Introduction to data visualisation in Python

In [18]:
```
import os

directory = "C:/Users/cepedazk/Jupyter Notebook/Datasets/"
os.chdir(directory)
```

There are lots of different packages for visualising data in Python, including:

- Matplotlib
- Seaborn
- Bokeh
- Built-in pandas plotting tools
- Plotly (Not covered in this module)
- Dash (Not covered in this module)
- Shiny for Python (Not covered in this module)

We will start today with matplotlib

- Matplotlib is the most widely used package for data visualisation in Python.
- It is convention to read matplotlib in as plt: `import matplotlib.pyplot as plt`
- Always use the command `plt.show()` to display the plot.
- Official documentation: https://matplotlib.org/stable/users/index.html

In [19]:
```
import matplotlib.pyplot as plt
import pandas as pd
```

In [20]:
```
# Set default figure size (width, height) in inches
plt.rcParams["figure.figsize"] = (5, 3)  # Change values as needed

 # Set width=5 inches, height=3 inches
```

Read in an IMDB dataset containing details on the top 100 films of 2016.

In [21]:
```
imdb = pd.read_csv('imdb.csv')
imdb.head()
```

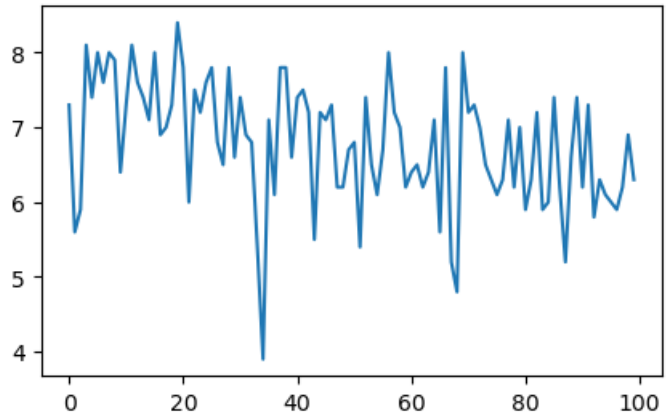
Out[21]:

	rank	title	desc	runtime	genre	rating	votes	director	metascore
0	1	13 Hours	During an attack on a U.S. compound in Libya, ...	144	Action	7.3	155234	Michael Bay	48.0
1	2	Terrifier	On Halloween night, Tara Heyes finds herself a...	85	Horror	5.6	48568	Damien Leone	NaN
2	3	Suicide Squad	A secret government agency recruits some of th...	123	Action	5.9	710994	David Ayer	40.0
3	4	Hacksaw Ridge	World War II American Army Medic Desmond T. Do...	139	Biography	8.1	573353	Mel Gibson	71.0
4	5	The Nice Guys	In 1970s Los Angeles, a mismatched pair of pri...	116	Action	7.4	358550	Shane Black	70.0

- We will now see how to produce some basic line plots, bar charts, histograms, boxplots and scatterplots using matplotlib.
- We will also see how to improve these plots by changing plot types, specifying axis labels and titles etc.

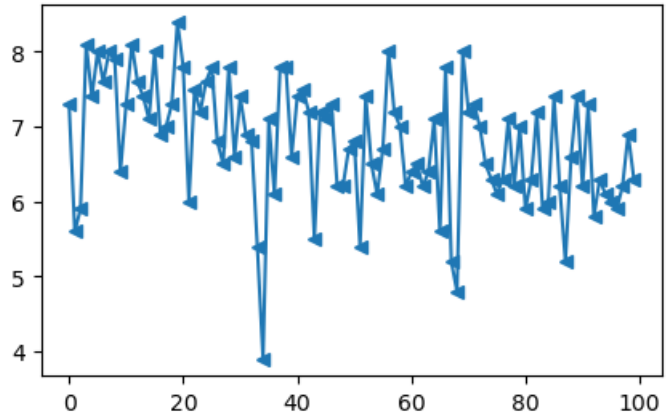
- Let's start with line plots, which are mainly used to display one variable using the basic `plt.plot()` function.
- Remember to specify `plt.show()` to output the plot.

In [22]:
```
plt.plot(imdb.rating) # lets send a Series first, lineal plot
plt.show()
```



Change point markers using `marker=` parameter

In [23]:
```
plt.plot(imdb.rating, marker = '<')
plt.show()
```

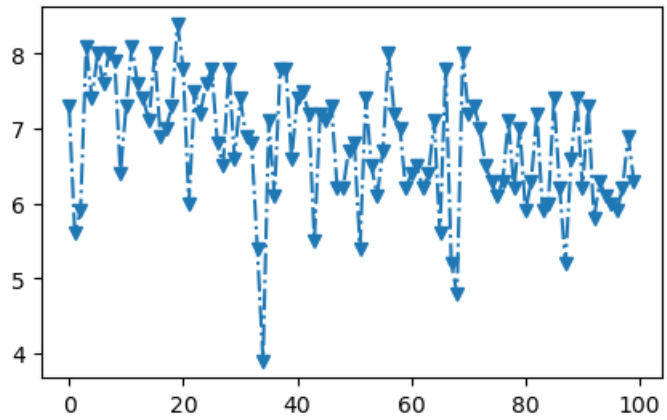


Markers

character	description
``'.'``	point marker
``'o'``	circle marker
``'v'``	triangle_down marker
``'^'``	triangle_up marker
``'<'``	triangle_left marker
``'>'``	triangle_right marker

Change to a dotted line using `linestyle =`

In [24]:
```
plt.plot(imdb.rating, linestyle = '-.', marker = "v")
plt.show()
```

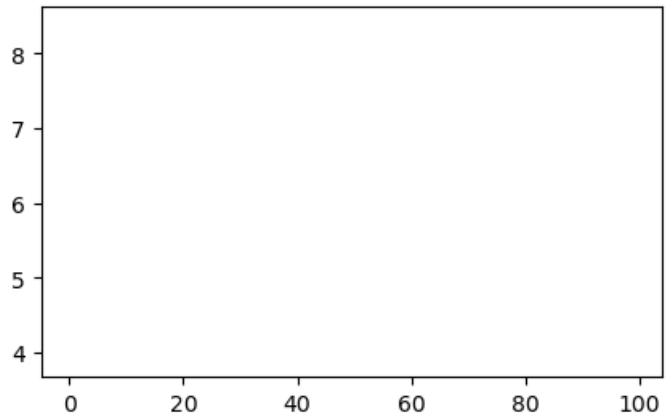


Line Styles

character	description
``'-'``	solid line style
``'--'``	dashed line style
``'-.'``	dash-dot line style
``':'``	dotted line style

- Specify that you do not want a line connecting the points using linestyle = 'None'.

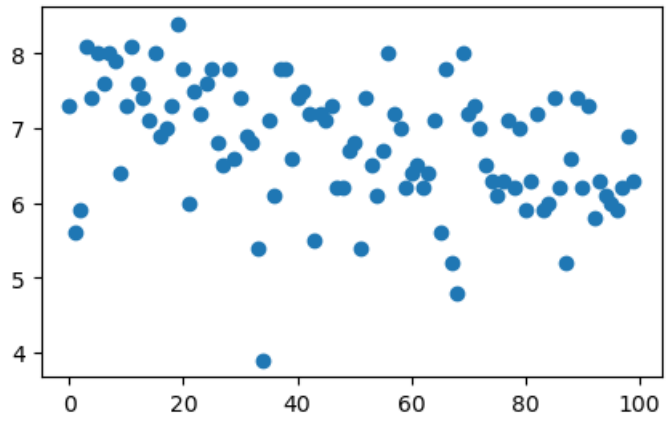
In [25]:
```
plt.plot(imdb.rating, linestyle = 'None')
plt.show()
```



We have not specified how the points should be marked, so they are not shown either.

