Matplotlib is a Python module for plotting First import matplotlib and numpy, these are useful for charting import pandas library if you required

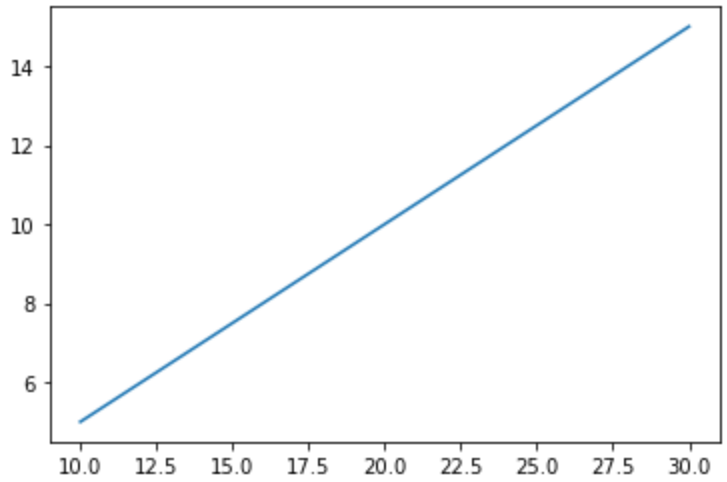
In [21]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

1. Line Chart You can use the plot(x,y) method to create a line chart

In [22]:
```
#Line chart example
x=[10,20,30]
y=[5,10,15]

plt.plot(x,y)
```

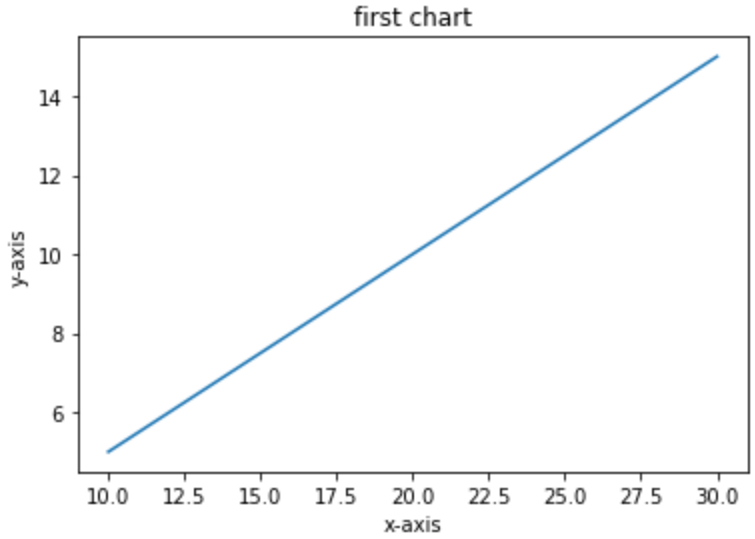
Out[22]:
```
[<matplotlib.lines.Line2D at 0x1caf9c07880>]
```



to have x-axis and y-axis name along with title use following

In [23]:
```
a=plt.plot(x,y)
plt.xlabel('x-axis') # x-axis label
plt.ylabel('y-axis') # y-axis label
plt.title("first chart")# it will give title to chart
```

Out[23]:
```
Text(0.5, 1.0, 'first chart')
```



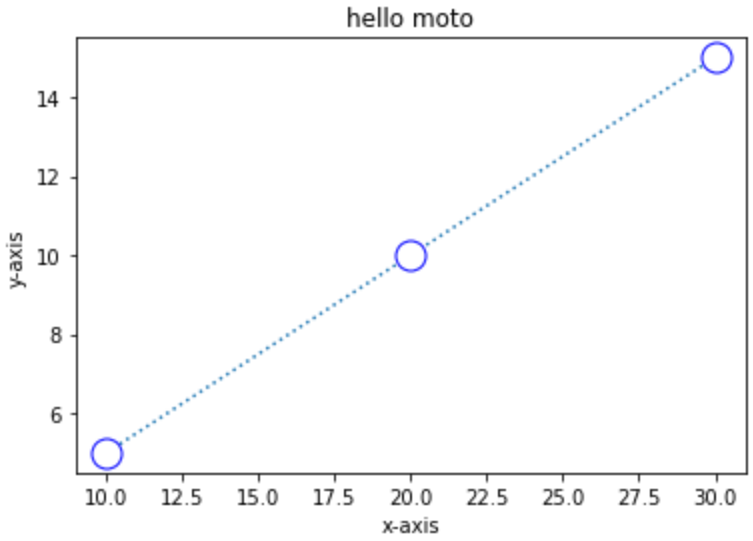
for example if you want different markers and different lines we have following parameters in plot() function

