matplotlib

May 7, 2025

1 Data Visualization

[ ]:
```
Data visualization is the graphical representation of information and data. It␣
 ↪uses visual elements like charts, graphs, maps, and dashboards
to make complex data more understandable and accessible.

Why Use Data Visualization?
Simplifies complex data - Makes patterns, trends, and outliers easier to spot.

Improves decision-making - Quick visual insights help stakeholders act faster.

Enhances storytelling - Communicates findings clearly and persuasively.

Supports data analysis - Complements statistical methods by making data more␣
 ↪intuitive.

Note - Pictures speak louder than number

Imp Points-  1) Human adapts Image 60000 Times Faster than Text
             2) Charts depends on the data
             3) Data Visualization Amplifyies Human Memory
```

2 Key terms

[ ]:
```
''' 1] Data Point - Single pair of value
    2] X-axis , Y-axis - Horizontal(input) , Vertical(output)
    3] Figure - All Graph are lies
    4] Axes - Graph box (actual area where the graph is plot)
    5] plot- Actual line or points which in the graph
    6] Market - A symbol or dot
    7] Line Style - continuous connection between the data points and his style␣
 ↪(use for style line)
    8] color - used to define the color
    9] Legend - A small Box that shows which line what's represent
    10] Label - Used for explaining what x-axis do and what y-axis do
    11] Grid - Used to draw horizontal and vertical line in the background
```

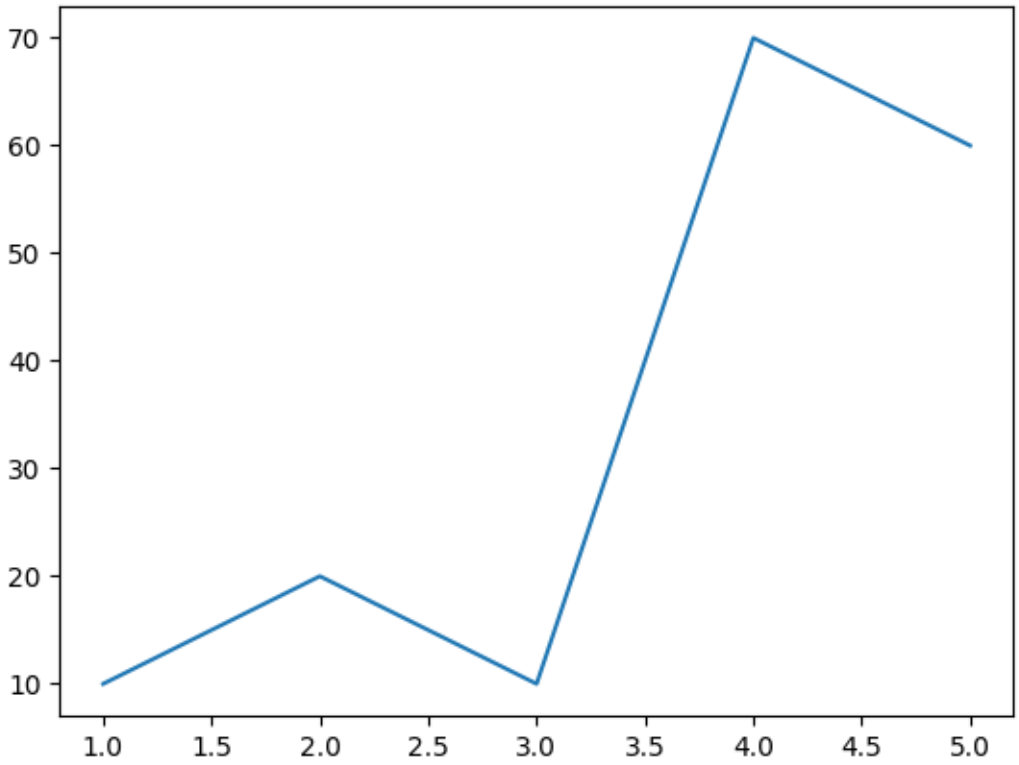
```
   12] Function - Block of code which perform a specific task
   13] Object-Oriented API - Advance techinque to create plots
   14] Methods - A method is a function that belongs to an object or a class
   15] Parameters - A parameter is a variable used to pass information into a␣
↪function or method
   16] Keyword Arguments - Keyword arguments (also called named arguments) are␣
↪a way to pass values to a function using the parameter name
   17] DPI-(Dots per Inch) uesd to control the resolution of dots (clear␣
↪Image)
   18] The system Matplotlib uses to actually draw the plots on the screen or␣
↪used to save it [example - TKAGG,QT AGG,WEBGG]
   19] Pyplot - It is a function that provide various features
   20] Symbols
       1] 'o'-circle
       2] 's'-square
       3] '^'-triangle
       4] 'x'-cross
       5] 'D'-Diamond
       6] '+'- + sign

   '''
```

3 Importing Matplotlib

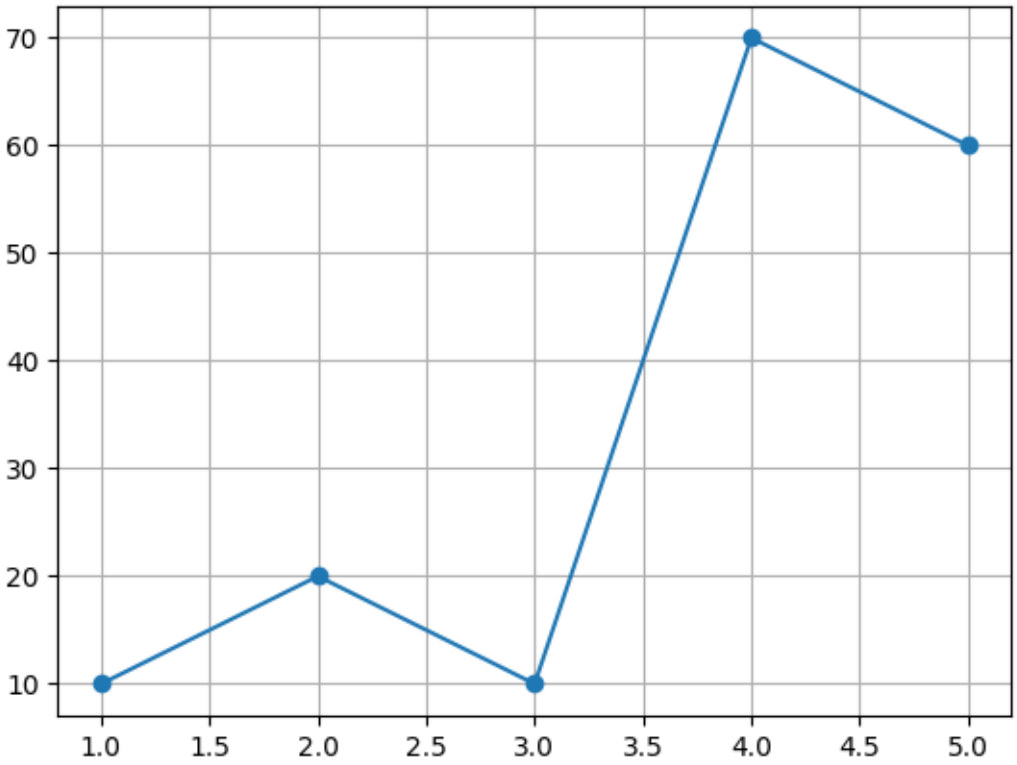
[2]:
```
import matplotlib.pyplot as plt
```

4 Plot function

[17]:
```
# Creating a List-
X=[1,2,3,4,5]  #- x axis Horizontal
Y=[10,20,10,70,60] #- y -axis Vertical

plt.plot(X,Y)
plt.show()

# It like as 1=10 , 2=20 , 3=10 , 4=70 , 5=60.
```



