import numpy as np - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices

import pandas as pd - it offers data structures and operations for manipulating numerical tables and time series.

import matplotlib as mpl

import matplotlib.pyplot as plt - Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

import seaborn as sns - Seaborn is an amazing data visualization library for statistical graphics plotting in Python

import warnings; warnings.filterwarnings(action='once') - Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program.

%matplotlib inline - it sets the backend of matplotlib to the 'inline' backend: With this backend, the output of plotting commands is displayed inline within frontends like the Jupyter notebook, directly below the code cell that produced it.

df = pd.read_csv("https://raw.githubusercontent.com/tigershiva02/dataset/main/ct3clouddataset.csv") –

in this line we are importing the data set we mentioned before to our prgm by uploading that csv file to github and then using the raw file link from github here

klass

klass is a common convention in OOP code when referring to classes, because "class" and "Class" are often reserved already.

def newline

The character or character sequence that indicates the end of a line of text and transition to the next line;

ax = plt.gca()

The gca() function in pyplot module of matplotlib library is used to get the current Axes instance on the current figure

l = mlines.Line2D([p1[0],p2[0]], [p1[1],p2[1]], color='red' if p1[1]-p2[1] > 0 else 'green', marker='o', markersize=6)

here in this line we are essentially assigning the color , markersize of the lines or slopes that go across the before and after column

fig, ax = plt.subplots(1,1,figsize=(14,14), dpi= 80)

This line basically helps us sacle the end graph and assign the necessary pixels for it

 # Vertical Lines

ax.vlines(x=1, ymin=5, ymax=10, color='black', alpha=0.7, linewidth=1, linestyles='dotted')

ax.vlines(x=3, ymin=5, ymax=10, color='black', alpha=0.7, linewidth=1, linestyles='dotted')

these lines of the code adds the specifics like color, type of line like either solid, dotted or dashed for the verticle lines where our GPA's lie in both the before and after coloumn. We have assigned the color to be black and the type of line to be dotted.

#Points

ax.scatter(y=df['1st'], x=np.repeat(1, df.shape[0]), s=10, color='black', alpha=0.7)

ax.scatter(y=df['4th'], x=np.repeat(3, df.shape[0]), s=10, color='black', alpha=0.7)

these lines of the code add the specifics of the point that corresponds with each value of the data on the line before and after

here as you can see the size is marked as 10 and the alpha parameter is for the transparency/opacity of the points

 # Line Segmentsand Annotation

```
for p1, p2, c in zip(df['1st'], df['4th'], df['Student']):
    newline([1,p1], [3,p2])
    ax.text(1-0.05, p1, c + ', ' + str(round(p1)), horizontalalignment='right',
verticalalignment='center', fontdict={'size':14})
    ax.text(3+0.05, p2, c + ', ' + str(round(p2)), horizontalalignment='left', verticalalignment='center',
fontdict={'size':14})
```

These lines are for the specifics of the annotations of each point in the graphs like for example in our output it says the name and the gpa of the student in both lines and these lines are for specifying what size and alignment they are supposed to be in, here u can see that the allignemnet is set at its desired place and the font size has been made 12

 # 'Before' and 'After' Annotations

```
ax.text(1-0.05, 12, 'BEFORE', horizontalalignment='right', verticalalignment='center',
fontdict={'size':18, 'weight':700})
ax.text(3+0.05, 12, 'AFTER', horizontalalignment='left', verticalalignment='center',
fontdict={'size':18, 'weight':700})
```

these are for the annotations that appear at the top of the graph below the heading , that is the annotations BEFORE and AFTER

these lines make it possible so that they are aligned perfectly with their respective lines andwhat size they are supoosed to be in

 # Decoration

```
ax.set_title("Slopechart: Comparing CGPA of students when they were in year 1 and 4",
fontdict={'size':22})
ax.set(xlim=(0,4), ylim=(0,13), ylabel='CGPA')
ax.set_xticks([1,3])
ax.set_xticklabels(["1st Year", "4th Year"])
plt.yticks(np.arange(5, 11, 1), fontsize=15)
```

These lines are for assigning the headings, x axis text and y axis text of the graph and the specifics so that they are aligned perfectly

 # Lighten borders

plt.gca().spines["top"].set_alpha(.0)

plt.gca().spines["bottom"].set_alpha(.0)

plt.gca().spines["right"].set_alpha(.0)

plt.gca().spines["left"].set_alpha(.0)

Spines are the lines connecting the axis tick marks and noting the boundaries of the data area. They can be placed at arbitrary positions and these are so that the spines are at the positions of all 4

plt.show()

And the final line here is to print the graph for us to see