# NumPy vs Pandas, Tickers, Plotting & Imports — Q&A (Set 25)

## Q1. NumPy array vs Pandas DataFrame + conversion

NumPy ndarray: homogeneous n-D numeric container, no row/col labels.  
Pandas DataFrame: 2-D, labeled axes, mixed dtypes, rich I/O/groupby/merge/rolling.  
Convert: df = pd.DataFrame(ndarray); ndarray2 = df.to\_numpy(); s = pd.Series(ndarray[:,0]).

## Q2. Stock ticker input issues and handling

Problems: typos, unknown/delisted symbols, wrong exchange suffix, case/whitespace, missing data, network errors.  
Solutions: normalize input (.strip().upper()), validate against symbol directory, provider format mapping, retry/backoff for API calls, did-you-mean suggestions, clear error messages.

## Q3. Stock chart plotting techniques

Line/area charts of Close; candlestick/OHLC via mplfinance; volume bars; moving averages (20/50/200); bands/indicators (Bollinger, RSI, MACD); log scale; annotations for splits/earnings.

## Q4. Why legend is essential

Multiple series (price, MAs, volume) look similar; legend clarifies, avoids misinterpretation, improves accessibility and documents plotted signals.

## Q5. Limiting DataFrame to less than a year

Use datetime filtering with DateOffset:  
today = pd.Timestamp.today(); one\_year = today - pd.DateOffset(years=1);  
df\_year = df.loc[df.index >= one\_year]  
Or approximate trading year: df\_252 = df.tail(252).

## Q6. Definition of 180-day moving average

Mean of last 180 observations (≈ 9 months trading days).  
df['MA180'] = df['Close'].rolling(180).mean().  
Calendar window: df['MA180c'] = df['Close'].rolling('180D').mean().

## Q7. Indirect importing in Python

Dynamic import of module by string name at runtime:  
import importlib; mod = importlib.import\_module('pkg.module');  
func = getattr(mod, 'main', None); if callable(func): func().