Python Backtesting Frameworks for Options Trading Evaluation Report

Introduction

In the realm of options trading, having a robust backtesting framework is essential for testing and refining strategies in a risk-free environment. While Python offers many libraries for backtesting equities and futures, fewer are tailored to the unique complexities of options. This report evaluates four Python-based backtesting frameworks—Backtrader, QuantConnect, Optopsy, and a Custom Pandas Simulation—focusing on their suitability for options trading based on ease of use, feature set, flexibility, and community support.

Backtrader

a. Ease of use and documentation quality

Backtrader is relatively beginner-friendly for equity strategies, thanks to its modular class-based design and accessible documentation. It comes with many built-in indicators and strategy templates. However, documentation for options trading is limited and lacks detailed examples. Users often need to rely on community forums and GitHub issues for support.

b. Support for options backtesting

Backtrader does not natively support options trading. There is no dedicated options instrument type. Users must manually simulate option pricing and execution. It lacks features like options chains, strike selection, Greeks, and expiration handling. This makes it a poor choice for full-featured options strategy testing unless extended with custom logic.

c. Flexibility and customization options

Backtrader is flexible, allowing for deep customization of indicators, strategies, and brokers. You can implement your own options pricing models, create synthetic instruments, or inject external pricing data, although this adds significant development overhead. Some advanced users simulate option legs by feeding synthetic prices into Backtrader.

d. Community support and ongoing maintenance

Backtrader has a large community of quant enthusiasts and is frequently discussed on forums like Quantopian and Reddit. However, the original repository is not actively maintained, and many users rely on forks. Examples and extensions are often shared across personal blogs and GitHub gists.

Detailed Analysis of Suitability for Options

Backtrader is not well-suited out of the box for options trading. It was built with equities and futures in mind and does not support key options-specific features. However, it can be repurposed for options through extensive customization, such as simulating option legs using custom data feeds or integrating external libraries for Black-Scholes pricing. This makes it suitable only for advanced users comfortable with significant engineering work.

Strengths:

- Excellent for equities and strategies involving indicators or moving averages
- Great modular design for customization

Limitations for Options:

- No built-in support for option chains, Greeks, or multi-leg strategies
- Requires manual engineering of everything options-related

Conclusion: Backtrader is not ideal for options unless you are building a hybrid simulation manually. It is flexible, but not option-aware.

Example Code:

```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
aapl = yf.download('AAPL', period='90d', interval='1d')
if isinstance(aapl.columns, pd.MultiIndex):
    aapl.columns = aapl.columns.droplevel(1)
aapl = aapl[['Open', 'High', 'Low', 'Close', 'Volume']].dropna()
aapl.tail()
```

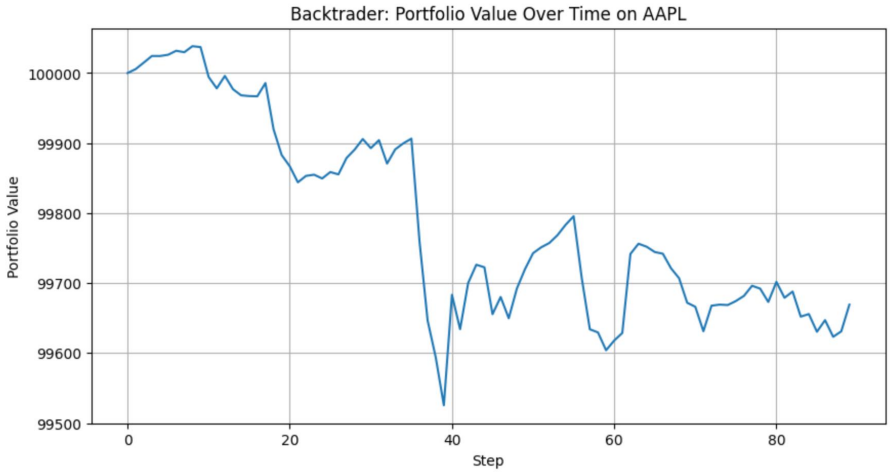
```
/var/folders/c6/z7pdynp14jd16wp1d1c7cvsc0000gn/T/ipykernel_53891/307384540.py:7: FutureWarning: YF.download() has changed argument auto_adjus
t default to True
  aapl = yf.download('AAPL', period='90d', interval='1d')
[*********************100%***********************]  1 of 1 completed
```

