Yahoo Finance (yfinance) Python Tutorial

This tutorial will guide you through using the Yahoo Finance library (`yfinance`) in Python to analyze stock market data, with a special focus on the Turkish stock market (BIST) and specific stocks like THYAO (Turkish Airlines).

1. Installation and Setup

First, let's install the required libraries:

```python
# Install required libraries
!pip install yfinance pandas matplotlib seaborn plotly
```

Now let's import the libraries we'll need:

```python
# Import libraries
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from datetime import datetime, timedelta

# Set plotting style
plt.style.use('fivethirtyeight')
sns.set_style('darkgrid')
```

2. Basic Stock Data Retrieval

2.1 Getting Information for a Single Stock

Let's start by retrieving basic information about a stock. For example, let's look at Turkish Airlines (THYAO.IS):

python

```python
# Create a Ticker object
thyao = yf.Ticker("THYAO.IS")  # .IS suffix is for Istanbul Stock Exchange

# Get basic info
info = thyao.info
print(f"Company Name: {info.get('longName', 'N/A')}")
print(f"Industry: {info.get('industry', 'N/A')}")
print(f"Sector: {info.get('sector', 'N/A')}")
print(f"Country: {info.get('country', 'N/A')}")
print(f"Website: {info.get('website', 'N/A')}")
print(f"Current Price: {info.get('currentPrice', 'N/A')} {info.get('currency', '')}")
print(f"52 Week High: {info.get('fiftyTwoWeekHigh', 'N/A')}")
print(f"52 Week Low: {info.get('fiftyTwoWeekLow', 'N/A')}")
```

2.2 Downloading Historical Data

Now let's download historical price data for Turkish Airlines:

python

```python
# Download historical data for the past year
thyao_data = thyao.history(period="1y")

# Display the first few rows
print(thyao_data.head())

# Basic information about the dataset
print("\nDataset Information:")
print(f"Start Date: {thyao_data.index[0].strftime('%Y-%m-%d')}")
print(f"End Date: {thyao_data.index[-1].strftime('%Y-%m-%d')}")
print(f"Number of Trading Days: {len(thyao_data)}")
```

2.3 Plotting Stock Price History

Let's visualize the stock price history:

python

```python
# Plot the closing price
plt.figure(figsize=(14, 7))
plt.plot(thyao_data.index, thyao_data['Close'], label='THYAO Close Price')
plt.title('Turkish Airlines (THYAO) Stock Price - Last 1 Year')
plt.xlabel('Date')
plt.ylabel('Price (TRY)')
plt.legend()
plt.tight_layout()
plt.show()

# Plot with trading volume
fig, ax1 = plt.subplots(figsize=(14, 7))

# Plot price on the first axis
ax1.plot(thyao_data.index, thyao_data['Close'], 'b-', label='Close Price')
ax1.set_xlabel('Date')
ax1.set_ylabel('Price (TRY)', color='b')
ax1.tick_params(axis='y', labelcolor='b')

# Create a second y-axis for volume
ax2 = ax1.twinx()
ax2.bar(thyao_data.index, thyao_data['Volume'], alpha=0.3, color='g', label='Volume')
ax2.set_ylabel('Volume', color='g')
ax2.tick_params(axis='y', labelcolor='g')

plt.title('THYAO Price and Volume - Last 1 Year')
fig.tight_layout()
plt.show()
```

3. Advanced Stock Analysis

3.1 Calculating Returns

Let's calculate daily, monthly, and annual returns:

```python
# Calculate daily returns
thyao_data['Daily_Return'] = thyao_data['Close'].pct_change()

# Plot daily returns
plt.figure(figsize=(14, 7))
plt.plot(thyao_data.index, thyao_data['Daily_Return'], label='Daily Returns')
plt.title('THYAO Daily Returns')
plt.xlabel('Date')
plt.ylabel('Return')
plt.legend()
plt.tight_layout()
plt.show()

# Calculate cumulative returns
thyao_data['Cumulative_Return'] = (1 + thyao_data['Daily_Return']).cumprod() - 1

# Plot cumulative returns
plt.figure(figsize=(14, 7))
plt.plot(thyao_data.index, thyao_data['Cumulative_Return'] * 100, label='Cumulative Re
plt.title('THYAO Cumulative Returns')
plt.xlabel('Date')
plt.ylabel('Cumulative Return (%)')
plt.legend()
plt.tight_layout()
plt.show()

# Monthly returns
monthly_return = thyao_data['Close'].resample('M').ffill().pct_change()
print("\nMonthly Returns:")
print(monthly_return)

# Annual return (assuming we have at least one year of data)
annual_return = thyao_data['Close'].iloc[-1] / thyao_data['Close'].iloc[0] - 1
print(f"\nAnnual Return: {annual_return * 100:.2f}%")
```

3.2 Calculating Moving Averages

Moving averages are a common technical indicator:

python

```python
# Calculate 20-day and 50-day moving averages
thyao_data['MA20'] = thyao_data['Close'].rolling(window=20).mean()
thyao_data['MA50'] = thyao_data['Close'].rolling(window=50).mean()

# Plot the moving averages
plt.figure(figsize=(14, 7))
plt.plot(thyao_data.index, thyao_data['Close'], label='THYAO Close Price')
plt.plot(thyao_data.index, thyao_data['MA20'], label='20-Day MA')
plt.plot(thyao_data.index, thyao_data['MA50'], label='50-Day MA')
plt.title('THYAO Stock Price with Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price (TRY)')
plt.legend()
plt.tight_layout()
plt.show()
```

3.3 Calculating Volatility

Let's calculate and visualize the stock's volatility:

python

```python
# Calculate 20-day rolling standard deviation of returns (volatility)
thyao_data['Volatility'] = thyao_data['Daily_Return'].rolling(window=20).std() * np.sq

# Plot volatility
plt.figure(figsize=(14, 7))
plt.plot(thyao_data.index, thyao_data['Volatility'], label='20-Day Rolling Volatility
plt.title('THYAO Volatility Over Time')
plt.xlabel('Date')
plt.ylabel('Volatility')
plt.legend()
plt.tight_layout()
plt.show()
```

4. Working with Multiple BIST Stocks

Let's analyze multiple stocks from the BIST together:

```python
# List of some major BIST stocks
bist_tickers = ['THYAO.IS', 'EREGL.IS', 'GARAN.IS', 'AKBNK.IS', 'KCHOL.IS']

# Download data for multiple stocks
bist_data = yf.download(bist_tickers, period='1y')

# Display the first few rows
print(bist_data.head())

# Plot closing prices for multiple stocks
plt.figure(figsize=(14, 7))
for ticker in bist_tickers:
    plt.plot(bist_data.index, bist_data['Close'][ticker], label=ticker.replace('.IS',
plt.title('Major BIST Stocks - Last 1 Year')
plt.xlabel('Date')
plt.ylabel('Price (TRY)')
plt.legend()
plt.tight_layout()
plt.show()
```

4.1 Calculating Correlation Between Stocks

Let's find out how these stocks move relative to each other:

```python
# Extract the closing prices
close_prices = bist_data['Close']

# Calculate the correlations
correlations = close_prices.pct_change().corr()

# Plot the correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlations, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Between BIST Stocks')
plt.tight_layout()
plt.show()
```

4.2 Calculating and Comparing Returns

Let's compare the returns of different BIST stocks:

```python
# Calculate daily returns for each stock
returns = close_prices.pct_change()

# Calculate cumulative returns
cumulative_returns = (1 + returns).cumprod() - 1

# Plot cumulative returns
plt.figure(figsize=(14, 7))
for ticker in bist_tickers:
    plt.plot(cumulative_returns.index, cumulative_returns[ticker] * 100, label=ticker.
plt.title('Cumulative Returns of BIST Stocks (%)')
plt.xlabel('Date')
plt.ylabel('Cumulative Return (%)')
plt.legend()
plt.tight_layout()
plt.show()

# Calculate and display total returns for each stock
total_returns = cumulative_returns.iloc[-1] * 100
print("Total Returns (%):")
for ticker in bist_tickers:
    print(f"{ticker.replace('.IS', '')}: {total_returns[ticker]:.2f}%")
```

5. Fundamental Analysis

5.1 Getting Financial Statements

Yahoo Finance also provides financial statement data:

python

```python
# Get income statement data for THYAO
income_stmt = thyao.income_stmt
print("\nIncome Statement:")
print(income_stmt)

# Get balance sheet data
balance_sheet = thyao.balance_sheet
print("\nBalance Sheet:")
print(balance_sheet)

# Get cash flow data
cash_flow = thyao.cashflow
print("\nCash Flow Statement:")
print(cash_flow)
```

5.2 Fundamental Ratios

Let's calculate some key financial ratios:

python

```python
# Get some key financial ratios
print("\nKey Financial Ratios for THYAO:")
print(f"P/E Ratio: {info.get('trailingPE', 'N/A')}")
print(f"Forward P/E: {info.get('forwardPE', 'N/A')}")
print(f"Price-to-Book (P/B): {info.get('priceToBook', 'N/A')}")
print(f"Dividend Yield: {info.get('dividendYield', 'N/A') * 100 if info.get('dividendY
print(f"Return on Equity (ROE): {info.get('returnOnEquity', 'N/A') * 100 if info.get('
print(f"Profit Margins: {info.get('profitMargins', 'N/A') * 100 if info.get('profitMar
```

6. Advanced Technical Analysis

6.1 Relative Strength Index (RSI)

RSI is a momentum oscillator that measures the speed and change of price movements:

python

```python
def calculate_rsi(data, window=14):
    """Calculate Relative Strength Index (RSI)"""
    delta = data.diff()
    gain = delta.where(delta > 0, 0).rolling(window=window).mean()
    loss = -delta.where(delta < 0, 0).rolling(window=window).mean()

    rs = gain / loss
    rsi = 100 - (100 / (1 + rs))
    return rsi

# Calculate RSI for THYAO
thyao_data['RSI'] = calculate_rsi(thyao_data['Close'])

# Plot RSI
plt.figure(figsize=(14, 7))
plt.plot(thyao_data.index, thyao_data['RSI'], label='RSI', color='purple')
plt.axhline(y=70, color='r', linestyle='--', alpha=0.5)
plt.axhline(y=30, color='g', linestyle='--', alpha=0.5)
plt.title('THYAO Relative Strength Index (RSI)')
plt.xlabel('Date')
plt.ylabel('RSI')
plt.legend()
plt.tight_layout()
plt.show()
```

6.2 Bollinger Bands

Bollinger Bands are a volatility indicator:

python

```python
# Calculate Bollinger Bands
thyao_data['MA20'] = thyao_data['Close'].rolling(window=20).mean()
thyao_data['Upper_Band'] = thyao_data['MA20'] + 2 * thyao_data['Close'].rolling(window
thyao_data['Lower_Band'] = thyao_data['MA20'] - 2 * thyao_data['Close'].rolling(window

# Plot Bollinger Bands
plt.figure(figsize=(14, 7))
plt.plot(thyao_data.index, thyao_data['Close'], label='Close Price')
plt.plot(thyao_data.index, thyao_data['MA20'], label='20-Day MA', color='orange')
plt.plot(thyao_data.index, thyao_data['Upper_Band'], label='Upper Band', color='green'
plt.plot(thyao_data.index, thyao_data['Lower_Band'], label='Lower Band', color='red')
plt.fill_between(thyao_data.index, thyao_data['Upper_Band'], thyao_data['Lower_Band'],
plt.title('THYAO Bollinger Bands')
plt.xlabel('Date')
plt.ylabel('Price (TRY)')
plt.legend()
plt.tight_layout()
plt.show()
```

6.3 MACD (Moving Average Convergence Divergence)

MACD is a trend-following momentum indicator:

```python
# Calculate MACD
def calculate_macd(data, fast=12, slow=26, signal=9):
    """Calculate MACD, MACD Signal, and MACD Histogram"""
    ema_fast = data.ewm(span=fast, adjust=False).mean()
    ema_slow = data.ewm(span=slow, adjust=False).mean()
    macd = ema_fast - ema_slow
    macd_signal = macd.ewm(span=signal, adjust=False).mean()
    macd_hist = macd - macd_signal
    return macd, macd_signal, macd_hist

# Calculate MACD for THYAO
thyao_data['MACD'], thyao_data['MACD_Signal'], thyao_data['MACD_Hist'] = calculate_mac

# Plot MACD
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 10), gridspec_kw={'height_ratios': [

# Plot price in the first subplot
ax1.plot(thyao_data.index, thyao_data['Close'], label='Close Price')
ax1.set_title('THYAO Stock Price')
ax1.set_ylabel('Price (TRY)')
ax1.legend()

# Plot MACD in the second subplot
ax2.plot(thyao_data.index, thyao_data['MACD'], label='MACD', color='blue')
ax2.plot(thyao_data.index, thyao_data['MACD_Signal'], label='Signal Line', color='red'
ax2.bar(thyao_data.index, thyao_data['MACD_Hist'], label='Histogram', color='green', a
ax2.set_title('MACD')
ax2.set_xlabel('Date')
ax2.set_ylabel('MACD')
ax2.legend()

plt.tight_layout()
plt.show()
```

