-Based Backtesting Framework?

A framework that applies **vectorized operations** to historic data all at the same time.

Pros:

- Fast computations with vectorized operations
- Easily scalable for large datasets
- Efficient multi-asset backtesting capabilities

Cons:

- Struggles with complex execution logic
- High memory demand for datasets
- Less flexible for custom events

What is a Vector-Based Backtesting Framework?

Feature	Vector-Based	Event-Based
Processing Method	Vectorized operations	Sequentially
Computational Efficiency	Highly efficient, especially for large datasets	Less efficient due to the need to process each event individually
Complexity in Coding	Simpler and more concise	More detailed programming to simulate market events accurately
Suitability for Strategy Type	Without complex state or path dependencies.	Ideal for complex, state-dependent, and path-dependent strategies
Order Execution Modeling	Less accurate for modeling real market conditions like slippage and delays.	More accurately models order execution dynamics, including delays, slippage, and partial fills.
Risk Management Simulation	Basic risk management features; may not capture dynamic risk adjustments effectively.	Detailed risk management with realistic simulations of stop-losses and other real-time adjustments.
Scalability	Highly scalable for large-scale data analysis and multiple assets.	Challenging with large-scale tick-by-tick data due to computational demands.
Risk of Overfitting	Higher risk of overfitting	Risk exists but can be better managed
Development Cycle	Faster development and testing cycles due to computational efficiency and simpler code.	Potentially slower due to increased complexity and computational demands.
Realism	Less realistic as it does not simulate the sequential flow of market events.	Higher realism in simulating real-world market conditions and trader actions.

But you said I shouldn’t optimize parameters!

Testing different combinations of input parameters to identify the most effective configuration for maximizing performance metrics while maintaining robustness.

Consider optimizing:

- Stop-loss levels, position sizing, take-profit levels
- Timing parameters, signal thresholds

Parameters independent of random time series, like prices

Do not optimize:

- Complex combinations of indicators
- Purely price-dependent indicators

Parameters dependent on random time series, like prices

Let's code!

Section 08: Building an Algorithmic Trading App with Interactive Brokers

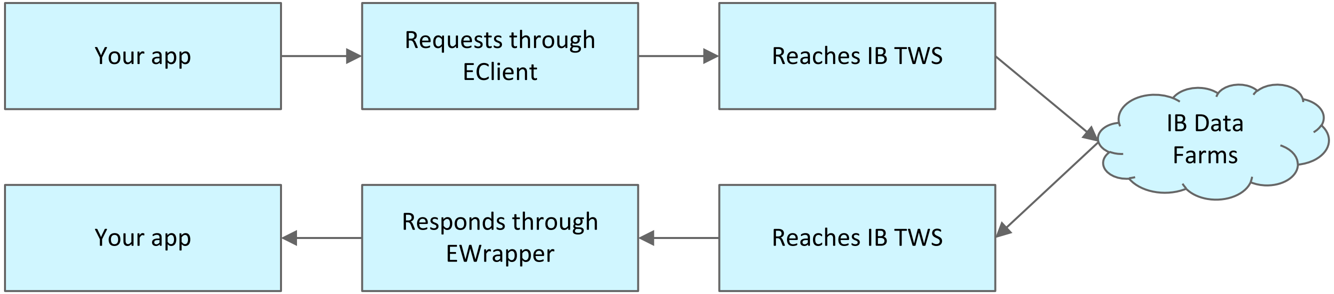
Why Interactive Brokers (IB)?

Interactive Brokers offers access to global markets, advanced trading tools, and competitive pricing for individual and institutional investors.

Why we ❤️ IB:

1. Access to over 150 markets across 33 countries
2. Low commissions and competitive fees
3. Advanced trading tools and analytics
4. Direct exchange access (i.e. does not sell order flow)
5. Integrated APIs for programmatic trading

The Interactive Brokers API Architecture



Let's code!

Section 09: Where to Improve Execution of the Strategy

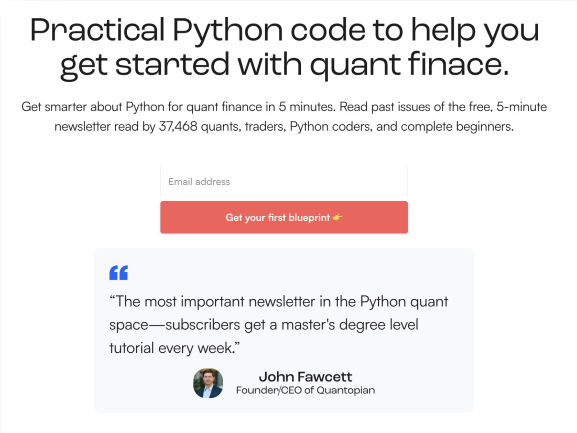
Areas for Improvement in the Trading Logic

1. **Avoiding Repeated Data Requests in Tight Loops**
   - Current Issue: The code continuously requests historical data in a tight loop without any delay or checks for significant market changes, leading to excessive data requests that could overwhelm the API or violate rate limits.
   - Improvement: Implement a more strategic approach to data fetching, such as incorporating a reasonable time delay between requests or triggering data requests based on market events or specific intervals. This can help reduce unnecessary API calls and enhance system performance.

2. **Improving Breakout Confirmation Logic**
   - Current Issue: The strategy triggers trades immediately upon detecting a breakout above or below the channel bands, which can lead to false signals, especially in volatile markets where prices frequently touch or briefly exceed the bands.
   - Improvement: Add breakout confirmation criteria, such as waiting for a candle close outside the bands, using additional indicators (e.g., volume, RSI), or checking if the breakout holds for a certain number of periods. This would help filter out false signals and improve trade accuracy.

3. Refining Trade Execution and Position Management
   - Current Issue: The current logic executes market orders immediately upon breakout detection without considering the existing position, risk limits, or trade size adjustments. This could lead to overtrading or violating risk management rules.
   - Improvement: Introduce position management logic that checks current positions before placing new trades, and incorporate risk management measures such as position sizing, stop-loss, and take-profit orders. This approach would help control risk, prevent overexposure, and enhance overall strategy robustness.

Section 10: Questions and Resources



Practical Python code to help you get started with quant finace.

Get smarter about Python for quant finance in 5 minutes. Read past issues of the free, 5-minute newsletter read by 37,468 quants, traders, Python coders, and complete beginners.

> “The most important newsletter in the Python quant space—subscribers get a master's degree level tutorial every week.”
>
> John Fawcett
> Founder/CEO of Quantopian

The PyQuant Newsletter

Free Python Code Tutorials for Algorithmic Trading

Start using Python with a course that keeps you motivated

A complete system for **learning Python from scratch**. Set up the Python Quant Stack and start automating your trading strategies.

No more errors. No more frustration. Everything in one place.

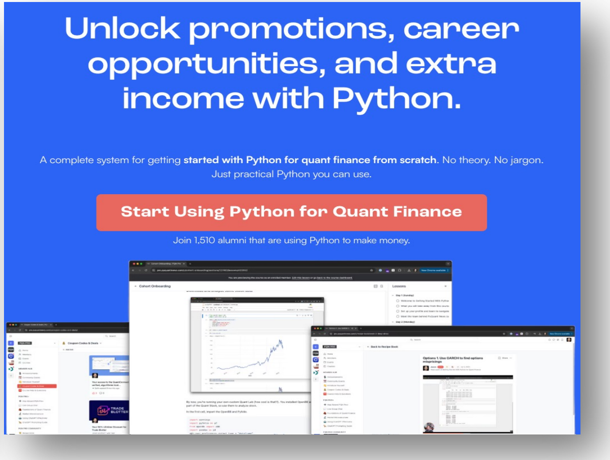
★★★★★

"This was the best course I have taken in my 28 year career in finance."

Zarko
Finance Professional

Python Foundations

A complete system for learning Python from scratch.



Unlock promotions, career opportunities, and extra income with Python.

A complete system for getting **started with Python for quant finance from scratch**. No theory. No jargon. Just practical Python you can use.

Join 1,510 alumni that are using Python to make money.

Getting Started With Python for Quant Finance

A complete step-by-step roadmap for learning Python for trading.

Do things at your own pace.
Life's not a race.

- Shawn
Garcia

Thank You