14/02/2025

**Python for Data Professionals in Finance**

This comprehensive learning path is designed for data professionals in the finance industry who want to leverage the power of Python. Learn best practices for performing calculations; ingesting and preparing financial data; integrating Python with Excel for data manipulation, analysis, and reporting; creating dynamic financial models; and using advanced machine learning models to enhance your trading strategies.

* **Explore**, analyze, and visualize financial data with Python.
* **Unlock** the power of Python functionality within Excel.
* **Build** dynamic financial models for trading strategies.

Course 1 : **Getting Started with Python for Finance**

**What you should know**

* **Basic Python Knowledge**: You should have a basic understanding of Python, including concepts like variables, loops, and functions.
* **Experience with Jupyter Notebooks**: Familiarity with Jupyter Notebooks will be helpful as it allows you to create and share code, visualizations, and other content in a single document.
* **Excel Formulas**: Knowledge of basic Excel formulas (like sum, average, and if) will be beneficial, even though Excel is not directly used in this course.

**Loading data**

* **Importing Libraries**: The video demonstrates how to import essential libraries like matplotlib, NumPy, Pandas, and Y Finance, and the use of aliasing for convenience.
* **Loading Data**: The instructor shows how to use the Y Finance library to download historical data for the S&P 500 and Apple from 2010 to 2019, resulting in a Pandas DataFrame.
* **Inspecting Data**: The video covers inspecting the DataFrame, understanding its structure, and using the pipe method to chain operations for better readability and maintainability of the code.

These steps are crucial for preparing financial data for analysis using Python.

**raw.pipe? – pulls the documentation for pipe in Jupyter notebook. Whenever we put a ? after a method or function Jupyter will pull a documentation for it.**

15/02/2025

**Line plots**

* **Creating Line Plots**: The video demonstrates how to create line plots using the Pandas library, which is built on top of Matplotlib.
* **Plotting Data**: It explains how to plot data with the date index on the X-axis and each column as its own line.
* **Customizing Plots**: The video shows how to customize plots, such as changing the figure size and plotting specific columns like the Close or Volume columns.

**Chaining operations in Pandas is important for several reasons:**

* **Readability**: Chaining makes the code easier to read and understand, as it flows like a recipe. Each step is clear and sequential.
* **Efficiency**: It allows for more concise code, reducing the need for intermediate variables and making the code more efficient.
* **Maintainability**: Chained operations are easier to maintain and debug, as each step of the data manipulation process is clearly defined.

By chaining, you can perform complex data transformations in a streamlined and organized manner, which is particularly useful when working with financial data in your field.

Bottom of Form

**Resampling data**

* **Resampling Data**: The video explains how to change the granularity of data using the Pandas library, such as converting daily data to monthly or quarterly data.
* **Using Pandas**: It demonstrates how to use the resample method in Pandas with different offset aliases (e.g., 'M' for month, 'Q' for quarter) to aggregate data.
* **Aggregation Methods**: The video shows how to apply aggregation methods like mean to compute average values for the resampled data.
* **Plotting Resampled Data**: Once resampled, the data can be easily plotted to visualize trends over different time periods.

The steps taught in the video:

1. **Load Data**: Load your data into a Pandas DataFrame, ensuring it includes a date column.
2. **Select Columns**: Use the iloc method to select the columns you need.
3. **Fix Columns**: Apply the fixed\_columns method to collapse hierarchical columns.
4. **Resample Data**: Use the resample method with an offset alias (e.g., 'M' for month) to change the data granularity.
5. **Aggregate Data**: Apply an aggregation method, such as mean, to compute the average values for the resampled data.
6. **Plot Data**: Plot the resampled data using the plot method to visualize the results.

**Offset aliases in Pandas are strings that specify the frequency for resampling data.**

* **'M'**: Represents month. Aggregates data at the monthly level.
* **'2M'**: Represents two months. Aggregates data at the two monthly level.
* **'Q'**: Represents quarter. Aggregates data at the quarterly level.

For example, using 'M' will resample the data to monthly values, while 'Q' will resample it to quarterly values. These aliases help you easily change the granularity of your data.

Bottom of Form

**Granularity refers to the level of detail in the data.** Specifically, it describes the frequency at which data points are recorded. For example:

* **Daily Granularity**: Data points are recorded every day.
* **Monthly Granularity**: Data points are aggregated and recorded every month.
* **Quarterly Granularity**: Data points are aggregated and recorded every quarter.

The video demonstrates how to change the granularity of data using the Pandas library, allowing you to resample daily data to monthly or quarterly data for different levels of analysis.