7. Portfolio Analysis with BIST Stocks

Let's perform a simple portfolio analysis with BIST stocks:

```python
# Let's create a hypothetical portfolio with equal weights
portfolio_tickers = ['THYAO.IS', 'EREGL.IS', 'GARAN.IS', 'AKBNK.IS', 'KCHOL.IS']
weights = np.array([0.2, 0.2, 0.2, 0.2, 0.2])  # Equal weights

# Get the data
portfolio_data = yf.download(portfolio_tickers, period='1y')['Close']

# Calculate returns
returns = portfolio_data.pct_change().dropna()

# Calculate individual expected returns (mean daily returns)
individual_expected_returns = returns.mean()
print("\nExpected Daily Returns for Each Stock:")
for ticker in portfolio_tickers:
    print(f"{ticker.replace('.IS', '')}: {individual_expected_returns[ticker] * 100:.4

# Calculate portfolio expected return
portfolio_return = np.sum(individual_expected_returns * weights) * 100
print(f"\nPortfolio Expected Daily Return: {portfolio_return:.4f}%")
print(f"Portfolio Expected Annual Return: {portfolio_return * 252:.2f}%")  # Assuming

# Calculate portfolio risk (volatility)
covariance_matrix = returns.cov() * 252  # Annualized
portfolio_variance = np.dot(weights.T, np.dot(covariance_matrix, weights))
portfolio_volatility = np.sqrt(portfolio_variance) * 100
print(f"Portfolio Annual Volatility: {portfolio_volatility:.2f}%")

# Calculate Sharpe Ratio (assuming risk-free rate of 4%)
risk_free_rate = 0.04  # 4%
sharpe_ratio = (portfolio_return * 252 / 100 - risk_free_rate) / (portfolio_volatility
print(f"Portfolio Sharpe Ratio: {sharpe_ratio:.2f}")
```

7.1 Portfolio Optimization (Efficient Frontier)

Let's perform a simple portfolio optimization:

python

```python
# We'll use Monte Carlo simulation to find the efficient frontier
import tqdm

# Number of simulations
num_portfolios = 10000

# Set up arrays to hold results
results = np.zeros((4, num_portfolios))
weights_record = np.zeros((len(portfolio_tickers), num_portfolios))
annual_returns = returns * 252

for i in tqdm.tqdm(range(num_portfolios)):
    # Generate random weights
    weights = np.random.random(len(portfolio_tickers))
    weights = weights / np.sum(weights)
    weights_record[:, i] = weights

    # Expected return
    portfolio_return = np.sum(annual_returns.mean() * weights) * 100
    results[0, i] = portfolio_return

    # Expected volatility
    portfolio_volatility = np.sqrt(np.dot(weights.T, np.dot(annual_returns.cov(), weigl
    results[1, i] = portfolio_volatility

    # Sharpe Ratio
    results[2, i] = (portfolio_return / 100 - risk_free_rate) / (portfolio_volatility ,

    # Store weights
    results[3, i] = i

# Plot efficient frontier
plt.figure(figsize=(14, 7))
plt.scatter(results[1, :], results[0, :], c=results[2, :], cmap='viridis', marker='o',
plt.colorbar(label='Sharpe Ratio')
plt.title('Efficient Frontier')
plt.xlabel('Volatility (%)')
plt.ylabel('Return (%)')

# Find portfolio with highest Sharpe Ratio
max_sharpe_idx = np.argmax(results[2, :])
max_sharpe_return = results[0, max_sharpe_idx]
max_sharpe_volatility = results[1, max_sharpe_idx]
max_sharpe_ratio = results[2, max_sharpe_idx]

# Find minimum volatility portfolio
```

```
min_vol_idx = np.argmin(results[1, :])
min_vol_return = results[0, min_vol_idx]
min_vol_volatility = results[1, min_vol_idx]
min_vol_ratio = results[2, min_vol_idx]

# Plot optimal portfolios
plt.scatter(max_sharpe_volatility, max_sharpe_return, marker='*', color='r', s=200, la
plt.scatter(min_vol_volatility, min_vol_return, marker='*', color='g', s=200, label='M

plt.legend()
plt.tight_layout()
plt.show()

# Print optimal portfolio allocations
print("\nOptimal Portfolio Allocations:")
print("\nMax Sharpe Ratio Portfolio:")
print(f"Annual Return: {max_sharpe_return:.2f}%")
print(f"Annual Volatility: {max_sharpe_volatility:.2f}%")
print(f"Sharpe Ratio: {max_sharpe_ratio:.2f}")
print("Allocations:")
for i, ticker in enumerate(portfolio_tickers):
    print(f"{ticker.replace('.IS', '')}: {weights_record[i, max_sharpe_idx] * 100:.2f}

print("\nMinimum Volatility Portfolio:")
print(f"Annual Return: {min_vol_return:.2f}%")
print(f"Annual Volatility: {min_vol_volatility:.2f}%")
print(f"Sharpe Ratio: {min_vol_ratio:.2f}")
print("Allocations:")
for i, ticker in enumerate(portfolio_tickers):
    print(f"{ticker.replace('.IS', '')}: {weights_record[i, min_vol_idx] * 100:.2f}%")
```

8. Working with BIST Index

Let's analyze the BIST 100 Index:

```python
# Download BIST 100 Index data (XU100.IS is the ticker for BIST 100)
bist100 = yf.Ticker("^XU100")
bist100_data = bist100.history(period="1y")

# Plot BIST 100 Index
plt.figure(figsize=(14, 7))
plt.plot(bist100_data.index, bist100_data['Close'], label='BIST 100 Index')
plt.title('BIST 100 Index - Last 1 Year')
plt.xlabel('Date')
plt.ylabel('Index Value')
plt.legend()
plt.tight_layout()
plt.show()

# Calculate and plot BIST 100 returns
bist100_data['Daily_Return'] = bist100_data['Close'].pct_change()
bist100_data['Cumulative_Return'] = (1 + bist100_data['Daily_Return']).cumprod() - 1

plt.figure(figsize=(14, 7))
plt.plot(bist100_data.index, bist100_data['Cumulative_Return'] * 100, label='Cumulativ
plt.title('BIST 100 Index Cumulative Return')
plt.xlabel('Date')
plt.ylabel('Cumulative Return (%)')
plt.legend()
plt.tight_layout()
plt.show()
```

8.1 Comparing BIST Stocks to Index

Let's compare our BIST stocks to the BIST 100 Index:

```python
# First, download all data together
all_tickers = portfolio_tickers + ['^XU100']
all_data = yf.download(all_tickers, period='1y')['Close']

# For BIST 100, rename the column
if '^XU100' in all_data.columns:
    all_data.rename(columns={'^XU100': 'BIST100'}, inplace=True)

# Calculate returns
all_returns = all_data.pct_change().dropna()

# Calculate cumulative returns
all_cum_returns = (1 + all_returns).cumprod() - 1

# Plot comparison
plt.figure(figsize=(14, 7))
for ticker in portfolio_tickers:
    plt.plot(all_cum_returns.index, all_cum_returns[ticker] * 100, label=ticker.replac
plt.plot(all_cum_returns.index, all_cum_returns['BIST100'] * 100, label='BIST 100', li
plt.title('BIST Stocks vs BIST 100 Index')
plt.xlabel('Date')
plt.ylabel('Cumulative Return (%)')
plt.legend()
plt.tight_layout()
plt.show()

# Calculate beta (measure of volatility compared to the market)
def calculate_beta(stock_returns, market_returns):
    """Calculate beta of a stock relative to the market"""
    covariance = stock_returns.cov(market_returns)
    market_variance = market_returns.var()
    return covariance / market_variance

# Calculate and display beta for each stock
print("\nBeta Values (Relative to BIST 100):")
for ticker in portfolio_tickers:
    beta = calculate_beta(all_returns[ticker], all_returns['BIST100'])
    print(f"{ticker.replace('.IS', '')}: {beta:.2f}")
```

9. News and Announcements

Yahoo Finance also provides news information:

python

```python
# Get recent news for THYAO
thyao_news = thyao.news
print("\nRecent News for THYAO:")
for i, news in enumerate(thyao_news[:5]):  # Display the first 5 news items
    print(f"{i+1}. {news['title']}")
    print(f"   Published: {datetime.fromtimestamp(news['providerPublishTime']).strftim
    print(f"   Link: {news['link']}")
    print()
```

10. Options Data (if available)

If options data is available for THYAO, we can analyze it:

python

```python
# Get options expiration dates
try:
    expirations = thyao.options

    if expirations:
        print("\nOptions Expiration Dates:")
        print(expirations)

        # Get option chain for the first expiration date
        option_chain = thyao.option_chain(expirations[0])

        # Display call options
        print("\nCall Options:")
        print(option_chain.calls.head())

        # Display put options
        print("\nPut Options:")
        print(option_chain.puts.head())
except:
    print("\nOptions data not available for THYAO.")
```

11. Interactive Visualizations with Plotly

Let's create some interactive visualizations:

python

```python
# Create an interactive candlestick chart for THYAO
thyao_data_recent = thyao.history(period="6mo")

fig = go.Figure(data=[go.Candlestick(
    x=thyao_data_recent.index,
    open=thyao_data_recent['Open'],
    high=thyao_data_recent['High'],
    low=thyao_data_recent['Low'],
    close=thyao_data_recent['Close'],
    name='THYAO'
)])

fig.update_layout(
    title='THYAO Candlestick Chart - Last 6 Months',
    xaxis_title='Date',
    yaxis_title='Price (TRY)',
    xaxis_rangeslider_visible=True
)

fig.show()

# Create an interactive line chart comparing multiple BIST stocks
fig = go.Figure()

for ticker in portfolio_tickers:
    ticker_data = yf.download(ticker, period='1y')
    # Normalize to 100 at the start for easier comparison
    normalized_data = ticker_data['Close'] / ticker_data['Close'].iloc[0] * 100
    fig.add_trace(go.Scatter(
        x=normalized_data.index,
        y=normalized_data,
        mode='lines',
        name=ticker.replace('.IS', '')
    ))

fig.update_layout(
    title='Normalized Performance of BIST Stocks (Base=100)',
    xaxis_title='Date',
    yaxis_title='Normalized Price',
    legend_title='Stocks'
)

fig.show()
```

12. Real-time Data (Intraday)

Yahoo Finance also provides intraday data:

python

```python
# Get intraday data for THYAO (1 minute intervals for the last 7 days)
thyao_intraday = yf.download("THYAO.IS", period="7d", interval="1m")

# Display the first few rows
print(thyao_intraday.head())

# Plot intraday data
plt.figure(figsize=(14, 7))
plt.plot(thyao_intraday.index, thyao_intraday['Close'], label='THYAO 1-min Close Price
plt.title('THYAO Intraday Prices - Last 7 Days')
plt.xlabel('Time')
plt.ylabel('Price (TRY)')
plt.legend()
plt.tight_layout()
plt.show()
```

13. Earnings and Dividends

Let's check earnings and dividends data:

python

```python
# Get earnings dates and estimates
earnings = thyao.earnings_dates
print("\nEarnings Dates and Estimates:")
print(earnings.head())

# Get dividends history
dividends = thyao.dividends
print("\nDividend History:")
print(dividends)

# Plot dividends if available
if len(dividends) > 0:
    plt.figure(figsize=(14, 7))
    plt.bar(dividends.index, dividends, color='green')
    plt.title('THYAO Dividend History')
    plt.xlabel('Date')
    plt.ylabel('Dividend Amount')
    plt.tight_layout()
    plt.show()
```

14. Analysis of Shareholders and Institutional Ownership

Let's look at the major holders of THYAO:

python

```python
# Get major holders
major_holders = thyao.major_holders
print("\nMajor Holders:")
print(major_holders)

# Get institutional holders
institutional_holders = thyao.institutional_holders
print("\nInstitutional Holders:")
print(institutional_holders)
```

15. Putting It All Together: Creating a Dashboard for BIST Stocks

Finally, let's create a comprehensive dashboard for analyzing BIST stocks:

python

```python
def analyze_stock(ticker, period="1y"):
    """Create a comprehensive analysis for a given stock"""
    stock = yf.Ticker(ticker)
    data = stock
```