WEEK 3

Data visualization using python

Data visualization refers to representing information in the form of visuals

Data Visualization can make your data speak! There is no doubt that when information is represented in the form of a picture like a graph or a chart, it can provide a much better understanding of the Data. Meaningful, effective, and aesthetically pleasing.

The key skill of a Data Scientist is to tell a compelling story after finding useful patterns and information from data. The plots and graphs can provide a clear description of the data. The Visuals can help support any claims you make based on the Data at hand.

They can be understood by any non-technical personnel which is the major advantage offered by them. While doing so, they let us convey most information while being very compact.

Data Visualization offers:

- Efficiency
- Clarity
- Accuracy

List of Python data visualization tools

1. Matplotlib:comprehensive library for creating static, animated, and interactive visualizations in Python.
2. Seaborn:Python data visualization library based on matplotib. It provides a high-level interface for drawing attractive and informative statistical graphics.
3. Folium:Folium is a powerful Python library that helps you create several types of Leaflet maps.
4. Plotly:The plotly python library is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases

Why Visualization

- The importance of data visualization is simple: it helps people see, interact with, and better understand data.

- Whether simple or complex, the right visualization can bring everyone on the same page, regardless of their level of expertise.
- It's hard to think of a professional industry that doesn't benefit from making data more understandable.
- A data visualization first and fore- most has to accurately convey the data. It must not mislead or distort

How to use the right visualization?

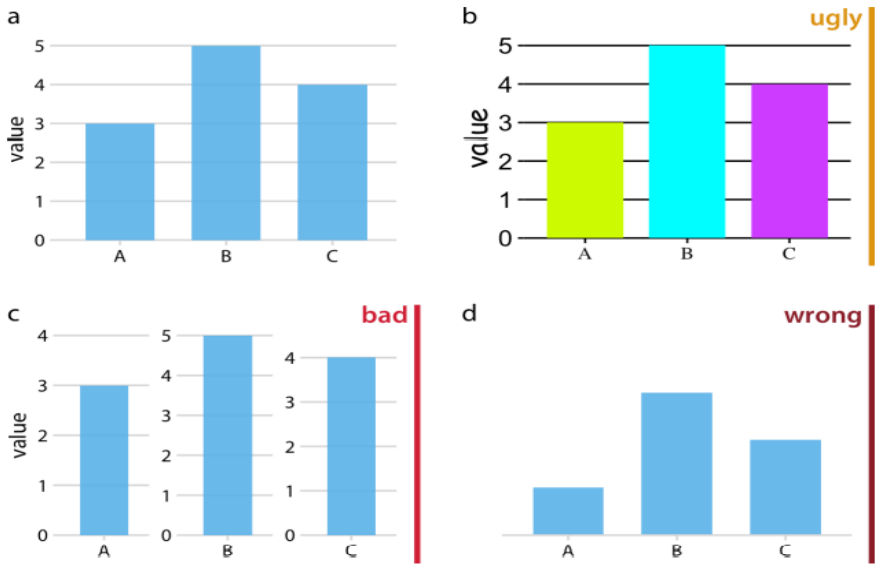
To extract the required information from the different visuals we create, it is quintessential that we use the correct representation based on the type of data and the questions that we are trying to understand. We will go through a set of most widely used representations below and how we can use them in the most effective manner.

Ugly, Bad, and Wrong Figures

Ugly: A figure that has aesthetic problems but otherwise is clear and informative

Bad :A figure that has problems related to perception; it may be unclear, confusing, overly complicated, or deceiving

Wrong : A figure that has problems related to mathematics; it is objectively incorrect

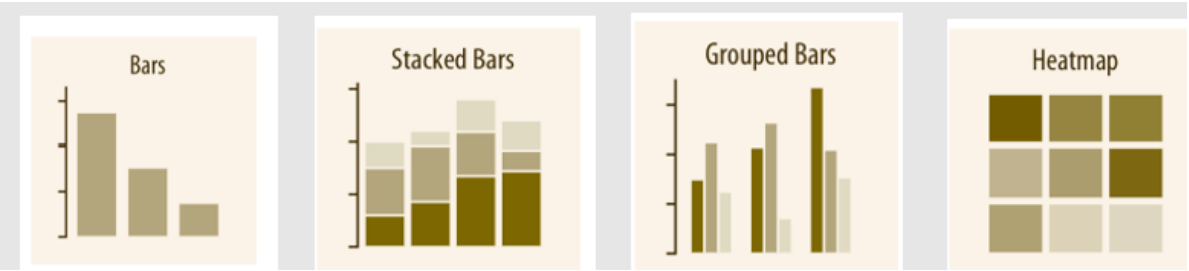


Amount, Distribution, Proportion, X-Y Relationships and Uncertainty

Amount : Numerical values shown for some set of categories usually done by bar charts

If there are two or more sets of categories for which we want to show amounts, we can group or stack the bars.

