

In [Getting Started With Python for Quant Finance](#) one of the foundational frameworks I teach is how to quickly iterate through trading ideas.

Start with a real problem, learn just enough Python to solve it, and iterate quickly.

This is my proven method for:



Avoiding meaningless puzzles and Tutorial Hell



Ensuring you can apply the Python you learn to real-life



Avoiding the overwhelm of hundreds of millions of online resources

But there's a way to supercharge the whole thing even more.

And because you'll test and discard dozens (or 100s) of trading ideas before you find a profitable one, iterating quickly is critical.

And one of the hardest things people tell me is how to generate trading ideas to test.

That's where AI and ChatGPT come in.

Most aspiring algorithmic traders don't realize how much of their craft is boring, repetitive work.

And most people totally misunderstand how AI can be used in trading.

There are advances in how AI and ML are used for finding signals and alpha factors, but most people have not even considered it for generating ideas before that part.

Let alone writing code for you.

This is where I want to help you.

Step 1: Macro-Level Idea

In [Getting Started With Python for Quant Finance](#), I teach you that trading ideas are best linked to economic reality.

When that's true, a trading edge is more robust and persistent.

Before you even open ChatGPT you need to gain some clarity yourself.

Start by coming up with some macro-level themes about what you want to explore.

Here's an example:

The crack spread against refiners.

You need to come to the table with ideas. You wouldn't expect your new intern to just know how to do the job. You can't expect an AI to know either.

Step 2: One Macro-Level Idea + Market Inefficiencies

Model-based systems start with a model of market inefficiency.

Inefficiencies can be based on trader psychology, economics, market microstructure, company events, or anything else that affects the price.

These inefficiencies cause patterns that deviate from the normal randomness of the market.

Sometimes, these patterns repeat and can be detected, predicted, and traded.

So let's see if we can combine our macro-level idea with a mean reversion.

Mean reversion is an inefficiency that occurs when asset prices and historical returns eventually revert to their long-term mean or average level.

Mean reversion does not happen in a random process.

Step 3: Train ChatGPT To Become A Trading Idea Generation Machine

We'll use the RISE Prompt framework for this task.

It works like this:

Role: *Assume the role of a world-class trader with expertise in trading energy futures. Specifically, you excel in trading the crack spread against refiner stocks.*

Instructions: *Write 5 trading ideas related to trading a crack spread position in futures against a refiner stock.*

Steps: *1. Start with the composition of the crack spread. Proceed with a description of how to trade it against a refiner stock. Proceed with explaining the economic rationale why the crack spread and refiner stocks should be related. Finish with the description of a trade that the difference between the crack spread and refiner stock.*

End Goal: *The goal is a concise description of a trade I can execute.*

And the response from ChatGPT:



Composition of the Crack Spread



The crack spread is a trading strategy in the energy sector that captures the price differential between crude oil and its refined products, most commonly gasoline and heating oil. To construct a simple 3:2:1 crack spread, you would buy 3 futures contracts of crude oil and simultaneously sell 2 contracts of gasoline and 1 contract of heating oil.

Trading Crack Spread Against a Refiner Stock

When trading the crack spread against a refiner stock, you are essentially comparing the profitability of refining operations to the market's valuation of a specific refining company. Refiner stocks should theoretically benefit from a widening crack spread, as their margins improve when the price difference between crude and its products increases.

Economic Rationale

The economic rationale behind the correlation between the crack spread and refiner stocks is straightforward. 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, although other factors such as operational efficiency, regulatory changes, and broader market sentiment can cause divergence.

Trade Ideas

1. Long Crack Spread, Short Refiner Stock - Mean Reversion

If the crack spread widens substantially but a refiner's stock hasn't reacted proportionally, you may execute a mean-reversion strategy. Long the crack spread and short the refiner stock, expecting the stock to catch up to the increased profitability suggested by the crack spread.

