1 GBA 465 Module 10 Part 01 - Data Visualization with Matplotlib

2 A picture is worth a thousand words.

You and your audience need to understand inferences from data by viewing it in a **visual context**.

A visual context helps to detect:

- Trends
- Patterns
- Correlations
- Useful Insights

3 Fundametal Data Visualizations

We will start by learning to create **fundamental** data visualizations.

- Scatter Plots
- Line Charts
- Bar Charts
- Pie Charts
- Histograms

We can refer to these generally as plots, graphs, or charts.

4 Advanced Data Visualizations

In the later parts today, we will also explore more **advanced** data visualizations:

- Box Plots
- Heat Maps
- Faceting
- Pair Plots

As your skill progresses, you can move into additional advanced plots which are live, interactive, or highly customized.

5 Python Packages for Data Visualizations

We will use the following **Python packages** to make data visualizations:

- Matplotlib
- Seaborn
- Pandas

Given our course learning objectives for programming, we will focus on **syntax**, not on graph **interpretation**.

6 Matplotlib

Matplotlib is the most popular plotting library for Python.

- Low-level library
- Requires more coding, but offers flexibility

Matplotlib is most useful for basic graphs and charts.

- Scatter Plots
- Line Chart
- Bar Charts
- Pie Charts
- Histograms

You can find documentation, examples, and tutorials on the Matplotlib web site:

- https://matplotlib.org/ (https://matplotlib.org/)

7 Installing Matplotlib

You should already have Matplotlib on your machine from Anaconda.

If not, you can install it using either of the following:

- `pip install matplotlib` (from Python command line)
- `conda install matplotlib` (from Anaconda Powershell Prompt)

8 Importing Matplotlib

Like Pandas, you must **import** matplotlib before you can use it.

Let's import both pandas and matplotlib:

In [3]:
```
# import the Pandas package so we can use it
import pandas as pd

# import matplotlib
import matplotlib.pyplot as plot
```

8.1 Data Set 1: Iris

Let's import two popular data sets for learning data analytics, using Pandas.

The first is the **Iris** data set, used to classify iris flowers by four factors.

In [5]:
```
# import the Iris data set into a Pandas DataFrame
iris = pd.read_csv ("iris.csv")

# print some records from the DataFrame to make sure it worked
iris.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

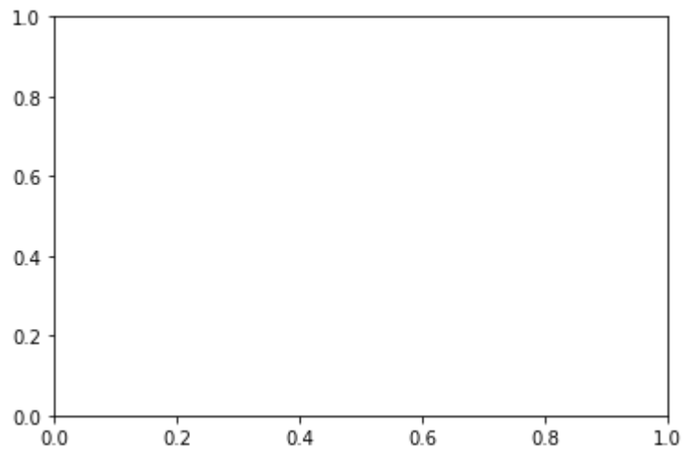
9 Matplotlib: Scatter Plot - Axis

Once matplotlib is imported, you can create **axis subplot** for a **scatterplot**.

An axis includes the **data**.

If it works, we should just get an empty plot on the screen, which is a good start!

In [6]:
```
# create an axis for the scatterplot
axis = plot.subplots()
```



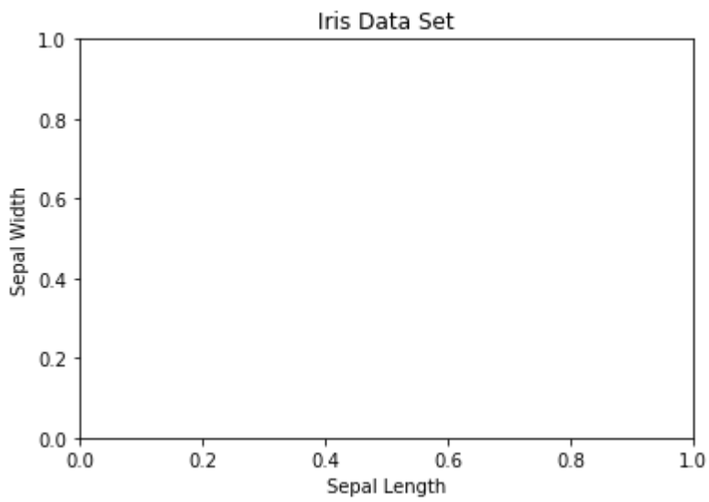
10 Matplotlib: Scatter Plot - Figure

Now that we have our axis subplot, let's try again but also add a **figure subplot**.

A figure includes the **title** and **labels** for both the x and y axis.

In [7]:
```
# create an axis and a figure for the scatterplot
figure, axis = plot.subplots()

# set title and labels for the figure
axis.set_title ("Iris Data Set")
axis.set_xlabel ("Sepal Length")
axis.set_ylabel ("Sepal Width")
```

Out[7]:
```
Text(0, 0.5, 'Sepal Width')
```



11 Matplotlib: Scatter Plot - Data

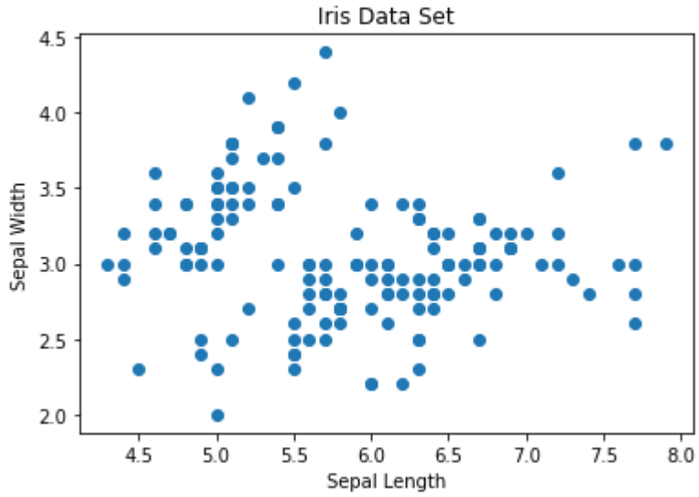
Now that we have our figure and our axis subplots, it's time to add our **data**.

In this case, we want to plot:

- `Sepal Length` along the x-axis
- `Sepal Width` along the y-axis

In [8]:
```
# create an axis and a figure for the scatterplot
figure, axis = plot.subplots()

# set title and labels for the figure
axis.set_title ("Iris Data Set")
axis.set_xlabel ("Sepal Length")
axis.set_ylabel ("Sepal Width")

# create a scatter plot of the Sepal Length (x-axis) vs. the Sepal Width (y-axis)
axis.scatter (iris ["sepal_length"], iris ["sepal_width"])
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x26e9c40a310>
```

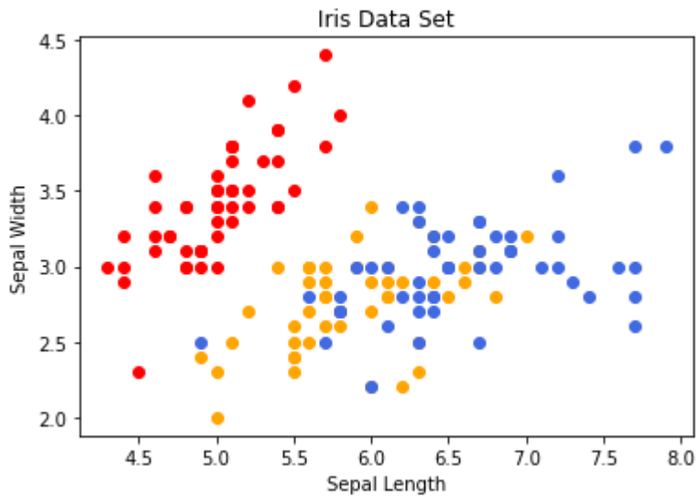


12 Matplotlib: Scatter Plot - Colors

Let's use the color of dots to show the **different iris flowers** for the data points.

To use colors, we will **plot each point individually** by iterating over data in the Pandas DataFrame.

In [6]:
```
# create an axis and a figure for the scatterplot
figure, axis = plot.subplots()

# set title and labels for the figure
axis.set_title ("Iris Data Set")
axis.set_xlabel ("Sepal Length")
axis.set_ylabel ("Sepal Width")

# create a color dictionary
colors = { "Iris-setosa": "red", "Iris-versicolor": "orange", "Iris-virginica": '

# use the index to iterate over the rows
for i in iris.index:

    axis.scatter (iris ["sepal_length"][i], iris ["sepal_width"][i], color = colo
```



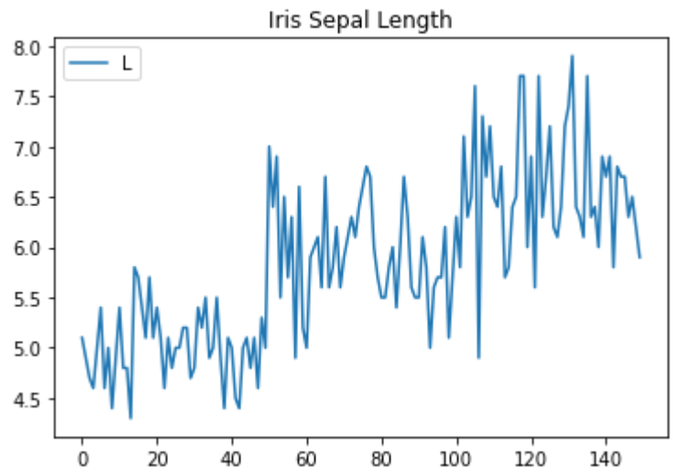
13 Matplotlib: Line Chart

Next let's learn how to create a **line chart**.

We'll start with a single line.

In [7]:
```
# create the x-axis data (range from 0 to the number of rows)
x_axis_data = range (0, iris.shape[0])

# create the figure and axis
figure, axis = plot.subplots()

# get the data
columns = iris["sepal_length"]

# plot each data point
axis.plot (x_axis_data, columns)

axis.set_title ("Iris Sepal Length")

axis.legend("L")
```

Out[7]:
```
<matplotlib.legend.Legend at 0x27601deac40>
```



14 Data Set 2: Wine Reviews

The second data set we will use is **Wine**, used to correlate wine prices with reviews.

In [10]:
```
# import the Wine Reviews data set into a Pandas DataFrame
wine = pd.read_csv ("wine.csv")

# print some records to make sure it worked
wine.head()
```

Out[10]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	tast
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	R(
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Pa(
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	
4	4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Pa(

15 Matplotlib: Histogram

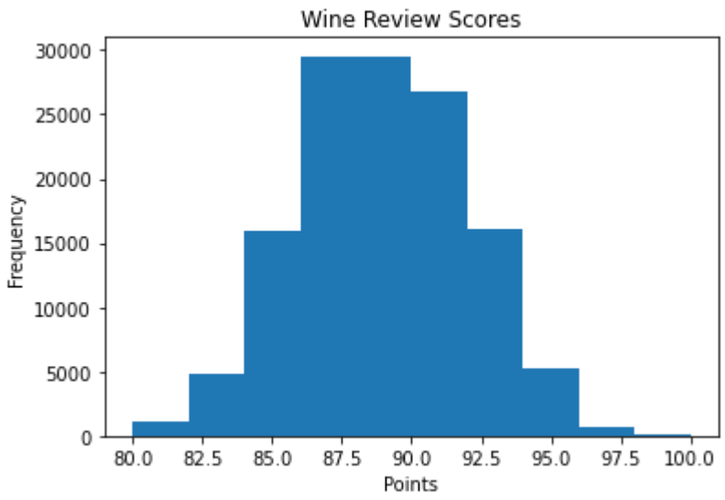
A histogram displays the frequency (number of times) that a data point falls into a bucket of possible values.

We can make a histogram using the `hist` method.

To do so, we pass **categorical data** to this method

In [9]:
```
# create the figure and axis
figure, axis = plot.subplots()

# plot the histogram
axis.hist(wine["points"])

# set the title and labels
axis.set_title ("Wine Review Scores")
axis.set_xlabel ("Points")
axis.set_ylabel ("Frequency")
```

Out[9]:
```
Text(0, 0.5, 'Frequency')
```



16 Matplotlib: Bar Chart

We can create bar charts using the `bar` method.

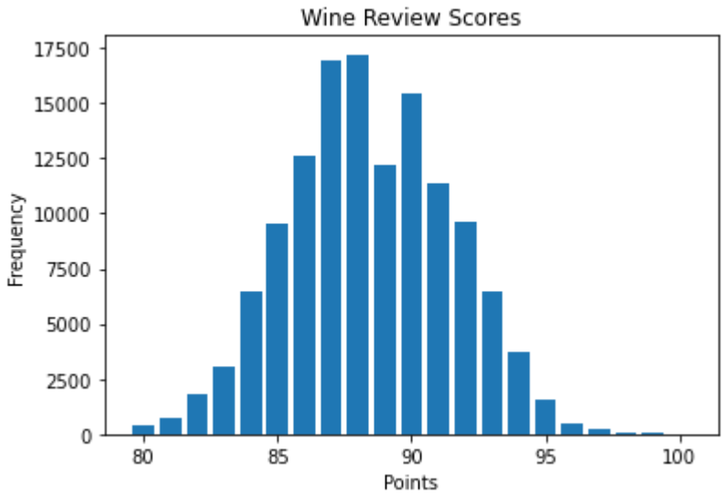
Unlike the histogram, the bar chart will not automatically calculate the category frequency.

Instead, we need to make the counts data ourselves.

Thankfully, Pandas can make this quick work of this.

In [11]:
```
# create a figure and axis
figure, axis = plot.subplots()

# count the occurrence of each score
count_data = wine["points"].value_counts()

# get the x and y data

# x axis data
points = count_data.index

# y axis data
frequencies = count_data.values

# create the bar chart
axis.bar(points, frequencies)

# set the title and labels
axis.set_title ("Wine Review Scores")
axis.set_xlabel ("Points")
axis.set_ylabel ("Frequency")
```

Out[11]: Text(0, 0.5, 'Frequency')

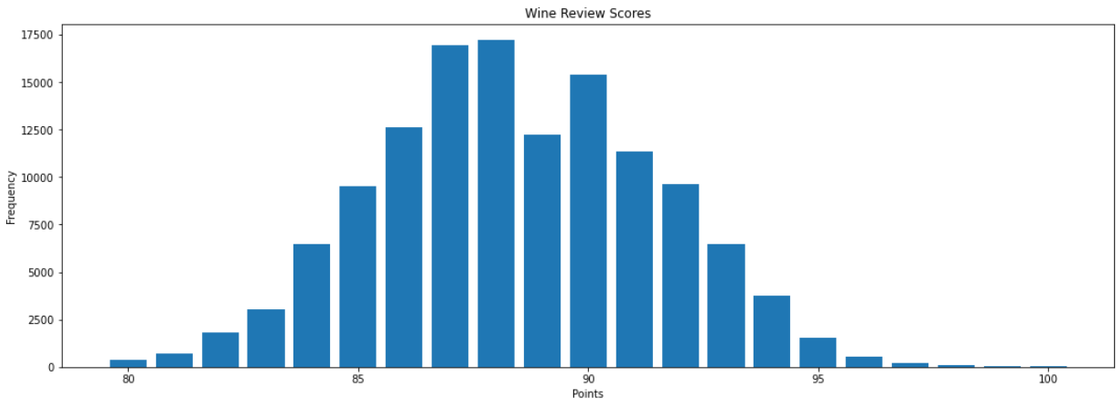


17 Matplotlib: Graph Size

If you want to change the dimensions of the chart size, use the optional `figsize` argument.

In [13]:
```
# create a figure and axis
figure, axis = plot.subplots(figsize = (18, 6))

# count the occurrence of each class
count_data = wine["points"].value_counts()

# get the x and y data
points = count_data.index
frequencies = count_data.values

# create the bar chart
axis.bar(points, frequencies)

# set the title and labels
axis.set_title ("Wine Review Scores")
axis.set_xlabel ("Points")
axis.set_ylabel ("Frequency")
```

Out[13]: Text(0, 0.5, 'Frequency')



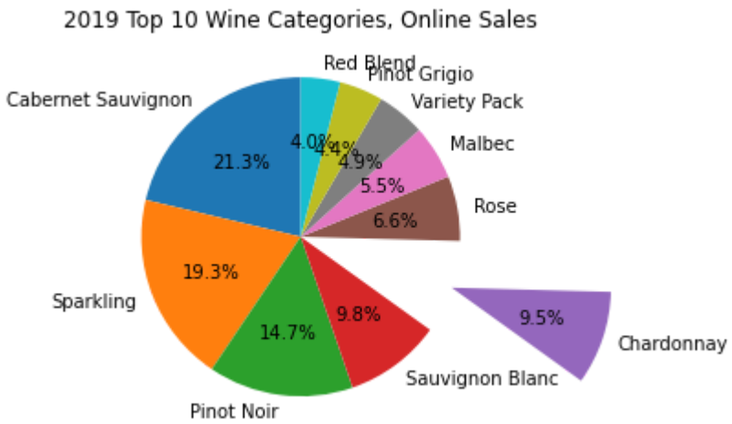
18 Matplotlib: Pie Chart

Pie charts show percentage breakdowns in **slices**.

We can create pie charts using the `pie` method.

In [15]:
```
# create a figure and axis
figure, axis = plot.subplots()

# create the labels for the data points
slice_labels = [ "Cabernet Sauvignon", "Sparkling", "Pinot Noir", "Sauvignon Blar

# create the percent sizes for the slices (slices will be ordered and plotted cou
slice_sizes = [ 13.9, 12.6, 9.6, 6.4, 6.2, 4.3, 3.6, 3.2, 2.9, 2.6 ]

s = 0

for size in slice_sizes:

    s = s + size

print ("Total size: " + str (s))

# set the title and labels
axis.set_title ("2019 Top 10 Wine Categories, Online Sales")

# setting an "equal" aspect ration ensures that the pie chart is drawn as a circl
axis.axis ("equal")

# explode one of the slices
explode_slices = ( 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 )

# set the formatting for the chart values
value_formatting = "%.1f%%"

# set the starting angle of the first slice
first_slice_start_angle = 90

# create the pie chart
axis.pie (slice_sizes, labels = slice_labels, autopct = value_formatting, explode

plot.show()
```

```
Total size: 65.3
```



19 Matplotlib: Colors

Matplotlib has its own set of named colors you can use.

There are many ways to specify color, including by Cascading Style Sheets (CSS) color name or Red-Green-Blue (RGB) value (plus transparency).

You can find a list of them here:

- https://matplotlib.org/examples/color/named_colors.html (https://matplotlib.org/examples/color/named_colors.html)
- https://matplotlib.org/3.1.0/gallery/color/named_colors.html (https://matplotlib.org/3.1.0/gallery/color/named_colors.html)

Or, you can run the code in the following cell to generate a table of colors.

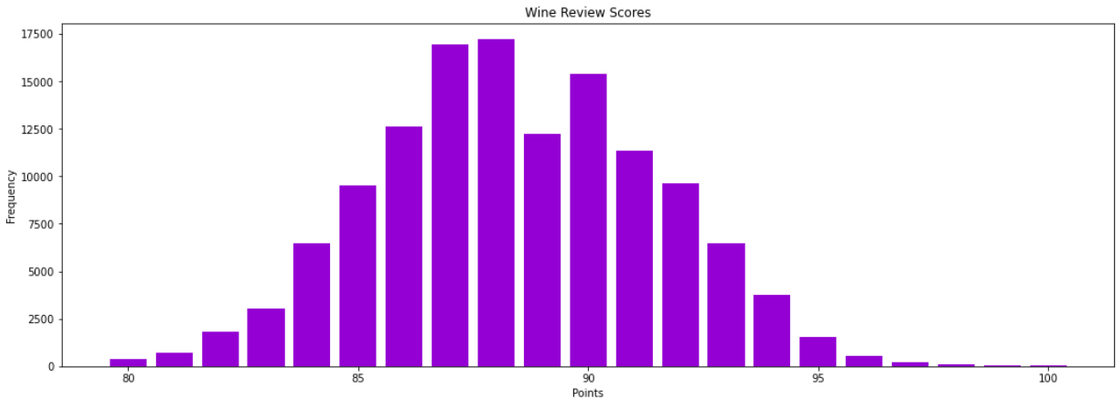
In [16]:
```
from __future__ import division

from matplotlib import colors as mcolors

colors = dict(mcolors.BASE_COLORS, **mcolors.CSS4_COLORS)

# Sort colors by hue, saturation, value and name.
by_hsv = sorted((tuple(mcolors.rgb_to_hsv(mcolors.to_rgba(color)[:3])), name)
                for name, color in colors.items())
sorted_names = [name for hsv, name in by_hsv]

n = len(sorted_names)
ncols = 4
nrows = n // ncols + 1

fig, ax = plot.subplots(figsize=(8, 5))

# Get height and width
X, Y = fig.get_dpi() * fig.get_size_inches()
h = Y / (nrows + 1)
w = X / ncols

for i, name in enumerate(sorted_names):
    col = i % ncols
    row = i // ncols
    y = Y - (row * h) - h

    xi_line = w * (col + 0.05)
    xf_line = w * (col + 0.25)
    xi_text = w * (col + 0.3)

    ax.text(xi_text, y, name, fontsize=(h * 0.8),
            horizontalalignment='left',
            verticalalignment='center')

    ax.hlines(y + h * 0.1, xi_line, xf_line,
              color=colors[name], linewidth=(h * 0.6))

ax.set_xlim(0, X)
ax.set_ylim(0, Y)
ax.set_axis_off()

fig.subplots_adjust(left=0, right=1,
                    top=1, bottom=0,
                    hspace=0, wspace=0)
plot.show()
```

20 Matplotlib: Uniform Color

You can set the uniform color for all the bars in a Matplotlib chart...

In [17]:
```
# create a figure and axis
figure, axis = plot.subplots(figsize = (18, 6))

# count the occurrence of each class
count_data = wine["points"].value_counts()

# get the x and y data
points = count_data.index
frequencies = count_data.values

# create the bar chart
axis.bar(points, frequencies, color = "darkviolet")

# set the title and labels
axis.set_title ("Wine Review Scores")
axis.set_xlabel ("Points")
axis.set_ylabel ("Frequency")
```

Out[17]: Text(0, 0.5, 'Frequency')



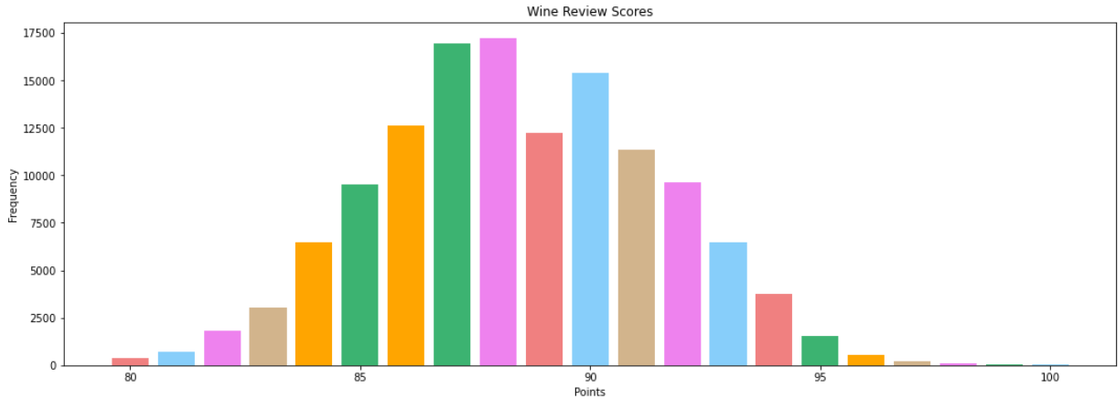
21 Matplotlib: Different Color for Each Bar

You can set the color for **each bar** in a bar chart by specifying a **list of color values**.

In the event that you have more bars than colors specified, Matplotlib will iterate over your colors and reuse them.

In [18]:
```
# create a figure and axis
figure, axis = plot.subplots(figsize = (18, 6))

# count the occurrence of each class
count_data = wine["points"].value_counts()

# get the x and y data
points = count_data.index
frequencies = count_data.values

colors = ["violet", "mediumseagreen", "lightskyblue", "orange", "lightcoral", "ta

# create the bar chart
axis.bar(points, frequencies, color = colors)

# set the title and labels
axis.set_title ("Wine Review Scores")
axis.set_xlabel ("Points")
axis.set_ylabel ("Frequency")
```

Out[18]:
```
Text(0, 0.5, 'Frequency')
```

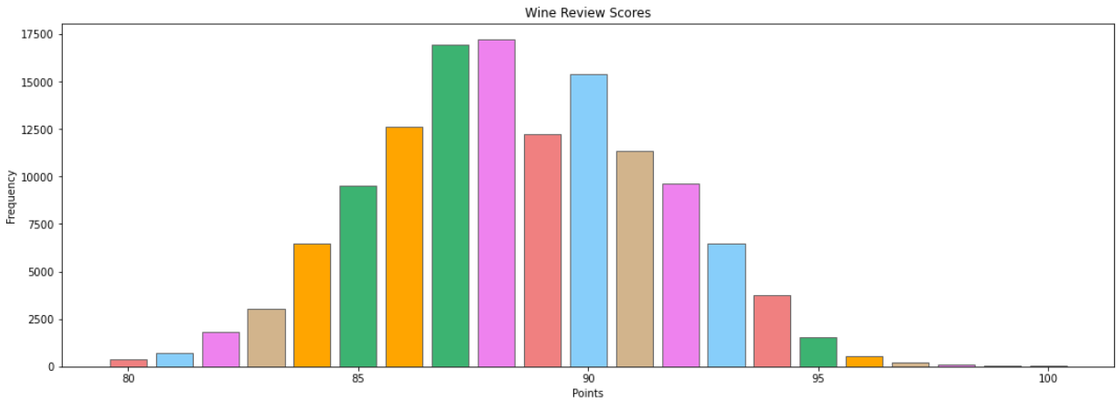


22 Matplotlib: Border Color

You can also set the border color by specifying the `edgecolor` argument...

In [19]:
```
# create a figure and axis
figure, axis = plot.subplots(figsize = (18, 6))

# count the occurrence of each class
count_data = wine["points"].value_counts()

# get the x and y data
points = count_data.index
frequencies = count_data.values

# create the bar chart
axis.bar(points, frequencies, color = ["violet", "mediumseagreen", "lightskyblue"

# set the title and labels
axis.set_title ("Wine Review Scores")
axis.set_xlabel ("Points")
axis.set_ylabel ("Frequency")
```

Out[19]:
```
Text(0, 0.5, 'Frequency')
```

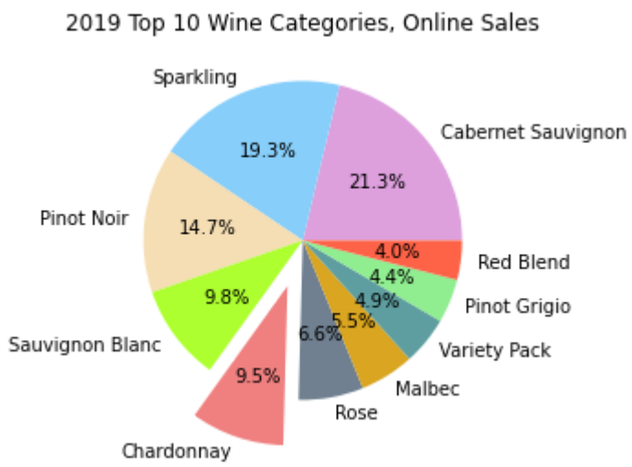


23 Matplotlib: Pie Chart Colors

You can also set the individual pie chart slice colors by specifying the `colors` optional parameter.

In [20]:
```
# create a figure and axis
figure, axis = plot.subplots()

# create the colors for the slices
slice_colors = [ "plum", "lightskyblue", "wheat", "greenyellow", "lightcoral", "s

# create the labels for the data points
slice_labels = [ "Cabernet Sauvignon", "Sparkling", "Pinot Noir", "Sauvignon Blar

# create the percent sizes for the slices (slices will be ordered and plotted cou
slice_sizes = [ 13.9, 12.6, 9.6, 6.4, 6.2, 4.3, 3.6, 3.2, 2.9, 2.6 ]

# set the title and labels
axis.set_title ("2019 Top 10 Wine Categories, Online Sales")

# setting an "equal" aspect ration ensures that the pie chart is drawn as a circl
axis.axis ("equal")

# explode one of the slices
explode_slices = ( 0, 0, 0, 0, 0.3, 0, 0, 0, 0, 0 )

# set the formatting for the chart values
value_formatting = "%.1f%%"

# set the starting angle of the first slice
first_slice_start_angle = 0

# create the pie chart
axis.pie (slice_sizes, labels = slice_labels, colors = slice_colors, autopct = va
```

Out[20]:
```
([<matplotlib.patches.Wedge at 0x26e9eaf4940>,
  <matplotlib.patches.Wedge at 0x26e9eb01130>,
  <matplotlib.patches.Wedge at 0x26e9eb01850>,
  <matplotlib.patches.Wedge at 0x26e9eb01f70>,
  <matplotlib.patches.Wedge at 0x26e9eb0e6d0>,
  <matplotlib.patches.Wedge at 0x26e9eb0edf0>,
  <matplotlib.patches.Wedge at 0x26e9eb1c550>,
  <matplotlib.patches.Wedge at 0x26e9eb1cc70>,
  <matplotlib.patches.Wedge at 0x26e9eb283d0>,
  <matplotlib.patches.Wedge at 0x26e9eb28af0>],
 [Text(0.8630699826237275, 0.681989886357473, 'Cabernet Sauvignon'),
  Text(-0.40070177323506084, 1.0244208553745273, 'Sparkling'),
  Text(-1.090732682847083, 0.14248583989016164, 'Pinot Noir'),
  Text(-0.8823941542965795, -0.6567956732981914, 'Sauvignon Blanc'),
  Text(-0.44669228678960854, -1.3268255352240814, 'Chardonnay'),
  Text(0.19737924925193495, -1.0821466776573048, 'Rose'),
  Text(0.5847548466337867, -0.9316983252846904, 'Malbec'),
  Text(0.8531369894148958, -0.6943754584459967, 'Variety Pack'),
  Text(1.017528514730907, -0.41789439061743194, 'Pinot Grigio'),
  Text(1.0914055389295723, -0.13723683759781907, 'Red Blend')],
 [Text(0.47076544506748763, 0.3719944834677125, '21.3%'),
  Text(-0.21856460358276045, 0.5587750120224694, '19.3%'),
  Text(-0.5949450997347725, 0.07771954903099726, '14.7%'),
  Text(-0.4813059023435887, -0.3582521854353771, '9.8%'),
  Text(-0.28715932722189114, -0.8529592726440522, '9.5%'),
  Text(0.10766140868287359, -0.5902618241767117, '6.6%'),
  Text(0.31895718907297455, -0.508199086518922, '5.5%'),
  Text(0.46534744877176126, -0.37875025006145274, '4.9%'),
  Text(0.5550155534895855, -0.22794239488223558, '4.4%'),
  Text(0.595312112143403, -0.07485645687153766, '4.0%')])
```



24 Congratulations!

You have just learned the basics of data visualization in **Matplotlib**.

There are so many more ways to configure these charts and graphs, but for now you have the ability the get data onto the screen to tell your analysis story.

This is the end of Part 01.