1. line style in shortcut you can write as ls 'solid' (default) '-' 'dotted' ':' 'dashed ' '--'
   'dashdot' '-.'
   'None' '' or ' ' example ls='dotted'
2. markers : You can use the keyword argument marker to emphasize each point with a specified marker Marker Description 'o' Circle
   '*' Star
   '.' Point
   ',' Pixel
   'x' X
   'X' X (filled)
   '+' Plus
   'P' Plus (filled)
   's' Square
   'D' Diamond 'd' Diamond (thin)
   'p' Pentagon
   'H' Hexagon 'h' Hexagon 'v' Triangle Down
   '^' Triangle Up '<' Triangle Left
   '>' Triangle Right
   '1' Tri Down
   '2' Tri Up
   '3' Tri Left
   '4' Tri Right
   '|' Vline
   '_' Hline

To increase the marker size use ms=size(int) example ms='15' To give color to marker use mfc To give color to marker outside use mec

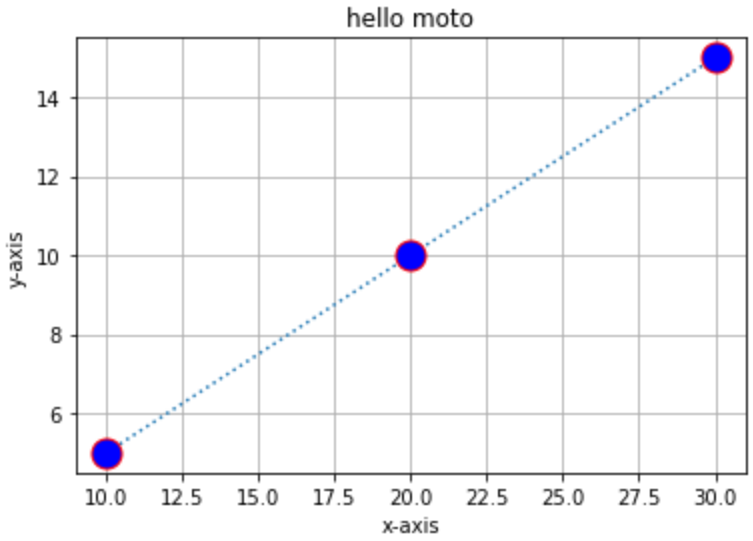
you can observe the following line how to use all these

In [24]:
```
plt.plot(x,y,marker="o",mec='blue',mfc='white',ms="15",ls='dotted')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('hello moto')
plt.show()
```



to show grid over your chart use grid function. in the following line we explained about grid function

In [25]:
```
plt.plot(x,y,marker="o",mec='red',mfc='blue',ms="15",ls='dotted')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('hello moto')
plt.grid(axis='x')
plt.grid(axis='y')
plt.show()
```



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

The subplot() Function

The subplot() function takes three arguments that describes the layout of the figure.

The layout is organized in rows and columns, which are represented by the first and second argument.

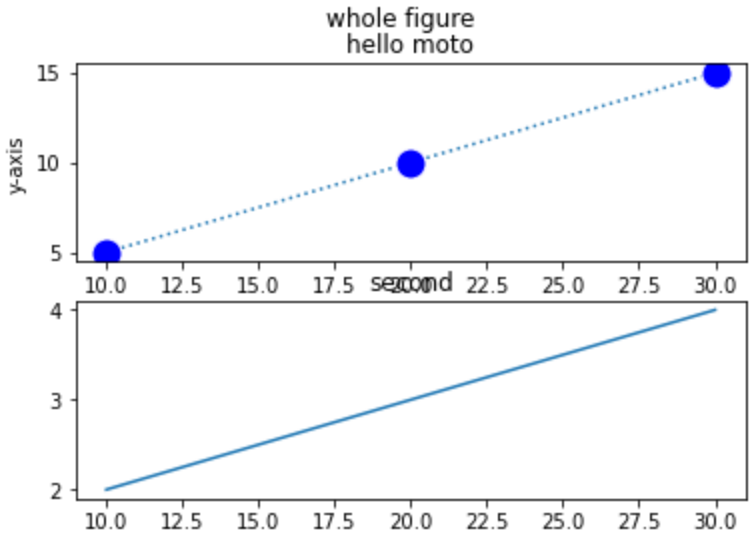
The third argument represents the index of the current plot.

subplot(r,c,i)

use suptitle() to give common name to all figures use title() to give name to individual subplot figure refer the below code for subplot() example

In [26]:
```
y1=[2,3,4]
```

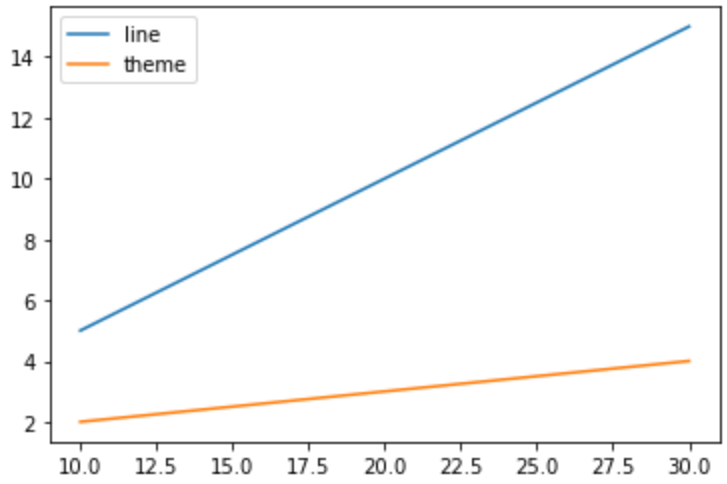
In [27]:
```
plt.suptitle('whole figure')
plt.subplot(2,1,1)
plt.plot(x,y,marker="o",mec='white',mfc='blue',ms="15",ls='dotted')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('hello moto')
plt.subplot(2,1,2)
plt.plot(x,y1)
plt.title('second')
plt.show()
```



legend() To describe about functionality of line in graph we use legend i.e what the line is explaining use legend() function to show

In [28]:
```
plt.plot(x,y,label='line')
plt.plot(x,y1,label='theme')
plt.legend()
```

Out[28]:
```
<matplotlib.legend.Legend at 0x1cafb026a00>
```



Scatter plot():

```
With Pyplot, you can use the scatter() function to draw a scatter plot.
```

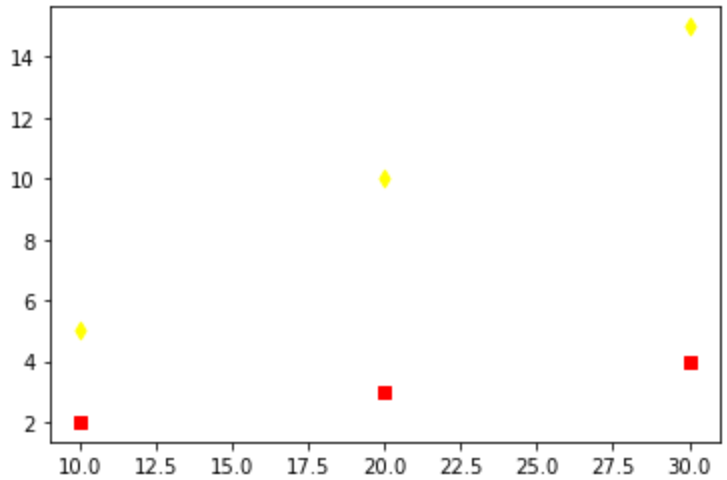
--The scatter() function plots one dot for each observation. --It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis Colors You can set your own color for each scatter plot with the color or the c argument Color Each Dot You can even set a specific color for each dot by using an array of colors as value for the c argument:

Note: You cannot use the color argument for this, only the c argument.

refer below code for scatterplot

In [29]:
```
#scatterplot
plt.scatter(x,y,color='yellow',marker='d')
plt.scatter(x,y1,color='red',marker='s')
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x1cafb0b0280>
```



Barplot():

Creating Bars

With Pyplot, you can use the bar() function to draw bar graphs The bar() function takes arguments that describes the layout of the bars. The categories and their values represented by the first and second argument as arrays.

