Learning Objectives

Students will be able to...

- **Create column and bar plots**
- **Create a stacked column and bar charts**

definition

Assumptions

- Learners are comfortable working around Jupyter and matplotlib.

Limitations

In this assignment, we only work with the matplotlib library.

Column Charts and Chart Titles

Column charts are one of the most common visualizations. Often, column charts and **bar charts** are used interchangeably, but for the purposes of this course, we will refer to column charts as charts containing **vertical** bars and bar charts as charts containing **horizontal** bars.

In order to work on our visualizations, we are going to use matplotlib:

```
import matplotlib.pyplot as plt
```

To create a column chart, you use the following syntax:

```
plt.bar(xAxis,yAxis)
plt.show()
```

- `xAxis` represents the data on the x-axis.
- `yAxis` represents the data on the y-axis.

Now let's try it out!

Creating a Column Chart

Add the following code to the Jupyter Notebook on the left to create some data that can be used to create a **column chart**.

```
days =["Sunday", "Monday", "Tuesday", "Wednesday",
          "Thursday", "Friday", "Saturday"]
max_temp = [65, 54, 32, 35, 44, 40, 58]
```

The code above creates lists: `days` and `max_temp`.

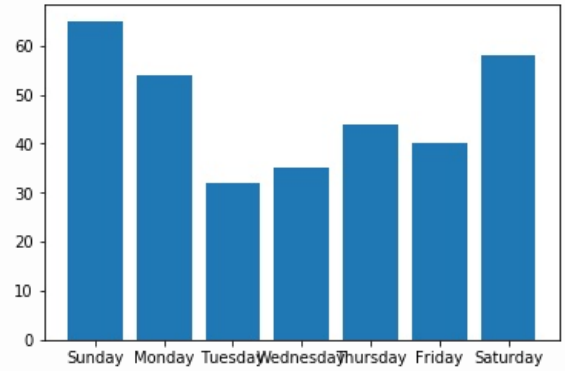
Let's create our column chart:

```
import matplotlib.pyplot as plt

plt.bar(days,max_temp)
plt.show()
```

Remember that your independent variable goes **first**. The independent variable should be on the x-axis or the horizontal axis.

Plot Result:

Click the `Run` button to display your chart.



.guides/img/column_chart_1

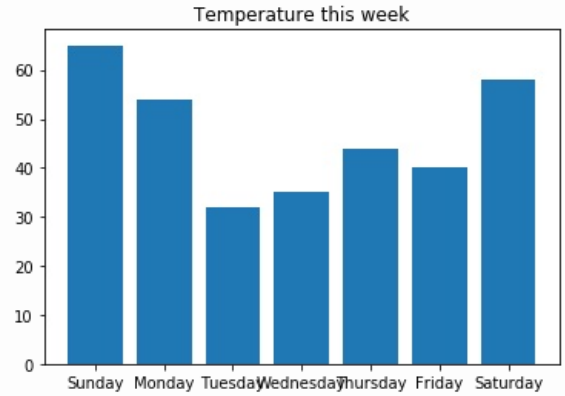
Customizing your Chart: Chart Titles

We can add a title to our column charts to make them understandable to others. Best practice is to always include a title.

Let's start by adding a title! Add the following line of code to your existing code cell and re-run it:

```
import matplotlib.pyplot as plt

plt.bar(days,max_temp)
plt.title('Temperature this week') #add this line
plt.show()
```

The `.title()` method takes a string of text and sets it as the title.



.guides/img/column_chart_2

challenge

Try these variations:

- `python   plt.title('Temperature this week', size='large')`
- `python   plt.title('Temperature this week', weight='bold')`
- You can even combine the properties! If your code gets too long, you can put each property on a new line for readability:
  ```
  python   plt.title('Temperature this week',
  size='large',              weight='bold')
  ```

There are dozens of properties you can set – check out the official documentation for the complete list.

Bar Charts and Axis Labels

The next chart we will learn about is a **bar chart**. Remember that we differentiate between a column chart and bar chart based on the orientation of their bars. Column charts contain **vertical** bars while bar charts contain **horizontal** bars.

info

Creating bar charts

We can show the exact same chart horizontally using `plt.barh()`. We can think of the **h** as meaning **h**orizontal bar charts.

To create a bar chart, you use the following syntax:

```
plt.barh(xAxis,yAxis)
plt.show()
```

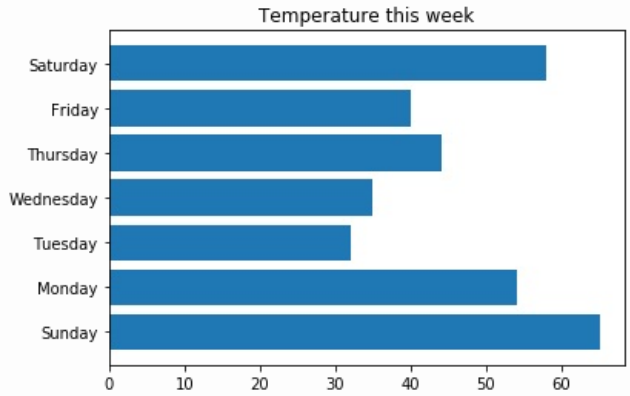
Now let's try it out!

Creating Bar Charts

We will revisit the data we wrangled earlier by re-adding the following code into the text editor:

```
import matplotlib.pyplot as plt

days =["Sunday", "Monday", "Tuesday", "Wednesday",
          "Thursday", "Friday", "Saturday"]
max_temp = [65, 54, 32, 35, 44, 40, 58]

plt.barh(days,max_temp)
plt.title('Temperature this week')
plt.show()
```

Please Note: You should use `plt.barh` instead of `plt.bar`.



.guides/img/bar_chart_1

Customizing your Chart: Axis Labels

We can add labels to our axes to make them understandable to others. Best practice is to always include axis labels – particularly if there is any kind of unit associated with the data.

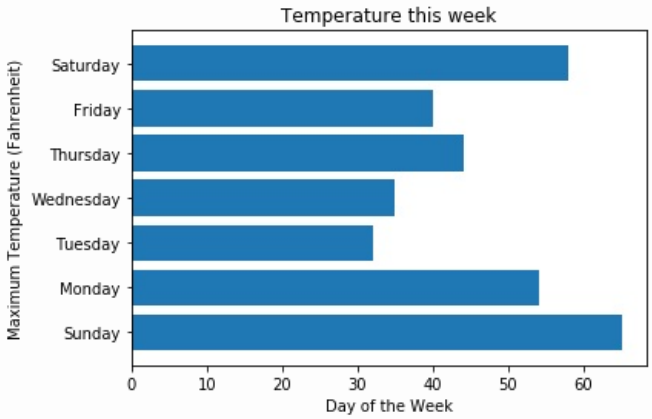
Let's start by adding an x-axis label! Add the following line of code to your existing code cell and re-run it:

```
plt.barh(days,max_temp)
plt.title('Temperature this week')
plt.xlabel('Day of the Week') # add this line
plt.show()
```

The `.xlabel()` method takes a string of text and sets it as the x-axis label.

Now let's add the y-axis label! Add the following line of code to your existing code cell and re-run it:

```
plt.barh(days,max_temp)
plt.title('Temperature this week')
plt.xlabel('Day of the Week')
plt.ylabel('Maximum Temperature (Fahrenheit)') # add this line
plt.show()
```



.guides/img/bar_chart_2

The `.ylabel()` method takes a string of text and sets it as the y-axis label. Notice that because temperature has a unit (in this case Fahrenheit), we can include this information in the label!

In this particular example, it's pretty clear, but what if the temperatures were 22, 30, 28, 27, 24, 29, 27? In Celsius, that is a hot summer week, and in Fahrenheit that is a cold winter week – the unit is very important!

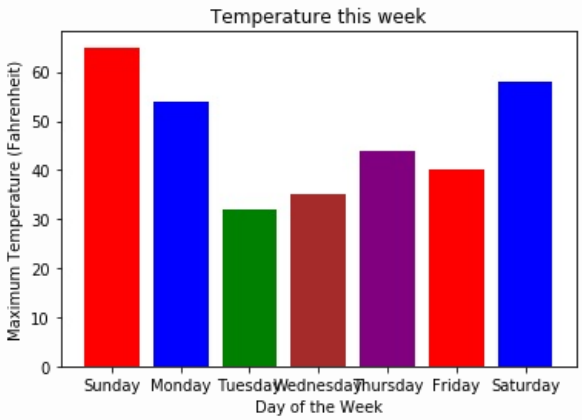
challenge

Try these variations:

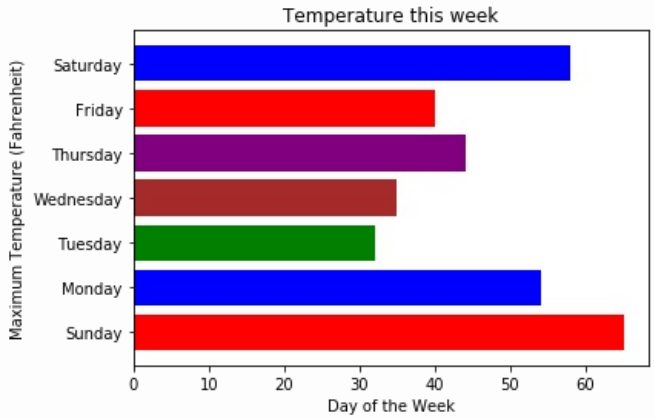
- ```
  plt.ylabel('Maximum Temperature (Fahrenheit)',
             family='monospace')
  ```
- ```
  plt.ylabel('Maximum Temperature (Fahrenheit)',
             style='italic')
  ```
- ```
  plt.ylabel('Maximum Temperature (Fahrenheit)',
            family='monospace',
            style='italic')
  ```

There are dozens properties you can set – check out the official documentation for the complete list.
  ```

Adding Color to Column and Bar Charts

Now that you can add a title and labels to your charts, let's get to something really fun - colors!

We can use the `color` property of the `bar` method so that each day is represented by a different color. Try out the code below in a new code cell to create a colorful column chart:

```
import matplotlib.pyplot as plt

days =["Sunday", "Monday", "Tuesday", "Wednesday",
          "Thursday", "Friday", "Saturday"]
max_temp = [65, 54, 32, 35, 44, 40, 58]
color_list = ['red','blue','green','brown','purple']

plt.bar(days,max_temp,color=color_list)
plt.title('Temperature this week')
plt.xlabel('Day of the Week')
plt.ylabel('Maximum Temperature (Fahrenheit)')
plt.show()
```



.guides/img/column_chart_colors

The `barh` method has the same `color` parameter. Try out the code below in a new code cell to create a colorful bar chart:

```
import matplotlib.pyplot as plt

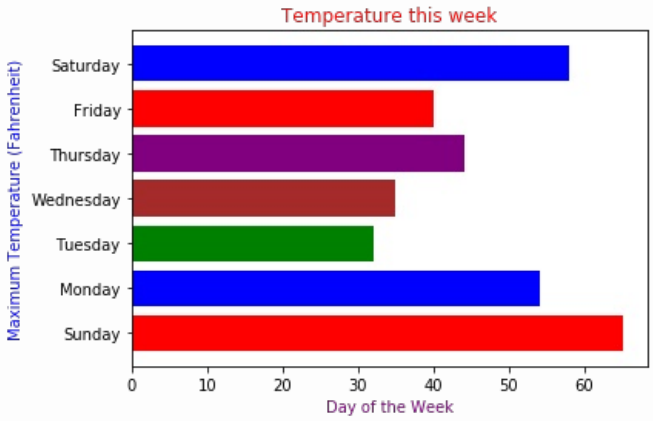
days =["Sunday", "Monday", "Tuesday", "Wednesday",
          "Thursday", "Friday", "Saturday"]
max_temp = [65, 54, 32, 35, 44, 40, 58]
color_list = ['red','blue','green','brown','purple']

plt.barh(days,max_temp,color=color_list)
plt.title('Temperature this week')
plt.xlabel('Day of the Week')
plt.ylabel('Maximum Temperature (Fahrenheit)')
plt.show()
```



.guides/img/bar_chart_colors

Specifying Colors

So far, we have been using pretty basic color names, but there are a lot of options:

We have been using **X11/CSS4** colors in the examples above, but you can use the `xkcd` colors by prefacing the color with `xkcd:`. For example, you swap out your `color_list` for the following:

```
color_list = ['xkcd:hot pink','xkcd:apple green','xkcd:puke
        green',
              'xkcd:baby blue','xkcd:bright pink']
```

There is actually a long list of xkcd colors.

challenge

Try these variations:

- ```
  color_list = ['#0f0f0f','#000000']
  ```
- ```
  color_list = ['b','c']
  ```
- ```
  color_list = ['(0.1, 0.2, 0.5)','(0.5, 0.1, 0.2)']
  ```

There are many ways to set colors – check out the official documentation for the complete list.

Setting Title and Axis Label Colors

The `title`, `xlabel`, and `ylabel` methods also have a `color` property you can set. Try setting the text color by running the code below:

```
import matplotlib.pyplot as plt

days =["Sunday", "Monday", "Tuesday", "Wednesday",
          "Thursday", "Friday", "Saturday"]
max_temp = [65, 54, 32, 35, 44, 40, 58]
color_list = ['red','blue','green','brown','purple']

plt.barh(days,max_temp,color=color_list)
plt.title('Temperature this week', color='red')
plt.xlabel('Day of the Week', color='purple')
plt.ylabel('Maximum Temperature (Fahrenheit)', color='blue')
plt.show()
```
```

You should see a plot similar to:



Stacked Bar Charts

We have now plotted our bar charts horizontally or vertically and learned how to specify colors, so we can now create **stacked bar charts.** Stacked bar charts are used to describe a parts to whole comparison.

For this example, our data is the total number of university students in each class or grade:

```
grades = ['Freshman','Sophomore','Junior','Senior']
in_state=[40,31,15,14]
out_of_state=[60,40,33,18]
```

Stacked Column and Bar Charts

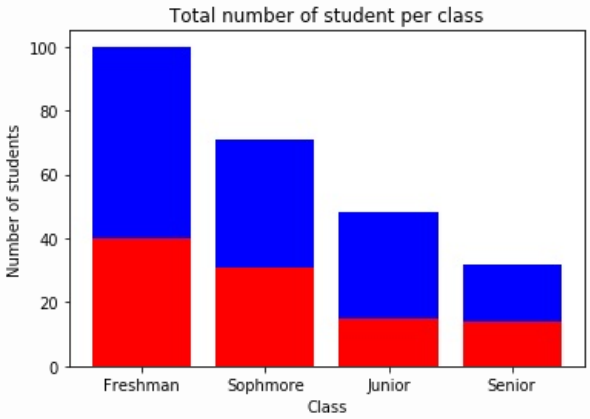
We can use the basic syntax below to create our stacked column chart:

```
import matplotlib.pyplot as plt

plt.bar(grades, in_state, color='r')
plt.bar(grades, out_of_state, bottom=in_state, color='b')
plt.show()
```

Remember to add a title and label on to your plot:

```
plt.xlabel("Class")
plt.ylabel("Number of students")
plt.title("Total number of student per class")
```

Plot Result:

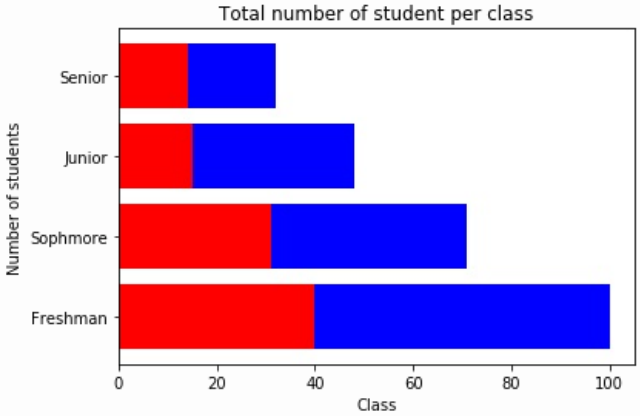


Stacked Bar Chart

The syntax is similar, however we need to replace `bottom` with `left`:

```
import matplotlib.pyplot as plt

grades = ['Freshman','Sophomore','Junior','Senior']
in_state=[40,31,15,14]
out_of_state=[60,40,33,18]

plt.barh(grades, in_state, color='r')
plt.barh(grades, out_of_state, left=in_state, color='b')

plt.xlabel("Class")
plt.ylabel("Number of students")
plt.title("Total number of student per class")

plt.show()
```

Plot Result:



Customizing your Chart: Adding a Legend

The above plots are nice – but it is unclear what the colors mean. To remedy this we can add a legend.

Let's revisit our column chart. To add a legend, we need to add a value to the `label` property for each segment of the `bar`. We also need to add the `plt.legend()` method call:

```
import matplotlib.pyplot as plt

grades = ['Freshman','Sophomore','Junior','Senior']
in_state=[40,31,15,14]
out_of_state=[60,40,33,18]

plt.bar(grades, in_state, color='r',
        label='In-State Students')          # add label property
plt.bar(grades, out_of_state, bottom=in_state, color='b',
        label='Out-of-State Students')      # add label property

plt.legend()                                #add this line

plt.xlabel("Class")
plt.ylabel("Number of students")
plt.title("Total number of student per class")

plt.show()
```

Plot Result: