EXPERIMENT NO : 6

Python programs to implement different types of plots using Numpy & Matplotlob.

NAME : AKASH RAMKRIT YADAV		***ID.NO: VU4F2122016***
BATCH : A	***BRANCH : IT***	***DIV : A***

Aim :- python programs to implement different types of plots using Numpy and Matplotlob.

THEORY:

OUTPUT:

Python 3.11.0a4 (main, Mar 13 2023, 10:57:32) [MSC v.1929 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
#AKASH YADAV ID.NO:VU4F2122016 EXP:6 DATE:6/4/2023

Matplotlib:

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

source code for Matplotlib:

located at this github repository https://github.com/matplotlib/matplotlib

Installation of Matplotlib :

If you have Python and PIP already installed on a system, then installation of Matplotlib is very easy.

Install it using this command:

```
C:\Users\Your Name>pip install matplotlib
```

If this command fails, then use a python distribution that already has Matplotlib installed, like Anaconda, Spyder etc.

Import Matplotlib :

Once Matplotlib is installed, import it in your applications by adding the `import module` statement:

```
import matplotlib
```

Now Matplotlib is imported and ready to use:

Checking Matplotlib Version :

The version string is stored under `__version__` attribute.

Example:

```
import matplotlib

print(matplotlib.__version__)
```

1]Matplotlib Pyplot :

Pyplot

Most of the Matplotlib utilities lies under the `pyplot` submodule, and are usually imported under the `plt` alias:

```
import matplotlib.pyplot as plt
```

Now the Pyplot package can be referred to as `plt`.

Plotting x and y points

The `plot()` function is used to draw points (markers) in a diagram.

By default, the `plot()` function draws a line from point to point.

The function takes parameters for specifying points in the diagram.

Parameter 1 is an array containing the points on the **x-axis**.

Parameter 2 is an array containing the points on the **y-axis**.

Plotting Without Line

To plot only the markers, you can use *shortcut string notation* parameter 'o', which means 'rings'.

Multiple Points

You can plot as many points as you like, just make sure you have the same number of points in both axis.

Default X-Points

If we do not specify the points on the x-axis, they will get the default values 0, 1, 2, 3 (etc., depending on the length of the y-points.

So, if we take the same example as above, and leave out the x-points, the diagram will look like this:

Markers

You can use the keyword argument `marker` to emphasize each point with a specified marker:

Marker Reference

You can choose any of these markers:

Marker	Description	Marker	Description
'o'	Circle	'1'	Tri Down
'*'	Star	'2'	Tri Up
'.'	Point	'3'	Tri Left
','	Pixel	'4'	Tri Right
'x'	X	'|'	Vline
'X'	X (filled)	'_'	Hline
'+'	Plus	'>'	Triangle Right
'P'	Plus (filled)		
's'	Square		
'D'	Diamond		
'd'	Diamond (thin)		
'p'	Pentagon		
'H'	Hexagon		
'h'	Hexagon		
'v'	Triangle Down		
'^'	Triangle Up		
'<'	Triangle Left		

Format Strings `fmt`

You can use also use the *shortcut string notation* parameter to specify the marker.

This parameter is also called `fmt`, and is written with this syntax:

marker|line|color

The marker value can be anything from the Marker Reference above.

The line value can be one of the following:

Line Reference

Line Syntax	Description
'-'	Solid line
':'	Dotted line
'--'	Dashed line
'-.'	Dashed/dotted line

Note: If you leave out the line value in the fmt parameter, no line will be plotted.

Marker Size

You can use the keyword argument `markersize` or the shorter version, `ms` to set the size of the markers:

Marker Color

You can use the keyword argument `markeredgecolor` or the shorter `mec` to set the color of the *edge* of the markers:

You can use the keyword argument `markerfacecolor` or the shorter `mfc` to set the color inside the edge of the markers:

Use *both* the `mec` and `mfc` arguments to color the entire marker:

Matplotlib Line

Linestyle

You can use the keyword argument `linestyle`, or shorter `ls`, to change the style of the plotted line

Use a dashed line:

```
plt.plot(ypoints, linestyle = 'dashed')
```

Shorter Syntax

The line style can be written in a shorter syntax:

`linestyle` can be written as `ls`.

`dotted` can be written as `:`.

`dashed` can be written as `--`.

Line Styles

You can choose any of these styles:

Style	Or
'solid' (default)	'-'
'dotted'	':'
'dashed'	'--'
'dashdot'	'-.'
'None'	'' or ' '

Line Color

You can use the keyword argument `color` or the shorter `c` to set the color of the line:

Line Width

You can use the keyword argument `linewidth` or the shorter `lw` to change the width of the line.

The value is a floating number, in points:

Multiple Lines

You can plot as many lines as you like by simply adding more `plt.plot()` functions:

Matplotlib Labels and Title

Create Labels for a Plot

With Pyplot, you can use the `xlabel()` and `ylabel()` functions to set a label for the x- and y-axis.

Create a Title for a Plot

With Pyplot, you can use the `title()` function to set a title for the plot.

Set Font Properties for Title and Labels

You can use the `fontdict` parameter in `xlabel()`, `ylabel()`, and `title()` to set font properties for the title and labels.

Position the Title

You can use the `loc` parameter in `title()` to position the title.

Legal values are: 'left', 'right', and 'center'. Default value is 'center'.

Matplotlib Adding Grid Lines

Add Grid Lines to a Plot

With Pyplot, you can use the `grid()` function to add grid lines to the plot.

Specify Which Grid Lines to Display

You can use the `axis` parameter in the `grid()` function to specify which grid lines to display.

Legal values are: 'x', 'y', and 'both'. Default value is 'both'

Set Line Properties for the Grid

You can also set the line properties of the grid, like this: grid(color = '*color*', linestyle = '*linestyle*', linewidth = *number*).

Matplotlib Subplot

Display Multiple Plots

With the `subplot()` function you can draw multiple plots in one figure:

CODE :

```
import matplotlib.pyplot as plt

import numpy as np


x = np.array(["CNND\n MS.SEEMA LADHE","AT\n MR.NILESH MAIL","MATHS\n MS.DAIMI MARIYA","COA \n MS.NEETA INGLE","OS\n MR.PRAVIN PATIL"])

y = np.array([75, 79,70,68,65])


plt.bar(x, y)

plt.bar(x, y, color = "#4CAF50",width = 0.3)

plt.ylabel("MARKS  SCORE")

plt.xlabel("\n SUBJECTS  TEACHER")

plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)


plt.show()
```

OUTPUT:

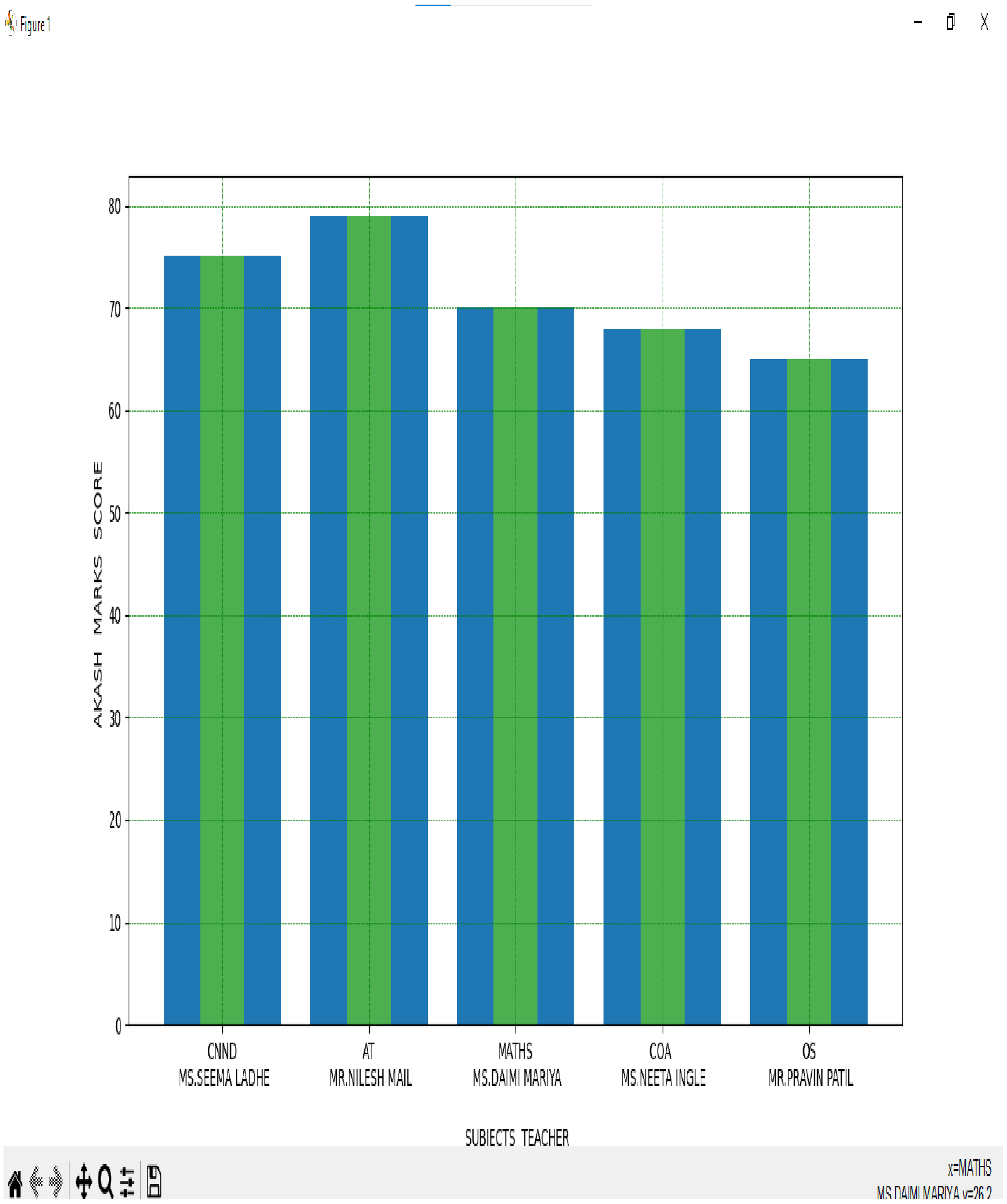
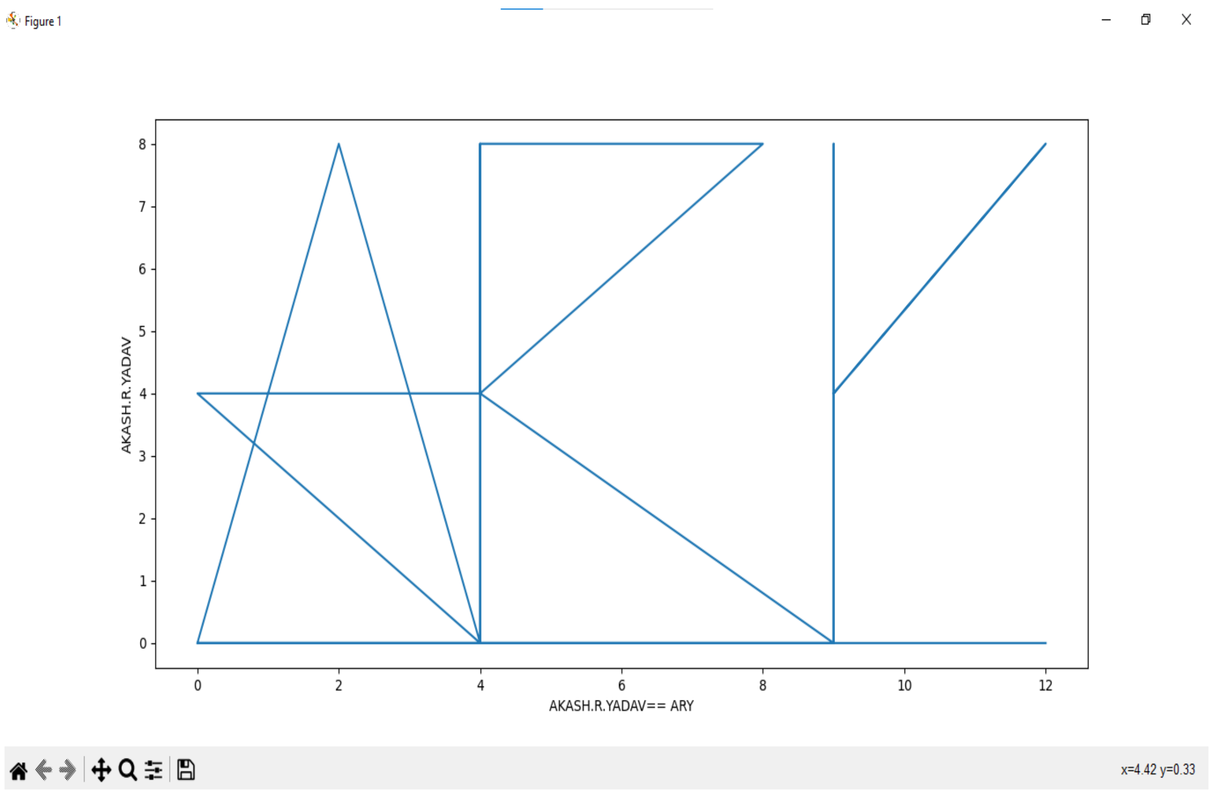


Figure 1
80
70
60
50
40
30
20
10
0
AKASH MARKS SCORE
CNND
MS.SEEMA LADHE
AT
MR.NILESH MAIL
MATHS
MS.DAIMI MARIYA
COA
MS.NEETA INGLE
OS
MR.PRAVIN PATIL
SUBJECTS TEACHER
x=MATHS
MS.DAIMI MARIYA y=26.2

CODE:

```
import matplotlib.pyplot as plt

import numpy as np


xpoints = np.array([0, 2, 4, 0,4,4,4,4,8,4,4,9,9,9,12,9,9,0,12])

ypoints = np.array([0, 8, 0, 4,4,0,8,4,8,8,4,0,8,4,8,4,0,0,0])


plt.plot(xpoints, ypoints)

plt.ylabel("AKASH.R.YADAV")

plt.xlabel("AKASH.R.YADAV== ARY")

plt.show()
```

OUTPUT:



2]Creating Pie Charts :

With Pyplot, you can use the `pie()` function to draw pie charts:

By default the plotting of the first wedge starts from the x-axis and moves *counterclockwise*:

Note: The size of each wedge is determined by comparing the value with all the other values, by using this formula:

The value divided by the sum of all values: `x/sum(x)`

Labels :

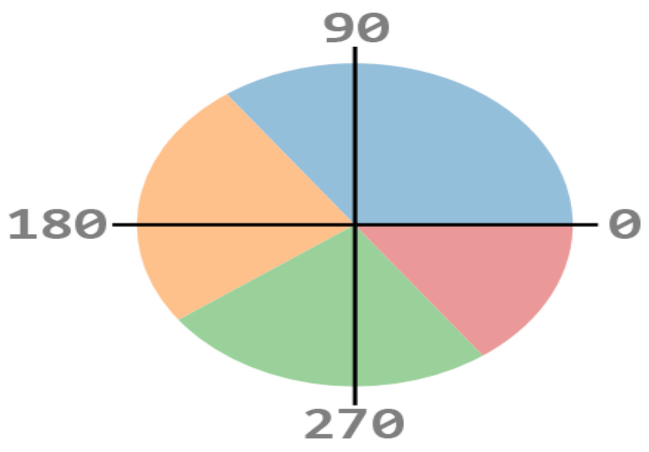
Add labels to the pie chart with the `label` parameter.

The `label` parameter must be an array with one label for each wedge:

Start Angle :

As mentioned the default start angle is at the x-axis, but you can change the start angle by specifying a `startangle` parameter.

The `startangle` parameter is defined with an angle in degrees, default angle is 0:



Explode :

Maybe you want one of the wedges to stand out? The `explode` parameter allows you to do that.

The `explode` parameter, if specified, and not `None`, must be an array with one value for each wedge.

Each value represents how far from the center each wedge is displayed:

Shadow :

Add a shadow to the pie chart by setting the `shadows` parameter to `True`:

Colors :

You can set the color of each wedge with the `colors` parameter.

The `colors` parameter, if specified, must be an array with one value for each wedge:

You can use Hexadecimal color values, any of the 140 supported color names, or one of these shortcuts:

`'r'` - Red
`'g'` - Green
`'b'` - Blue
`'c'` - Cyan
`'m'` - Magenta
`'y'` - Yellow
`'k'` - Black
`'w'` - White

Legend :

To add a list of explanation for each wedge, use the `legend()` function:

Legend With Header

To add a header to the legend, add the `title` parameter to the `legend` function.

Code:

```
import matplotlib.pyplot as ary

import numpy as a1


y=a1.array([75, 79,70,68,65])


mylabels = ["CNND\n MS.SEEMA LADHE","AT\n MR.NILESH
MAIL","MATHS\n MS.DAIMI MARIYA","COA \n MS.NEETA
INGLE","OS\n MR.PRAVIN PATIL"]

mycolors = [ "hotpink","r", "b", "#4CAF50","orange"]

myexplode = [0, 0.4, 0, 0,0]




ary.pie(y, labels = mylabels,autopct='%1.2f%%',colors =
mycolors, explode = myexplode, shadow = True,startangle = -
90)


ary.legend(title = "  % of AKASH YADAV \n   in TOTAL five
subjects:")


#ary.pie(y, labels = mylabels,autopct='%1.2f%%')

ary.show()
```

OUTPUT:

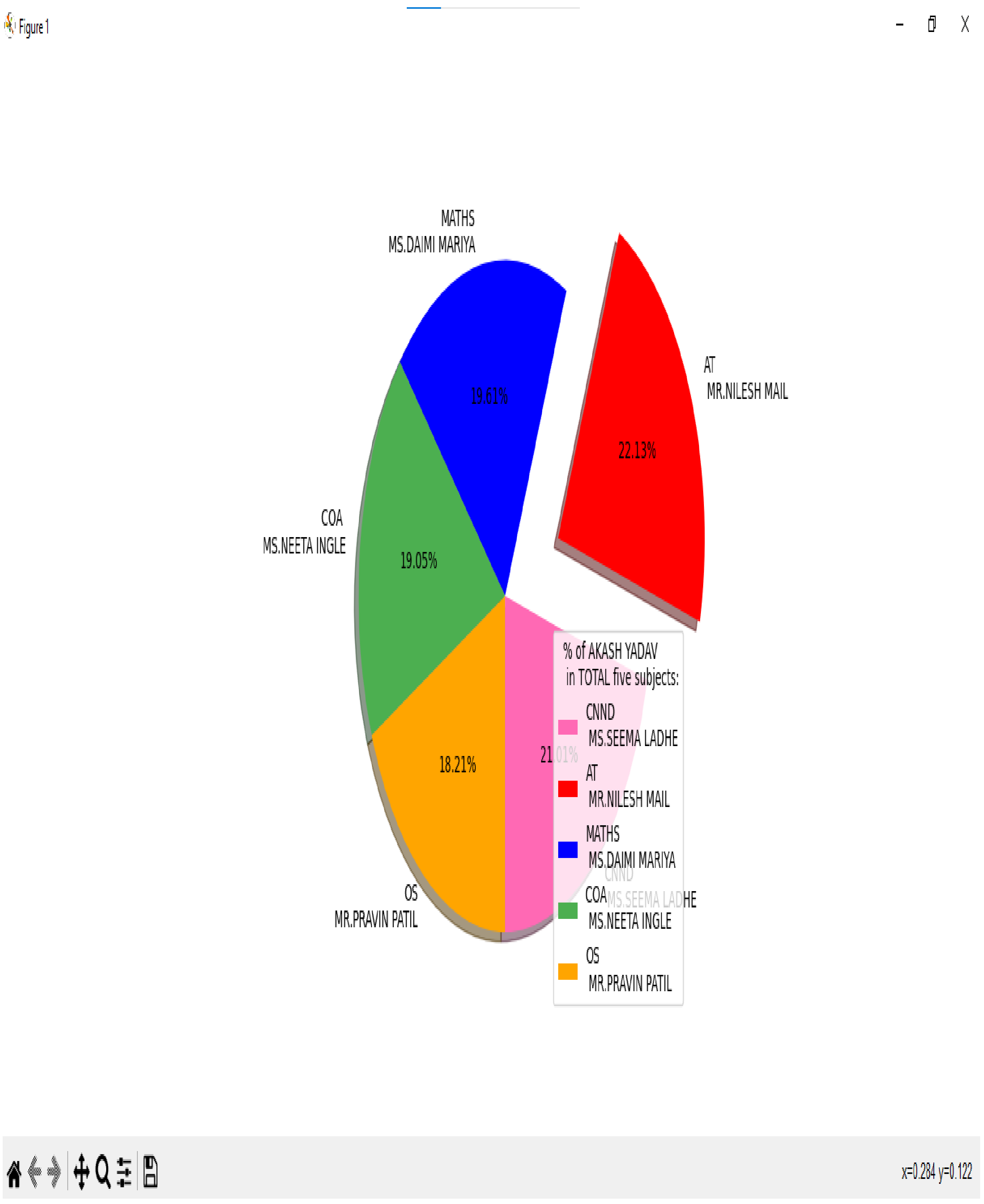


Figure 1
MATHS
MS.DAIMI MARIYA
19.61%
AT
MR.NILESH MAIL
22.13%
COA
MS.NEETA INGLE
19.05%
18.21%
21.01%
OS
MR.PRAVIN PATIL
CNND
MS.SEEMA LADHE
% of AKASH YADAV
in TOTAL five subjects:
CNND
MS.SEEMA LADHE
AT
MR.NILESH MAIL
MATHS
MS.DAIMI MARIYA
COA
MS.NEETA INGLE
OS
MR.PRAVIN PATIL
x=0.284 y=0.122