2. Short Crack Spread, Long Refiner Stock - Divergence Trade

If the crack spread narrows but a refiner's stock price remains resilient, it could suggest that the market is overly optimistic about the refiner's profit margins. Short the crack spread and go long on the refiner stock, anticipating that the stock will correct downward as margins compress.

3. Long Crack Spread, Long Refiner Stock - Trend Confirmation

In an environment where both the crack spread and the refiner stock are trending upward, take a long position in both to capitalize on the sustained strength in refining margins and investor sentiment towards the sector.

4. Relative Strength Trade - Pairing Strong with Weak

Identify two refiner stocks: one showing strong performance and another showing weak performance. If the crack spread is expanding, go long on the strong stock and short the weak stock, expecting the strong stock to outperform further.

5. Arbitrage Opportunities - Geographical Spreads

Some refiners have a more significant presence in specific geographic regions where the crack spread may be widening more quickly. Long the local crack spread and short a global crack spread index while simultaneously long the refiner stock most affected by the local crack spread.

Trade Execution: Long Crack Spread, Short Refiner Stock

1. Buy 3 futures contracts of WTI crude oil.
2. Sell 2 futures contracts of RBOB gasoline.
3. Sell 1 futures contract of Heating Oil.
4. Short shares of a specific refiner stock that has not yet reacted to the widening crack spread.

This strategy aims to capture the eventual rise in the refiner stock to align more closely with the improved economics implied by a wider crack spread. Keep a close eye on market developments that might affect either the crack spread or the refiner stock and adjust your positions accordingly.

Resumen Técnico de Proyectos de Finanzas Cuantitativas y *Machine Learning* con Python

Este documento resume los enfoques, librerías y desafíos técnicos encontrados en varios proyectos de *quant finance* y *backtesting* para guiar la implementación de un *framework* predictivo.

1. Backtesting Ultrarrápido y Validación Estadística (VectorBT)

Este proyecto se centra en la **ejecución de millones de simulaciones** y la validación estadística de estrategias para **evitar el *overfitting***.

Componente	Descripción y Uso	Referencias
Librería Principal	VectorBT (Vector BT) es una librería de <i>backtesting</i> ultrarrápida.	
Estrategia	Media Móvil Simple (SMA): Compra cuando la SMA rápida supera la SMA lenta, y vende viceversa.	
Metodología Clave	Análisis Walk Forward (similar a la validación cruzada en ciencia de datos). Se utiliza para aplicar parámetros optimizados en una sección de datos de entrenamiento (azul) a una sección de datos no vistos (<i>out-of-sample</i> , naranja).	
Optimización	Se optimizan los parámetros de las ventanas de la SMA (rápida y lenta) para maximizar el Sharpe Ratio en la muestra de entrenamiento (<i>in-sample</i>).	
Validación Estadística	Se utiliza la librería SciPy para realizar un T-Test unilateral . El objetivo es determinar si la media poblacional del <i>Sharpe Ratio out-of-sample</i> es estadísticamente mayor que la media <i>in-sample</i> .	
Criterio de Rechazo	Un valor P (P-value) menor a 0.05 se requiere para considerar el experimento estadísticamente significativo y rechazar la hipótesis de <i>overfitting</i> .	

2. Modelado de Factores de Riesgo para Acciones (Fama-French)

El objetivo es obtener las exposiciones factoriales (*factor exposures*) y el riesgo idiosincrático de una acción (ej. Apple).

Componente	Descripción y Uso	Referencias
Librerías/Herramientas	Pandas Data Reader (para recopilar datos financieros), Matplotlib (para graficar), Y Finance (como <i>fix</i> para problemas de conexión con Yahoo Finance en Data Reader), Statsmodels (para regresión OLS).	
Adquisición de Datos	Se obtienen series de tiempo de retornos de acciones y retornos de factores de riesgo. Se utiliza Fama-French (<i>Fama French Reader</i>) para obtener datos de factores (3 factores Fama-French + Momento).	
Frecuencia de Datos	Se prefiere la frecuencia mensual ('m') sobre la diaria ('d'), ya que los retornos diarios de los factores suelen ser demasiado ruidosos para medir las sensibilidades factoriales.	

