This code performs the following tasks:

1. Imports necessary modules: **pandas**, **numpy**, and **matplotlib.pyplot**.
2. Reads in a CSV file called **startup\_funding.csv** and creates a Pandas DataFrame called **data**.
3. Creates a copy of the original DataFrame called **df**.
4. Prints the shape and head of the DataFrame using the **shape** and **head** methods.
5. Checks the data types of the DataFrame using the **dtypes** method.
6. Cleans the data by removing any rows where the **CityLocation** or **AmountInUSD** columns are null, converting the **AmountInUSD** column from a string to a float, replacing periods in the **Date** column with forward slashes, and converting the **Date** column to a Pandas datetime format.
7. Checks the data types of the DataFrame after cleaning using the **dtypes** method.
8. Replaces certain city names to ensure consistency.
9. Defines a function called **valid\_city** that takes a city name as input and returns a standardized version of the city name.
10. Applies the **valid\_city** function to the **CityLocation** column using the **apply** method.
11. Sorts and displays the unique city names in the **CityLocation** column to check for consistency after cleaning.

Overall, this code reads in and cleans a dataset of startup funding information, ensuring consistency of city names and converting relevant columns to appropriate data types for further analysis.

This code segment aims to achieve the following objectives:

1. Select data for specific cities that are required for analysis (Bangalore, Mumbai, Gurgaon, Noida, New Delhi).
2. Count the number of funding rounds for each city that was selected.
3. Extract the city names and the number of funding rounds into NumPy arrays.
4. Create a bar chart that shows the number of funding rounds received for each city.
5. Add labels to the chart showing the exact number of funding rounds received for each city.

Overall, this code segment helps to determine the location with the most number of funding rounds and creates a visual representation of the same using a bar chart.

This code analyzes the investor funding data by extracting individual investor names and counting the number of investments made by each investor. It also replaces similar investor names for consistency and creates a horizontal bar chart of the top 5 investors by number of fundings.

First, the code selects the rows that have a non-empty 'InvestorsName' column and extracts individual investor names from it. The investor names are added to a list and the number of investments made by each investor is counted and stored in a dictionary. This dictionary is then converted to a Pandas dataframe, sorted by the number of investments made, and the top 5 investors are selected.

The code then searches for similar investor names to the top 5 investors, such as 'sequoia' and 'accel', and prints out the names of investors containing those substrings. It then creates a list of substrings to search for and a list of corresponding replacements, and loops through them to replace the investor names in the dataframe. The code then creates arrays for the top 5 investors and the number of investments they made, and creates a horizontal bar chart to visualize the data.

Overall, this code provides an analysis of investor funding data by counting the number of investments made by each investor and visualizing the top 5 investors using a horizontal bar chart. It also provides a way to replace similar investor names for consistency.

This code performs data cleaning and analysis on a dataset of startup investments in India.

The code imports the necessary libraries, reads the dataset into a pandas dataframe, and selects the columns of interest ('InvestorsName', 'StartupName', 'InvestmentType'). It then drops any rows where the 'InvestorsName' or 'InvestmentType' columns are missing.

Next, the startup names are standardized by replacing variations of the same name with a single name ('Ola', 'Flipkart', 'Oyo', 'Paytm').

The code then creates a new dataframe, 'investors\_data', which has two columns: 'Investor\_Name' and 'StartupName'. It loops through each row of the original dataframe and splits the 'InvestorsName' column into individual investor names. It then adds a row to 'investors\_data' for each investor/startup pair. The investor names are standardized by replacing variations of the same name using a list of substrings to search for and a list of corresponding replacements.

The code then creates a dictionary to count the number of investments made by each investor and creates a new dataframe, 'investor\_funding', that contains the top 5 investors by number of investments. It then extracts the investor names and number of investments as numpy arrays and plots a bar chart of the number of investments made by each investor.

Finally, the code adds the number of investments as labels on top of each bar and displays the plot.

It looks like you have written code to analyze startup investment data and create a bar chart of the top 5 investors by number of investments in startups through seed and crowd funding.

First, you renamed certain values in the 'InvestmentType' column to standardize the data. Then, you created a new dataframe 'investors\_data\_new' with columns 'Investor\_Name', 'StartupName', and 'Invest\_type'. You looped through each row in the 'df\_new' dataframe, split the 'InvestorsName' column by ',', and looped through each investor name. If the investor name does not contain the word 'undisclosed', you added a new row to 'investors\_data\_new' with the investor name, startup name, and investment type.

You standardized investor names by creating a list of substrings to search for and a list of corresponding replacements, and then looping through the substrings and replacements to replace the investor names.

You created a new DataFrame to only include Seed Funding and Crowd Funding investments, dropped any duplicate investor names in the new DataFrame, and created a dictionary of investor names and the number of investments they've made.

You then created a DataFrame of the top 5 investors by number of investments and created a bar chart of the top 5 investors by number of investments. Finally, you displayed the chart using the 'plt.show()' function.

Top of Form

Bottom of Form

This code analyzes Private Equity investments in startups and plots the top 5 investors by number of investments.

First, the code filters the **investors\_data\_new** dataframe to keep only the rows with "Private Equity" in the "Invest\_type" column, and then selects the "Investor\_Name" and "StartupName" columns. The resulting dataframe is called **investors\_data\_new\_private\_fund**.

Next, the code drops any duplicate rows in **investors\_data\_new\_private\_fund** based on the "Investor\_Name" column, keeping only the first occurrence of each unique investor. It then creates a dictionary called **dic**, where the keys are investor names and the values are the number of investments they have made. The code iterates over the "Investor\_Name" column of **investors\_data\_new\_private\_fund** and updates the corresponding key in the **dic** dictionary for each investor.

The code then creates a new dataframe called **investor\_funding** with the top 5 investors by number of investments, sorted in descending order. The dataframe has two columns: "Investor\_Name" and "No\_of\_investments".

After that, the code creates two NumPy arrays: **investor\_names**, which contains the top 5 investor names, and **no\_of\_fundings**, which contains the corresponding number of investments for each investor.

The code then plots a bar chart using **plt.bar()**, with "Investor\_Name" on the x-axis and "No\_of\_investments" on the y-axis. The chart is titled "Number of Fundings by Investor in different startups through Private Equity" and the y-axis is labeled "Number of Fundings". The x-axis labels are rotated 90 degrees for readability.

Finally, the code adds labels to each bar using the **addlabels()** function, which takes the **investor\_names** and **no\_of\_fundings** arrays as arguments. The plot is displayed using **plt.show()**.