[20]:
```
# If we use grid the the at the background the box pattern is created


X=[1,2,3,4,5]  #- x axis Horizontal
Y=[10,20,10,70,60] #- y -axis Vertical
plt.plot(X,Y,marker='o')  # for points use marker ='o'
plt.grid(True)
plt.show()
```



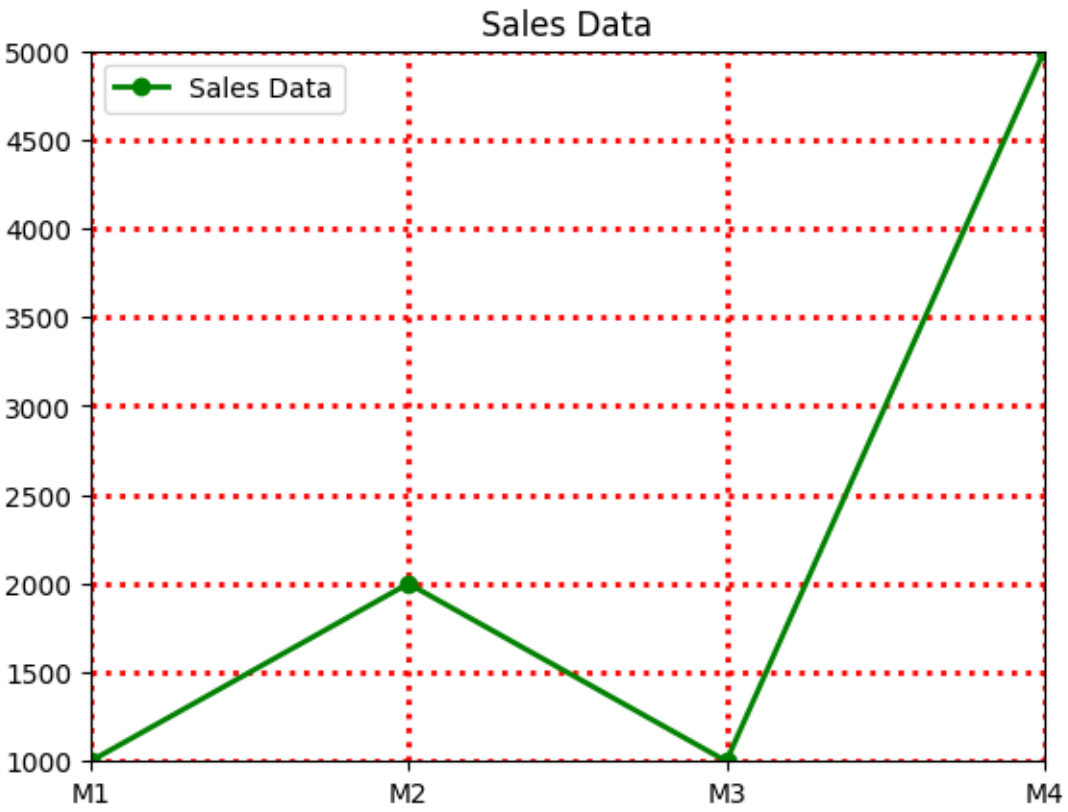
[ ]:

[25]:
```
x=['Mon','Tues','Wed','Thu','Fri']
y=[10,30,20,50,15]
plt.plot(x,y,marker='o')
plt.title('Bakery Sales This Week')
plt.xlabel('Days')
plt.ylabel('Sales of cake')
plt.grid(True)
plt.show()
```

[30]:
```
Month=[1,2,3,4,5,6,7]
sales=[1000,2000,1000,5000,10000,3000,200]

plt.plot(Month,sales ,color='green',linewidth='2',marker='o',label='Sales␣
 ↪Data')  # Used For Display

plt.grid(color='red',linestyle=':',linewidth='2') # color - Specifies the color
                                                  # Linestyle - Style the line␣
 ↪in different ways
                                                  #linewidth - Increase the␣
 ↪width of line

plt.title('Sales Data')

plt.legend() # Default
             # pass the parameter like loc='upper left' , loc='lower right'
             # Use only when pass the label

plt.xlim(1,4) # Using to apply limit upto
```

```
plt.ylim(1000,5000)

plt.xticks([1,2,3,4],['M1','M2','M3','M4']) # used to replace numbers  with␣
 ↪meaningful texts
plt.yticks([],[])
```

[30]:
```
([<matplotlib.axis.XTick at 0x26c44b37890>,
  <matplotlib.axis.XTick at 0x26c449c5310>,
  <matplotlib.axis.XTick at 0x26c449c5a90>,
  <matplotlib.axis.XTick at 0x26c449c6210>],
 [Text(1, 0, 'M1'), Text(2, 0, 'M2'), Text(3, 0, 'M3'), Text(4, 0, 'M4')])
```

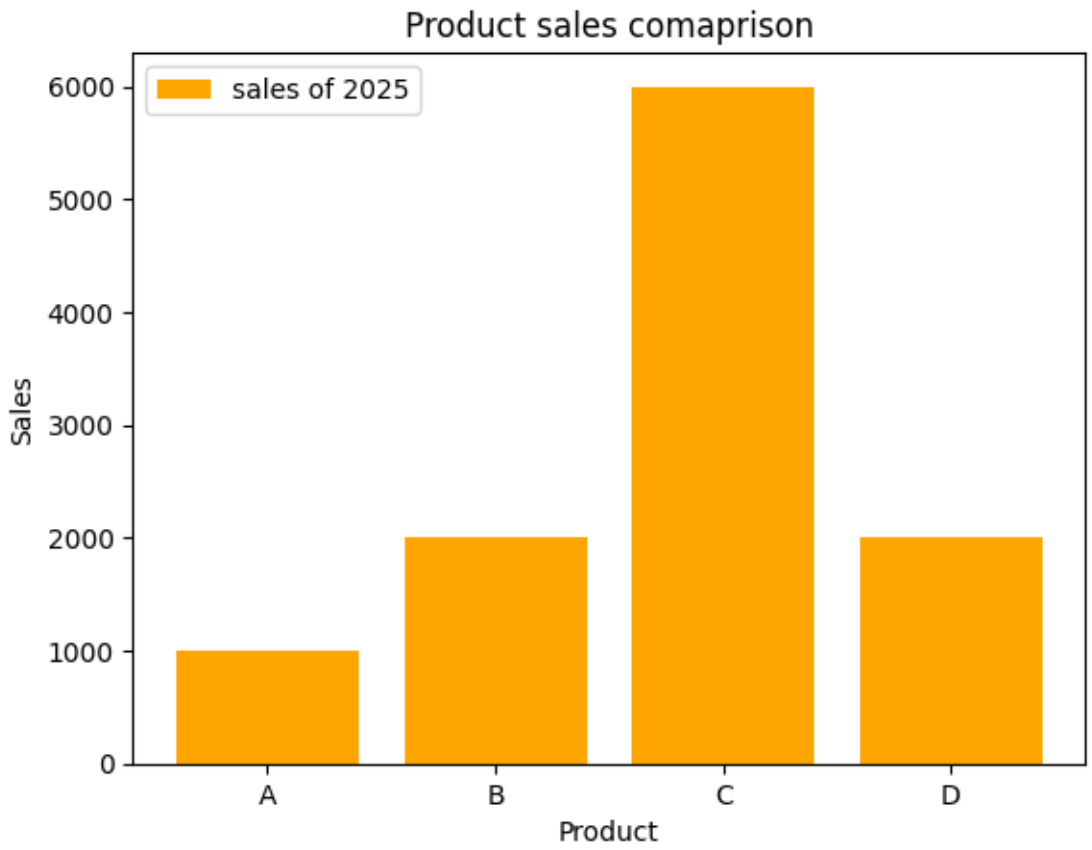


5 Bars

[44]:
```
''' Bar -A bar in Matplotlib is a visual element that displays the magnitude of␣
 ↪a variable for a
given category using a rectangular shape, typically constructed using the plt.
 ↪bar() or plt.barh() function.
'''
```

```
products=['A','B','C','D']
sales=[1000,2000,6000,2000]
plt.bar(products,sales,color='orange',label='sales of 2025') # Passing␣
 ↪different parameters -  width,label,color
plt.title('Product sales comaprison')
plt.xlabel('Product')
plt.ylabel('Sales')
plt.legend()
```