Limpieza y Preparación de Datos	Incluye la instalación de <code>yfinance</code> , la importación y uso de <code>yfinance</code> para acceder a Yahoo Finance, el cálculo de retornos (usando <code>percent_change</code>), la conversión de retornos a unidades de porcentaje, la sustracción de la tasa libre de riesgo (<i>risk-free rate</i>) para obtener retornos en exceso.
Desafío de Integración	La dificultad clave es fusionar datos con índices de fecha inconsistentes (ej. <code>Timestamp</code> vs. <code>Period</code> , con o sin día). Esto requiere crear una variable identificadora común (<code>dt_date</code> como cadena 'Año-Mes') en ambos <i>DataFrames</i> para realizar la fusión (<code>pd.merge</code> con <i>inner join</i>).
Modelado (Regresión)	Se usa Ordinary Least Squares (OLS) de statsmodels . La regresión es de los retornos en exceso del activo (Y) sobre los cuatro factores de riesgo (X).
Corrección de OLS	Python no incluye automáticamente la constante (Alpha) ; se debe agregar manualmente una columna de unos al <i>DataFrame</i> de las variables X, utilizando <code>statsmodels.tools.add_constant</code> .
Resultado	La regresión produce coeficientes (exposiciones/betas), <i>t-stats</i> , <i>R-cuadrado</i> , e Alpha (retorno en exceso no explicado por los factores).

3. Tutorial Completo de *Algorithmic Trading* (Conceptos Básicos)

Este material proporciona una base en Python para *algorithmic trading*, cubriendo desde la descarga de datos hasta la creación de estrategias y el uso de programación orientada a objetos (POO).

Componente	Descripción y Uso	Referencias
Configuración Base	Se recomienda instalar Anaconda y lanzar Jupyter Notebook . Se requiere instalar <code>yfinance</code> (<code>pip3 install yfinance</code>).	
Librerías Fundamentales	<code>yfinance</code> (descarga de datos), <code>pandas</code> (manipulación de datos, <code>pd</code>), <code>numpy</code> (cálculos matemáticos, <code>np</code>), <code>matplotlib.pyplot</code> (gráficos, <code>plt</code>).	
Manipulación de Datos	Descarga de datos de múltiples <i>tickers</i> . Funciones de <i>Pandas</i> : <code>stocks.head()</code> , <code>stocks.info()</code> , <code>stocks.columns</code> . Conversión de índices múltiples a tuplas planas y viceversa. Filtrado por columna: <code>stocks.loc[:, 'Close']</code> .	
Análisis de Retornos	Normalización de precios a base 100 para comparar el rendimiento de diferentes activos. Cálculo de retornos logarítmicos (<code>np.log(df / df.shift(1))</code>) como medida de retorno más precisa que el retorno simple, ya que los retornos simples pueden ser engañosos al calcular medias.	

Métricas de Riesgo	Cálculo de Retorno Anualizado (retorno diario * 252 días hábiles). Cálculo de Desviación Estándar Anualizada (<i>risk</i>) (std dev diaria * 252).
Drawdowns	Se calcula el <i>drawdown</i> a partir de la diferencia entre el Máximo Acumulativo (<i>cumulative max</i>) y los Retornos Acumulativos (<i>cumulative returns</i>). Esto permite encontrar la caída más alta desde un pico.
Estrategia SMA Crossover	Se utilizan las funciones <code>df.rolling(window).mean()</code> para calcular SMA de periodos cortos (ej. 50) y largos (ej. 100). La posición se define con <code>np.where</code> : 1 (Long) si SMA corta > SMA larga, -1 (Short) si no.
Optimización de Estrategia	Se muestra cómo ajustar la estrategia SMA a un sesgo largo (<i>long bias</i>) (<code>np.where</code> devuelve 1 o 0, en lugar de 1 o -1) para reducir el riesgo (desviación estándar) y potencialmente mejorar los retornos.
Programación Avanzada	Creación de Funciones (<code>def</code>) para encapsular la lógica de <i>backtesting</i> . Creación de Clases (<code>class</code>) (ej. <code>SMABackTester</code>) con la función <code>__init__</code> (para inicializar variables) y métodos (<code>get_data</code> , <code>test_results</code> , <code>plot_results</code>) para POO y reutilización de código.

4. Implementación de una Estrategia de Factor de Baja Volatilidad (Zipline Reloaded)

El proyecto demuestra cómo construir y probar un *factor* de inversión que favorece las acciones de baja volatilidad, utilizando la librería Zipline Reloaded.

Componente	Descripción y Uso	Referencias
Librería Principal	Zipline Reloaded (motor de <i>backtesting</i>).	
Variables de Estrategia	Semanas de retrospectiva (<i>look back weeks</i>), tamaño del universo (<i>universe size</i>), y <i>quantile</i> (grupo de acciones a comprar).	
Definición del Factor	Uso de la maquinaria interna de Zipline (<i>Custom Factor</i>) que interactúa con la Pipeline API . El factor calcula la desviación estándar o volatilidad de los retornos semanales en una ventana definida.	
Manejo de Data Leakage	Se implementa un método para eliminar superposiciones (<i>overlaps</i>) utilizando bloques de 5 días (fragmentos no superpuestos) para evitar el sesgo conocido como fuga de datos (<i>data leakage</i>).	
Pipeline API (Selección)	Se utiliza el <i>pipeline</i> para: 1. Definir el universo base (ej. <i>top</i> 3,000 activos por volumen promedio en dólares como <i>proxy</i> de tamaño). 2. Aplicar el factor de volatilidad semanal. 3. Seleccionar las acciones con la volatilidad más baja (<i>lowest volatility stocks</i>) dentro del universo base.	

Backtest Setup	La función <code>initialize</code> instala el <i>pipeline</i> y programa la función de reequilibrio (<code>rebalance</code>) para que se ejecute al comienzo de cada mes . El <i>pipeline</i> se recalcula diariamente para actualizar el universo negociable.
Lógica de Negociación	Se verifica que los activos sean negociables (<code>can trade</code>). Si se toma una posición, se construye una cartera de peso igual (<i>equal weight</i>). Se gestionan las órdenes para liquidar posiciones fuera del objetivo y abrir posiciones <i>long</i> .
Ingesta de Datos	Se requiere una clave API de NASDAQ Data Link (anteriormente Quandl) para ingestar y agrupar (<i>bundle</i>) los datos de mercado que Zipline Reloaded necesita para ejecutarse.

5. Análisis Rápido de Estrategias con Python Mínimo Viable (MVP)

Se describe el uso de "**Python Mínimo Viable**" (MVP) para probar rápidamente si una estrategia tiene un *edge* potencial, utilizando principalmente `pandas` y `yfinance`.

Componente	Descripción y Uso	Referencias
Librerías	<code>yfinance</code> (adquisición de datos) y <code>pandas</code> .	
Estrategia de Ejemplo	<i>Flow effects</i> en el ETF de bonos TLT. Hipótesis: los gestores de fondos rotan a activos seguros a fin de mes (compran TLT) y rotan a activos de riesgo a principio de mes (venden TLT).	
Implementación MVP	Se descargan datos y se calculan retornos logarítmicos . Se manipula el <i>DataFrame</i> para agregar la columna del día del mes . Se agrupan los retornos por día del mes para ver el rendimiento promedio.	
Definición de Posiciones	Posición Corta (<i>Short</i>) en la primera semana del mes (Día del mes < 7). Posición Larga (<i>Long</i>) en la última semana del mes (Día del mes >= 23).	
Cálculo de Retornos de Estrategia	Se calcula la diferencia entre los retornos de la última semana (Long) y los retornos de la primera semana (Short).	
Resultados	Visualización de retornos anuales y comparación de la curva de capital acumulada de la estrategia contra la estrategia <i>Buy and Hold</i> .	

