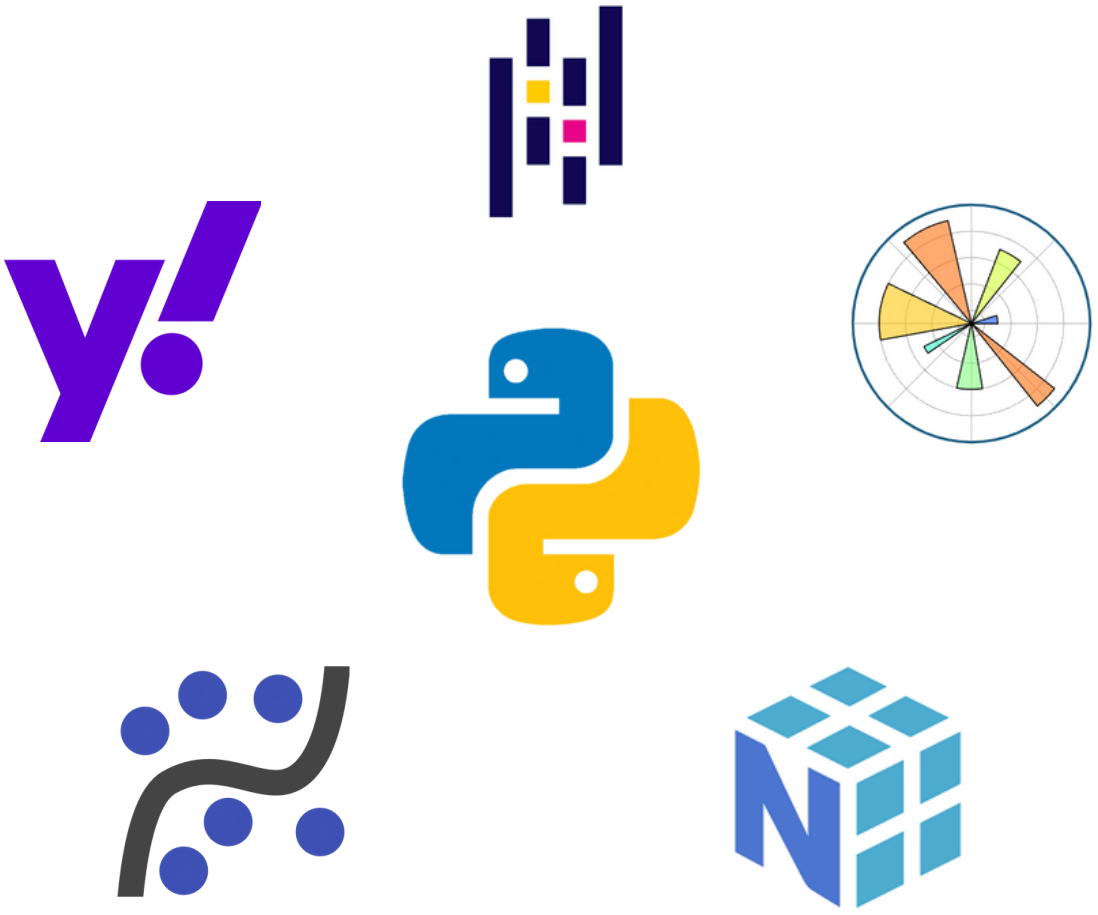
5 librerías de Python que todo analista financiero debería conocer

APRENDE A CALCULAR RENDIMIENTOS, MODELAR RIESGOS Y VISUALIZAR DATOS COMO UN PROFESIONAL.



5 librerías explicadas paso a paso

📊5 librerías de Python que todo analista Financiero debería conocer🚀

Autor:JoseDiego CazaresMinjares

Cntacto:LinkedIn SitioWeb:betafinanciera.com

Pandas:El Exceldelos analistas en Python

Introducción: En finanzas trabajamos con datos tabulares: precios,

balances, ratios,
etc. Pandas es la librería que convierte Python en una hoja de cálculo superpoderosa. Fundamento: Permite organizar datos en estructuras

llamadas DataFrames, muy
similares a una tabla de Excel, pero con mayor eficiencia y flexibilidad.

Código de pandas

```
                        #Importante importar    la   paqueria a utilizar
import pandas as pd
# Simulación de precios de una acción
data = {
      "Fecha": pd.date_range(start="2025-01-01", periods=5, freq="D"),
      "Precio": [100, 102, 101, 105, 107]
}

df = pd.DataFrame(data)
print(f"{df}\n")
# Calcular rendimientos simples
df["Rendimiento"] = df["Precio"].pct_change() # Funcion para Calcula
el cambio fraccional de la fila inmediatamente anterior

print(df)
```

```
        Fecha Precio
0 2025-01-01      100
1 2025-01-02      102
2 2025-01-03      101
3 2025-01-04      105
4 2025-01-05      107
```

	Fecha	Precio	Rendimiento
0	2025-01-01	100	NaN
1	2025-01-02	102	0.020000
2	2025-01-03	101	-0.009804
3	2025-01-04	105	0.039604
4	2025-01-05	107	0.019048

En este código, primero creamos una tabla de datos con fechas y precios simulados de una acción usando pandas. Luego, calculamos el cambio porcentual diario de esos precios con la función .pct_change().

Hacemos esto para analizar la variación o el rendimiento que tiene el activo día a día.

Finalmente, el código muestra la tabla original con la nueva columna "Rendimiento"

Matplotlib: Visualizandola historia delosdatos

Introducción

En finanzas, una gráfica vale más que mil números. Matplotlib es la librería base para visualizar tendencias y comparaciones.

Código deMatplotlib

```
import matplotlib.pyplot as plt #Importante importar la paqueria a utilizar
plt.plot(df["Fecha"], df["Precio"], marker="o")
plt.title("Evolución del precio de la acción")
plt.xlabel("Fecha")
plt.ylabel("Precio")
plt.show()
```

En este código, usamos la biblioteca matplotlib pra crear un gráfico de líneas a partir de los datos de la tabla anterior. Colocamos las fechas en el eje horizontal (X), los precios en el eje vertical (Y) y le añadimos un título y etiquetas.

Hacemos esto para visualizar la información de la tabla, lo que permite entender de forma más intuitiva y rápida cómo ha cambiado el precio a lo largo del tiempo.

Finalmente, el código muestra una ventana con el gráfico de la evolución del precio, donde una línea conecta cada punto de dato.

NumPy: Matemáticas financieras a toda velocidad

Introducción

Las operaciones financieras suelen requerir mucho cálculo numérico: medias, desviaciones, tasas compuestas. Ahí entra NumPy.

Fundamento

Permite trabajar con vectores y matrices, optimizando cálculos repetitivos.

Ejemplo: Rendimiento compuesto

$$R_c = \left(\prod_{t=1}^{n} (1 + r_t) \right)^{-1}$$

Código de Numpy

```
import   numpy as np #Importante importar la paqueria a utilizar

rendimientos = np.array([0.02, -0.01, 0.04, 0.03])              # 4 periodos
rend_compuesto = np.prod(1 + rendimientos) - 1
print("Rendimiento compuesto:", round(rend_compuesto, 4))
```

```
Rendimiento compuesto: 0.0817
```

En este código, usamos la biblioteca numpy para calcular el rendimiento compuesto total a partir de una serie de rendimientos simples de varios periodos.

Hacemos esto para conocer la ganancia o pérdida acumulada de una inversión, considerando que las ganancias de un periodo se reinvierten y generan ganancias en el siguiente.

Finalmente,el código muestra en unasola línea el resultado de ese cálculototal,redondeado a 4 decimales.

yFinance: Datos financieros en tiempo real

Introducción

¿Quieres descargar datos de acciones, ETFs o índices en segundo?

yFinance conecta con Yahoo Finance y lo hace posible.