We can also map the categories onto the x and y axes and show amounts by color, via a heatmap



Distributions

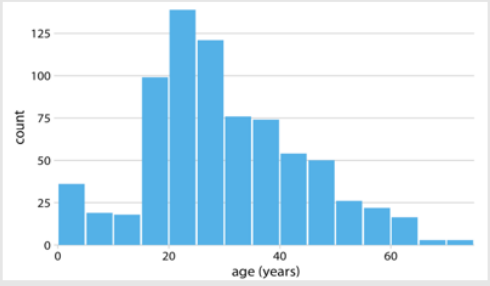
To understand how a particular variable is distributed in a dataset.

Numbers of passengers with known age on the Titanic

Ex: Histogram:

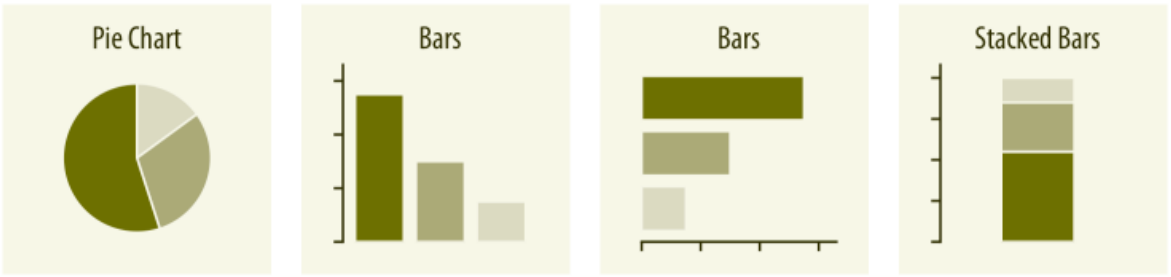
We can visualize this table by drawing filled rectangles whose heights correspond to the counts and whose widths correspond to the width of the age bins

Age range	Count	Age range	Count	Age range	Count
0–5	36	31–35	76	61–65	16
6–10	19	36–40	74	66–70	3
11–15	18	41–45	54	71–75	3
16–20	99	46–50	50		
21–25	139	51–55	26		
26–30	121	56–60	22		



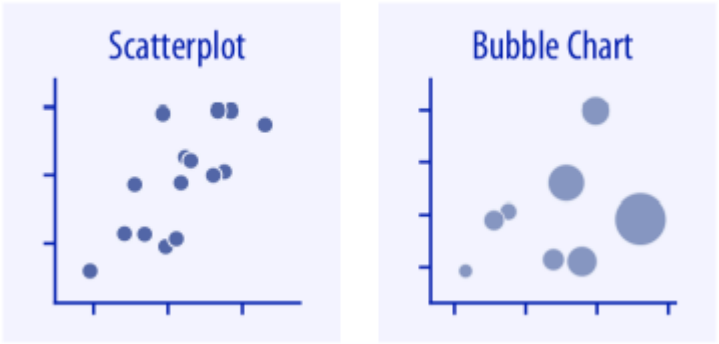
Proportions

Proportions can be visualized as pie charts, side-by-side bars.Examples include regional differences in happiness, economic indicators or crime, demographic differences in voting patterns



x–y relationships

For visualization when we want to show one quantitative variable relative to another. If we have three quantitative variables, we can map one onto the dot size, creating a variant of the scatterplot called a bubble chart.