Horizontal Bars

If you want the bars to be displayed horizontally instead of vertically, use the barh() function

Bar Color The bar() and barh() take the keyword argument color to set the color of the bars Bar Width The bar() takes the keyword argument width to set the width of the bars The default width value is 0.8

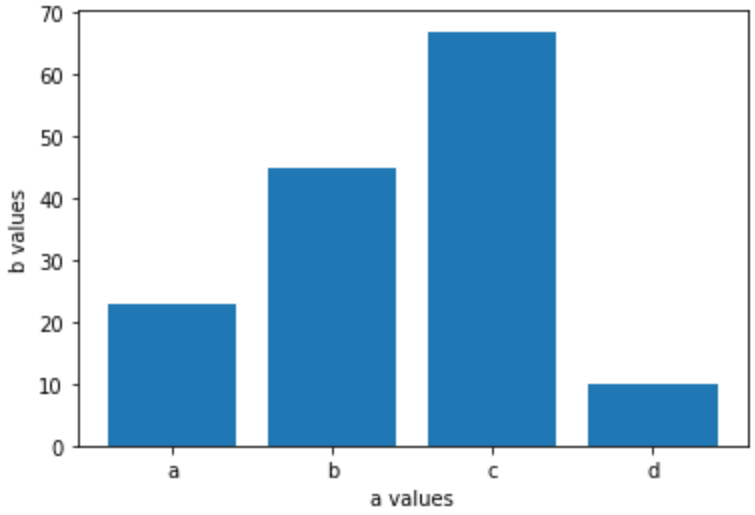
Note: For horizontal bars, use height instead of width.

Bar Height The barh() takes the keyword argument height to set the height of the bars The default height value is 0.8

Refer below codes for bar plot

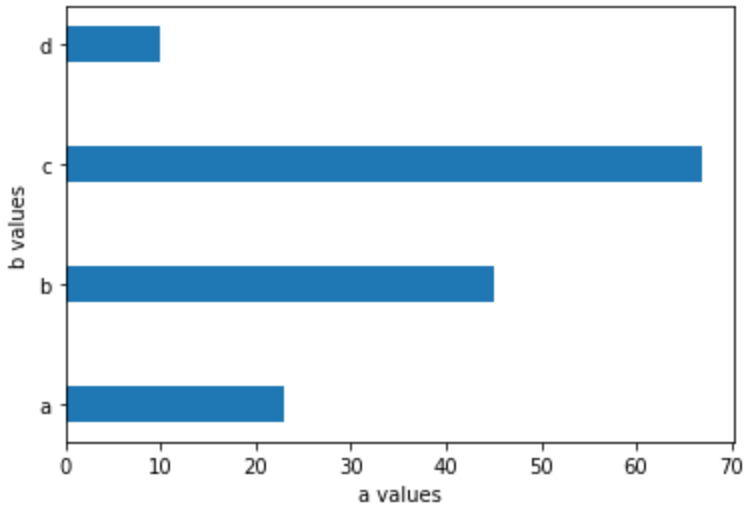
In [30]:
```
#bar chart
a=['a','b','c','d']
b=[23,45,67,10]
plt.bar(a,b)
plt.xlabel('a values')
plt.ylabel('b values')
```

Out[30]:
```
Text(0, 0.5, 'b values')
```



In [31]:
```
#horizontal bar chart
a=['a','b','c','d']
b=[23,45,67,10]
plt.barh(a,b,height=0.3)
plt.xlabel('a values')
plt.ylabel('b values')
```

Out[31]:
```
Text(0, 0.5, 'b values')
```



stacked bar charts:

A stacked bar chart is also known as a stacked bar graph. It is a graph that is used to compare parts of a whole. In a stacked bar chart each bar represents the whole, and the segments or parts in the bar represent categories of that whole. Different colors are used to represent these categories.

To plot the stacked bar graph in the bar function the bottom parameter is very important. As we have to drawbars one above the other, so the bottom of the next bar is equal to the value of the previous bar.

The following steps are used to plot the stacked bar chart in matplotlib is outlined below:

Defining Libraries: Import the important libraries which are required (For data creation and manipulation: Numpy and Pandas, For data visualization: pyplot from matplotlib).