Fundamento:

Con unas pocas líneas:

- Obtienes datos históricos.

- Puedes calcular indicadores como rendimientos o medias móviles.
- Sirve como base para modelos de predicción o backtesting.

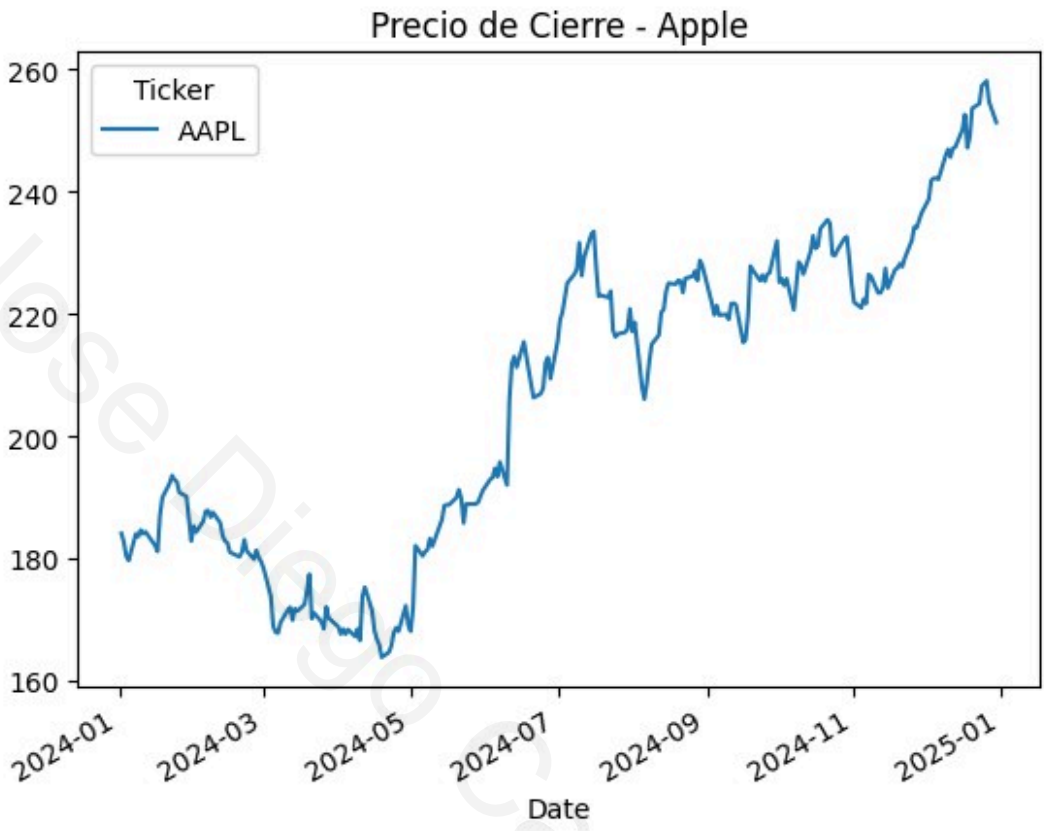
Códigodeyfinance

```
import yfinance as yf #Importante importar la paqueria a utilizar

# Descargar datos de Apple
apple = yf.download("AAPL", start="2024-01-01", end="2024-12-31",
auto_adjust=True)
print(apple.head())
# Graficar precios de cierre
apple["Close"].plot(title="Precio de Cierre - Apple")
plt.show()
```

```
[*********************100%***********************]  1 of 1 completed
```

Price	Close	High	Low	Open	Volume
Ticker	AAPL	AAPL	AAPL	AAPL	AAPL
Date					
2024-01-02	184.081497	186.857993	182.346189	185.578815	82488700
2024-01-03	182.703171	184.319491	181.890048	182.673424	58414500
2024-01-04	180.382812	181.552899	179.361461	180.620788	71983600
2024-01-05	179.658951	181.225688	178.657435	180.462163	62379700
2024-01-08	184.002182	184.041855	179.976269	180.561312	59144500



En este código, primero usamos la biblioteca yfinance para descargar los datos históricos del precio de la acción de Apple ("AAPL") durante el año 2024. Después, a partir de esos datos, creamos un gráfico de líneas que representa específicamente la evolución de los precios de cierre.

Hacemos esto para obtener datos reales del mercado y luego visualizarlos, lo que permite analizar de forma muy intuitiva la tendencia y el comportamiento del precio de la acción en ese periodo.

Statsmodels: Entendiendo relaciones con econometría

Introducción

En análisis financiero no solo observamos, también queremos explicar y predecir. Statsmodels permite hacer regresiones y análisis estadístico.

Fundamento

Ejemplo de regresión lineal:

$$R_{stock} = \alpha + \beta \cdot R_{market} + \epsilon$$

Donde:

- α = rendimiento independiente del mercado (alpha).
- β = sensibilidad al mercado.
- ϵ = erroraleatorio.

Código de Statsmodels

```
import statsmodels.api      as sm

# Datos simulados
rend_mercado = np.random.normal(0.01, 0.02, 100)
rend_accion = 0.002 + 1.2 * rend_mercado + np.random.normal(0, 0.01,
100)
X = sm.add_constant(rend_mercado)
modelo = sm.OLS(rend_accion, X).fit()
print(modelo.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:
0.804
Model:                            OLS   Adj. R-squared:
0.802
Method:                 Least Squares   F-statistic:
402.9
Date:                Fri, 26 Sep 2025   Prob (F-statistic):
1.72e-36
Time:                        07:52:31   Log-Likelihood:
310.24
```

```
No. Observations:                   100   AIC:
-616.5
Df Residuals:                        98   BIC:
-611.3
Df Model:                             1
Covariance Type:                 nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const          0.0030      0.001      2.366      0.020       0.000
0.005
x1             1.1472      0.057     20.072      0.000       1.034
1.261
==============================================================================
Omnibus:                        1.201   Durbin-Watson:
2.433
Prob(Omnibus):                  0.549   Jarque-Bera (JB):
0.872
Skew:                          -0.224   Prob(JB):
0.647
Kurtosis:                       3.095   Cond. No.
52.0
==============================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Interpretación

Si β > 1: la acción es más volátil que el mercado.

Si β < 1: la acción es más defensiva.

Si α > 0: la acción genera exceso de rendimiento independiente del mercado.

En este código, primero simulamos datos para los rendimientos de un mercado y de una acción, creando una relación lineal entre ellos. Después, usamos la biblioteca statsmodels para aplicar un modelo de regresión lineal (conocido como Mínimos Cuadrados Ordinarios - OLS).

Hacemos esto para medir estadísticamente la relación entre la acción y el mercado. El objetivo es estimar el:

Alpha (α): El intercepto, que representa el rendimiento de la acción que no es explicado por los movimientos del mercado.

Beta (β): La pendiente, que mide la sensibilidad o volatilidad de la acción en relación con los movimientos del mercado.

Finalmente, el código muestra una tabla de resumen estadístico con los resultados completos de la regresión. Esta tabla incluye valores clave como el R-cuadrado (qué tan bien el modelo explica los datos) y los coeficientes calculados para la constante (Alpha) y la variable del mercado (Beta).

🎯Conclusión

Con estas 5 librerías:

Pandas organiza y limpia tus datos.

NumPy hace los cálculos rápidos.

Matplotlib los convierte en gráficos.

yFinance trae información real.

Statsmodels te permite explicar y modelar relaciones.

👉Juntas forman una caja de herramientas esencial para cualquier analista financiero que quiera ir más allá de Excel