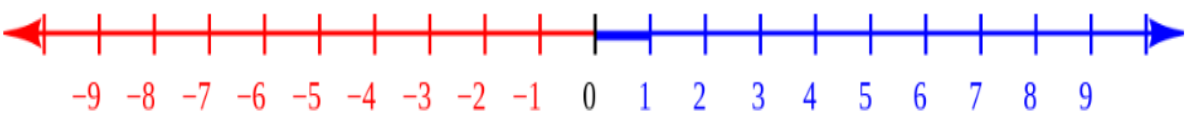
Uncertainty

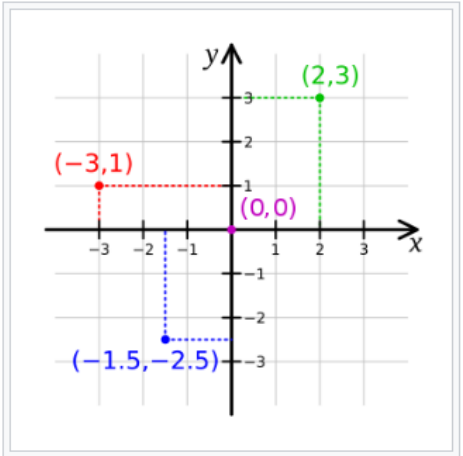
Error bars are meant to indicate the range of likely values for some estimate or measurement. They extend horizontally and/or vertically from some reference point representing the estimate or measurement

Coordinate Systems and Axes:

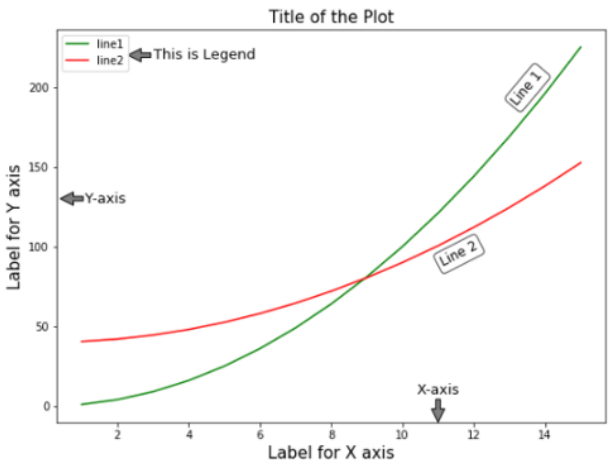
In geometry, a coordinate system is a system that uses one or more numbers or coordinates to uniquely determine the position of the points.

Example: The simplest example of a coordinate system is the identification of points on a line with real numbers using the number line.





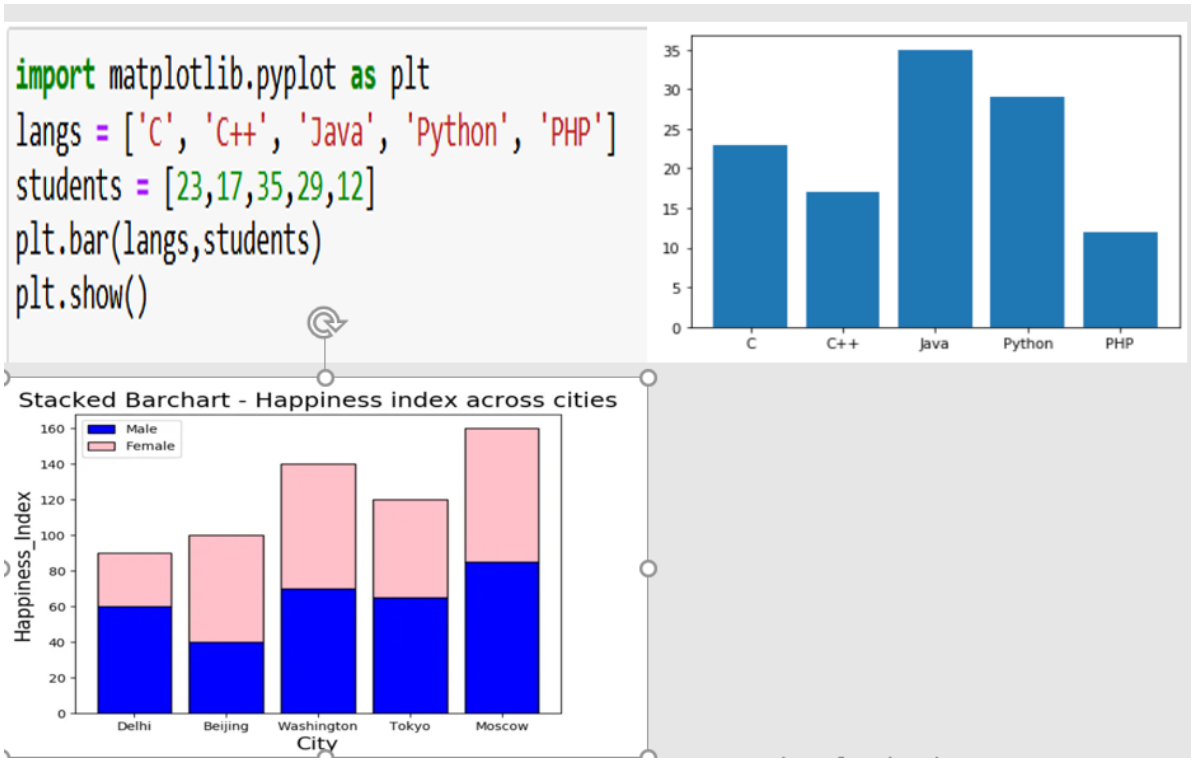
The Cartesian coordinate system in the plane.



Bar chart

A bar chart is used when we want to compare metric values across different subgroups of

the data. If we have a greater number of groups, a bar chart is preferred over a column chart. Column charts are mostly used when we need to compare a single category of data between individual sub-items, for example, when comparing revenue between regions

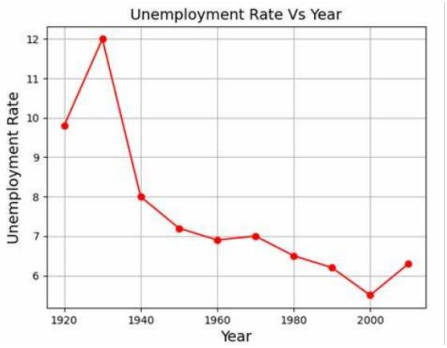


Line chart

A line chart is used for the representation of continuous data points. This visual can be effectively utilized when we want to understand the trend across time. Line charts are typically used to show the overall trend of a certain topic

Ex: To overall price movement of a stock or people's interest, Unemployment rate over years

Sample line chart looks like this



How to plot a line chart in Matplotlib?

Line charts are great to show trends in data by plotting data points connected with a line. In matplotlib, you can plot a line chart using pyplot's plot() function. The following is the syntax to plot a line chart:

```
import matplotlib.pyplot as plt

plt.plot(x_values, y_values)
```

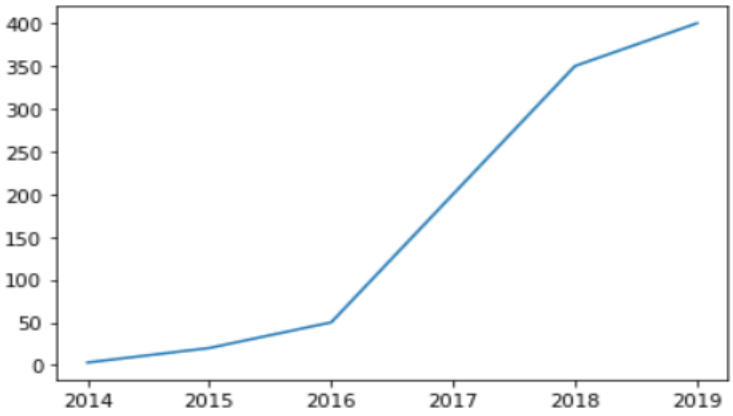
Here, x_values are the values to be plotted on the x-axis and y_values are the values to be plotted on the y-axis.

Example

Plot a line chart with default parameters

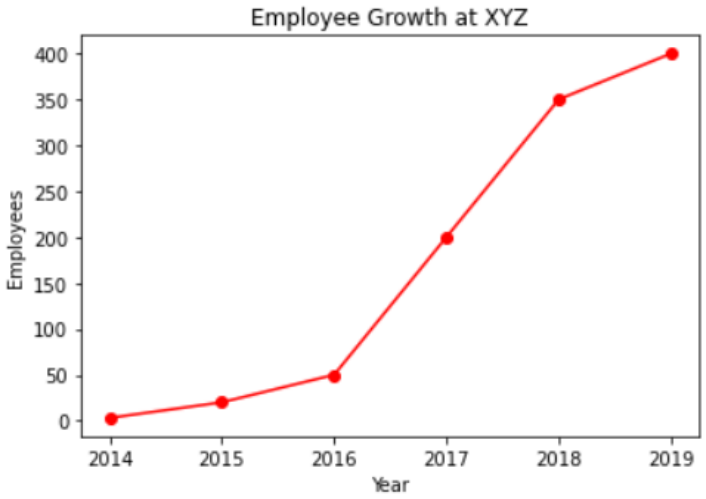
We have the data on the number of employees of a company XYZ, A year on year, and want to plot it on a line chart using matplotlib.

```
import matplotlib.pyplot as plt

# number of employees of XYZ
emp_count = [3, 20, 50, 200, 350, 400]
year = [2014, 2015, 2016, 2017, 2018, 2019]

# plot a line chart
plt.plot(year, emp_count)

plt.show().
```

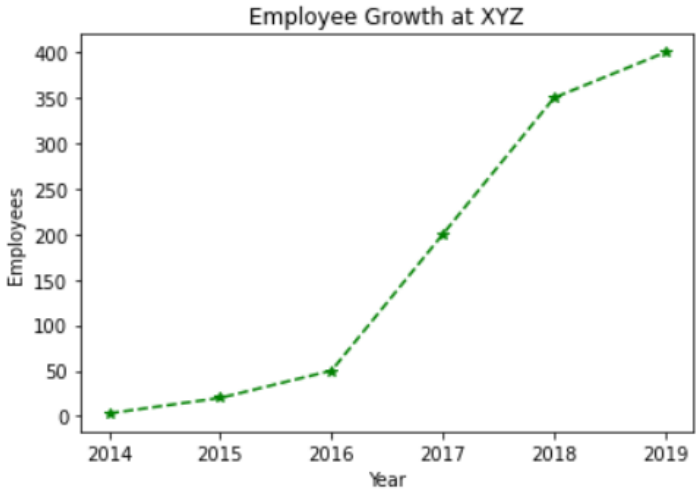


Customize the formatting of a line chart

```
import matplotlib.pyplot as plt

# number of employees of XYZ
emp_count = [3, 20, 50, 200, 350, 400]
year = [2014, 2015, 2016, 2017, 2018, 2019]

# plot a line chart
plt.plot(year, emp_count, 'o-r')
# set axis titles
plt.xlabel("Year")
plt.ylabel("Employees")
# set chart title
plt.title("Employee Growth at XYZ")
plt.show()
```

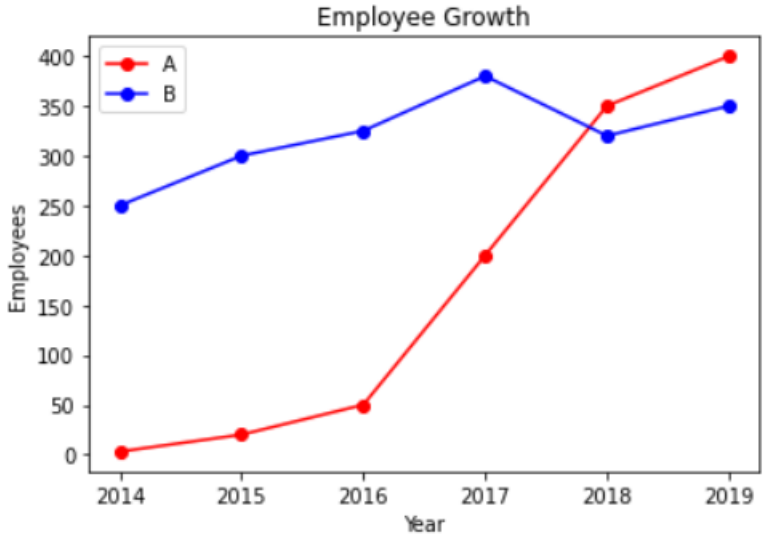


```
import matplotlib.pyplot as plt

# number of employees of XYZ
emp_count = [3, 20, 50, 200, 350, 400]
year = [2014, 2015, 2016, 2017, 2018, 2019]

# plot a line chart
plt.plot(year, emp_count, '*--g')
# set axis titles
plt.xlabel("Year")
plt.ylabel("Employees")
# set chart title
plt.title("Employee Growth at XYZ")
plt.show()
```



Line chart with multiple lines on the same scale

```
#Lines with same scale
import matplotlib.pyplot as plt

# number of employees
emp_countA = [3, 20, 50, 200, 350, 400]
emp_countB = [250, 300, 325, 380, 320, 350]
year = [2014, 2015, 2016, 2017, 2018, 2019]
# plot two lines
plt.plot(year, emp_countA, 'o-r')
plt.plot(year, emp_countB, 'o-b')
# set axis titles
plt.xlabel("Year")
plt.ylabel("Employees")
# set chart title
plt.title("Employee Growth")
# legend
plt.legend(['A', 'B'])
plt.show()
```

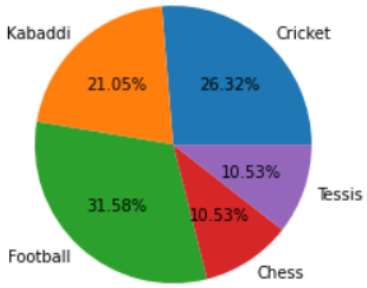


Pie chart

Pie charts can be used to identify proportions of the different components in a given whole. Pie charts are used to **present categorical data** in a format that highlights how each data point contributes to a whole, that is 100%.

pie chart showing interest of 200 class students in sports

```
from matplotlib import pyplot as plt
import numpy as np
langs = ['Cricket', 'Kabaddi', 'Football', 'Chess', 'Tessis']
students = [50,40,60,20,20]
plt.pie(students, labels = langs,autopct='%1.2f%%')
plt.show()
```



Plotting Histogram using the Matplotlib

It is a graph showing the number of observations within each given interval. A histogram is a graph that shows the frequency of numerical data using rectangles.

Example: Say you ask for the height of 250 people, you might end up with a histogram like

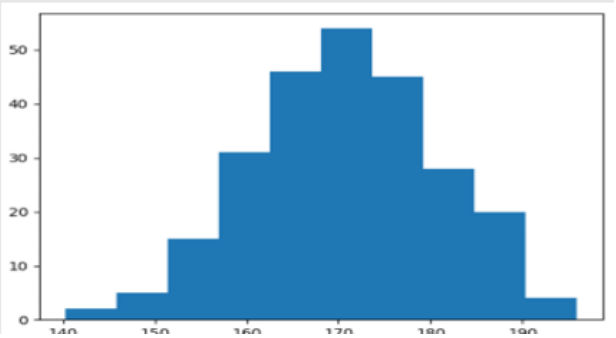
2 people from 140 to 145cm

5 people from 145 to 150cm

15 people from 151 to 156cm

For simplicity we use NumPy to randomly generate an array with 250 values, where the values will concentrate around 170, and the standard deviation is 10

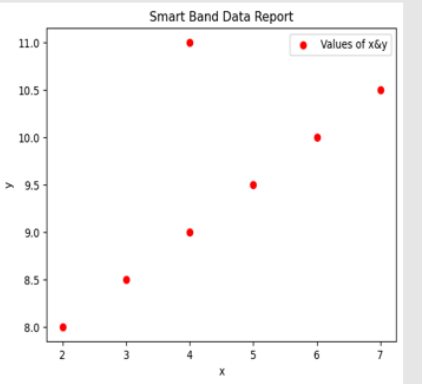
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```



Scatter plots

Scatter plots can be leveraged to identify relationships between two variables. It can be effectively used in circumstances where the dependent variable can have multiple values for the independent variable.

```
import matplotlib.pyplot as plt
#Scatter plot
x = [ 2, 3, 4, 5, 6, 7, 4]
y = [8, 8.5, 9, 9.5, 10, 10.5, 11]
plt.scatter(x,y,label='Values of x&y',color='r')
plt.title('Smart Band Data Report')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
# Print the chart
plt.show()
```



Saving Plot as image

```python
import matplotlib.pyplot as plt

# Creating data
year = ['2010', '2002', '2004', '2006', '2008']
production = [25, 15, 35, 30, 10]

# Plotting barchart
plt.bar(year, production)

# Saving the figure.
plt.savefig("output.jpg")
```