6. Predicción de Precios con *Machine Learning* (LSTM)

Este proyecto se centra en la aplicación de una red neuronal recurrente para la predicción de precios de acciones (Apple) basándose en 60 días de precios anteriores.

Componente	Descripción y Uso	Referencias
Entorno/Modelo	Google Colab . LSTM (<i>Long Short-Term Memory</i>), una Red Neuronal Recurrente Artificial (RNN) para el procesamiento de secuencias de datos.	

Librerías Clave	pandas_datareader (como web), numpy, sklearn.preprocessing.MinMaxScaler, keras.models.Sequential , keras.layers.Dense y keras.layers.LSTM .
Preparación de Datos	Se obtienen datos de precios de cierre. Se divide la data en conjuntos de entrenamiento (80%) y prueba.
Escalamiento (CRÍTICO)	Se utiliza MinMaxScaler para escalar los datos al rango de 0 a 1 . Esto es una práctica recomendada para redes neuronales.
Secuenciación de Datos	Se dividen los datos de entrenamiento en X_train (60 valores anteriores) y Y_train (el valor 61 a predecir). El <i>DataFrame</i> de X_train debe ser remodelado a una forma tridimensional (Número de muestras, Número de pasos de tiempo, Número de <i>features</i>) para cumplir con el requerimiento de la red LSTM.
Reshape para LSTM	Modelo <code>Sequential</code> . Capas LSTM (50 neuronas; <code>return_sequences=True</code> para la primera capa, <code>False</code> para la última capa LSTM). Capas Dense (25 neuronas y 1 neurona final de salida).
Arquitectura del Modelo	Optimizador Adam . Función de pérdida Mean Squared Error (MSE). Entrenamiento (<code>model.fit</code>) con parámetros como <i>batch size</i> y <i>epochs</i> .
Compilación y Entrenamiento	Se desescala la data de predicciones (utilizando <code>scaler.inverse_transform</code>) para compararla con los precios reales. Se calcula el Root Mean Squared Error (RMSE) para evaluar la precisión del modelo (valores más bajos indican mejor ajuste).
Evaluación	

7. Uso de Bases de Datos Relacionales y *DataFrames* de Alto Rendimiento
El objetivo es utilizar herramientas avanzadas de Python para el almacenamiento y análisis rápido de grandes volúmenes de datos.

Componente	Descripción y Uso	Referencias
Librerías/Herramientas	Polars (reemplazo ultrarrápido de Pandas). DuckDB (base de datos relacional en memoria <i>in-process</i> y extremadamente rápida).	
Flujo de Trabajo	Se convierte el <i>DataFrame</i> de Pandas (obtenido vía <code>yfinance</code>) a un <i>DataFrame</i> de Polars (<code>pl.from_pandas</code>).	
Integración DuckDB/Polars	Se crea una conexión a una base de datos DuckDB. La característica clave es que DuckDB puede leer directamente desde una variable de DataFrame de Polars (ej. <code>SELECT * FROM DF</code>) para crear una tabla sin necesidad de definir un esquema explícitamente.	

Análisis SQL

Permite ejecutar consultas SQL complejas, incluyendo funciones de ventana (*window functions*), como el **VWAP rodante de 20 días** (*rolling 20-day VWAP*).

Resultado

Las consultas SQL en DuckDB se ejecutan instantáneamente, devolviendo los resultados en un *DataFrame* de Polars.

Este conjunto de recursos le proporciona metodologías para: **(1) Backtesting y validación estadística (VectorBT/Zipline)**, **(2) Creación de *features* y exposición a factores de riesgo (Fama-French)**, **(3) Implementación de modelos predictivos de series de tiempo (LSTM)**, y **(4) Optimización del rendimiento de datos (Polars/DuckDB)**.