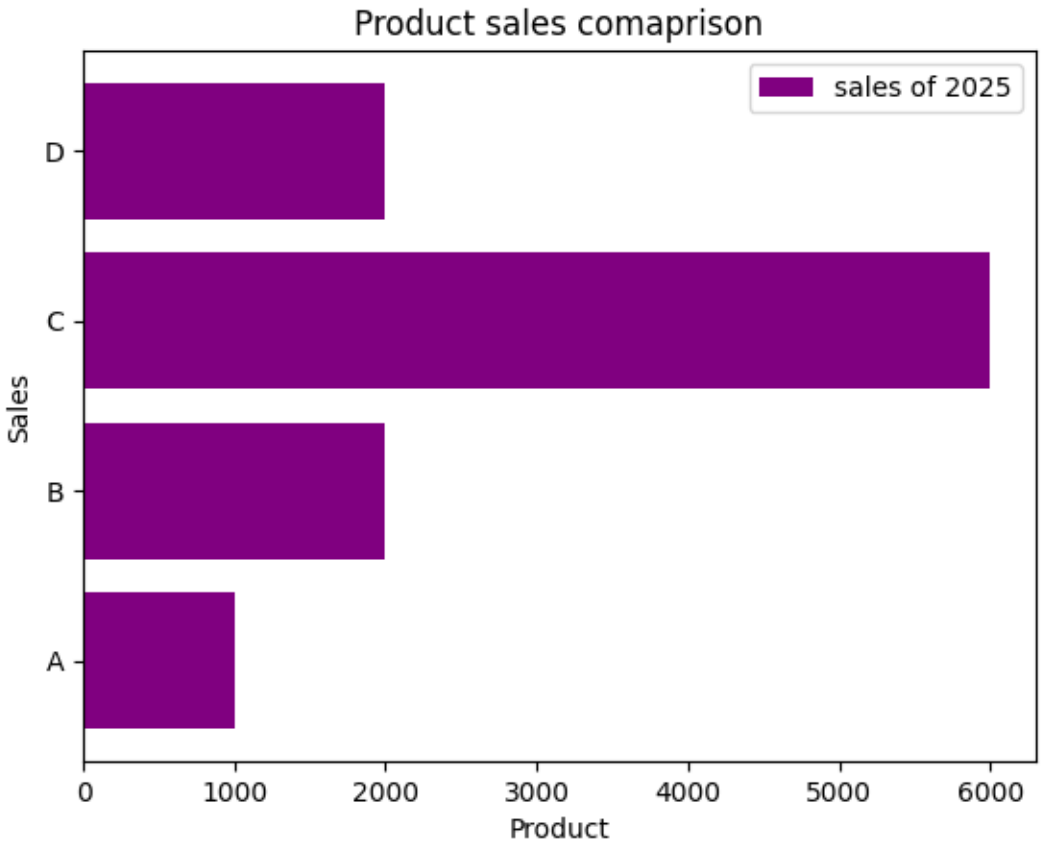
[44]: <matplotlib.legend.Legend at 0x26c46dc3b10>



[46]:
```
'Bars- It can be horizontal'

products=['A','B','C','D']
sales=[1000,2000,6000,2000]
plt.barh(products,sales,color='purple',label='sales of 2025') # Passing␣
 ↪different parameters -  width,label,color
plt.title('Product sales comaprison')
plt.xlabel('Product')
plt.ylabel('Sales')
```

```
plt.legend()
```

[46]: <matplotlib.legend.Legend at 0x26c46e9a0d0>

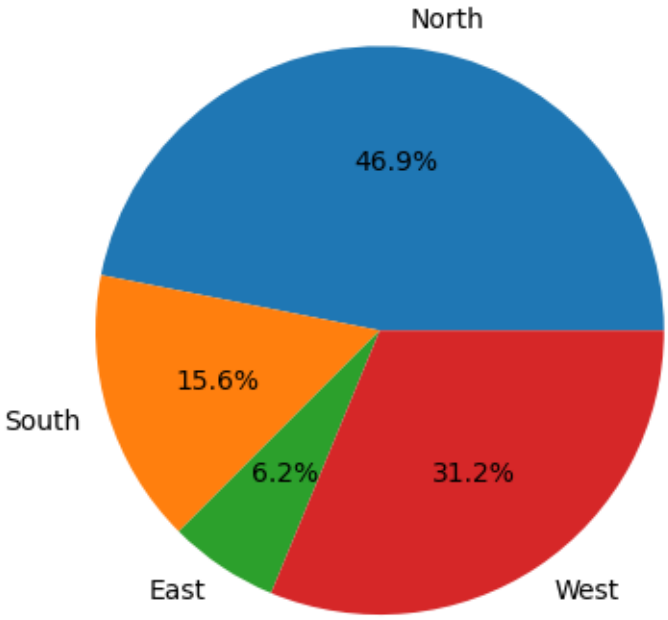


6 Piechart

[49]:
```
'''piechart - A pie chart is a circular statistical graphic divided into slices␣
 ↪to illustrate numerical proportions.
Each slice represents a category and its angle or area is proportional to the␣
 ↪percentage or share of that category
in the total dataset.'''

regions=['North','South','East','West']
revenu=[3000,1000,400,2000]
plt.pie(revenu,labels=regions,autopct='%1.1f%%')  # %1.1f%% shows the percentage
```

[49]:
```
([<matplotlib.patches.Wedge at 0x26c44226ba0>,
  <matplotlib.patches.Wedge at 0x26c47462210>,
  <matplotlib.patches.Wedge at 0x26c474625d0>,
```

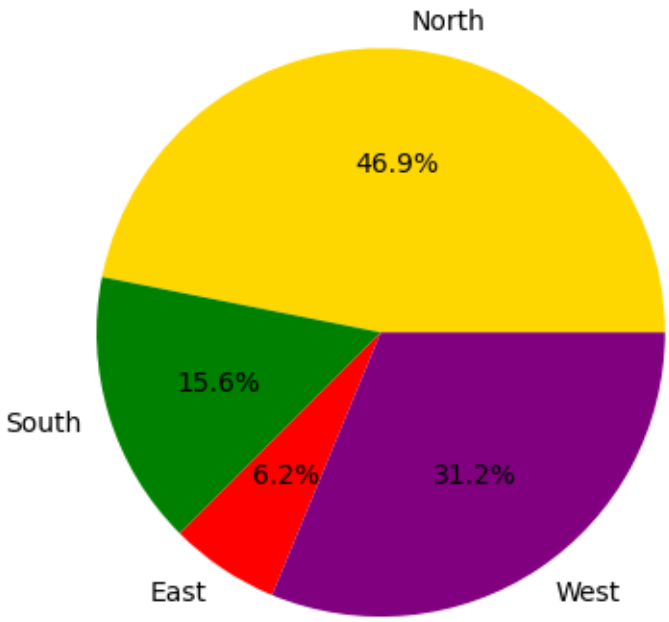
```
 <matplotlib.patches.Wedge at 0x26c47462990>],
[Text(0.10781885028307768, 1.0947031997412062, 'North'),
 Text(-1.0526343030870855, -0.3193133632724537, 'South'),
 Text(-0.6111270287166031, -0.9146167256135307, 'East'),
 Text(0.611127486653157, -0.9146164196301068, 'West')],
[Text(0.05881028197258782, 0.5971108362224761, '46.9%'),
 Text(-0.5741641653202284, -0.17417092542133836, '15.6%'),
 Text(-0.3333420156636016, -0.49888185033465304, '6.2%'),
 Text(0.3333422654471765, -0.49888168343460365, '31.2%')])
```



[52]:
```
# passing specific colors
regions=['North','South','East','West']
revenu=[3000,1000,400,2000]
plt.pie(revenu,labels=regions,autopct='%1.
  ↪1f%%',colors=['gold','green','red','purple'])  # %1.1f%% shows the percentage
```

[52]:
```
([<matplotlib.patches.Wedge at 0x26c47409a90>,
  <matplotlib.patches.Wedge at 0x26c474096d0>,
  <matplotlib.patches.Wedge at 0x26c4740a210>,
  <matplotlib.patches.Wedge at 0x26c47409e50>],
 [Text(0.10781885028307768, 1.0947031997412062, 'North'),
  Text(-1.0526343030870855, -0.3193133632724537, 'South'),
  Text(-0.6111270287166031, -0.9146167256135307, 'East'),
```

```
 Text(0.611127486653157, -0.9146164196301068, 'West')],
[Text(0.05881028197258782, 0.5971108362224761, '46.9%'),
 Text(-0.5741641653202284, -0.17417092542133836, '15.6%'),
 Text(-0.3333420156636016, -0.49888185033465304, '6.2%'),
 Text(0.3333422654471765, -0.49888168343460365, '31.2%')])
```

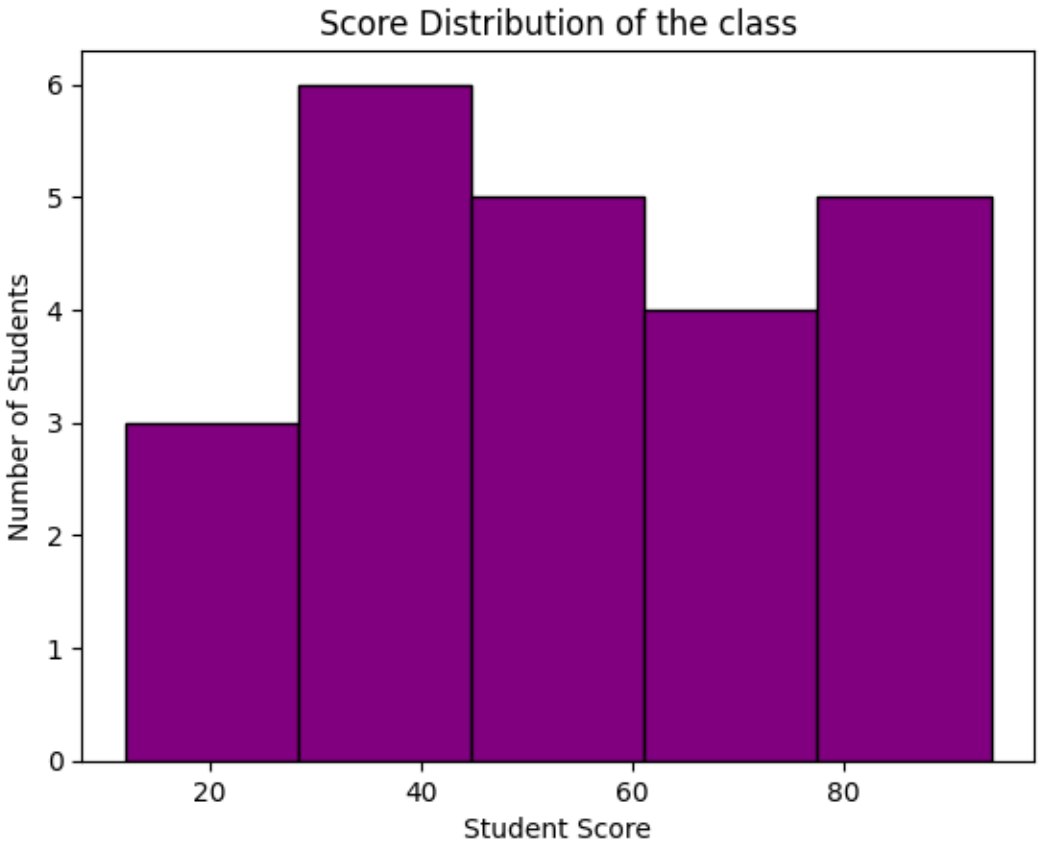


7 Histogram

[59]:
```
''' Histogram - A histogram is a graphical representation of the distribution␣
 ↪of numerical data using bars.
It groups data into intervals (called bins) and shows how many data points fall␣
 ↪into each interval.

Unlike a bar chart (which shows categorical data), a histogram is used for␣
 ↪continuous numerical data.
'''
# parameters- data , edgecolor=(border color) , bins=(number of bins)

scores=[92,94,58,38,83,34,56,76,65,84,34,23,64,12,43,48,59,79,12,34,43,54,67]
plt.hist(scores,bins=5,color='purple',edgecolor='black')
plt.xlabel('Student Score')
plt.ylabel('Number of Students')
```

```
plt.title('Score Distribution of the class')
```

[59]: Text(0.5, 1.0, 'Score Distribution of the class')



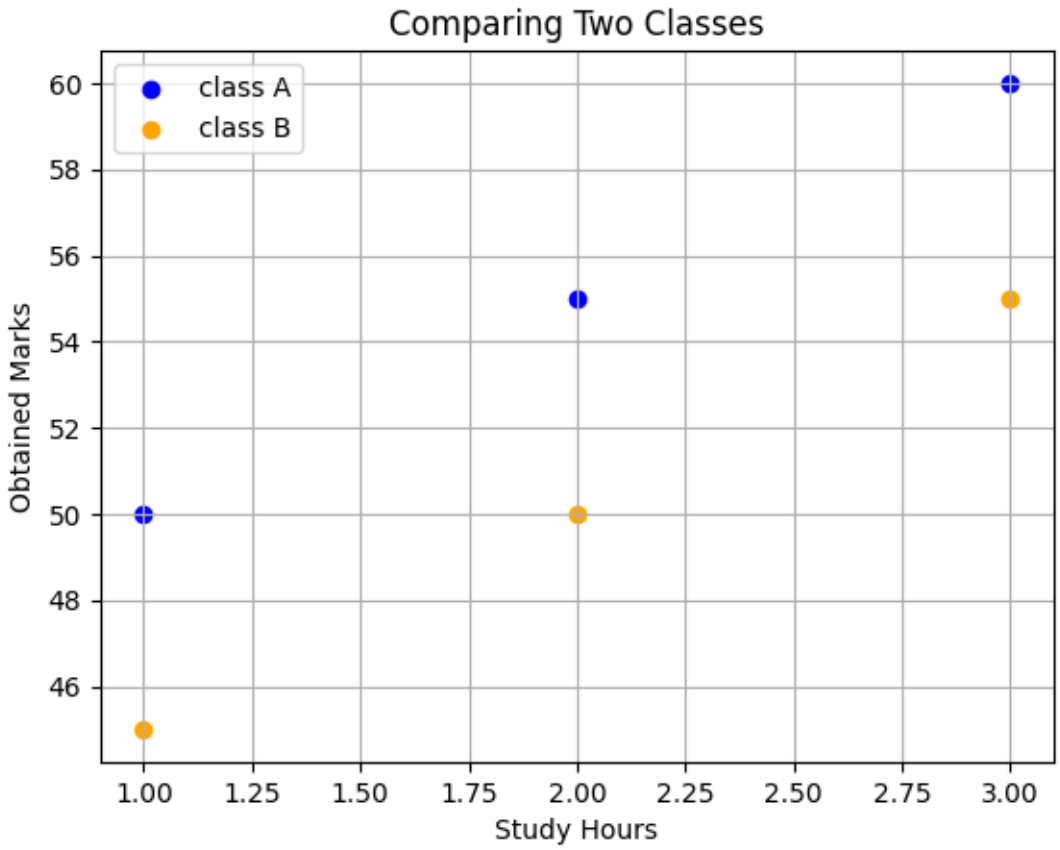
8 Scatter Plots

[ ]:
```
''' A scatter plot is a type of data visualization that shows the relationship␣
 ↪between two numerical variables using dots.
Each point on the plot represents a single observation with values for both␣
 ↪variables '''
```