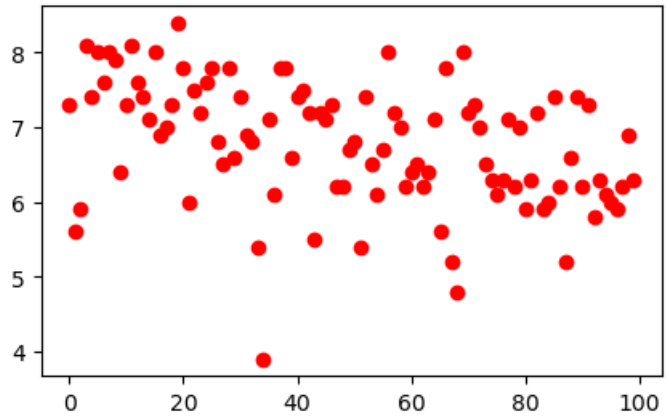
Specify the markers now.

In [26]:
```
plt.plot(imdb.rating, marker = 'o', linestyle = 'None')
plt.show()
```



Change the colour of the points using the `color=`.

In [27]:
```
plt.plot(imdb.rating, marker = 'o', linestyle = 'None', color = 'red')
plt.show()
```



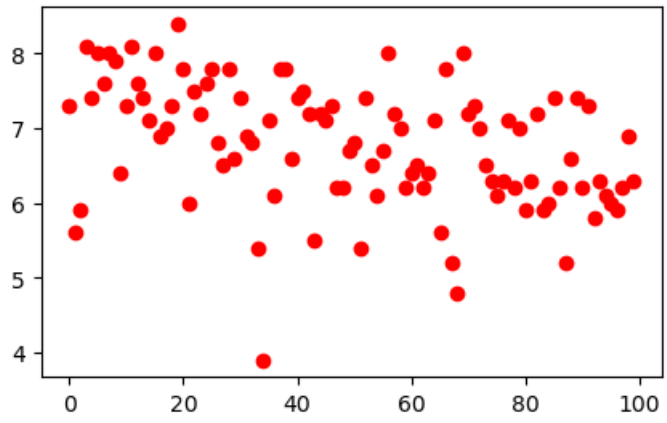
Colors

The supported color abbreviations are the single letter codes

character	color
``'b'``	blue
``'g'``	green
``'r'``	red
``'c'``	cyan
``'m'``	magenta
``'y'``	yellow
``'k'``	black
``'w'``	white

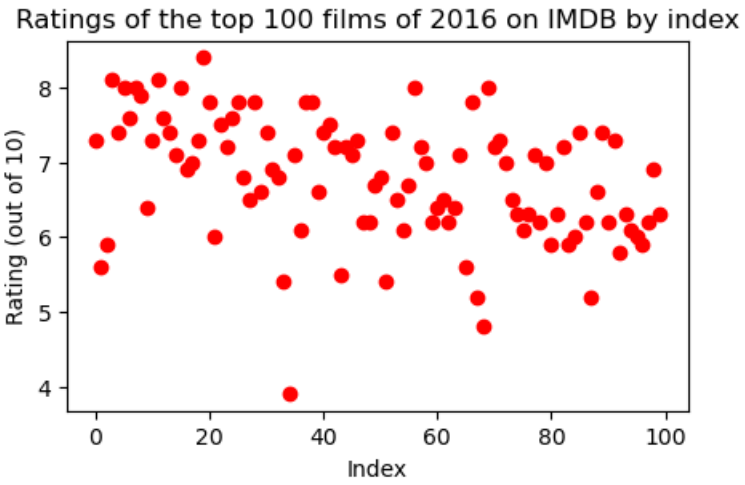
- What do you think it is happening in the figure below?

In [28]:
```
plt.plot(imdb.rating, marker = 'o', linestyle = 'None', color = 'r')
plt.show()
```



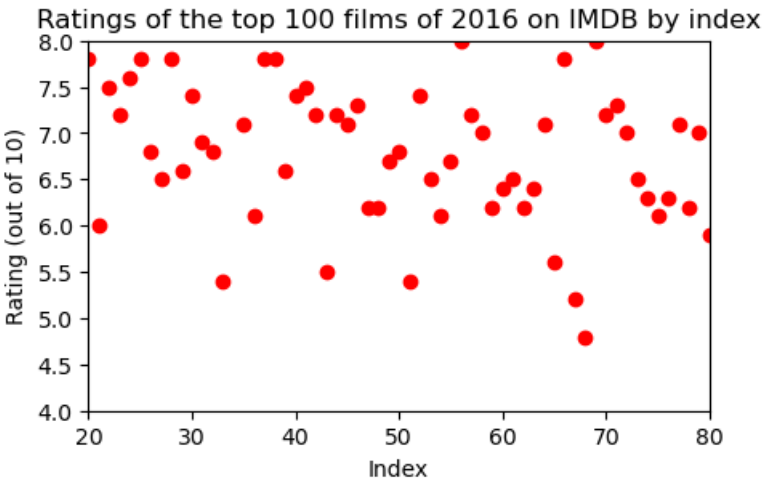
Add x and y axis labels, and a title using `plt.xlabel`, `plt.ylabel`, and `plt.title`

In [29]:
```
plt.plot(imdb.rating, marker = 'o', linestyle = 'None', color = 'r')
plt.xlabel('Index') # add a label in x axis
plt.ylabel('Rating (out of 10)') # add a label in y axis
plt.title('Ratings of the top 100 films of 2016 on IMDB by index') # adds title
plt.show()
```



You can also use `xlim` and `ylim` to zoom in/out

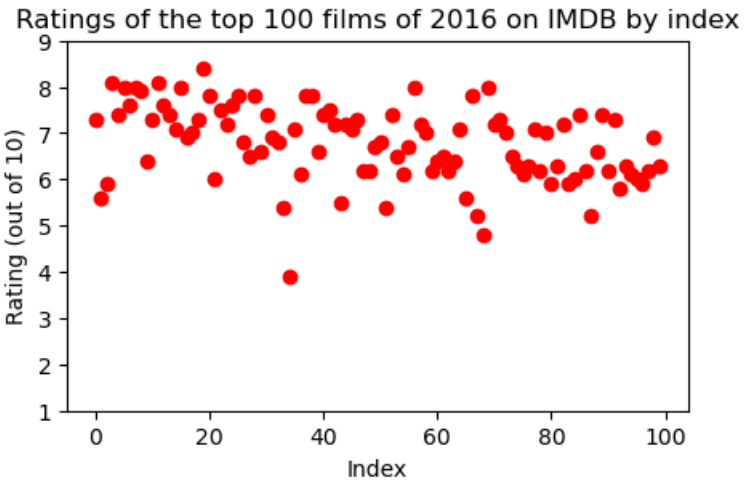
In [30]:
```
plt.plot(imdb.rating, marker = 'o', linestyle = 'None', color = 'r')
plt.xlabel('Index') # add a label in x axis
plt.ylabel('Rating (out of 10)') # add a label in y axis
plt.title('Ratings of the top 100 films of 2016 on IMDB by index') # adds title
plt.ylim([4,8])
plt.xlim([20,80])
plt.show()
```



Specify where the ticks on the y axis appear using `plt.yticks`

Note: ticks are the marks on the axes that indicate specific values

In [31]:
```
plt.plot(imdb.rating, marker = 'o', linestyle = 'None', color = 'r')
plt.xlabel('Index')
plt.ylabel('Rating (out of 10)')
plt.title('Ratings of the top 100 films of 2016 on IMDB by index')
plt.yticks([1,2,3,4,5,6,7,8,9]) # added more ticks
plt.show()
```



Bar charts

- Bar charts show information in a similar way to the line plots, but with bars instead of points.
- Use `plt.bar(x,y)` to plot a bar char
- They are used mainly to display one variable
- Specify the variable whose height you want to be represented by bars on the y axis (second argument of plt.bar).
- The x axis is usually the number index.

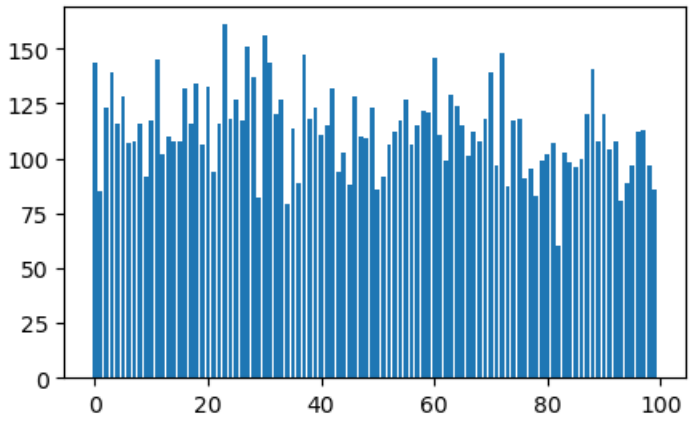
In [32]:
```
imdb.head()
```

Out[32]:

	rank	title	desc	runtime	genre	rating	votes	director	metascore
0	1	13 Hours	During an attack on a U.S. compound in Libya, ...	144	Action	7.3	155234	Michael Bay	48.0
1	2	Terrifier	On Halloween night, Tara Heyes finds herself a...	85	Horror	5.6	48568	Damien Leone	NaN
2	3	Suicide Squad	A secret government agency recruits some of th...	123	Action	5.9	710994	David Ayer	40.0
3	4	Hacksaw Ridge	World War II American Army Medic Desmond T. Do...	139	Biography	8.1	573353	Mel Gibson	71.0
4	5	The Nice Guys	In 1970s Los Angeles, a mismatched pair of pri...	116	Action	7.4	358550	Shane Black	70.0

In [33]:
```
# help(plt.bar)
```

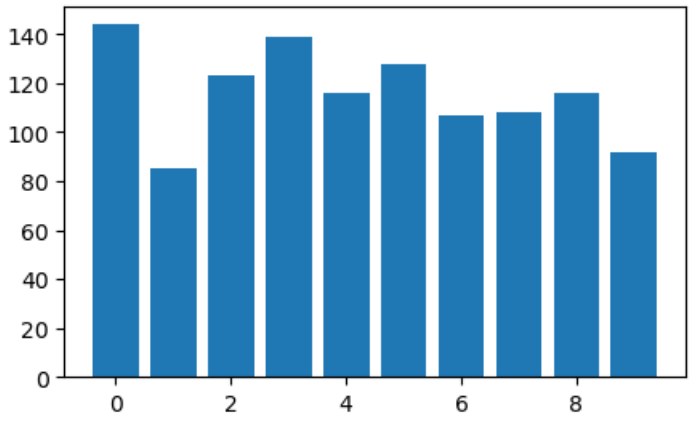
In [34]:
```
plt.bar(imdb.index, imdb.runtime)
plt.show()
```



There is a bit too much information here.

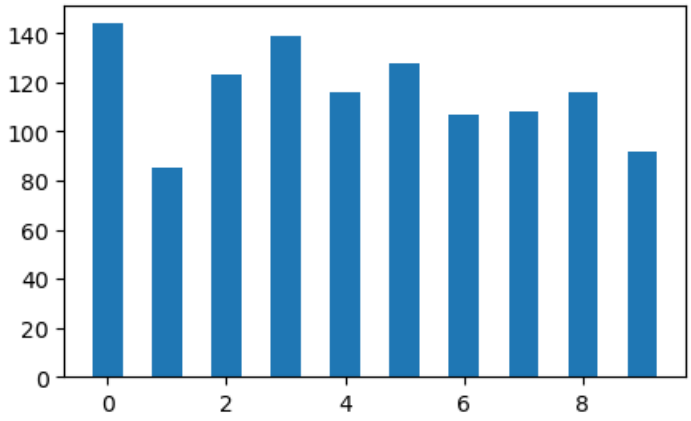
For illustration purposes, I will focus on the top 10 movies to show how barplots can be formatted.

In [35]:
```
plt.bar(imdb.index[0:10], imdb.runtime[0:10])
plt.show()
```



Change the width of the bars using the `width=`

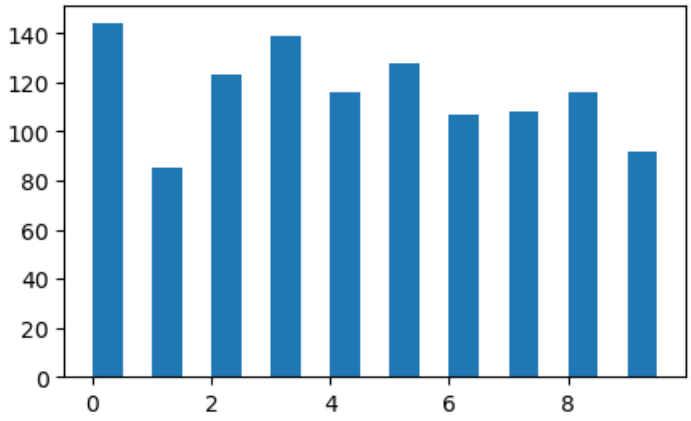
In [36]:
```
plt.bar(imdb.index[0:10], imdb.runtime[0:10], width = 0.5)
plt.show()
```



Change the alignment of the bars in relation to their x coordinate using the `align`

- 'center': Center the base on the *x* positions (default value).
- 'edge': Align the left edges of the bars with the *x* positions.

In [37]:
```
plt.bar(imdb.index[0:10], imdb.runtime[0:10], width = 0.5, align = 'edge')
plt.show()
```

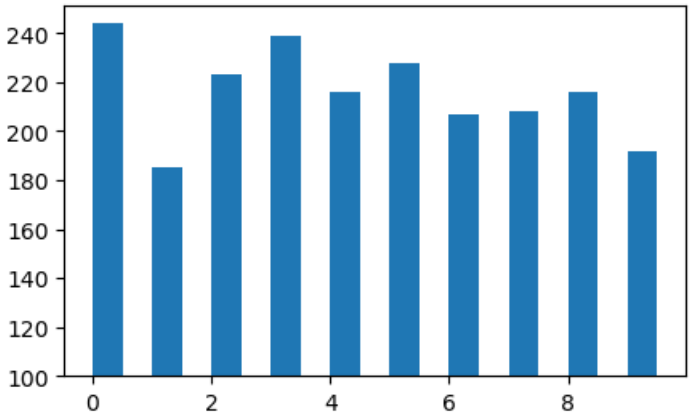


Specify the value at which the bars begin on the y axis using the `bottom=`

The y coordinate(s) of the bottom side(s) of the bars.

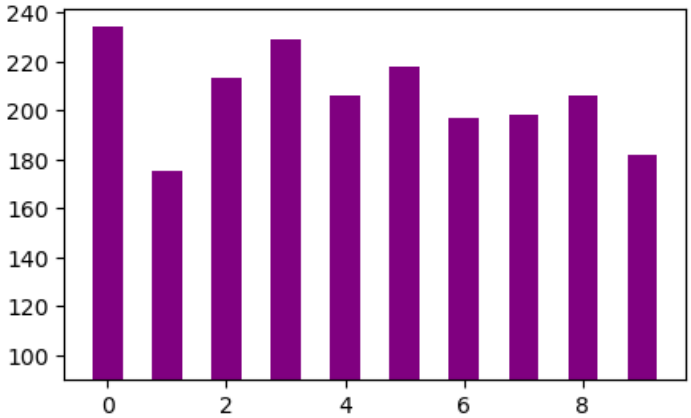
In [38]:
```
plt.bar(imdb.index[0:10], imdb.runtime[0:10],
        width = 0.5, align = 'edge', bottom = 100)
```

```
plt.show()
```

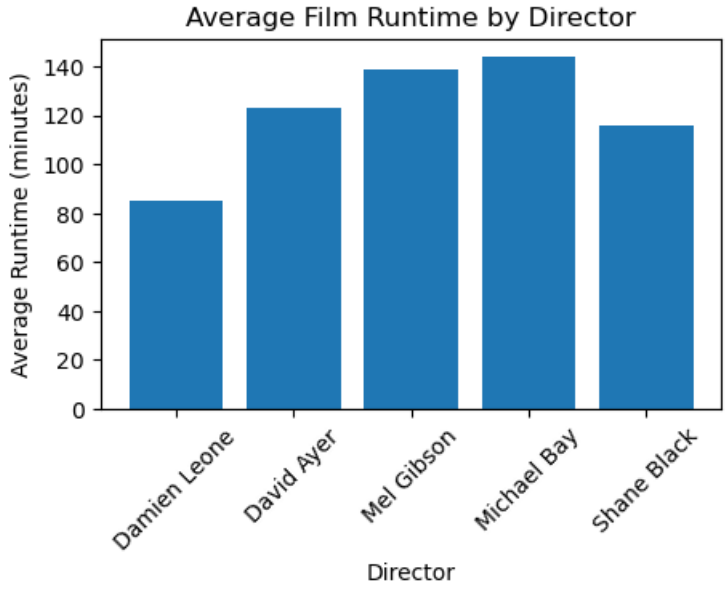


- Again, we can change the color of the bars using the color argument.
- Note that all of the arguments for these functions are specified on their help pages. I am not fully formatting every plot to be the finished article.
    - How could the plot below be improved?

In [39]:
```
plt.bar(imdb.index[0:10], imdb.runtime[0:10],
        width = 0.5, bottom = 90, color = 'purple')
plt.show()
```



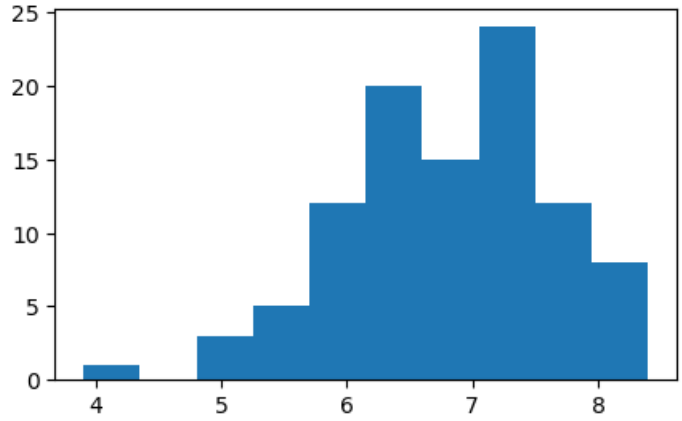
In [40]:
```
# Group by director and compute the average runtime

grouped_runtime = imdb[0:5].groupby("director")["runtime"].mean() # director is index, we can use it in x-ax

# Plot bar chart
plt.bar(grouped_runtime.index, grouped_runtime.values)

# Add labels
plt.xlabel("Director")
plt.ylabel("Average Runtime (minutes)")
plt.title("Average Film Runtime by Director")
plt.xticks(rotation=45)  # Rotate x-axis labels if needed

plt.show()
```



Histograms

- Histograms are used to visualise the distribution of a variable. They are mainly used to display one variable.
- We will use the `plt.hist` function.

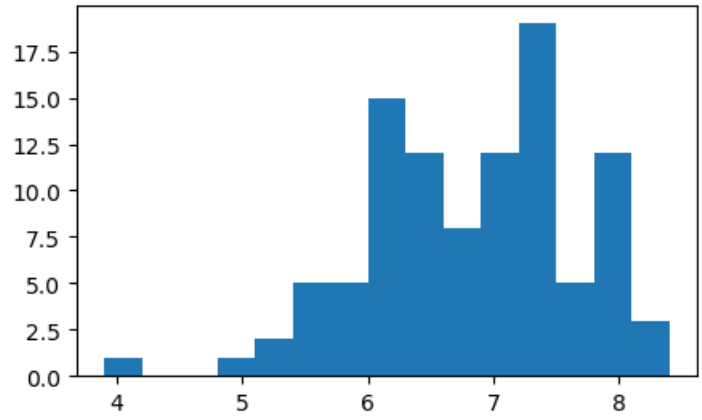
In [111…]
```
plt.hist(imdb.rating)
plt.show()
```



The default number of bins is 10 but you can change it with `bins =`.

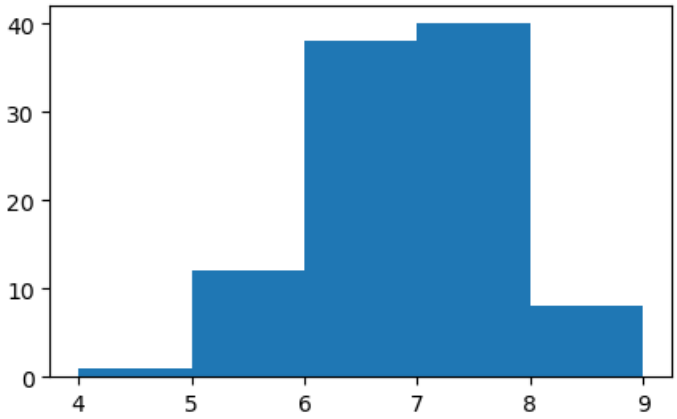
Note: bins are the intervals (or "buckets") into which your data is grouped

In [43]:
```
plt.hist(imdb.rating, bins = 15)
plt.show()
```



You can specify the exact tick marks by giving a list for the bins argument

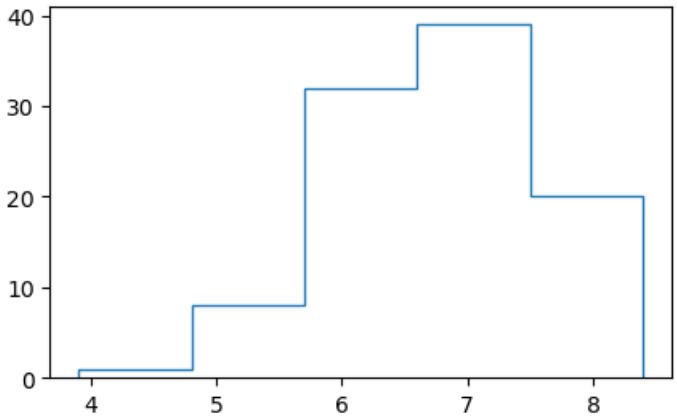
In [44]:
```
plt.hist(imdb.rating, bins = [4, 5, 6, 7, 8, 9])
plt.show()
```



You can change the formatting of histogram using the `histtype`

Where might this formatting be useful?

In [45]:
```
plt.hist(imdb.rating, bins = 5, histtype = 'step')
plt.show()
```



Boxplots

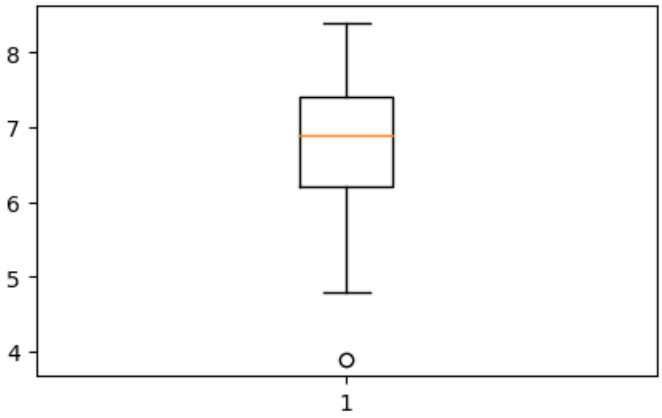
- Boxplots are used to show the distribution of one or more variables.
- Boxplots are often used to detect outliers as they are shown clearly on the pt.
- Boxplots also display the median and upper and lower quartiles of the data.
- The command for a boxplot is `plt.boxplot(values)` .

In [46]:
```
imdb.head()
```

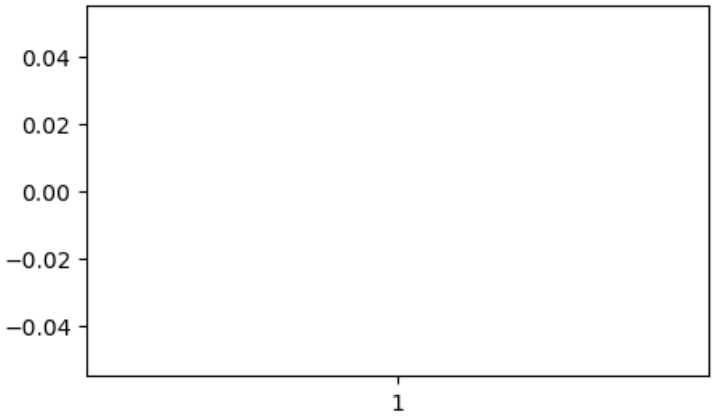
Out[46]:

	rank	title	desc	runtime	genre	rating	votes	director	metascore
0	1	13 Hours	During an attack on a U.S. compound in Libya, ...	144	Action	7.3	155234	Michael Bay	48.0
1	2	Terrifier	On Halloween night, Tara Heyes finds herself a...	85	Horror	5.6	48568	Damien Leone	NaN
2	3	Suicide Squad	A secret government agency recruits some of th...	123	Action	5.9	710994	David Ayer	40.0
3	4	Hacksaw Ridge	World War II American Army Medic Desmond T. Do...	139	Biography	8.1	573353	Mel Gibson	71.0
4	5	The Nice Guys	In 1970s Los Angeles, a mismatched pair of pri...	116	Action	7.4	358550	Shane Black	70.0

In [47]:
```
plt.boxplot(imdb.rating)
plt.show()
```



In [48]:
```
plt.boxplot(imdb.metascore)
plt.show()
```



- I got an empty plot when using metascore column from the imdb DataFrame as a boxplot.
- This is due to missing data.

In [49]:
```
sum(pd.isnull(imdb.metascore))
```

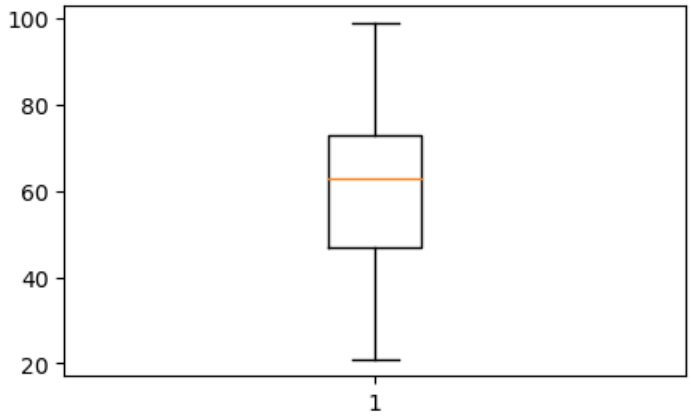
Out[49]:
```
6
```

Use `pd.notnull` as a "mask" to only plot the non-NaN values from metascore column

In [50]:
```
mask = pd.notnull(imdb.metascore)
imdb.metascore[mask]
```

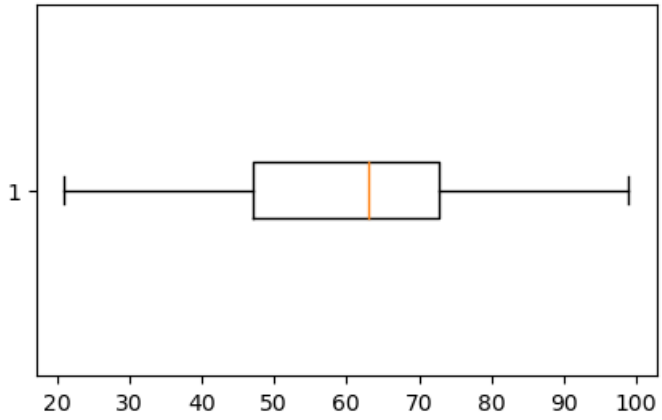
Out[50]:
```
0     48.0
2     40.0
3     71.0
4     70.0
5     94.0
      ...
95    42.0
96    40.0
97    34.0
98    64.0
99    59.0
Name: metascore, Length: 94, dtype: float64
```

In [51]:
```
plt.boxplot(imdb.metascore[mask])
plt.show()
```



Draw the boxplot horizontally using the `vert =` value `False`

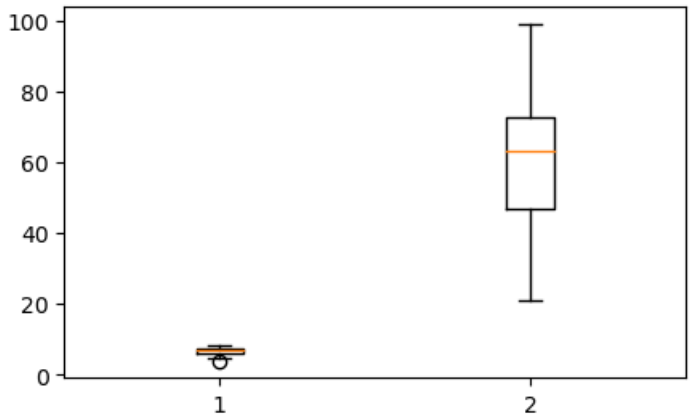
In [52]:
```
plt.boxplot(imdb.metascore[mask], vert = False)
plt.show()
```



Boxplots can be used to show two different continuous variables on the same plot.

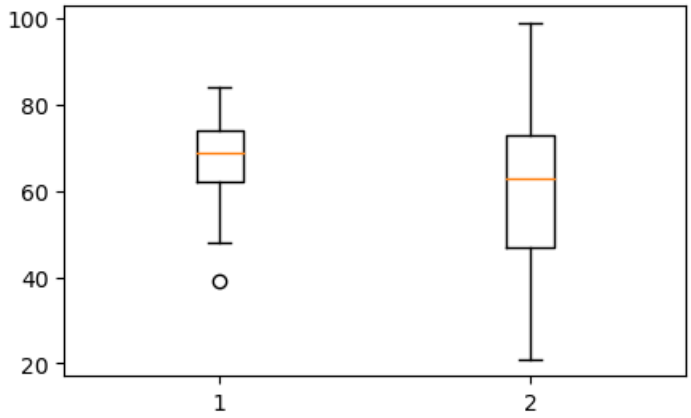
However, if the variables are not on similar scales, it may not be easy to compare their distributions:

In [53]:
```
plt.boxplot([imdb.rating, imdb.metascore[mask]])
plt.show()
```



Multiply rating by 10 to put them onto the same scale (out of 100).

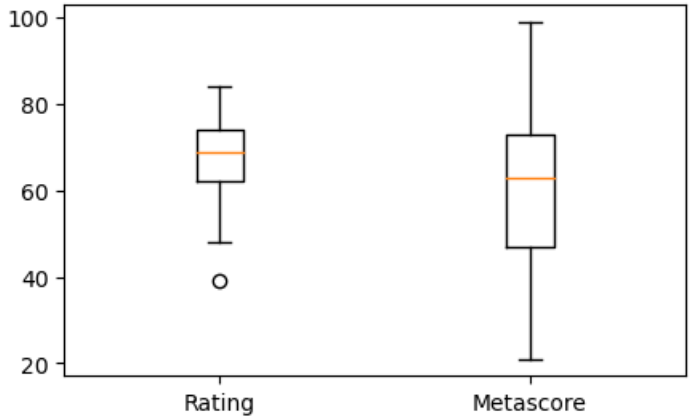
In [54]:
```
rating_100 = imdb.rating * 10

plt.boxplot([rating_100, imdb.metascore[mask]])
plt.show()
```



We now see that metascore column has a lower median and a wider distribution than rating.

Label the boxplots using `plt.xticks`. The first argument is a list of where the ticks appear. The second argument is a list of the labels.

In [55]:
```
plt.boxplot([rating_100, imdb.metascore[mask]])
plt.xticks([1,2], ['Rating', 'Metascore'])
plt.show()
```



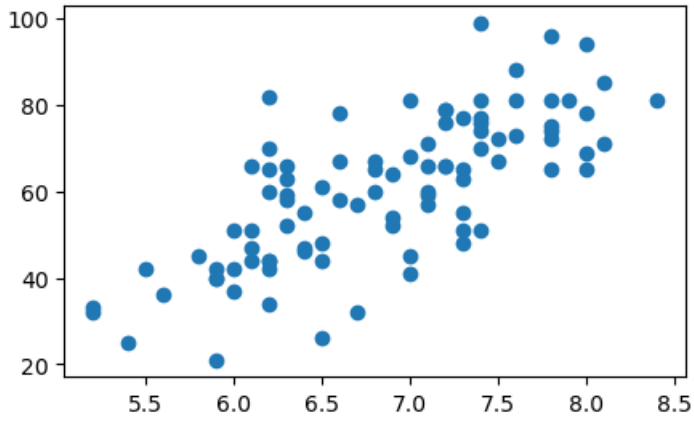
Boxplots for splitting a continuous variable by the levels of a categorical variable are not straightforward using matplotlib. They are easier in seaborn, which we will see next week.

Scatterplots

Scatterplots are used to see the relationship between two continuous variables, for example rating and metascore columns from imdb dataframe.

In [56]:
```
# help(plt.scatter)
```

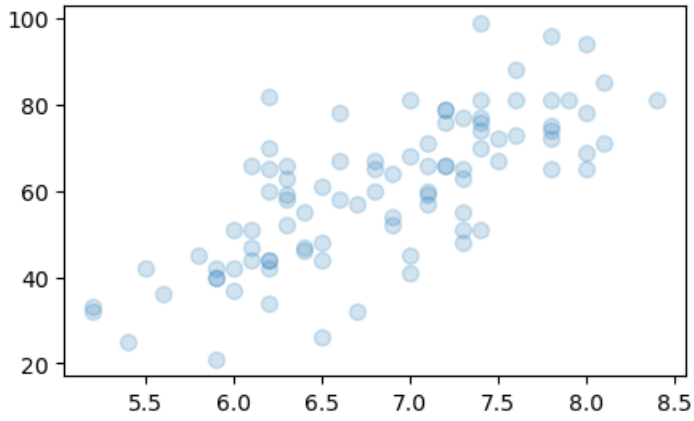
In [57]:
```
plt.scatter(imdb.rating, imdb.metascore) # plt.scatter(x,y)
plt.show()
```



Change the size of the points and their transparency using the `s=` and `alpha=` arguments.

alpha takes values between 0 and 1.

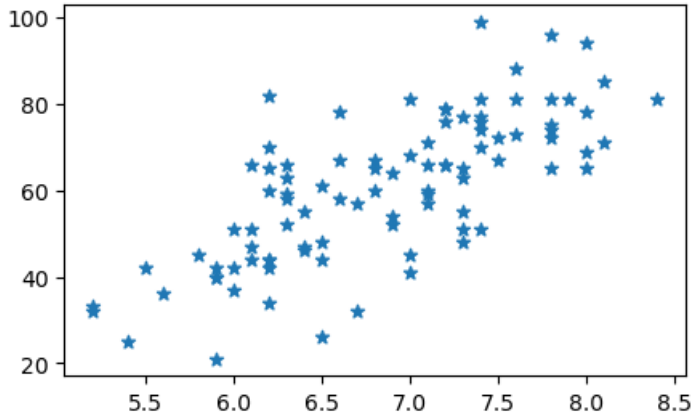
In [58]:
```
plt.scatter(imdb.rating, imdb.metascore, s = 50, alpha = 0.2)
plt.show()
```



Change the point type using marker

more info: https://matplotlib.org/stable/api/markers_api.html

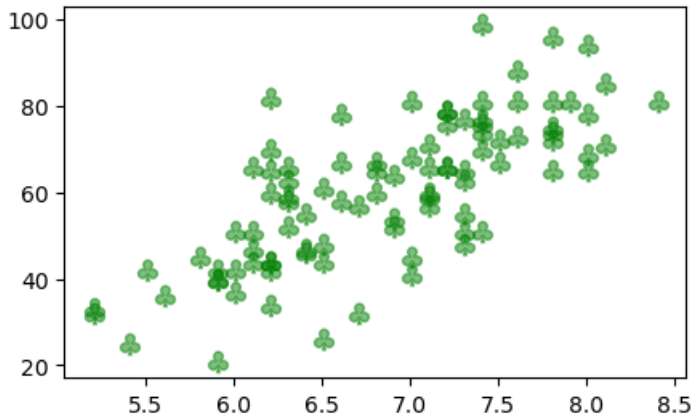
In [59]:
```
plt.scatter(imdb.rating, imdb.metascore, marker = '*')
plt.show()
```



There are even some novelty markers!

You can use latex symbols to show different markers https://artofproblemsolving.com/wiki/index.php/LaTeX:Symbols

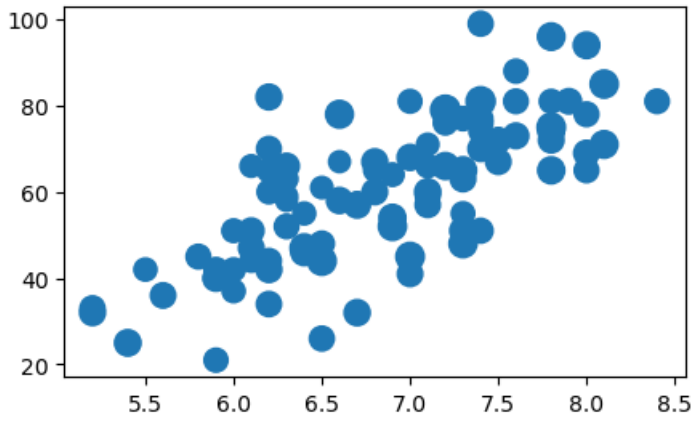
In [60]:
```
plt.scatter(imdb.rating, imdb.metascore, marker = r'$\clubsuit$', s = 100, c = 'g', alpha = 0.5)
plt.show()
```



Sizing points based on a continuous variable such as runtime allows us to give details on a third variable in our plot.

Larger points have longer runtimes.

In [61]:
```
plt.scatter(imdb.rating, imdb.metascore, s = imdb.runtime+10)
plt.show()
```



Coloring by a categorical variable is not as straightforward with matplotlib.

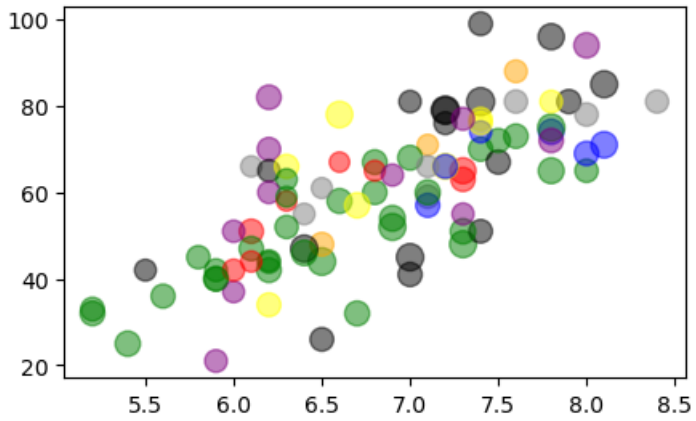
In [62]:
```
# this shows an error
# plt.scatter(imdb.rating, imdb.metascore, s = imdb.runtime, c = imdb.genre)
# plt.show()
```

Specify a dictionary in which each key is a category from the genre column, and each value is the color that will be used for that genre. Use it along with the map function to color points by genre.

In [63]:
```
imdb['genre'].unique()
```

Out[63]:
```
array(['Action', 'Horror', 'Biography', 'Comedy', 'Animation', 'Drama',
       'Crime', 'Adventure'], dtype=object)
```

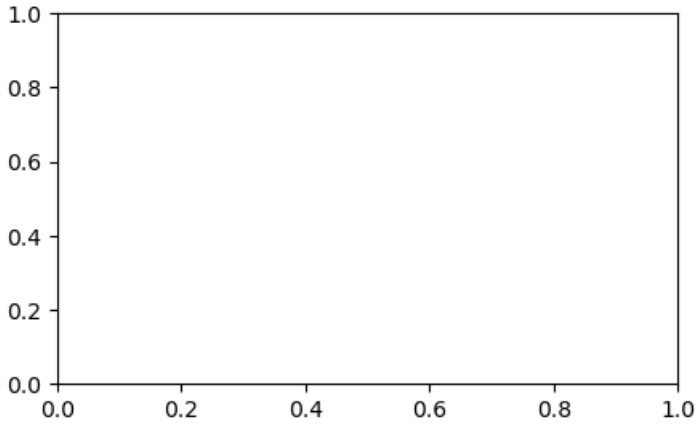
In [64]:
```
colors_genre = {'Horror':'red', 'Action':'green', 'Biography':'blue', 'Adventure':'yellow',
                'Drama': 'black', 'Crime': 'orange', 'Animation': 'grey', 'Comedy': 'purple'}

plt.scatter(imdb.rating, imdb.metascore, s = imdb.runtime, c = imdb['genre'].map(colors_genre), alpha=0.5)
plt.show()
```



Creating multiple plots in one window

`fig, ax = plt.subplots()` creates a figure with an array of axis objects that can be worked on.

In [65]:
```
fig, ax = plt.subplots()
plt.show()
```

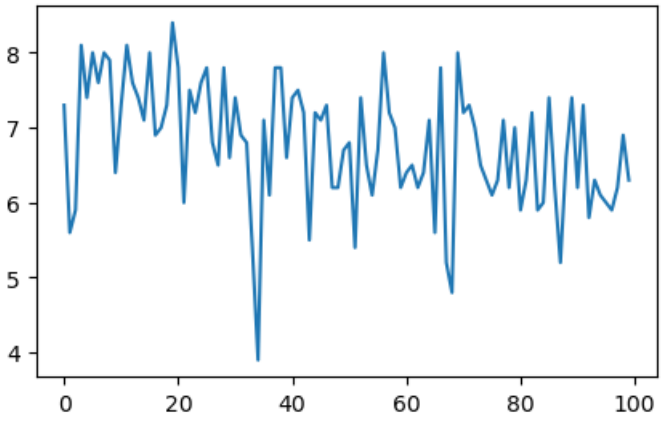


Create a figure and one subplot by specifying no argument in plt.subplots.

Use `ax.plot` to add a line plot to the figure. We are editing the axis object. We will see that certain functions are performed on the axis object (ax), while others are performed on the figure object (fig):

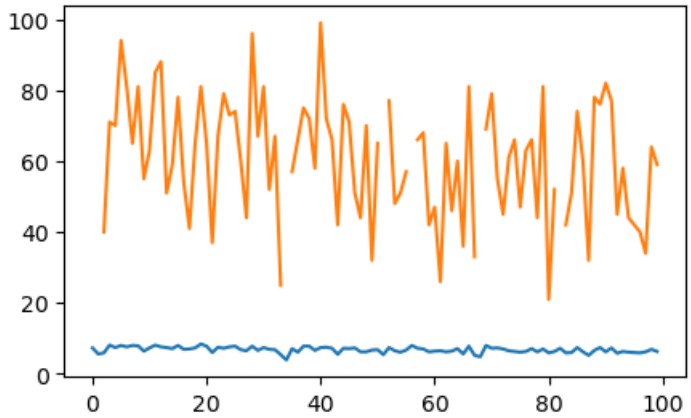
- fig: A reference to the Figure object, which is the "container" for your plots.
- ax: A reference to the Axes object, which represents a single subplot (the area where data is plotted).

In [66]:
```
fig, ax = plt.subplots()
ax.plot(imdb.rating)
plt.show()
```



Add a second line to the plot

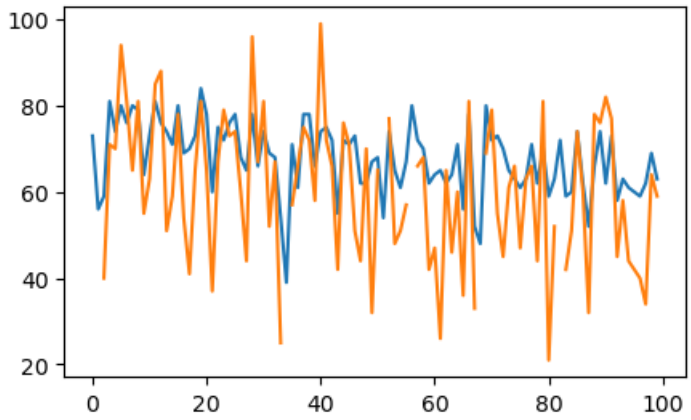
In [67]:
```
fig, ax = plt.subplots()
ax.plot(imdb.rating)
ax.plot(imdb.metascore)
plt.show()
```



In [ ]:
```

```

Plot the two lines on the same scale

In [68]:
```
rating_100 = imdb.rating * 10

fig, ax = plt.subplots()
ax.plot(rating_100)
ax.plot(imdb.metascore)
plt.show()
```

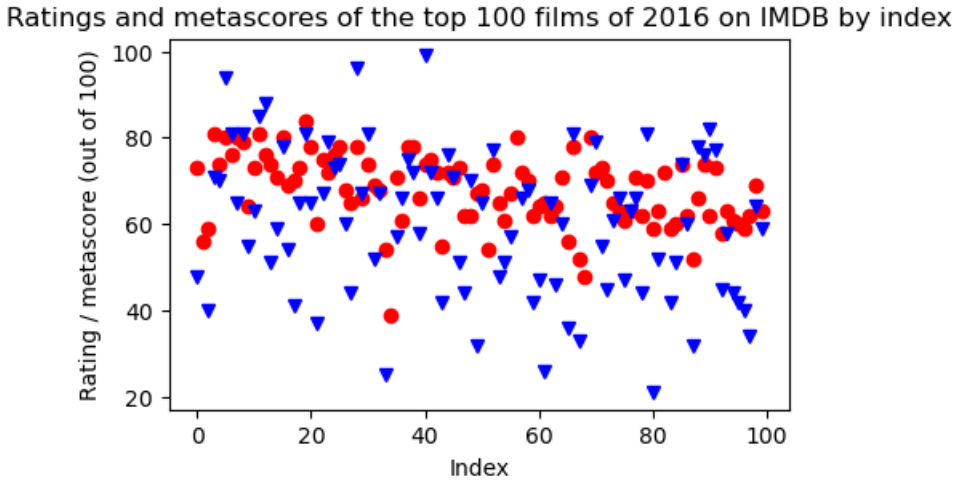


We can use a lot of the same arguments and functions as we have used before eg. `linestyle`, `color`, `marker`

However, notice that we use `ax.set_ylabel` instead of `plt.ylabel`.

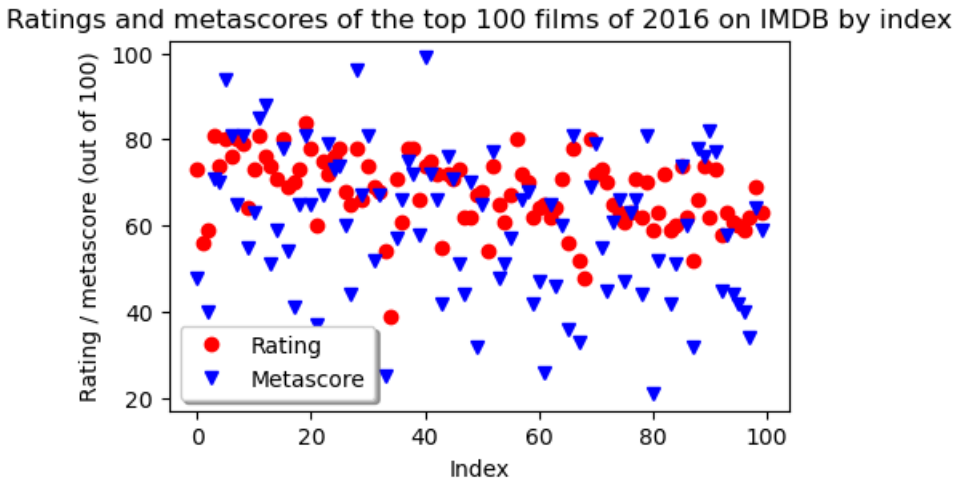
In [69]:
```
# help(plt.subplots)
```

In [70]:
```
fig, ax = plt.subplots()
ax.plot(rating_100, marker = 'o', linestyle = 'None', color = 'r')
ax.plot(imdb.metascore, marker = 'v', linestyle = 'None', color = 'b')
ax.set_ylabel('Rating / metascore (out of 100)')
ax.set_xlabel('Index')
ax.set_title('Ratings and metascores of the top 100 films of 2016 on IMDB by index')
plt.show()
```



This plot could do with a `legend`

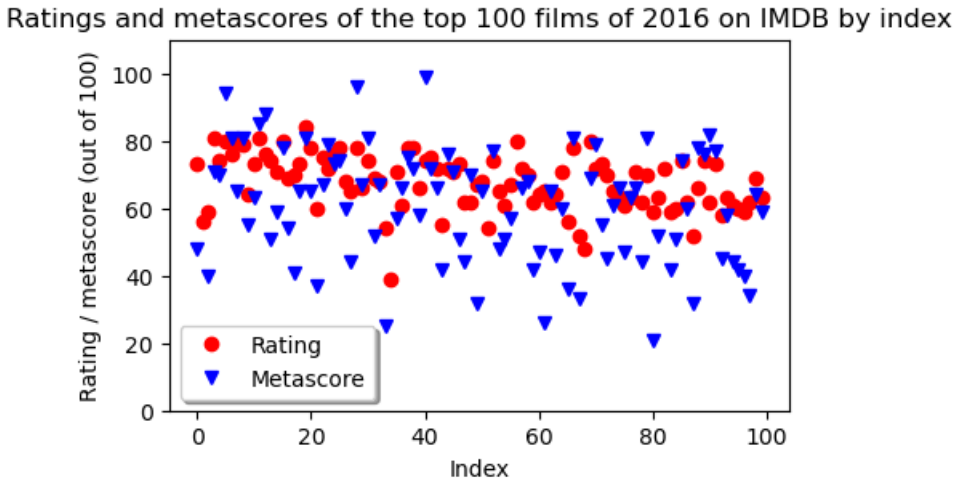
In [71]:
```
fig, ax = plt.subplots()
ax.plot(rating_100, marker = 'o', linestyle = 'None', color = 'r')
ax.plot(imdb.metascore, marker = 'v', linestyle = 'None', color = 'b')
ax.set_ylabel('Rating / metascore (out of 100)')
ax.set_xlabel('Index')
ax.set_title('Ratings and metascores of the top 100 films of 2016 on IMDB by index')
ax.legend(('Rating', 'Metascore'), loc = 'lower left', shadow = True)
plt.show()
```



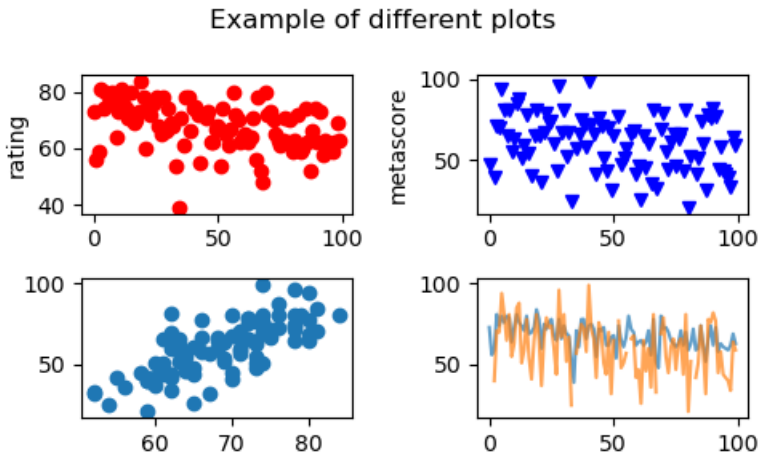
The legend may be blocking some points.

Maybe extend the y limits slightly using `ax.set_ylim`.

In [72]:
```
fig, ax = plt.subplots() # control to figure and axes
ax.plot(rating_100, marker = 'o', linestyle = 'None', color = 'r')
ax.plot(imdb.metascore, marker = 'v', linestyle = 'None', color = 'b')
ax.set_ylabel('Rating / metascore (out of 100)')
ax.set_xlabel('Index')
ax.set_title('Ratings and metascores of the top 100 films of 2016 on IMDB by index')
ax.legend(('Rating', 'Metascore'), loc = 'lower left', shadow = True)
ax.set_ylim([0,110])
plt.show()
```



In [110…
```
# using the variable axs for multiple Axes
fig, axs = plt.subplots(2, 2)

# Plot on each subplot
axs[0, 0].plot(rating_100, marker = 'o', linestyle = 'None', color = 'r')
axs[0, 0].set_ylabel("rating")

axs[0, 1].plot(imdb.metascore, marker = 'v', linestyle = 'None', color = 'b')
axs[0, 1].set_ylabel("metascore")

axs[1, 0].scatter(rating_100, imdb.metascore)
axs[1, 1].plot(rating_100, alpha=0.7)
axs[1, 1].plot(imdb.metascore, alpha=0.7)

fig.suptitle("Example of different plots", fontsize=12)
fig.tight_layout() # Adjust layout
plt.show()
```



Saving figures

- Suppose we are happy with this figure and we want to save it for inclusion in a paper or report.
- Use the `fig.savefig()` function. Specify the format you want to save it in.
- `.png` is a common format. A jpeg (.jpg) is good for inclusion on a website as it takes up less disk space.

The latest figure produced is saved to your current working directory as the name specified in the savefig function.

In [74]:
```
fig.savefig('rating_metascore.png')
fig.savefig('rating_metascore.jpg')
```

Short Exercises

- Import the top2018 dataset on Moodle. This is data on the top 100 songs on Spotify in 2018. Answer the following question using plots:
  - Which artists had the most Top 100 songs? (Note: Plot the first 10 artists, you can use Groupby to count the number of songs per artist)
  - Plot a scatter plot to see the relationship between loudness and energy. Is it a positive or negative relationship?
  - Display a boxplot on danceability. What is the median? How many outliers are shown?
  - Show the distribution of "acousticness", is it skewed? does it have a bell shape?
  - Formulate a question around the data and use a plot to answer that question. Be creative!

At this point, I would like you to complete the 'Introduction to Matplotlib' chapter in the 'Introduction to Data Visualization with Matplotlib' course on Data Camp.