Price	Open	High	Low	Close	Volume
Date					
2025-06-13	199.729996	200.369995	195.699997	196.449997	51447300
2025-06-16	197.300003	198.690002	196.559998	198.419998	43020700
2025-06-17	197.199997	198.389999	195.210007	195.639999	38856200
2025-06-18	195.940002	197.570007	195.070007	196.580002	45394700
2025-06-20	198.235001	201.699997	196.859604	201.000000	95316548

```
import backtrader as bt
class ExampleStrategy(bt.Strategy):
    def __init__(self):
        self.portfolio_values = []
    def next(self):
        if self.data.close[0] > self.data.open[0]:
            self.buy()
        else:
            self.sell()
        self.portfolio_values.append(self.broker.getvalue())
data = bt.feeds.PandasData(dataname=aapl.copy())
cerebro = bt.Cerebro()
cerebro.addstrategy(ExampleStrategy)
cerebro.adddata(data)
cerebro.broker.setcash(100000)
results = cerebro.run()
strat = results[0]
```

PnL:

```
plt.figure(figsize=(10, 5))
plt.plot(strat.portfolio_values)
plt.title("Backtrader: Portfolio Value Over Time on AAPL")
plt.xlabel("Step")
plt.ylabel("Portfolio Value")
plt.grid(True)
plt.show()
```



Documentation:

https://www.backtrader.com/docu/

QuantConnect (Lean Engine)

a. Ease of use and documentation quality

QuantConnect offers excellent documentation and many examples tailored to institutional-grade backtesting. While setup requires registration and use of their cloud IDE or local Lean CLI, the learning curve is reasonable due to helpful tutorials. Their Lean engine is open-source and can be run locally for free.

b. Support for options backtesting

QuantConnect has first-class options support, including options chains, expiration filters, strike range selection, option Greeks, and multi-leg strategy simulation.

Historical options data is also accessible via their data API. You can backtest strategies with rolling expiration and realistic slippage and commission modeling.

c. Flexibility and customization options

QuantConnect is extremely powerful, with support for equities, futures, forex, crypto, and options. It allows users to construct complex option strategies using LegHelpers and OptionStrategies objects. Custom indicators and data ingestion are supported through the API.

d. Community support and ongoing maintenance

QuantConnect has an active Slack, forum, and GitHub community. The Lean engine is professionally maintained with regular updates and contributions from developers worldwide.

Detailed Analysis of Suitability for Options

QuantConnect is very well suited for options trading. It offers all essential features for testing sophisticated strategies with multiple legs, rolling expiration, and volatility screens. It is also used by hedge funds and academic researchers.

Strengths:

- Professional-grade options support
- Access to real historical options chains
- Handles margin, slippage, commissions realistically

Limitations:

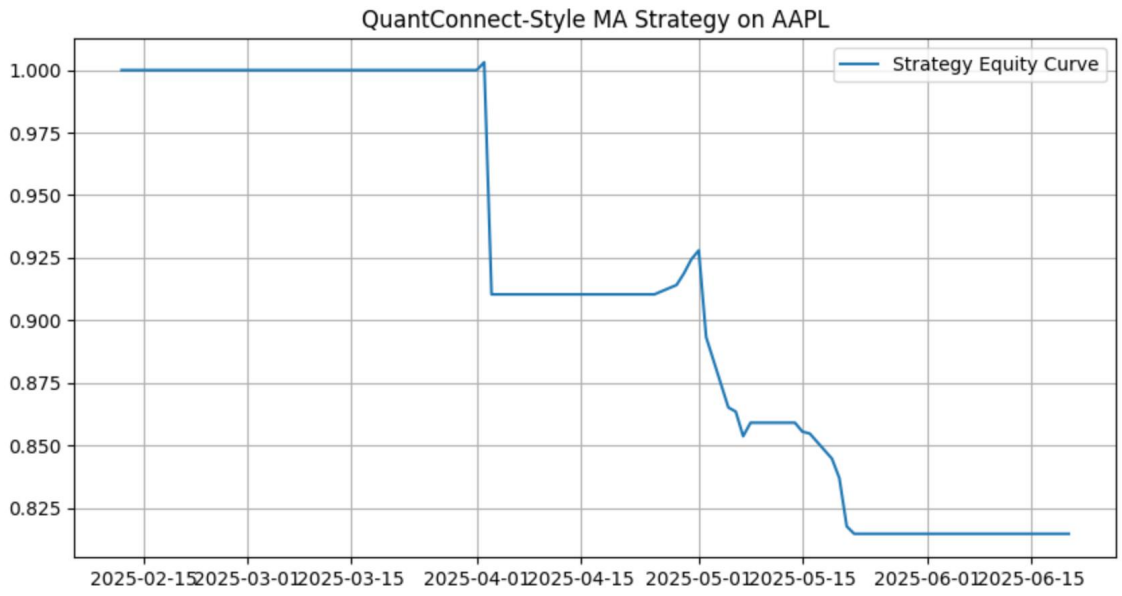
- Local setup for Lean engine is non-trivial
- Locked into their backtesting environment unless you replicate Lean

