```
# === avg_state_rain_pandasChart.ipynb ===

#%%
%matplotlib notebook
get_ipython().run_line_magic('matplotlib', 'notebook')


#%%
# Dependencies
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

#%% [markdown]
# ### Using MatplotLib to Chart a DataFrame

#%%
# ------------------------------------------------------------------------------
# So far we've been using PyPlot, and it took a lot of code to create a bar
# chart of average rainfall by state
# ------------------------------------------------------------------------------


#%%
# Load in csv
rain_df = pd.read_csv("../Resources/avg_rain_state.csv")
rain_df.head()


#%%
# Set x axis and tick locations
x_axis = np.arange(len(rain_df))
tick_locations = [value for value in x_axis]


#%%
# Create a list indicating where to write x labels and set figure size to adjust for space
plt.figure(figsize=(20,3))
plt.bar(x_axis, rain_df["Inches"], color='r', alpha=0.5, align="center")
plt.xticks(tick_locations, rain_df["State"], rotation="vertical")


#%%
# Set x and y limits
plt.xlim(-0.75, len(x_axis))
plt.ylim(0, max(rain_df["Inches"])+10)


#%%
# Set a Title and labels
plt.title("Average Rain per State")
plt.xlabel("State")
plt.ylabel("Average Amount of Rainfall in Inches")


```

```
#%%
# Save our graph and show the grap
plt.tight_layout()
plt.savefig("../Images/avg_state_rain.png")
plt.show()

#%% [markdown]
# ### Using Pandas to Chart a DataFrame

#%%
# ----------------------------------------------------------------------------
# The original DataFrame is being cut down to only those values which the
# application should chart. The index for the DataFrame is then set to the
# State column so that Pandas will use these values later on to create the chart
# ----------------------------------------------------------------------------


#%%
# Filter the DataFrame down only to those columns to chart
state_and_inches = rain_df[["State","Inches"]]

# Set the index to be "State" so they will be used as labels
state_and_inches = state_and_inches.set_index("State")

state_and_inches.head()


#%%
# Use DataFrame.plot() in order to create a bar chart of the data

# ----------------------------------------------------------------------------
# `DataFrame.plot()` is called and the parameters `kind="bar"` and
# `figsize=(20,3)` are passed into it. This tells Pandas to create a new bar
# chart using the values stored within the DataFrame. The values stored within
# the index will be the labels for the X axis while the values stored within
# the other column will be used to plot the Y axis

state_and_inches.plot(kind="bar", figsize=(20,3))

# ----------------------------------------------------------------------------
# The chart can still be edited just like any other kind of PyPlot as well
# For example, the title for the chart can still be set using `plt.title()

# Set a title for the chart

plt.title("Average Rain Per State")

plt.show()
plt.tight_layout()

# ----------------------------------------------------------------------------
# Note: the bar chart produced is automatically styled. The header for the
# index is now the label for the X axis while the header for the other
# column has been placed inside of a legend

```

```
111
112 #%%
113 # Pandas can also plot multiple columns if the DataFrame includes them
114 # *** AND change the "kind" that is being passed as a parameter
115
116 multi_plot = rain_df.plot(kind="bar", figsize=(20,5))
117
118 # ----------------------------------------------------------------------------
119 # It is also possible to modify a specific Pandas plot by storing the plot
120 # within a variable and then using built-in methods to modify it.
121 # For example: `PandasPlot.set_xticklabels()` will allow the user to
122 # modify the tick labels on the X axis without having to manually set the
123 # DataFrame's index
124
125 # PandasPlot.set_xticklabels() can be used to set the tick labels as well
126 multi_plot.set_xticklabels(rain_df["State"], rotation=45)
127
128 plt.show()
129 plt.tight_layout()
```