[5]:
```
# plt.scatter(x,y,color='name',marker='marker style',label='label name')

Study_hours=[1,2,3,4,5,6,7,8,9]
marks_obtained_in_exam=[50,55,60,65,70,75,80,85,95]
plt.
 ↪scatter(Study_hours,marks_obtained_in_exam,color='green',marker='o',label='student␣
 ↪data')
plt.xlabel('Study Hours')
```

```
plt.ylabel('Obtained Marks')
plt.title('Realtionship between Study time & Exam Score')
plt.legend()
plt.grid(True)
plt.show()
```

[9]:
```
# comparing two datasets so we compare multiple sets

plt.scatter([1,2,3],[50,55,60],color='blue',label='class A') # Group 1
plt.scatter([1,2,3],[45,50,55],color='orange',label='class B') # Group 2

plt.title('Comparing Two Classes')
plt.xlabel('Study Hours')
plt.ylabel('Obtained Marks')
plt.legend()
plt.grid(True)
plt.show()
```

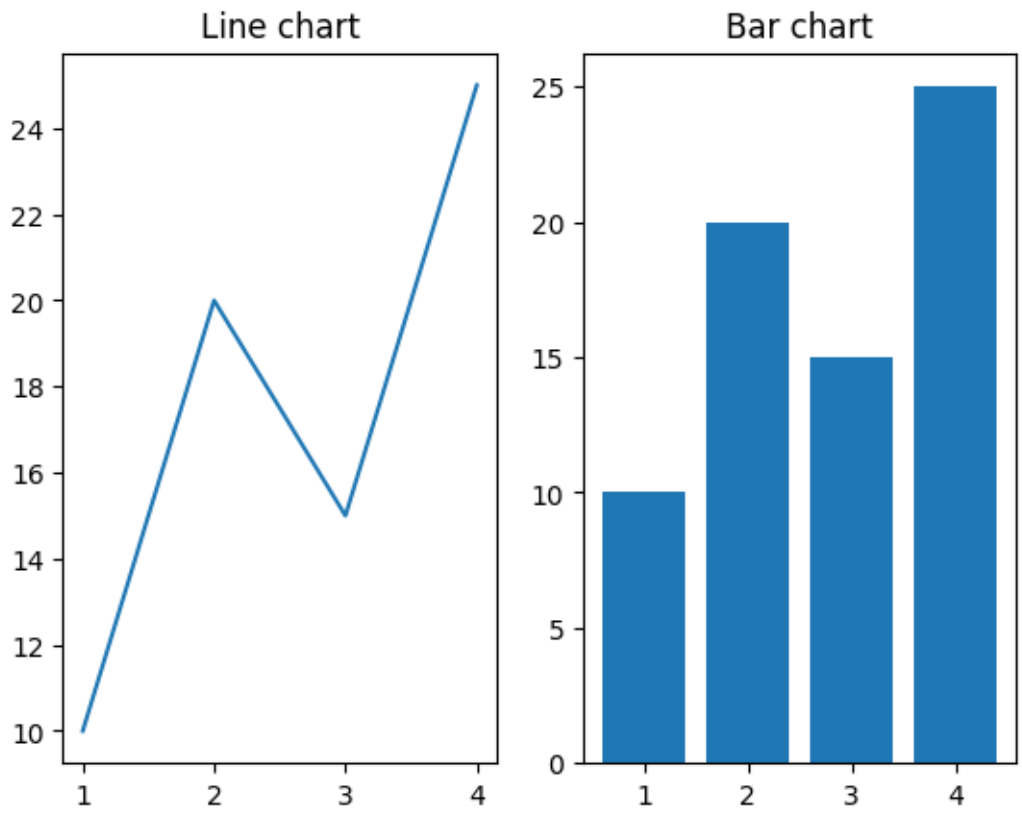


9 Subplots Functions and Layout Adjustment

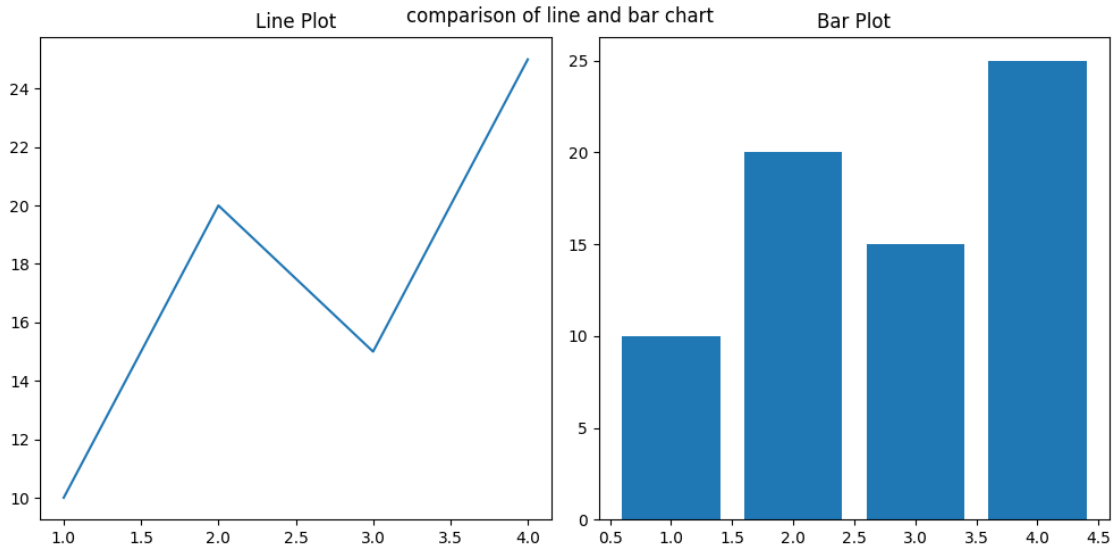
[ ]:
```
''' Subplots in Matplotlib allow you to display multiple plots (charts, graphs)␣
 ↪in a single figure,
arranged in rows and columns - like a grid of plots.
used to draw multiple plots in one canva'''

# plt.subplot(nrows,ncols,index)  nrows-number of rows ,ncols-number of columns␣
 ↪, index-Not Zero(0) based index its a one(1) based index
```

[10]:
```
x=[1,2,3,4]
y=[10,20,15,25]
plt.subplot(1,2,1) # 1-row , 2-column , 1-subplot
plt.plot(x,y)
plt.title('Line chart')

plt.subplot(1,2,2) # 1-row , 2-column , 2-subplot
plt.bar(x,y)
plt.title('Bar chart')
```

[10]: Text(0.5, 1.0, 'Bar chart')



[27]:
```
# professional way-
#  fig ax=plt.subplots(nrows,ncols,figsize=width,height)
fig,ax=plt.subplots(1,2,figsize=(10,5))

x=[1,2,3,4]
y=[10,20,15,25]

ax[0].plot(x,y)
ax[0].set_title('Line Plot')


ax[1].bar(x,y)
ax[1].set_title('Bar Plot')
plt.tight_layout()  # used to fit the graphs
fig.suptitle('comparison of line and bar chart')   # used for main title
plt.show()
```

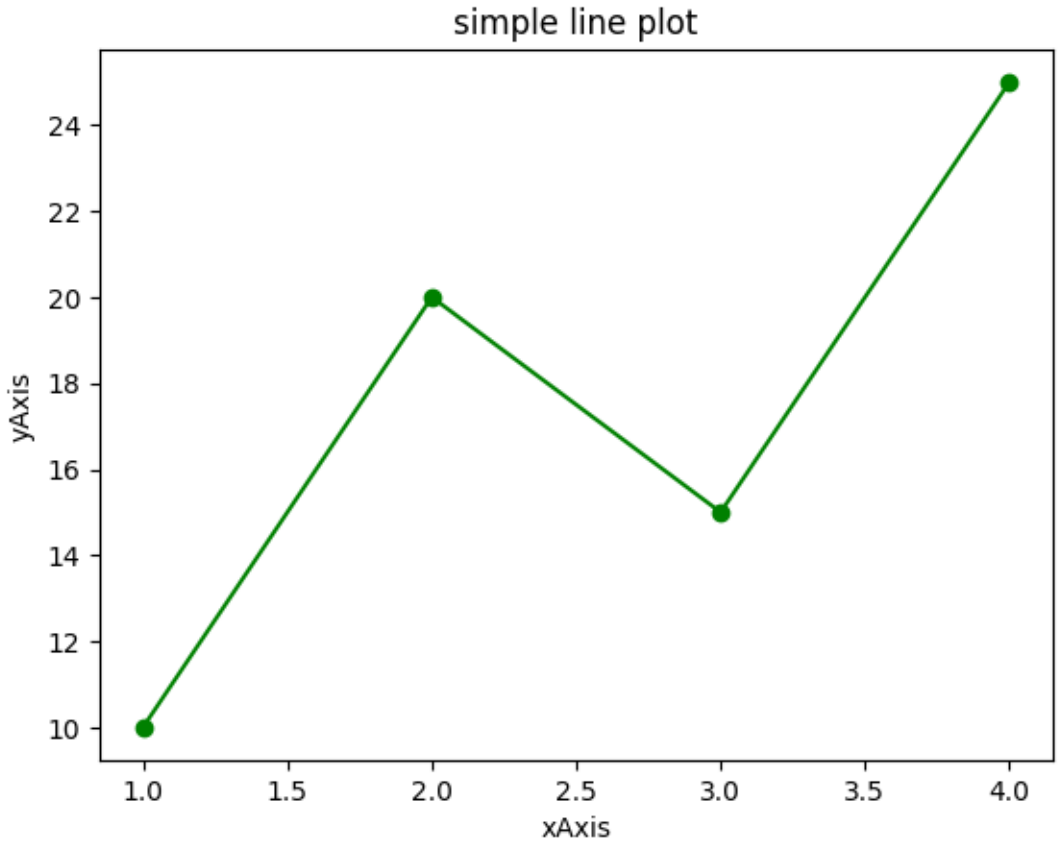


10 Saving the figure

[ ]:
```
''' why we need to save figure -

1] Permanent Storage
So your plot doesn't disappear when you close the window or shut down your␣
 ↪notebook.

2] Sharing
You can send the chart as an image in reports, presentations, emails, or papers.

3] Reusability
Saved plots can be reused in documents, blogs, or printed materials.

4] Automation
In data pipelines or automated reporting, you can generate and save plots␣
 ↪without manual work.
'''
# savefig('filename.extension',dpi=value,bbox_inches='tight')   bbox_inches -␣
 ↪crop the white space , doi - dot per inches
```

[28]:
```
x=[1,2,3,4]
y=[10,20,15,25]

# create plot
plt.plot(x,y,color='green',marker='o')
plt.title('simple line plot')
```

```
plt.xlabel('xAxis')
plt.ylabel('yAxis')
plt.savefig('Line_plot.png',dpi=300,bbox_inches='tight') # always first save␣
 ↪then show the file
plt.show()
```



11 Working on Netflix Data

[ ]:
```
''' Ratings- Content Rating

    1] PG - parential Guidence content
    2] R - Restricted Adult content
    3] TV-MA - Mature Audience only
    4] TV-14 - Parent Strongly questioned

'''
```

[32]:
```
# importing libraries
import pandas as pd
import matplotlib.pyplot as plt
```

[37]:
```
# load dataset
df=pd.read_csv(r'D:\Data Science\Datatsets\netflix1.csv')

# checking null values
df.isnull().sum()
```

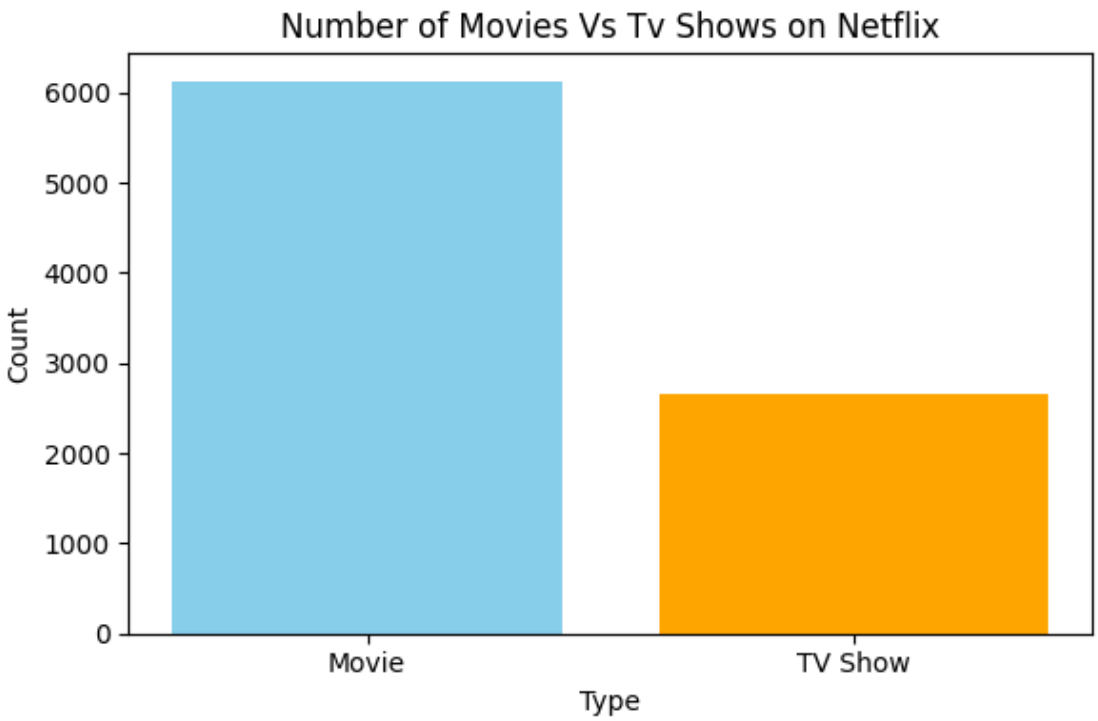
[37]:
```
show_id         0
type            0
title           0
director        0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
dtype: int64
```

[38]:
```
# checking all columns
df.columns
```

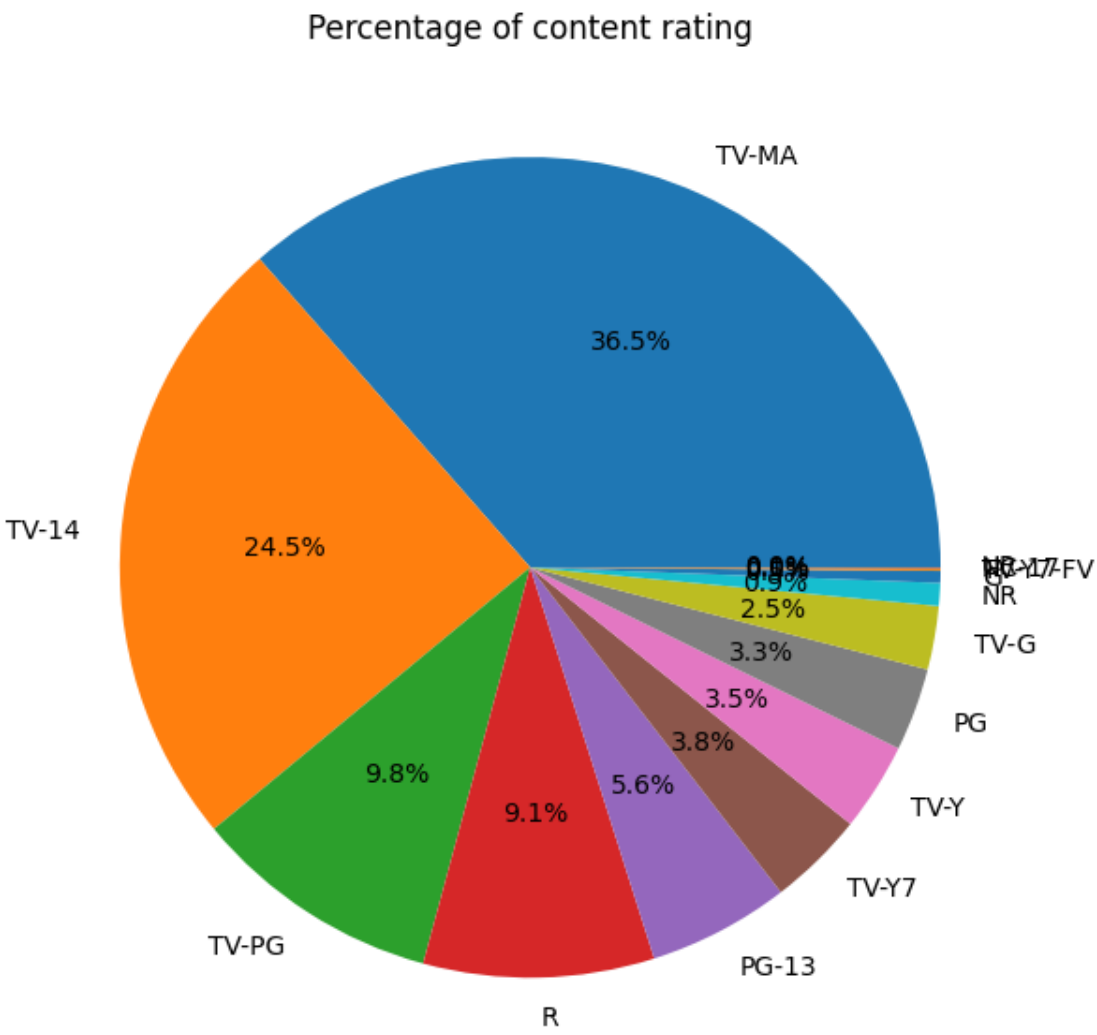
[38]:
```
Index(['show_id', 'type', 'title', 'director', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in'],
      dtype='object')
```

[39]:
```
# if Missing values present then
df=df.dropna(subset=['show_id', 'type', 'title', 'director', 'country',␣
 ↪'date_added',
       'release_year', 'rating', 'duration', 'listed_in'])
```

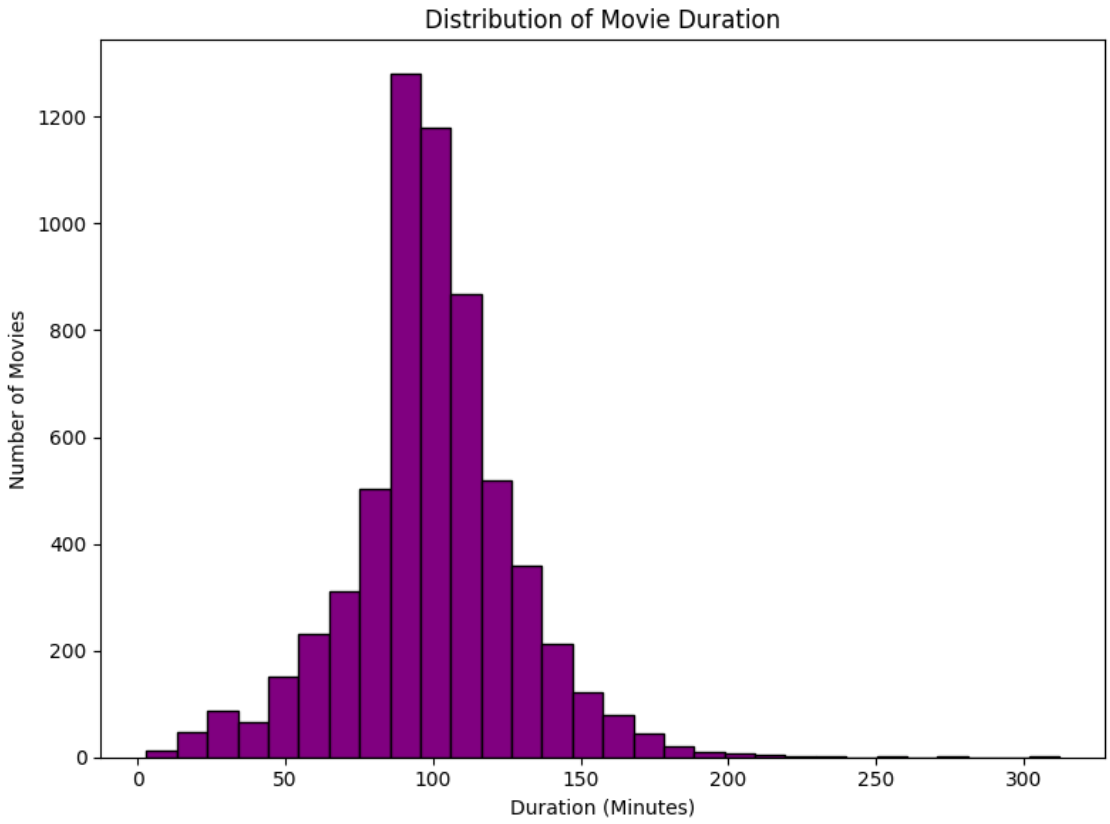
[43]:
```
# Bar chart

type_counts=df['type'].value_counts()
plt.figure(figsize=(6,4))
plt.bar(type_counts.index,type_counts.values,color=['skyblue','orange'])
plt.title('Number of Movies Vs Tv Shows on Netflix')
plt.xlabel('Type')
plt.ylabel('Count')
plt.tight_layout()
plt.savefig('Movies Vs Tv shows.png')
```



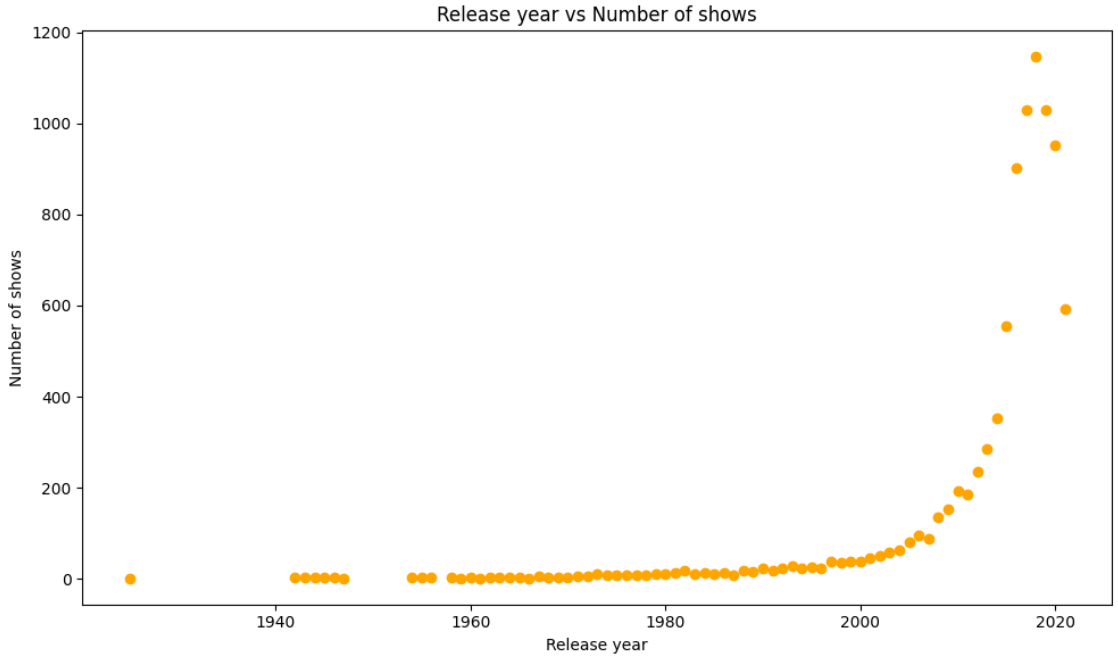
[46]:
```
# Rating counting

rating_counts=df['rating'].value_counts()
plt.figure(figsize=(8,6))
plt.pie(rating_counts,labels=rating_counts.index,autopct='%1.1f%%')
plt.title('Percentage of content rating')

plt.tight_layout()
plt.savefig('Content_rating.png')
plt.show()
```



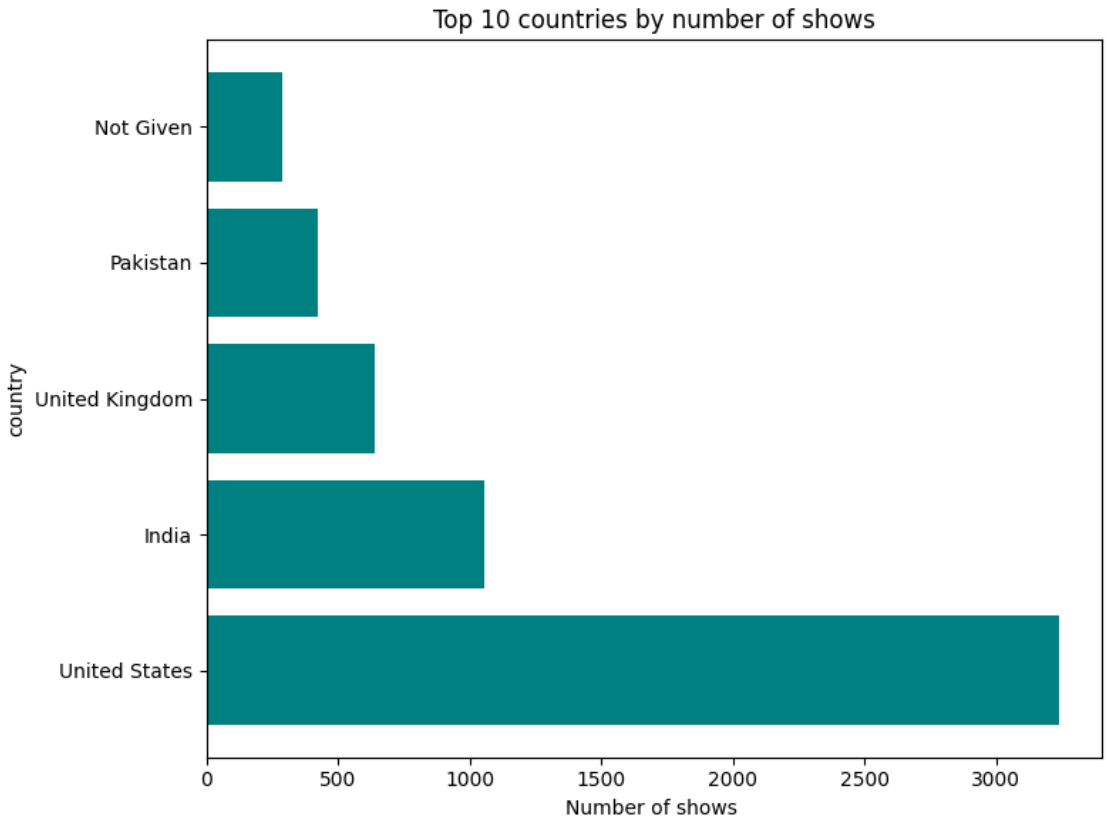
[52]:
```
# Duration of Movies Distribution Display using histogram

movie_df=df[df['type']=='Movie'].copy()
movie_df['duration_int']=movie_df['duration'].str.replace('min','').astype(int)
plt.figure(figsize=(8,6))
plt.hist(movie_df['duration_int'],bins=30,color='purple', edgecolor='black')
plt.title('Distribution of Movie Duration')
plt.xlabel('Duration (Minutes)')
plt.ylabel('Number of Movies')
plt.tight_layout()
plt.savefig('Movies Duration Histogram.png')
plt.show()
```



[56]:
```
# Release Year vs Number of Shows using scatterplot

release_counts=df['release_year'].value_counts().sort_index()
plt.figure(figsize=(10,6))
plt.scatter(release_counts.index,release_counts.values,color='orange')
plt.title('Release year vs Number of shows')
plt.xlabel('Release year')
plt.ylabel('Number of shows')
plt.tight_layout()
plt.savefig('Release year.png')
plt.show()
```

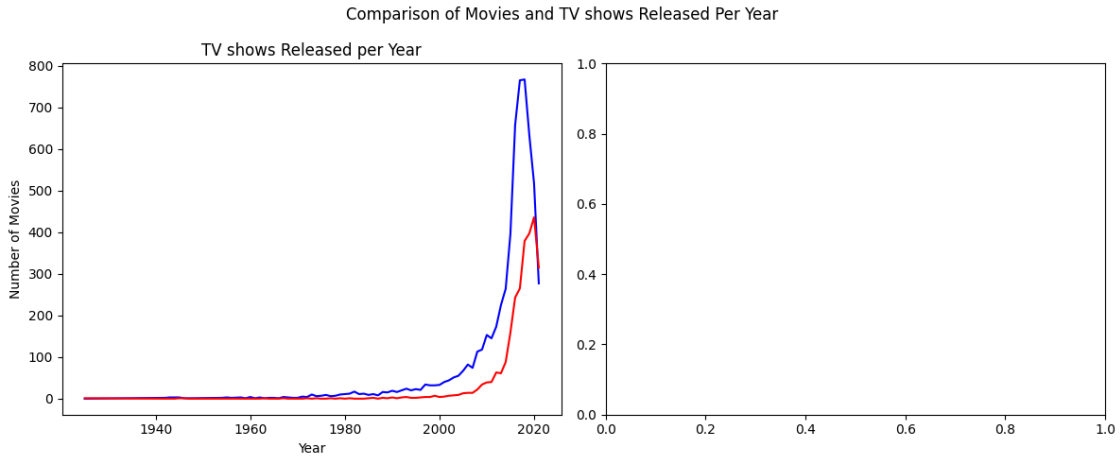


[59]:
```
# Top 10 countries whose publish number of shows using horizontal  barchart

country_counts=df['country'].value_counts().head()
plt.figure(figsize=(8,6))
plt.barh(country_counts.index,country_counts.values,color='teal')
plt.title('Top 10 countries by number of shows')
plt.xlabel('Number of shows ')
plt.ylabel('country')
plt.tight_layout()
plt.savefig('Top 10 country shows numbers.png')
plt.show()
```



[64]:
```
# Moves vs Tv shows By year using subplot

content_by_year=df.groupby(['release_year','type']).size().unstack().fillna(0)
fig,ax=plt.subplots(1,2,figsize=(12,5))

# first subplot-Movies
ax[0].plot(content_by_year.index,content_by_year['Movie'],color='blue')
ax[0].set_title('Movies Released per Year')
ax[0].set_xlabel('Year')
ax[0].set_ylabel('Number of Movies')


# Second subplot-TV shows
ax[0].plot(content_by_year.index,content_by_year['TV Show'],color='red')
ax[0].set_title('TV shows Released per Year')
ax[0].set_xlabel('Year')
ax[0].set_ylabel('Number of Movies')

# Title
fig.suptitle('Comparison of Movies and TV shows Released Per Year')
plt.tight_layout()
```

```
plt.savefig('Comparison of Movies and TV shows Released Per Year.png')
plt.show()
```



[ ]: