Working hard

Exercise

Several police officers have been working hard to help us solve the mystery of Bayes, the kidnapped Golden Retriever. Their commanding officer wants to know exactly how hard each officer has been working on this case. Officer Deshaun has created DataFrames called deshaun to track the amount of time he spent working on this case. The DataFrame contains two columns:

day_of_week: a string representing the day of the week

hours_worked: the number of hours that a particular officer worked on the Bayes case'

Instructions

1-From matplotlib, import the module pyplot under the alias plt

2-Plot Officer Deshaun's hours worked using the columns day_of_week and hours_worked from deshaun.

3-Display Deshaun's plot

In [ ]:

```
# From matplotlib, import pyplot under the alias plt
from ____ import ____ as ____
# Plot Officer Deshaun's hours_worked vs. day_of_week
plt.____(deshaun.____, deshaun._____)

#_____________________#
#Solutions

# From matplotlib, import pyplot under the alias plt
from matplotlib import pyplot as plt

# Plot Officer Deshaun's hours_worked vs. day_of_week
plt.plot(deshaun['day_of_week'], deshaun['hours_worked']) #and we can write this code w
ith below..
#(plt.plot(deshaun.day_of_week, deshaun.hours_worked))

# Display Deshaun's plot
plt.show()
```

Or hardly working?

Exercise

Two other officers have been working with Deshaun to help find Bayes. Their names are Officer Mengfei and Officer Aditya. Deshaun used their time cards to create two more DataFrames: mengfei and aditya. In this exercise, we'll plot all three lines together to see who was working hard each day.

We've already loaded matplotlib under the alias plt.

Instructions

1-Plot Officer Aditya's time worked with day_of_week on the x-axis and hours_worked on the y-axis.

2-Plot Officer Mengfei's time worked with day_of_week on the x-axis and hours_worked on the y-axis.

In [ ]:

```
# Plot Officer Deshaun's hours_worked vs. day_of_week
plt.plot(deshaun.day_of_week, deshaun.hours_worked)

# Plot Officer Aditya's hours_worked vs. day_of_week
____(____, ____)

# Plot Officer Mengfei's hours_worked vs. day_of_week
____

# Display all three line plots
plt.show()

#_____________________________#
#Solutions

# Plot Officer Deshaun's hours_worked vs. day_of_week
plt.plot(deshaun.day_of_week, deshaun.hours_worked)

# Plot Officer Aditya's hours_worked vs. day_of_week
plt.plot(aditya.day_of_week, aditya.hours_worked)

# Plot Officer Mengfei's hours_worked vs. day_of_week
plt.plot(mengfei.day_of_week, mengfei.hours_worked)

# Display Deshaun's plot
plt.show()
```

Adding a legend

Exercise

Officers Deshaun, Mengfei, and Aditya have all been working with you to solve the kidnapping of Bayes. Their supervisor wants to know how much time each officer has spent working on the case.

Deshaun created a plot of data from the DataFrames deshaun, mengfei, and aditya in the previous exercise. Now he wants to add a legend to distinguish the three lines

Instructions

1-Using the keyword label, label Deshaun's plot as "Deshaun".

2-Add labels to Mengfei's ("Mengfei") and Aditya's ("Aditya") plots.

In [ ]:

```
# Officer Deshaun
plt.plot(deshaun.day_of_week, deshaun.hours_worked, ....)

# Add a label to Aditya's plot
plt.plot(aditya.day_of_week, aditya.hours_worked)

# Add a label to Mengfei's plot
plt.plot(mengfei.day_of_week, mengfei.hours_worked)

# Add a command to make the legend display
____

# Display plot
plt.show()

#_____________________________#
#Solutions

# Officer Deshaun
plt.plot(deshaun.day_of_week, deshaun.hours_worked, label='Deshaun')

# Add a label to Aditya's plot
plt.plot(aditya.day_of_week, aditya.hours_worked,label='Aditya')

# Add a label to Mengfei's plot
plt.plot(mengfei.day_of_week, mengfei.hours_worked,label='Mengfei')

# Add a command to make the legend display
plt.legend()

# Display plot
plt.show()
```

Adding labels

Exercise

If we give a chart with no labels to Officer Deshaun's supervisor, she won't know what the lines represent.

We need to add labels to Officer Deshaun's plot of hours worked.

Instructions

1-Add a descriptive title to the chart.

2-Add a label for the y-axis.

In [ ]:

```
# Lines
plt.plot(deshaun.day_of_week, deshaun.hours_worked, label='Deshaun')
plt.plot(aditya.day_of_week, aditya.hours_worked, label='Aditya')
plt.plot(mengfei.day_of_week, mengfei.hours_worked, label='Mengfei')

# Add a title
plt.____(____)

# Add y-axis label
plt.____(____)

# Legend
plt.legend()
# Display plot
plt.show()

#_____#
#Solutions

# Lines
plt.plot(deshaun.day_of_week, deshaun.hours_worked, label='Deshaun')
plt.plot(aditya.day_of_week, aditya.hours_worked, label='Aditya')
plt.plot(mengfei.day_of_week, mengfei.hours_worked, label='Mengfei')

# Add a title
plt.title('descriptive')

# Add y-axis label
plt.ylabel('y-axis')

# Legend
plt.legend()
# Display plot
plt.show()
```

Adding floating text

Exercise

Officer Deshaun is examining the number of hours that he worked over the past six months. The number for June is low because he only had data for the first week. Help Deshaun add an annotation to the graph to explain this.

Instructions

Place the annotation "Missing June data" at the point (2.5, 80)

In [ ]:

```
# Create plot
plt.plot(six_months.month, six_months.hours_worked)

# Add annotation "Missing June data" at (2.5, 80)
____.____(____)

# Display graph
plt.show()

#_____________________________#
#Solutions

# Create plot
plt.plot(six_months.month, six_months.hours_worked)

# Add annotation "Missing June data" at (2.5, 80)
plt.text(2.5,80,"Missing June data")

# Display graph
plt.show()
```

Tracking crime statistics

Exercise

Sergeant Laura wants to do some background research to help her better understand the cultural context for Bayes' kidnapping. She has plotted Burglary rates in three U.S. cities using data from the Uniform Crime Reporting Statistics.

She wants to present this data to her officers, and she wants the image to be as beautiful as possible to effectively tell her data story.

Recall:

You can change linestyle to dotted (':'), dashed('--'), or no line ('').

You can change the marker to circle ('o'), diamond('d'), or square ('s').

Instructions

1-Change the color of Phoenix to "DarkCyan".

2-Make the Los Angeles line dotted.

3-Add square markers to Philadelphia

In [ ]:

```
# Change the color of Phoenix to `"DarkCyan"`
plt.plot(data["Year"], data["Phoenix Police Dept"], label="Phoenix", ____)

# Make the Los Angeles line dotted
plt.plot(data["Year"], data["Los Angeles Police Dept"], label="Los Angeles", ____)

# Add square markers to Philedelphia
plt.plot(data["Year"], data["Philadelphia Police Dept"], label="Philadelphia", ____)

# Add a legend
plt.legend()

# Display the plot
plt.show()

#_____________________________#
#Solutions

# Change the color of Phoenix to `"DarkCyan"`
plt.plot(data["Year"], data["Phoenix Police Dept"], label="Phoenix", color='DarkCyan')

# Make the Los Angeles line dotted
plt.plot(data["Year"], data["Los Angeles Police Dept"], label="Los Angeles", linestyle=
':')

# Add square markers to Philedelphia
plt.plot(data["Year"], data["Philadelphia Police Dept"], label="Philadelphia", marker=
's')

# Add a legend
plt.legend()

# Display the plot
plt.show()
```

Playing with styles

Exercise

Help Sergeant Laura wants to try out a few different style options. Changing the plotting style is a fast way to change the entire look of your plot without having to update individual colors or line styles. Some popular styles include:

'fivethirtyeight' - Based on the color scheme of the popular website

'grayscale' - Great for when you don't have a color printer!

'seaborn' - Based on another Python visualization library

'classic' - The default color scheme for Matplotlib

Instructions

1-Change the plotting style to "fivethirtyeight".

2-Change the plotting style to "ggplot".

3-View all styles by typing print(plt.style.available) in the console

In [ ]:

```
# Change the style to fivethirtyeight
plt.____.____(____)

# Change the style to ggplot
plt.style.use(....)

# Choose any of the styles
print(plt.style.available)   #To show all styles
plt.style.use(...)        #Choose any style

# Plot lines
plt.plot(data["Year"], data["Phoenix Police Dept"], label="Phoenix")
plt.plot(data["Year"], data["Los Angeles Police Dept"], label="Los Angeles")
plt.plot(data["Year"], data["Philadelphia Police Dept"], label="Philadelphia")

# Add a legend
plt.legend()

# Display the plot
plt.show()


#_____________________________________#
#Solutins
# Change the style to fivethirtyeight
plt.style.use('fivethirtyeight')

# Change the style to ggplot
plt.style.use('ggplot')

# Choose any of the styles
print(plt.style.available)   #To show all styles
plt.style.use(...)           #Choose any style
```

Identifying Bayes' kidnapper

Exercise

We've narrowed the possible kidnappers down to two suspects:

Fred Frequentist (suspect1)

Gertrude Cox (suspect2)

The kidnapper left a long ransom note containing several unusual phrases. Help DataCamp by using a line plot to compare the frequency of letters in the ransom note to samples from the two main suspects.

Three DataFrames have been loaded:

ransom contains the letter frequencies for the ransom note.

suspect1 contains the letter frequencies for the sample from Fred Frequentist.

suspect2 contains the letter frequencies for the sample from Gertrude Cox.

Each DataFrame contain two columns letter and frequency.

Instructions

1-Plot the letter frequencies from the ransom note. The x-values should be ransom.letter. The y-values should be ransom.frequency. The label should be the string 'Ransom'. The line should be dotted and gray.

2-Plot a line for the data in suspect1. Use a keyword argument to label that line 'Fred Frequentist').

3-Plot a line for the data in suspect2 (labeled 'Gertrude Cox').

4-Label the x-axis (Letter) and the y-axis (Frequency), and add a legend

In [ ]:

```
# x should be ransom.letter and y should be ransom.frequency
plt.plot(____.____, ____.____,
         # Label should be "Ransom"
         ____="Ransom",
         # Plot the ransom letter as a dotted gray line
         ____=':', ____='gray')

# X-values should be suspect1.letter
# Y-values should be suspect1.frequency
# Label should be "Fred Frequentist"
plt.plot(____, ____, ____=____)

# X-values should be suspect2.letter
# Y-values should be suspect2.frequency
# Label should be "Gertrude Cox"
____

# Add x- and y-labels
plt.____("Letter")
plt.____("Frequency")

# Add a legend
plt.____()


# Display the plot
plt.show()


#_____#
#Solutions

# x should be ransom.letter and y should be ransom.frequency
plt.plot(ransom.letter, ransom.frequency,
         # Label should be "Ransom"
         label ="Ransom",
         # Plot the ransom letter as a dotted gray line
         linestyle=':', color='gray')
# X-values should be suspect1.letter
# Y-values should be suspect1.frequency
# Label should be "Fred Frequentist"
plt.plot(suspect1.letter, suspect1.frequency, label='Fred Frequentist')

# X-values should be suspect2.letter
# Y-values should be suspect2.frequency
# Label should be "Gertrude Cox"
plt.plot(suspect2.letter, suspect2.frequency,label='Gertrude Cox')

# Add x- and y-labels
plt.xlabel("Letter")
plt.ylabel("Frequency")

# Add a legend
plt.legend()

# Display the plot
plt.show()
```