Define X and Y: Define the data coordinated values used for the x-axis and y-axis or we can say that x-axis and height of the bar.

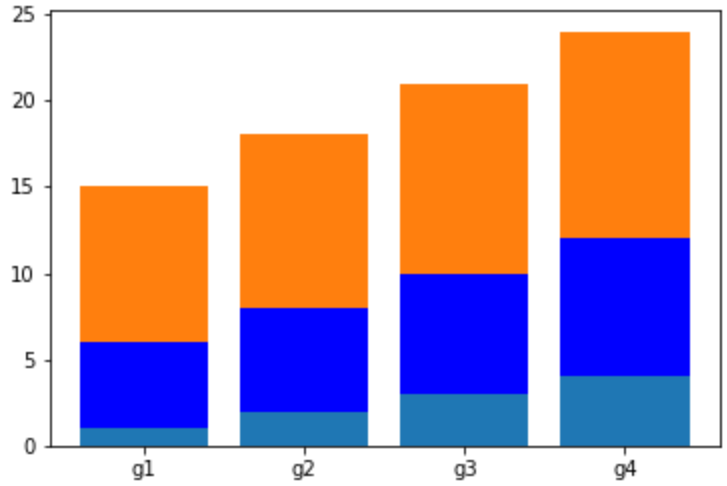
Plot bar chart: By using bar() method we can bar chart.

Set bottom: Set bottom of the next bar equalls to the values of the pervious bars.

Generate a Plot: Use the show() method to visulaize the plot on the user's windows.

here you need to find a way how to add values to bottom for plot

In [32]:
```
groups=['g1','g2','g3','g4']
values=[1,2,3,4]
values1=[5,6,7,8]
values2=[9,10,11,12]
values3=[13,14,15,16]
plt.bar(groups,values)
plt.bar(groups,values1,bottom=values,color='blue')
plt.bar(groups,values2,bottom=np.add(values,values1))
#plt.bar(groups,values3,bottom=np.add(values,values1,values2))
plt.show()
```



pie charts()

Creating Pie Charts With Pyplot, you can use the pie() function to draw pie charts

As you can see the pie chart draws one piece (called a wedge)

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

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

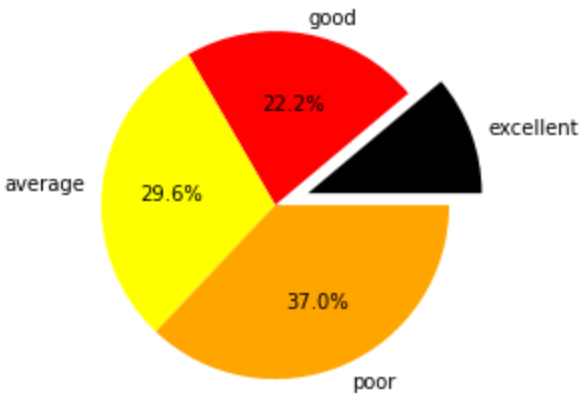
autopct

use autopct parameter to show perentage of each widge in total value syntax autopct='%value%%' example autopct='%2.1f%%'

you can observe pie chart and its related parameters in following cell

In [33]:
```
#pie chart
student_performance=['excellent','good','average','poor']
student_values=[30,60,80,100]
plt.pie(student_values,labels=student_performance,explode=[0.2,0,0,0],colors=["black","
```

Out[33]:
```
([<matplotlib.patches.Wedge at 0x1cafafabd60>,
  <matplotlib.patches.Wedge at 0x1cafafc6e50>,
  <matplotlib.patches.Wedge at 0x1cafafc6850>,
  <matplotlib.patches.Wedge at 0x1cafaff5fd0>],
 [Text(1.2216004058653225, 0.44462618950043836, 'excellent'),
  Text(0.19101298416420226, 1.083288530300532, 'good'),
  Text(-1.092562196394516, 0.12770218091164742, 'average'),
  Text(0.4356877338869101, -1.010037721345341, 'poor')],
 [Text(0.7517540959171214, 0.2736161166156544, '11.1%'),
  Text(0.10418890045320121, 0.5908846528911992, '22.2%'),
  Text(-0.5959430162151905, 0.06965573504271677, '29.6%'),
  Text(0.2376478548474055, -0.5509296661883678, '37.0%')])
```



In [ ]:

In [ ]: