5_Exercise _Visualize_data_with_Matplotlib

October 16, 2021

1 Exploring data with Python - visualize data

In this notebook, we'll apply basic techniques to analyze data with basic statistics and visualise using graphs.

1.1 Loading our data

Before we being, lets load the same data about study hours that we analysed in the previous notebook. We will also recalculate who passed in the same way as last time Run the code in the cell below by clicking the **Run** button to see the data.

[ ]:
```
import pandas as pd

# Load data from a text file
#!wget https://raw.githubusercontent.com/MicrosoftDocs/
 ↪mslearn-introduction-to-machine-learning/main/Data/ml-basics/grades.csv
df_students = pd.read_csv('grades.csv',delimiter=',',header='infer')

# Remove any rows with missing data
df_students = df_students.dropna(axis=0, how='any')

# Calculate who passed, assuming '60' is the grade needed to pass
passes  = pd.Series(df_students['Grade'] >= 60)

# Save who passed to the Pandas dataframe
df_students = pd.concat([df_students, passes.rename("Pass")], axis=1)



# Print the result out into this notebook
df_students
```

[ ]:
```
         Name  StudyHours  Grade   Pass
0         Dan       10.00   50.0  False
1       Joann       11.50   50.0  False
2       Pedro        9.00   47.0  False
3       Rosie       16.00   97.0   True
4       Ethan        9.25   49.0  False
5       Vicky        1.00    3.0  False
```

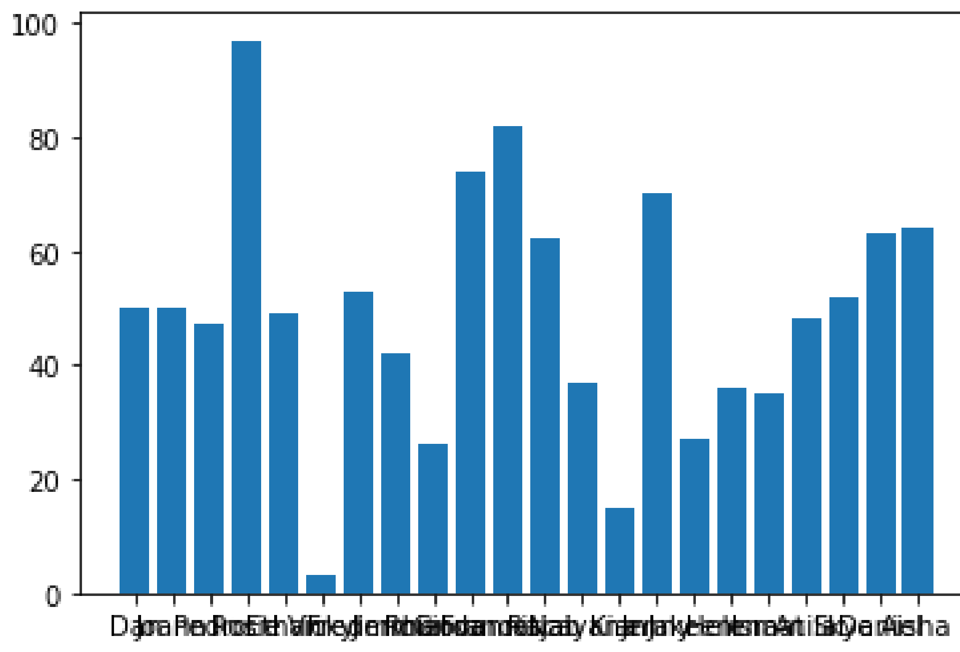
```
6    Frederic       11.50   53.0  False
7      Jimmie        9.00   42.0  False
8      Rhonda        8.50   26.0  False
9    Giovanni       14.50   74.0   True
10  Francesca       15.50   82.0   True
11      Rajab       13.75   62.0   True
12    Naiyana        9.00   37.0  False
13       Kian        8.00   15.0  False
14      Jenny       15.50   70.0   True
15     Jakeem        8.00   27.0  False
16     Helena        9.00   36.0  False
17      Ismat        6.00   35.0  False
18      Anila       10.00   48.0  False
19       Skye       12.00   52.0  False
20     Daniel       12.50   63.0   True
21      Aisha       12.00   64.0   True
```

1.2 Visualizing data with Matplotlib

DataFrames provide a great way to explore and analyze tabular data, but sometimes a picture is worth a thousand rows and columns. The **Matplotlib** library provides the foundation for plotting data visualizations that can greatly enhance your ability the analyze the data.

Let's start with a simple bar chart that shows the grade of each student.

[ ]:
```
# Ensure plots are displayed inline in the notebook
%matplotlib inline

from matplotlib import pyplot as plt

# Create a bar plot name vs grade
plt.bar(x=df_students.Name, height=df_students.Grade)

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



Well, that worked; but the chart could use some improvements to make it clearer what we're looking at.

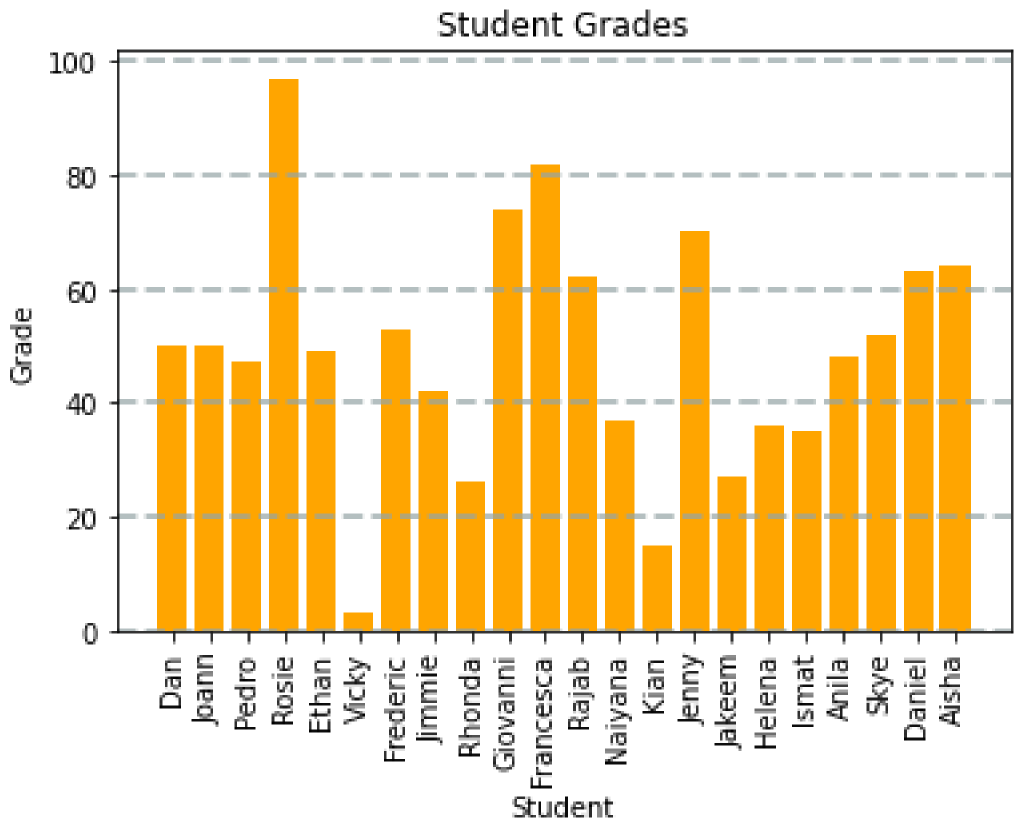
Note that you used the **pyplot** class from Matplotlib to plot the chart. This class provides a whole bunch of ways to improve the visual elements of the plot. For example, the following code:

- Specifies the color of the bar chart.
- Adds a title to the chart (so we know what it represents)
- Adds labels to the X and Y (so we know which axis shows which data)
- Adds a grid (to make it easier to determine the values for the bars)
- Rotates the X markers (so we can read them)

[ ]:
```
df_students.Grade.describe()
# height of the plot below is between 0 and 100 approx
# because of min of 2 and max of 97
```

[ ]:
```
count    22.000000
mean     49.181818
std      21.737912
min       3.000000
25%      36.250000
50%      49.500000
75%      62.750000
max      97.000000
Name: Grade, dtype: float64
```

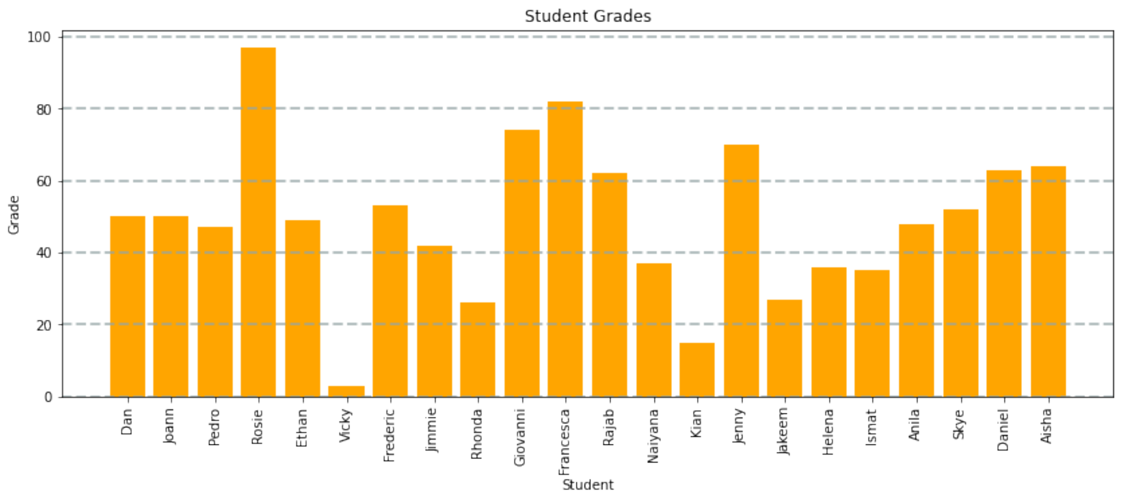
[ ]:
```python
# Create a bar plot of name vs grade
plt.bar(x=df_students.Name, height=df_students.Grade, color='orange')

# Customize the chart
plt.title('Student Grades')
plt.xlabel('Student')
plt.ylabel('Grade')
plt.grid(color='#95a5a6', linestyle='--',linewidth=2, axis='y', alpha=0.7)
plt.xticks(rotation=90)

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



A plot is technically contained with a Figure. In the previous examples, the figure was created implicitly for you; but you can create it explicitly. For example, the following code creates a figure with a specific size.

[ ]:
```python
# Create a Figure
fig = plt.figure(figsize=(14,5))

# Create a bar plot of name vs grade
```

```
plt.bar(x=df_students.Name, height=df_students.Grade, color='orange')

# Customize the chart
plt.title('Student Grades')
plt.xlabel('Student')
plt.ylabel('Grade')
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
plt.xticks(rotation=90)

# Show the figure
plt.show()
```



A figure can contain multiple subplots, each on its own *axis*.

For example, the following code creates a figure with two subplots - one is a bar chart showing student grades, and the other is a pie chart comparing the number of passing grades to non-passing grades.

[ ]:
```
df_students['Pass'].value_counts()
```

[ ]:
```
False    15
True      7
Name: Pass, dtype: int64
```

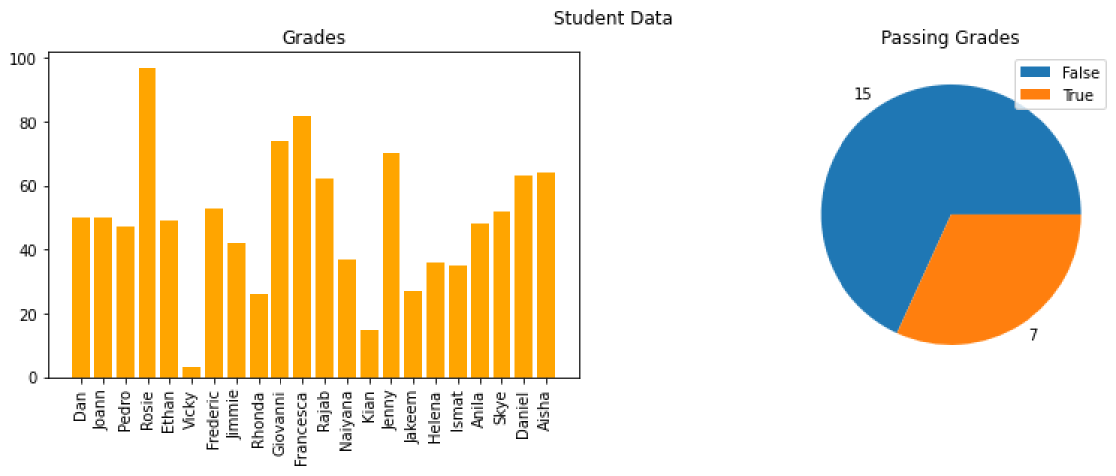
[ ]:
```
pass_counts = df_students['Pass'].value_counts()
pass_counts.keys().tolist()
```

[ ]:
```
[False, True]
```

[ ]:
```
# Create a figure for 2 subplots (1 row, 2 columns)
fig, ax =plt.subplots(1,2, figsize =(14,4))
```

```
# Create a bar plot of name vs grade on the first axis
ax[0].bar(x=df_students.Name, height=df_students.Grade, color='orange')
ax[0].set_title('Grades')
ax[0].set_xticklabels(df_students.Name, rotation=90)

# Create a pie chart of pass counts on the second axis
pass_counts = df_students['Pass'].value_counts()
ax[1].pie(pass_counts, labels=pass_counts)
ax[1].set_title('Passing Grades')
ax[1].legend(pass_counts.keys().tolist()) # list of [False,True]

# Add a title to the Figure
fig.suptitle('Student Data')

# Show the figure
fig.show()
```

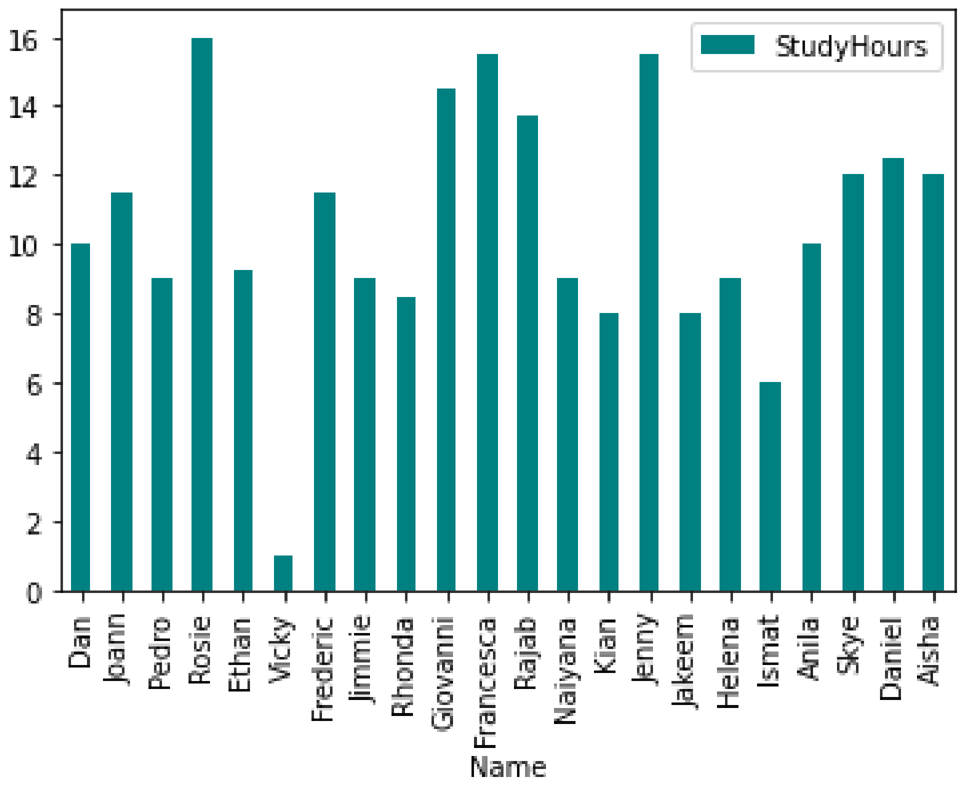
```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: UserWarning:
FixedFormatter should only be used together with FixedLocator
  import sys
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:19:
UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
C:\Users\aduzo\Anaconda3\lib\site-packages\IPython\core\pylabtools.py:132:
UserWarning: Creating legend with loc="best" can be slow with large amounts of
data.
  fig.canvas.print_figure(bytes_io, **kw)
```



Until now, you've used methods of the Matplotlib.pyplot object to plot charts. However, Matplotlib is so foundational to graphics in Python that many packages, including Pandas, provide methods that abstract the underlying Matplotlib functions and simplify plotting. For example, the DataFrame provides its own methods for plotting data, as shown in the following example to plot a bar chart of study hours.

```
[ ]: df_students.plot.bar(x='Name', y='StudyHours', color='teal', figsize=(6,4))
```

```
[ ]: <AxesSubplot:xlabel='Name'>
```



1.3 Getting started with statistical analysis

Now that you know how to use Python to manipulate and visualize data, you can start analyzing it.

A lot of data science is rooted in *statistics*, so we'll explore some basic statistical techniques.

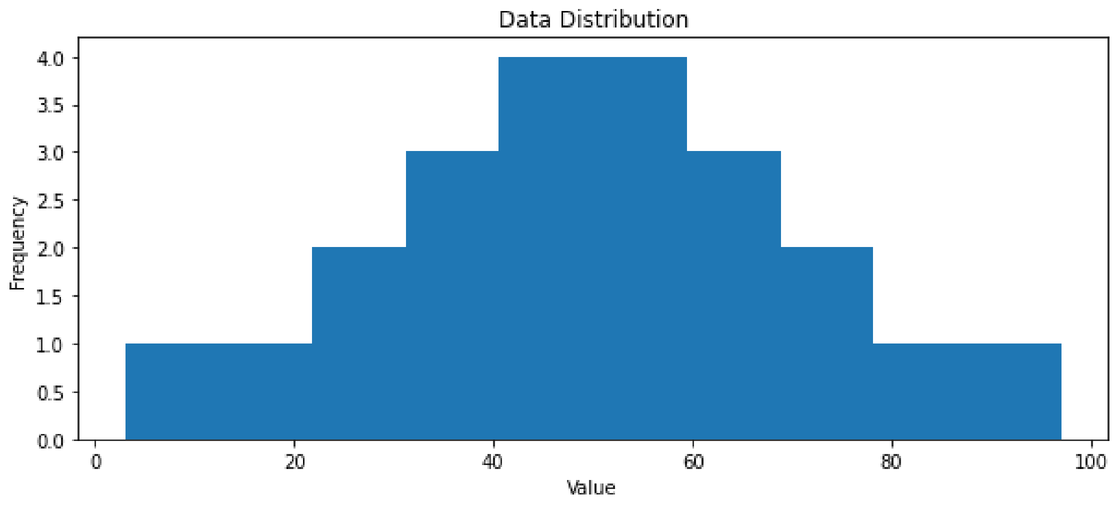
> **Note**: This is not intended to teach you statistics - that's much too big a topic for this notebook. It will however introduce you to some statistical concepts and techniques that data scientists use as they explore data in preparation for machine learning modeling.

1.3.1 Descriptive statistics and data distribution

When examining a *variable* (for example a sample of student grades), data scientists are particularly interested in its *distribution* (in other words, how are all the different grade values spread across the sample). The starting point for this exploration is often to visualize the data as a histogram, and see how frequently each value for the variable occurs.

```
[ ]: # Get the variable to examine
var_data = df_students['Grade']

# Create a Figure
fig = plt.figure(figsize=(10,4))

# Plot a histogram
plt.hist(var_data)

# Add titles and labels
plt.title('Data Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')

# Show the figure
fig.show()
```

```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:16:
UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  app.launch_new_instance()
```



The histogram for grades is a symmetric shape, where the most frequently occurring grades tend to be in the middle of the range (around 50), with fewer grades at the extreme ends of the scale.

Measures of central tendency To understand the distribution better, we can examine so-called *measures of central tendency*; which is a fancy way of describing statistics that represent the "middle" of the data. The goal of this is to try to find a "typical" value. Common ways to define the middle of the data include:

- The *mean*: A simple average based on adding together all of the values in the sample set, and then dividing the total by the number of samples.
- The *median*: The value in the middle of the range of all of the sample values.
- The *mode*: The most commonly occuring value in the sample set*.

Let's calculate these values, along with the minimum and maximum values for comparison, and show them on the histogram.

> *Of course, in some sample sets , there may be a tie for the most common value - in which case the dataset is described as *bimodal* or even *multimodal*.

```
[ ]: # Get the variable to examine
     var = df_students['Grade']
```

```
[ ]: print(var.mode())
     print(var.mode()[0])
```

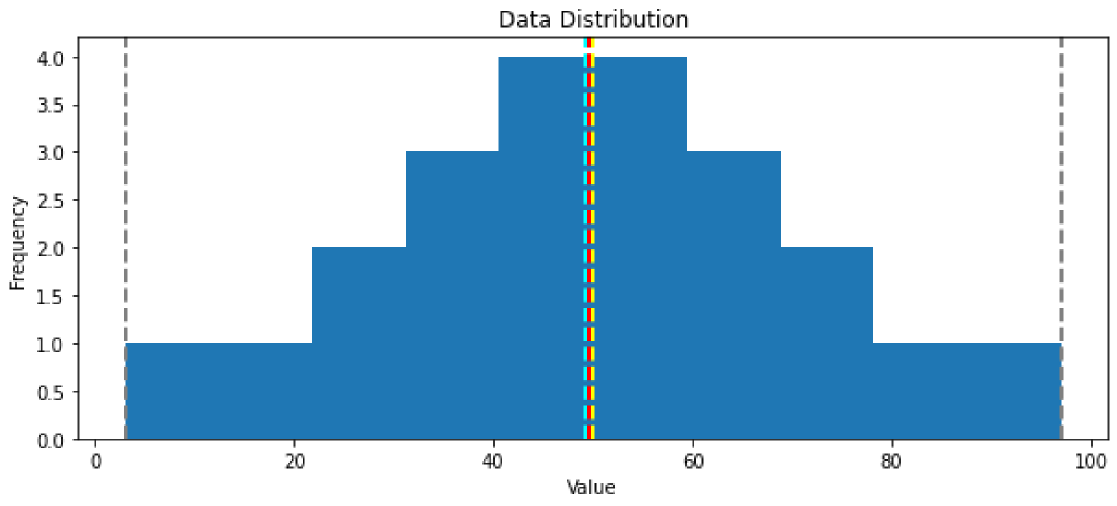
```
0    50.0
dtype: float64
50.0
```

```
[ ]: # Get statistics
     min_val = var.min()
     max_val = var.max()
     mean_val = var.mean()
     med_val = var.median()
     mod_val = var.mode()[0]

     print('Minimum:{:.2f}\nMean:{:.2f}\nMedian:{:.2f}\nMode:{:.2f}\nMaximum:{:.
      ↪2f}\n'.format(min_val,
                                                                                ␣
      ↪          mean_val,
                                                                                ␣
      ↪          med_val,
                                                                                ␣
      ↪          mod_val,
                                                                                ␣
      ↪          max_val))
```

```
Minimum:3.00
Mean:49.18
Median:49.50
Mode:50.00
Maximum:97.00
```

```
[ ]: # Create a Figure
     fig = plt.figure(figsize=(10,4))

     # Plot a histogram
     plt.hist(var)

     # Add lines for the statistics
     plt.axvline(x=min_val, color = 'gray', linestyle='dashed', linewidth = 2)
     plt.axvline(x=mean_val, color = 'cyan', linestyle='dashed', linewidth = 2)
     plt.axvline(x=med_val, color = 'red', linestyle='dashed', linewidth = 2)
     plt.axvline(x=mod_val, color = 'yellow', linestyle='dashed', linewidth = 2)
     plt.axvline(x=max_val, color = 'gray', linestyle='dashed', linewidth = 2)

     # Add titles and labels
     plt.title('Data Distribution')
     plt.xlabel('Value')
     plt.ylabel('Frequency')

     # Show the figure
     fig.show()
```

```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:20:
UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```

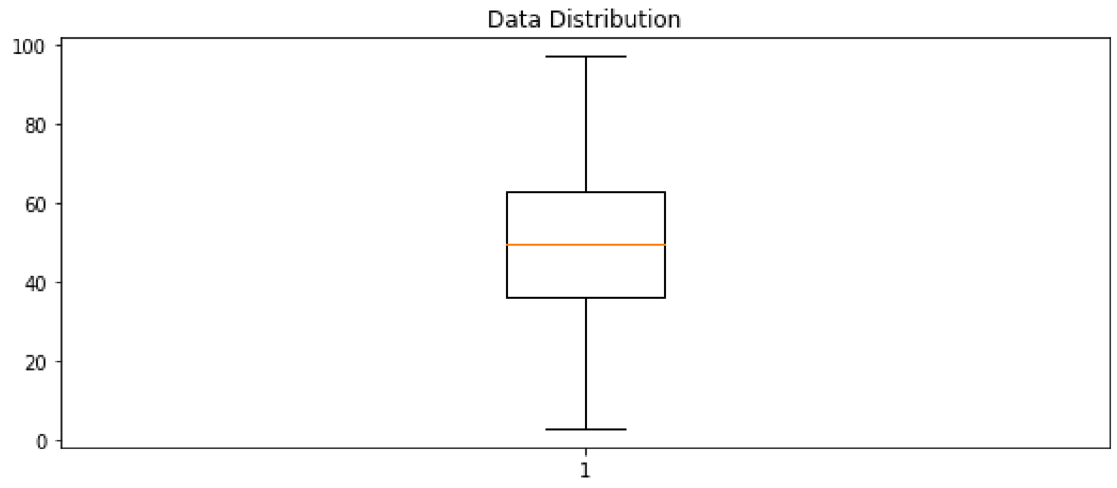


For the grade data, the mean, median, and mode all seem to be more or less in the middle of the minimum and maximum, at around 50.

Another way to visualize the distribution of a variable is to use a *box* plot (sometimes called a *box-and-whiskers* plot). Let's create one for the grade data.

```
[ ]: # Get the variable to examine
     var = df_students['Grade']

     # Create a Figure
     fig = plt.figure(figsize=(10,4))

     # Plot a histogram
     plt.boxplot(var)

     # Add titles and labels
     plt.title('Data Distribution')

     # Show the figure
     fig.show()
```

```
C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:14:
UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



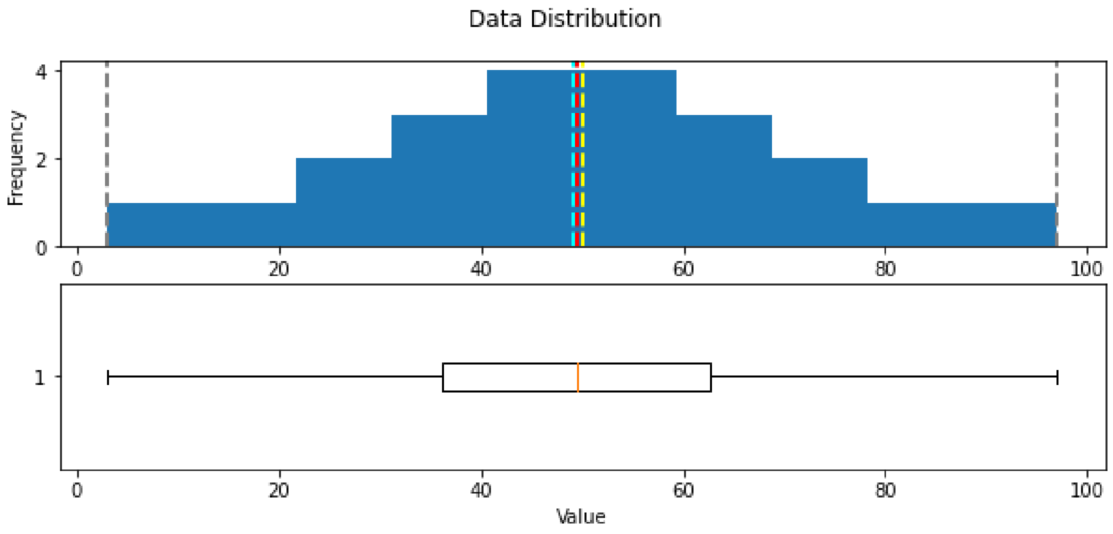
The box plot shows the distribution of the grade values in a different format to the histogram. The *box* part of the plot shows where the inner two *quartiles* of the data reside - so in this case, half of the grades are between approximately 36 and 63. The *whiskers* extending from the box show the outer two quartiles; so the other half of the grades in this case are between 0 and 36 or 63 and 100. The line in the box indicates the *median* value.

For learning, it can be useful to combine histograms and box plots, with the box plot's orientation changed to align it with the histogram (in some ways, it can be helpful to think of the histogram as a "front elevation" view of the distribution, and the box plot as a "plan" view of the distribution from above.)

[ ]:
```
# Create a function that we can re-use
def show_distribution(var_data):
    from matplotlib import pyplot as plt

    # Get statistics
    min_val = var_data.min()
    max_val = var_data.max()
    mean_val = var_data.mean()
    med_val = var_data.median()
    mod_val = var_data.mode()[0]

    print('Minimum:{:.2f}\nMean:{:.2f}\nMedian:{:.2f}\nMode:{:.2f}\nMaximum:{:.
 ↪2f}\n'.format(min_val,
                                                                                  ␣
 ↪           mean_val,
                                                                                  ␣
 ↪           med_val,
                                                                                  ␣
 ↪           mod_val,
                                                                                  ␣
 ↪           max_val))

    # Create a figure for 2 subplots (2 rows, 1 column)
    fig, ax = plt.subplots(2, 1, figsize = (10,4))

    # Plot the histogram
    ax[0].hist(var_data)
    ax[0].set_ylabel('Frequency')

    # Add lines for the mean, median, and mode
    ax[0].axvline(x=min_val, color = 'gray', linestyle='dashed', linewidth = 2)
    ax[0].axvline(x=mean_val, color = 'cyan', linestyle='dashed', linewidth = 2)
    ax[0].axvline(x=med_val, color = 'red', linestyle='dashed', linewidth = 2)
    ax[0].axvline(x=mod_val, color = 'yellow', linestyle='dashed', linewidth =␣
 ↪2)
    ax[0].axvline(x=max_val, color = 'gray', linestyle='dashed', linewidth = 2)

    # Plot the boxplot
    ax[1].boxplot(var_data, vert=False)
    ax[1].set_xlabel('Value')

    # Add a title to the Figure
    fig.suptitle('Data Distribution')

    # Show the figure
    fig.show()
```

```python
# Get the variable to examine
col = df_students['Grade']
# Call the function
show_distribution(col)
```

```
Minimum:3.00
Mean:49.18
Median:49.50
Mode:50.00
Maximum:97.00


C:\Users\aduzo\Anaconda3\lib\site-packages\ipykernel_launcher.py:40:
UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
```



All of the measurements of central tendency are right in the middle of the data distribution, which is symmetric with values becoming progressively lower in both directions from the middle.

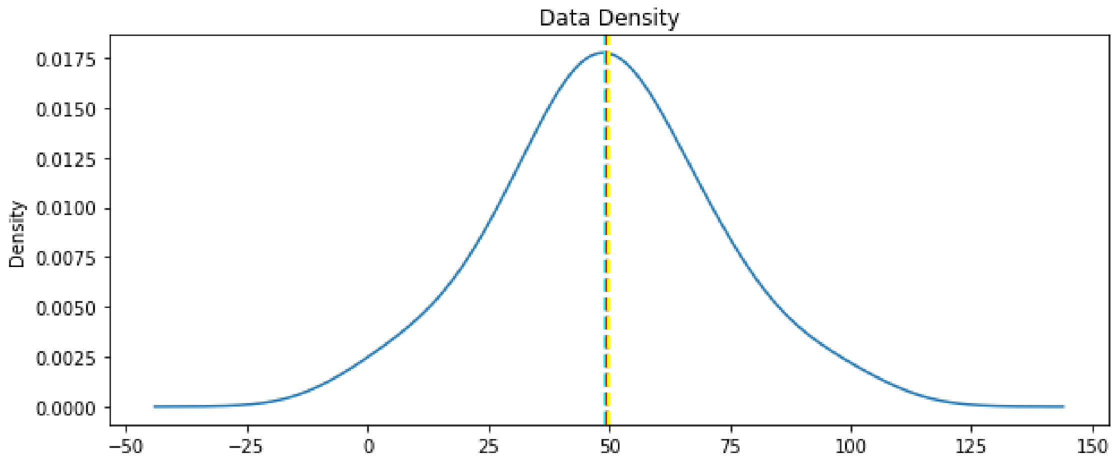
To explore this distribution in more detail, you need to understand that statistics is fundamentally about taking *samples* of data and using probability functions to extrapolate information about the full *population* of data.

What does this mean? *Samples* refer to the data we have on hand - such as information about these 22 students' study habits and grades. The *population* refers to all possible data we could collect - such as every student's grades and study habits across every educational institution throughout the history of time. Usually we're interested in the population but it's simply not practical to collect all of that data. Instead, we need to try estimate what the population is like from the small amount of data (samples) that we have.

If we have enough samples, we can calculate something called a *probability density function*, which estimates the distribution of grades for the full population.

The Pandas DataFrame class provides a helpful plot function to show this density.

```
[ ]: def show_density(var_data):
         from matplotlib import pyplot as plt

         fig=plt.figure(figsize=(10,4))

         # Plot density
         var_data.plot.density()

         # Add titles and labels
         plt.title('Data Density')
     # Show the mean, median, and mode
         plt.axvline(x=var_data.mean(), color = 'cyan', linestyle='dashed',␣
      ↪linewidth = 2)
         plt.axvline(x=var_data.median(), color = 'red', linestyle='dashed',␣
      ↪linewidth = 2)
         plt.axvline(x=var_data.mode()[0], color = 'yellow', linestyle='dashed',␣
      ↪linewidth = 2)

         # Show the figure
         plt.show()

     # Get the density of Grade
     col = df_students['Grade']
     show_density(col)
```



As expected from the histogram of the sample, the density shows the characteristic "bell curve" of what statisticians call a *normal* distribution with the mean and mode at the center and symmetric tails.

1.4 Summary

Well done! There were a number of new concepts in here, so let's summarise.

Here we have:

1. Made graphs with matplotlib
2. Seen how to customise these graphs
3. Calculated basic statistics, such as medians
4. Looked at the spread of data using box plots and histograms
5. Learned about samples vs populations
6. Estimated what the population of graphse might look like from a sample of grades.

In our next notebook we will look at spotting unusual data, and finding relationships between data.

1.5 Further Reading

To learn more about the Python packages you explored in this notebook, see the following documentation:

- NumPy
- Pandas
- Matplotlib