Conclusion: The best option among the frameworks for serious options strategy development and testing.

Example Code:

Simulated QuantConnect strategy: Buy when 5-day MA > 20-day MA

```
df = aapl.copy()
df['ma5'] = df['Close'].rolling(5).mean()
df['ma20'] = df['Close'].rolling(20).mean()
df['position'] = (df['ma5'] > df['ma20']).astype(int)
df['returns'] = df['Close'].pct_change().fillna(0)
df['pnl'] = df['position'].shift(1) * df['returns']
df['cumulative_pnl'] = (1 + df['pnl']).cumprod()
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['cumulative_pnl'], label='Strategy Equity Curve')
plt.title('QuantConnect-Style MA Strategy on AAPL')
plt.grid(True)
plt.legend()
plt.show()
```



Documentation:

https://www.quantconnect.com/docs/v2/cloud-platform/backtesting

Optopsy

a. Ease of use and documentation quality

Optopsy is designed specifically for options backtesting, with a clean interface and readable API. However, documentation is minimal, and there are limited working examples. The structure is intuitive once explored, but onboarding takes trial and error due to setup bugs.

b. Support for options backtesting

Optopsy is built for options trading—it supports real options chains, 30DTE filtering, IV rank, and other features that are difficult to simulate manually. It can simulate multi-leg strategies like straddles or iron condors, with expiration and roll logic.

c. Flexibility and customization options

The framework is extensible but somewhat rigid compared to Backtrader or QuantConnect. You must follow its design patterns, which limits flexibility for novel instruments or strategies. Advanced customization often requires modifying internal logic.

d. Community support and ongoing maintenance

The project is relatively new and maintained by a small team. The GitHub issues page shows some activity but lacks depth in community engagement.

Detailed Analysis of Suitability for Options

Optopsy is built for options and very well suited for vanilla strategies, like buying a straddle every 30 days based on IV rank. However, during testing, I encountered a major issue: importing core modules failed due to incorrect package structure. This significantly hinders its usability unless the user is able to dig into its internals or patch it.

Strengths:

- Tailored to options with real concepts like DTE and IV Rank
- Lightweight and easy to extend for common retail strategies

Limitations:

- Critical import bugs in the current version
- Small community and lack of real-time support

Conclusion: Very promising, but not currently reliable in production without fixes.

Example Code:

```
[ ]: import optopsy as op
class ExampleStrategy(op.Strategy):
    def should_enter(self, date, data_row):
        return data_row['iv_rank'] > 0.5 and data_row['dte'] == 30
    def should_exit(self, date, data_row):
        return data_row['dte'] <= 5
strats = op.run_backtest(df, strategy=ExampleStrategy(), leg_type="straddle")
```

Documentation:

https://github.com/michaelchu/optopsy

Custom Pandas Simulation

a. Ease of use and documentation quality

Using pandas for backtesting is extremely flexible and has excellent documentation. However, the burden of implementing every feature (order execution, slippage, rolling expiration) is on the developer. It is ideal for researchers who want full transparency.

b. Support for options backtesting

There is no built-in support—everything must be simulated. For example, you must generate synthetic options prices or import real data, handle DTE logic, and manage legs manually. Pricing models like Black-Scholes must be added.

c. Flexibility and customization options

Unmatched flexibility, but with full responsibility. Great for research or approximations of strategies like delta-neutral straddles or volatility filters. Allows users to use statistical modeling and EDA tools.

d. Community support and ongoing maintenance

While Pandas is widely used, it is not purpose-built for trading. There is little direct support for financial modeling unless extended with zipline, bt, or other layers. However, StackOverflow and GitHub are rich resources.

Detailed Analysis of Suitability for Options

Custom Pandas simulations are a good fallback when full-featured frameworks fall short or when educational experimentation is the goal. For real-world deployment or realistic slippage/margin modeling, it's not recommended.

Strengths:

- Fully customizable logic
- Great for prototyping and EDA

Limitations:

- No built-in execution logic
- All pricing and options modeling must be simulated

Conclusion: A decent prototyping tool, but insufficient for live-like options testing.

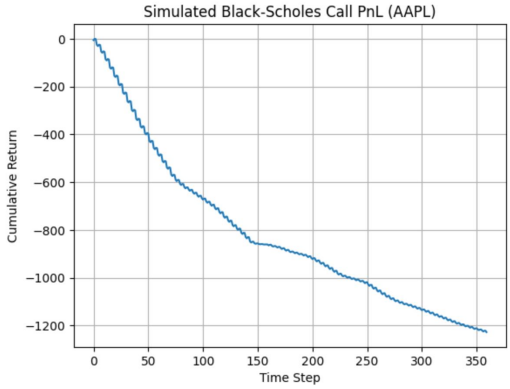
Example Code:

```
[ ]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm

def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma*np.sqrt(T)
    return S * norm.cdf(d1) - K * np.exp(-r*T) * norm.cdf(d2)
K = aapl.mean()
T = 30/365
r = 0.01
sigma = aapl.pct_change().std() * np.sqrt(252)
call_prices = [black_scholes_call(S, K, T, r, sigma) for S in aapl['Close']]
returns = np.diff(call_prices)
```

PnL:

```
plt.plot(np.cumsum(returns))
plt.title("Simulated Black-Scholes Call PnL (AAPL)")
plt.xlabel("Time Step")
plt.ylabel("Cumulative Return")
plt.grid(True)
plt.show()
```



Comparison Table:

Framework	Options Support	Ease of Use	Customization	Community
Pandas/Custom Sim	Simulated	Easy	High	None
Backtrader	Manual	Medium	High	Strong
QuantConnect	Full	Medium	Very High	Strong

Summary Table:

Criteria	Custom Pandas Sim	Backtrader	QuantConnect
a. Ease of Use	Yes	Yes	Yes
b. Options	Simulated	Manual	Full
c. Flexibility	High	Very High	Very High
d. Community	None	Strong	Strong

Conclusion

After evaluating four Python-based frameworks for backtesting options strategies, QuantConnect emerges as the most complete and production-ready tool. It offers extensive support for options chains, Greeks, expirations, and even margin modeling.

Backtrader, while very flexible and popular for equities, is not recommended for options trading unless heavily customized.

Optopsy is conceptually strong and designed specifically for options, but is currently hampered by package issues and lack of documentation.

Finally, pandas-based simulation remains useful for prototyping, research, and simple educational analysis, but lacks the realism needed for full-featured options strategy testing.

References

"Backtrader Official Documentation." *Backtrader*, https://www.backtrader.com/docu/.

"QuantConnect Documentation." *QuantConnect*, https://www.quantconnect.com/docs.

Supergrecko. *Optopsy: A Lightweight Options Backtester*. GitHub, https://github.com/supergrecko/optopsy.

"Pandas Documentation." *pandas.pydata.org*, https://pandas.pydata.org/docs/.

"Lean Algorithmic Trading Engine by QuantConnect." *GitHub*, https://github.com/QuantConnect/Lean.

"Options Trading Strategies on QuantConnect." *QuantConnect Tutorials*, https://www.quantconnect.com/learning.

"Backtrader GitHub Repository." *GitHub*, https://github.com/backtrader/backtrader.