DATA VISUALIZATION TOOLS

What is data visualization and why is it important?

Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from.

The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets.
Goal: Visualization is a simple technique that you can use to create a strong mental image of a future event. With good use of visualization, you can practice in advance for the event, so that you can prepare properly for it. And by visualizing success, you can build the self-confidence you need to perform well.

Data visualization methods refer to the creation of graphical representations of information. Visualization plays an important part in data analytics and helps interpret big data in a real-time structure by utilizing complex sets of numerical or factual figures.

Commonly used Data Visualizations Tools

- **Chart.js (javascript)**
- **Sigma.js (javascript)**
- **Google charts (javascript)**
- **Pandas, matplotlib, seaborn (python)**

Common Types of Data Visualizations

- Bar Chart
- Doughnut Chart or Pie Chart
- Line Graph or Line Chart
- Pivot Table
- Scatter Plot

Visualizations-

Using the dataset of finding probability of drawing 20 independent bulbs from a lot so that the bulb drawn works for more than 500 hrs.
Here are the implementations of the following problems using various different data visualization tools -

1. Python Libraries

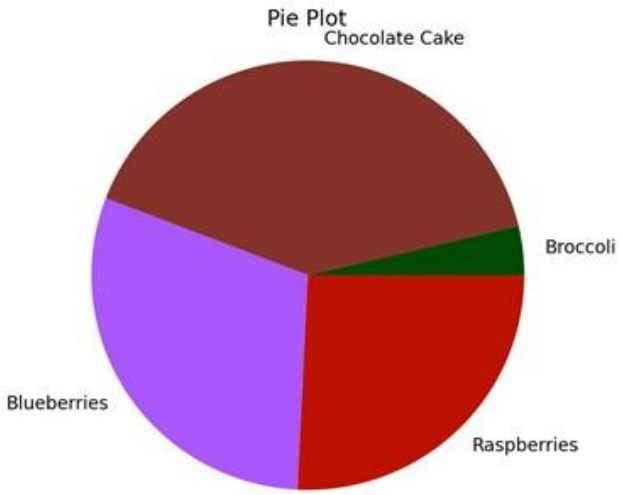
Here are the implementations of the following problems using various different data visualization tools -
Such implementations can be easily done using matplotlib library in python or using scilab , seaborn

Here are the implementations of the following problems using various different data visualization tools -
Such implementations can be easily done using matplotlib library in python or using scilab , seaborn

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB.
Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

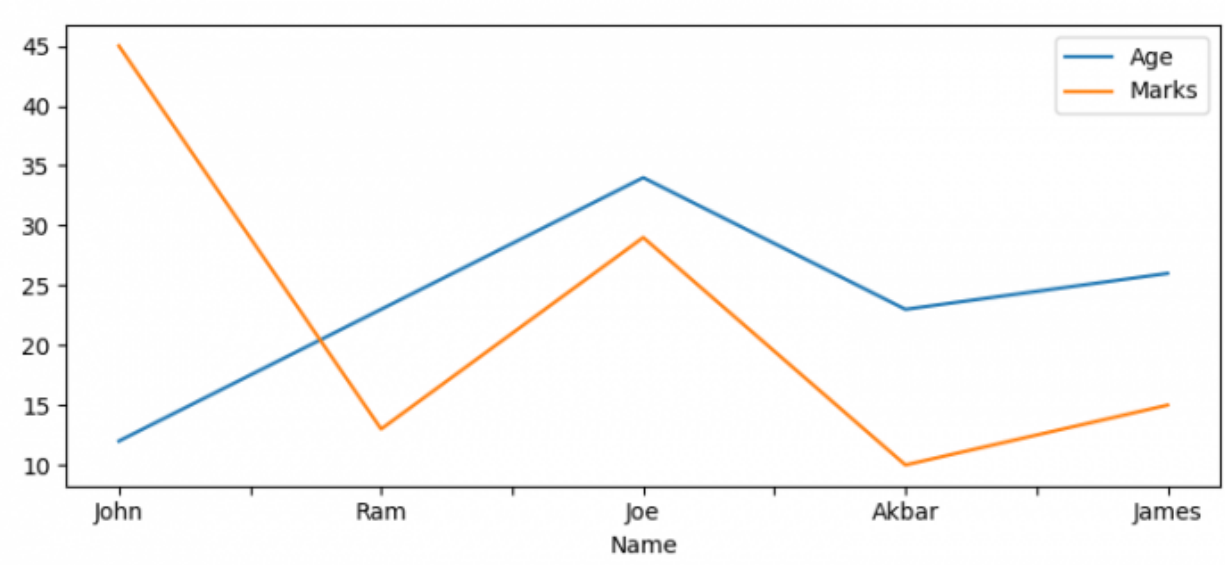
Implementing Pandas and Matplotlib

```
import matplotlib.pyplot as plt

# Data labels, sizes, and colors are defined:
labels = 'Broccoli', 'Chocolate Cake', 'Blueberries', 'Raspberries'
sizes = [30, 330, 245, 210]
colors = ['green', 'brown', 'blue', 'red']

# Data is plotted:
plt.pie(sizes, labels=labels, colors=colors)

plt.axis('equal')
plt.title("Pie Plot")
plt.show()
```

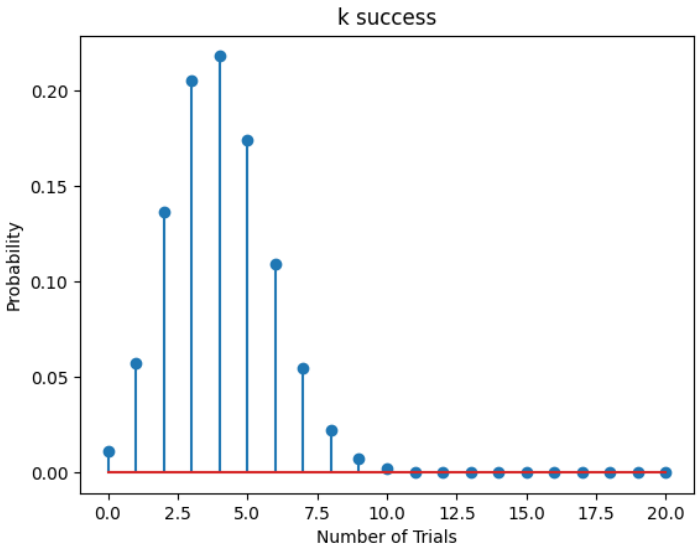


Pandas & Matplotlib with csv dataset

```
import pandas as pd
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True

headers = ['Name', 'Age', 'Marks']

df = pd.read_csv('student.csv', names=headers)

df.set_index('Name').plot()

plt.show()
```



```python
# import matplotlib as mtt
import matplotlib.pyplot as mt
numEvents = int(input("Enter number of times bulb is drawn: "))
p = 0.2
time = 500
Event = {}
# LOGIC
# nCr * p ^k  * (1-p)^n-k
def nCr(n, r):
    return (factorial(n) / (factorial(r) * factorial(n - r)))
def factorial(n):       You, yesterday • feat: assignment 1
    if n == 0:
        return 1
    result = 1
    for i in range(2, n+1):
        result = result * i
    return result
n = numEvents
r = 0
for k in range(numEvents+1):
    " Graph --- prob of success at kth trial vs trials"
    # z = (1-p)**(k)
    print(int(nCr(n, r)))
    ans = (int(nCr(n, r))) * p**(k) * (1-p)**(n-k)
    r = r + 1
    Event[k] = ans

x_axis = list(Event.keys())
y_axis = list(Event.values())
print(x_axis)
print(y_axis)
mt.bar(x_axis, y_axis)
# mt.plot(x_axis, y_axis)
mt.title("k success")
mt.xlabel("Number of Trials")
mt.ylabel("Probability")
mt.show()
```



2. Chart.js

Chart.js is a popular JavaScript charting toolkit that is open source. It has support for eight various types of charts, including pies, lines, and bars. Javascript being a very popular programming language, the library is actively maintained by a large community of developers.

With JavaScript

👉 Bar Chart

👉 Line Chart

👉 Scatter Chart

```
const labelsBarChart = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
const dataBarChart = {
  labels: labelsBarChart,
  datasets: [
    {
      label: "k success",
      backgroundColor: "hsl(252, 82.9%, 67.8%)",
      borderColor: "hsl(252, 82.9%, 67.8%)",
      data: [0.011529215046068483, 0.05764607523034241, 0.13690942867206324, 0.20536414300809488, 0.21819940194610077,
      0.17455952155688062, 0.1090997009730504, 0.05454985048652519, 0.022160876760150862, 0.007386958920050287,
      0.0020314137030138287, 0.0004616849325031429, 8.656592484433929e-05, 1.3317834591436814e-05, 1.6647293239296018e-06,
      1.6647293239296019e-07, 1.3005697843200014e-08, 7.65041049600001e-10, 3.187671040000004e-11, 8.388608000000009e-13,
      1.0485760000000012e-14],
    },
  ],
};
const configBarChart = {
  label: "label",
  type: "bar",
  data: dataBarChart,
  options: {},
};
const configLineChart = {
  label: "label",
  type: "line",
  data: dataBarChart,
  options: {},
};
const configRadarChart = {
  label: "label",
  type: "scatter",
  data: dataBarChart,
  options: {},
};
var chartBar = new Chart(
  document.getElementById("chartBar"),
  configBarChart
);
var chartLine = new Chart(
  document.getElementById("chartLine"),
  configLineChart
);
var chartBar = new Chart(
  document.getElementById("chartRadar"),
  configRadarChart
);      You, last month • feat: added is
```

Dataset Configuration

Name	Type	Description
`label`	`string`	The label for the dataset which appears in the legend and tooltips.
`clip`	`Number|object`	How to clip relative to chartArea. Positive value allows overflow, negative value clips that many pixels inside chartArea. 0 = clip at chartArea. Clipping can also be configured per side: clip: {left: 5, top: false, right: -2, bottom: 0}
`order`	`number`	The drawing order of the dataset. Also affects order for stacking, tooltip and legend.
`stack`	`string`	The ID of the group to which this dataset belongs to (when stacked, each group will be a separate stack). Defaults to dataset `type`.
`parsing`	`Boolean|object`	How to parse the dataset. The parsing can be disabled by specifying parsing: false at chart options or dataset. If parsing is disabled, data must be sorted and in the formats the associated chart type and scales use internally.
`hidden`	`boolean`	Configure the visibility of the dataset. Using `hidden: true` will hide the dataset from being rendered in the Chart.

The `data` property of a dataset can be passed in various formats. By default, that `data` is parsed using the associated chart type and scales. If the `labels` property of the main `data` property is used, it has to contain the same amount of elements as the dataset with the most values. These labels are used to label the index axis (default x axes). The values for the labels have to be provided in an array. The provided labels can be of the type string or number to be rendered correctly. In case you want multiline labels you can provide an array with each line as one entry in the array.