



Data Visualization with Python

Cheat Sheet : Data Preprocessing Tasks in Pandas

Task	Syntax	Description	Example
Load CSV data	pd.read_csv('filename.csv')	Read data from a CSV file into a Pandas DataFrame	df_can=pd.read_csv('data.csv')
Handling Missing Values	df.dropna()	Drop rows with missing values	df_can.dropna()
	df.fillna(value)	Fill missing values with a specified value	df_can.fillna(0)
Removing Duplicates	df.drop_duplicates()	Remove duplicate rows	df_can.drop_duplicates()
Renaming Columns	df.rename(columns={'old_name': 'new_name'})	Rename one or more columns	df_can.rename(columns={'Age': 'Years'})
Selecting Columns	df['column_name'] or df.column_name	Select a single column	df_can.Age or df_can['Age']
	df[['col1', 'col2']]	Select multiple columns	df_can[['Name', 'Age']]
Filtering Rows	df[df['column'] > value]	Filter rows based on a condition	df_can[df_can['Age'] > 30]
Applying Functions to Columns	df['column'].apply(function_name)	Apply a function to transform values in a column	df_can['Age'].apply(lambda x: x + 1)
Creating New Columns	df['new_column'] = expression	Create a new column with values derived from existing ones	df_can['Total'] = df_can['Quantity'] * df_can['Price']
Grouping and Aggregating	df.groupby('column').agg({'col1': 'sum', 'col2': 'mean'})	Group rows by a column and apply aggregate functions	df_can.groupby('Category').agg({'Total': 'mean'})
Sorting Rows	df.sort_values('column', ascending=True/False)	Sort rows based on a column	df_can.sort_values('Date', ascending=True)
Displaying First n Rows	df.head(n)	Show the first n rows of the DataFrame	df_can.head(3)
Displaying Last n Rows	df.tail(n)	Show the last n rows of the DataFrame	df_can.tail(3)
Checking for Null Values	df.isnull()	Check for null values in the DataFrame	df_can.isnull()
Selecting Rows by Index	df.iloc[index]	Select rows based on integer index	df_can.iloc[3]
	df.iloc[start:end]	Select rows in a specified range	df_can.iloc[2:5]
Selecting Rows by Label	df.loc[label]	Select rows based on label/index name	df_can.loc['Label']
	df.loc[start:end]	Select rows in a specified label/index range	df_can.loc['Age':'Quantity']
Summary Statistics	df.describe()	Generates descriptive statistics for numerical columns	df_can.describe()

Cheat Sheet : Plot Libraries

Library	Main Purpose	Key Features	Programming Language	Level of Customization	Dashboard Capabilities	Types of Plots Possible
Matplotlib	General-purpose plotting	Comprehensive plot types and variety of customization options	Python	High	Requires additional components and customization	Line plots, scatter plots, bar charts, histograms, pie charts, box plots, heatmaps, etc.
Pandas	Fundamentally used for data manipulation but also has plotting functionality	Easy to plot directly on Panda data structures	Python	Medium	Can be combined with web frameworks for creating dashboards	Line plots, scatter plots, bar charts, histograms, pie charts, box plots, etc.
Seaborn	Statistical data visualization	Stylish, specialized statistical plot types	Python	Medium	Can be combined with other libraries to display plots on dashboards	Heatmaps, violin plots, scatter plots, bar plots, count plots, etc.
Plotly	Interactive data visualization	interactive web-based visualizations	Python, R, JavaScript	High	Dash framework is dedicated for building interactive dashboards	Line plots, scatter plots, bar charts, pie charts, 3D plots, choropleth maps, etc.
Folium	Geospatial data visualization	Interactive, customizable maps	Python	Medium	For incorporating maps into dashboards, it can be integrated with other frameworks/libraries	Choropleth maps, point maps, heatmaps, etc.

Library	Main Purpose	Key Features	Programming Language	Level of Customization	Dashboard Capabilities	Types of Plots Possible
PyWaffle	Plotting Waffle charts	Waffle charts	Python	Low	Can be combined with other libraries to display waffle chart on dashboards	Waffle charts, square pie charts, donut charts, etc.