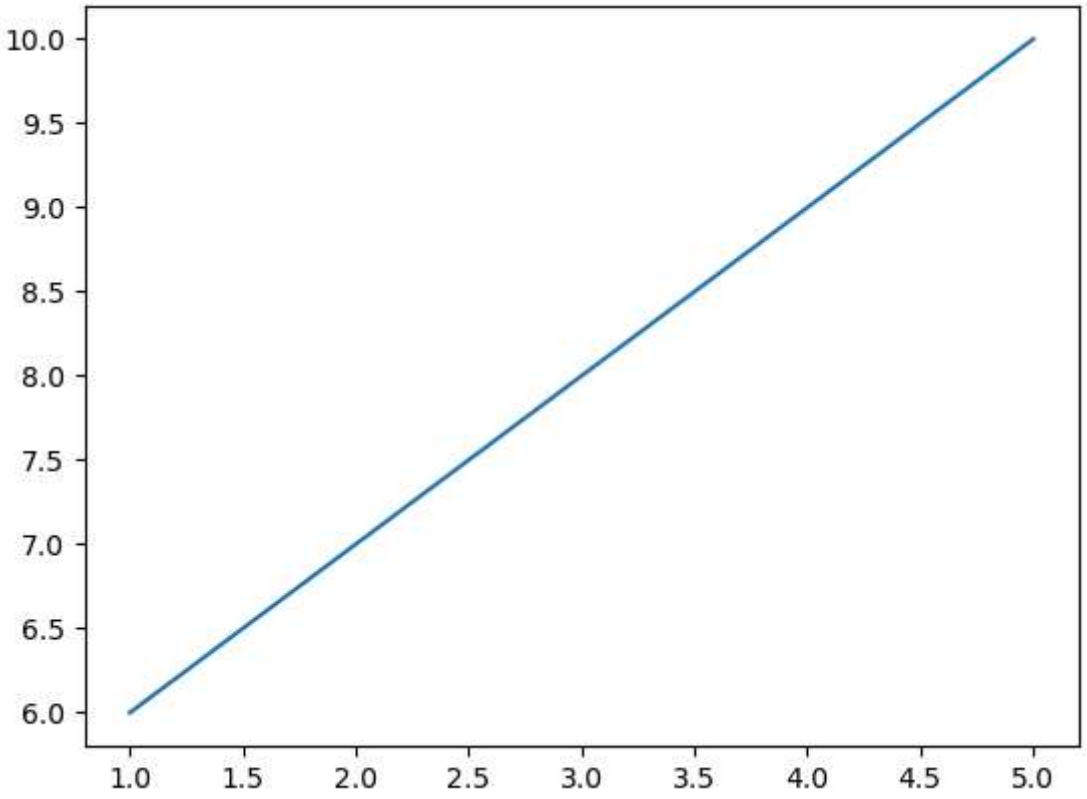
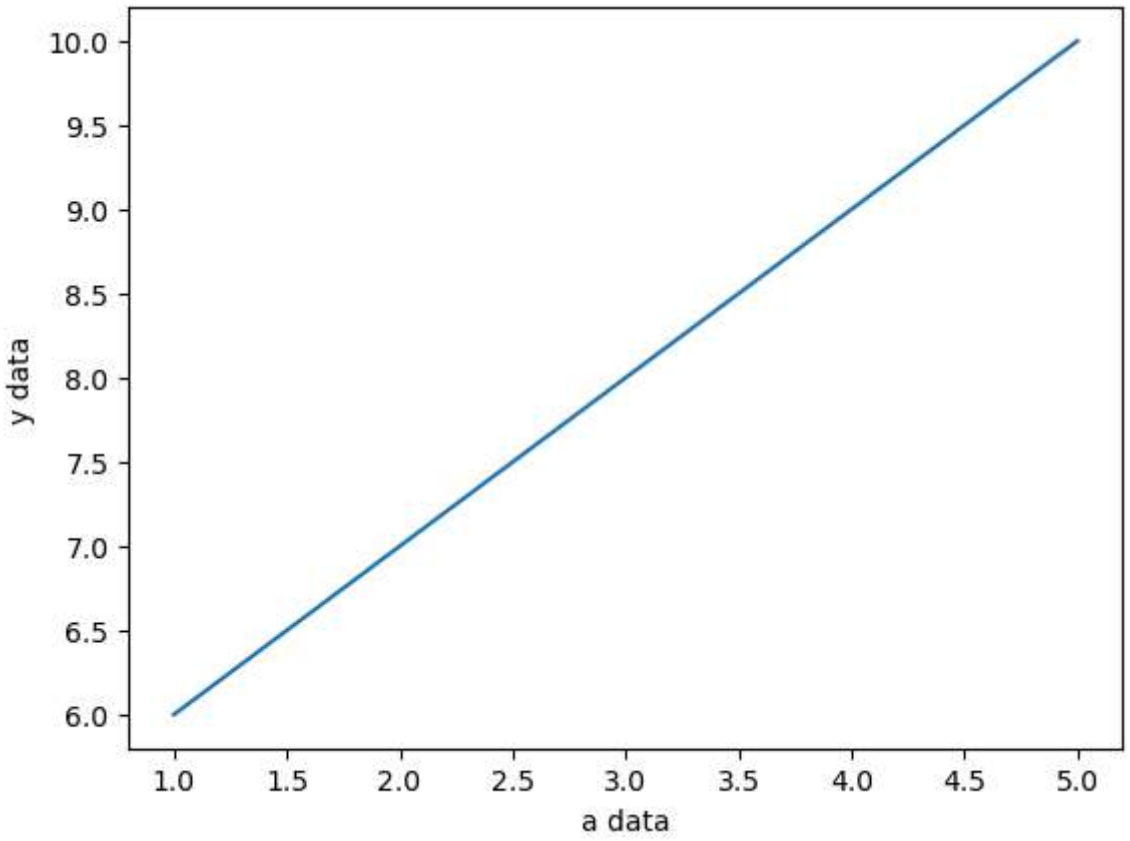
In [1]:
```
# VIVEK-CHAUHAN-ADVANCED-DATA-ANALYTICS-PLOTTING-WITH-DIFFERENT-CHARTS
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

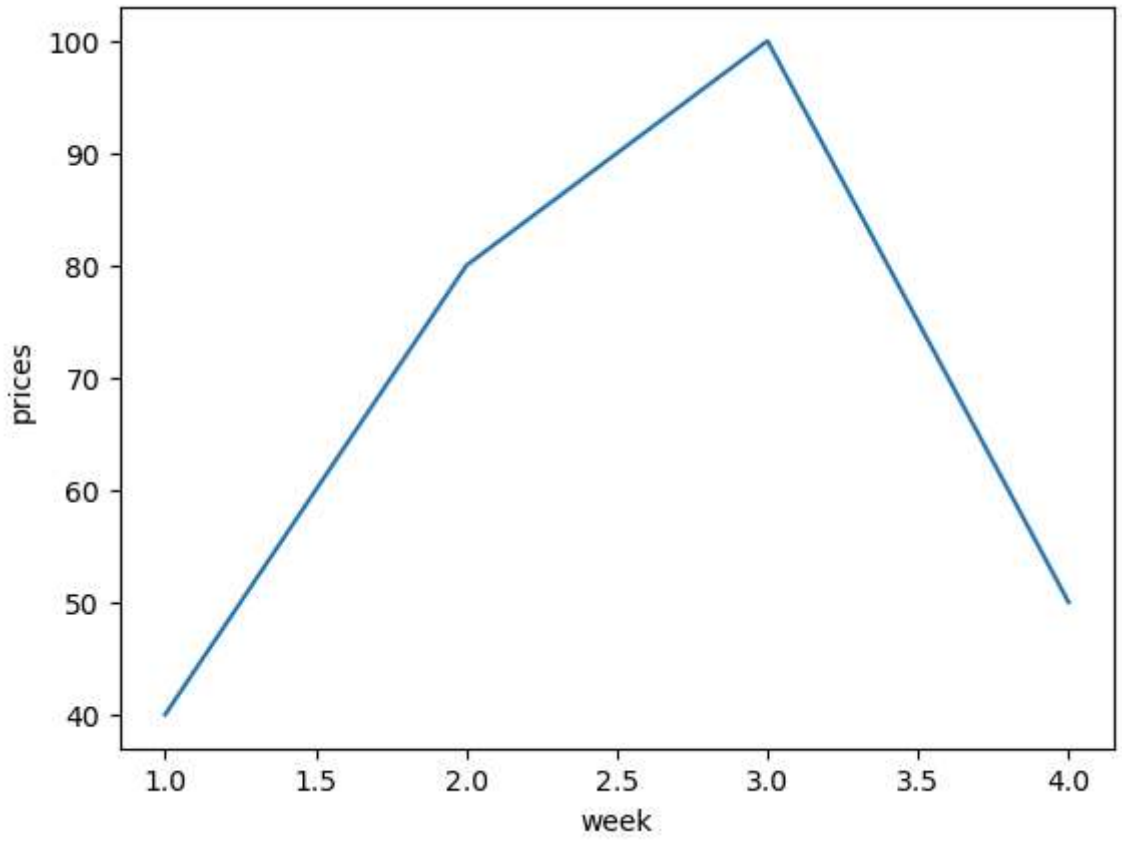
In [3]:
```
#by default plot will create a line chart
a = [1,2,3,4,5]
b = [6,7,8,9,10]

plt.plot(a,b)
plt.show()
```



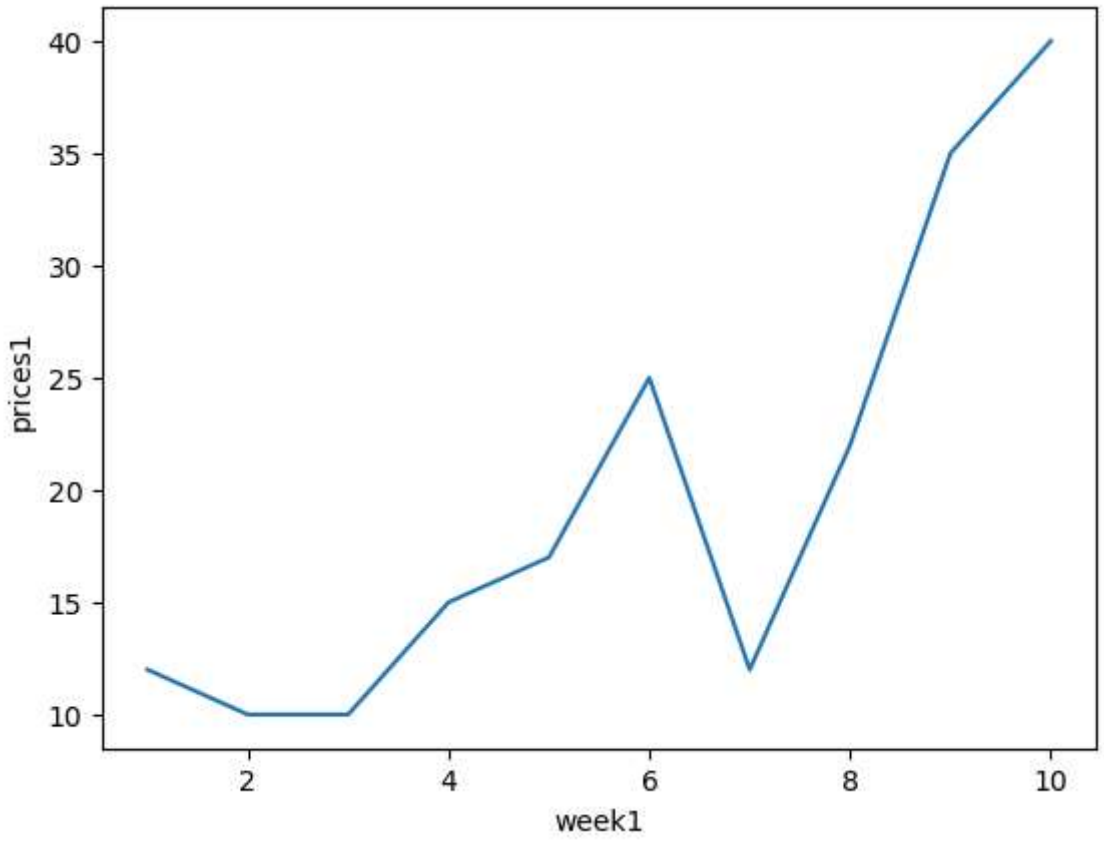
In [5]:
```
# we can give a name to the x and y axis by using xlabel and ylabel.

a = [1,2,3,4,5]
b = [6,7,8,9,10]

plt.plot(a,b)
plt.xlabel("a data")
plt.ylabel("y data")
plt.show()
```



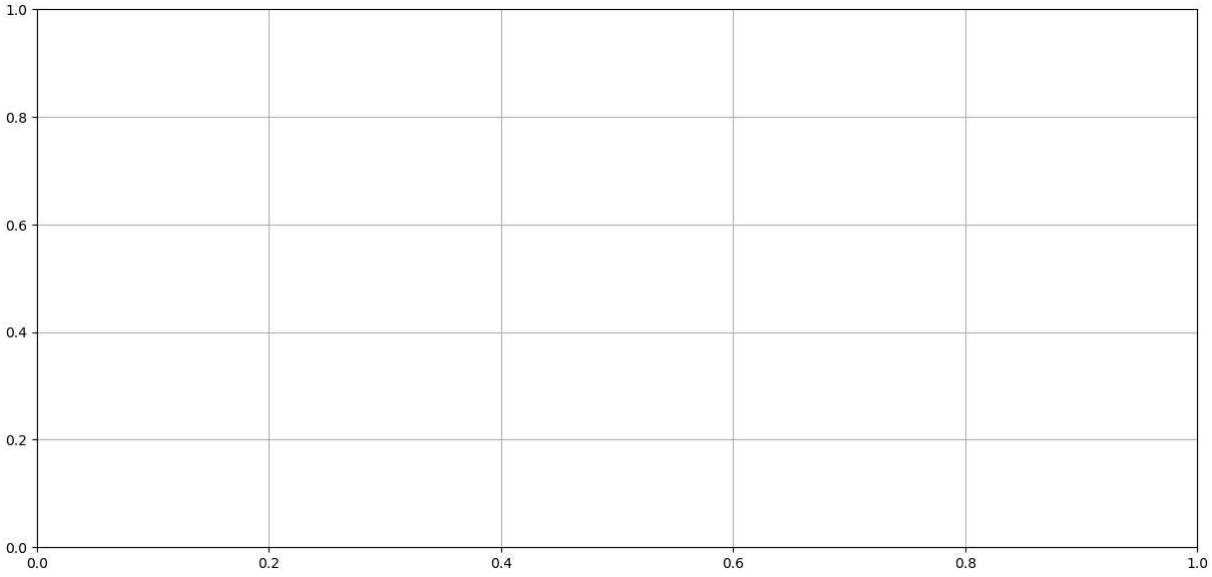
In [7]:
```
week = [1,2,3,4]
prices = [40,80,100,50]

plt.plot(week,prices)
plt.xlabel("week")
plt.ylabel("prices")
plt.show()
```



In [9]:
```
week1 = [1,2,3,4,5,6,7,8,9,10]
prices1 = [12,10,10,15,17,25,12,22,35,40]

plt.plot(week1,prices1)
plt.xlabel("week1")
plt.ylabel("prices1")
plt.show()
```

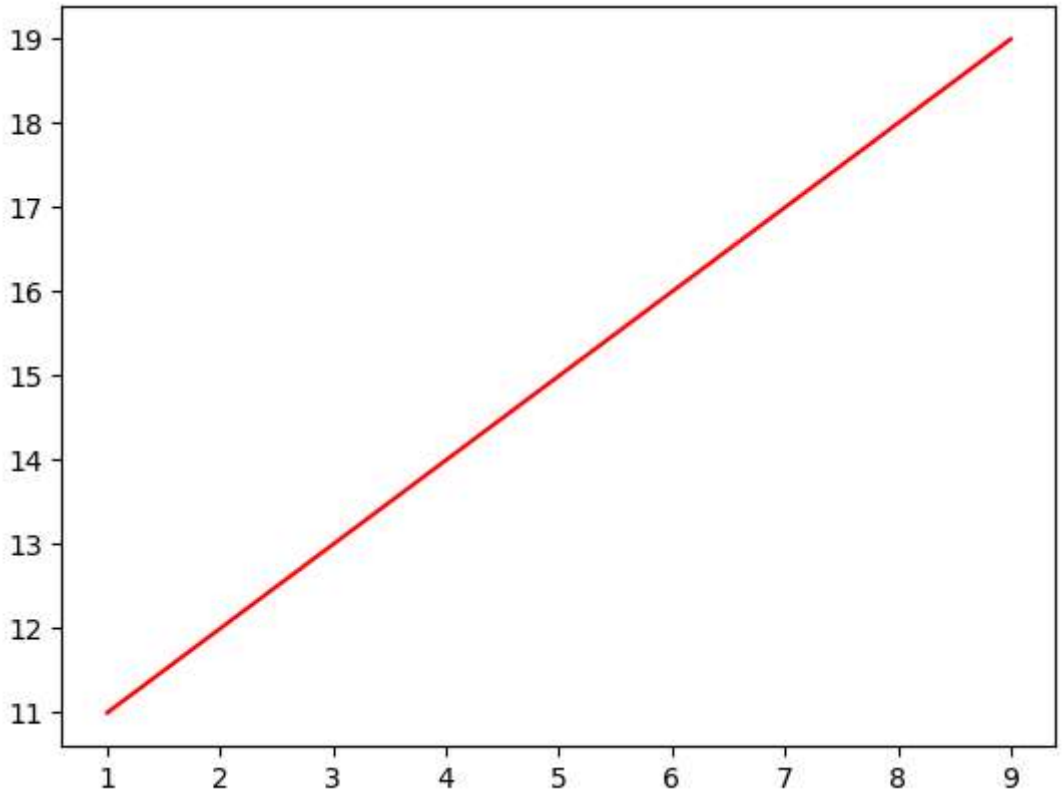


In [15]:
```
# specifying the plot size and grid using figsize
# grid is used for the exact point see purpose only like in excel

plt.figure(figsize=(15,7))
plt.grid(True)
plt.show()
```

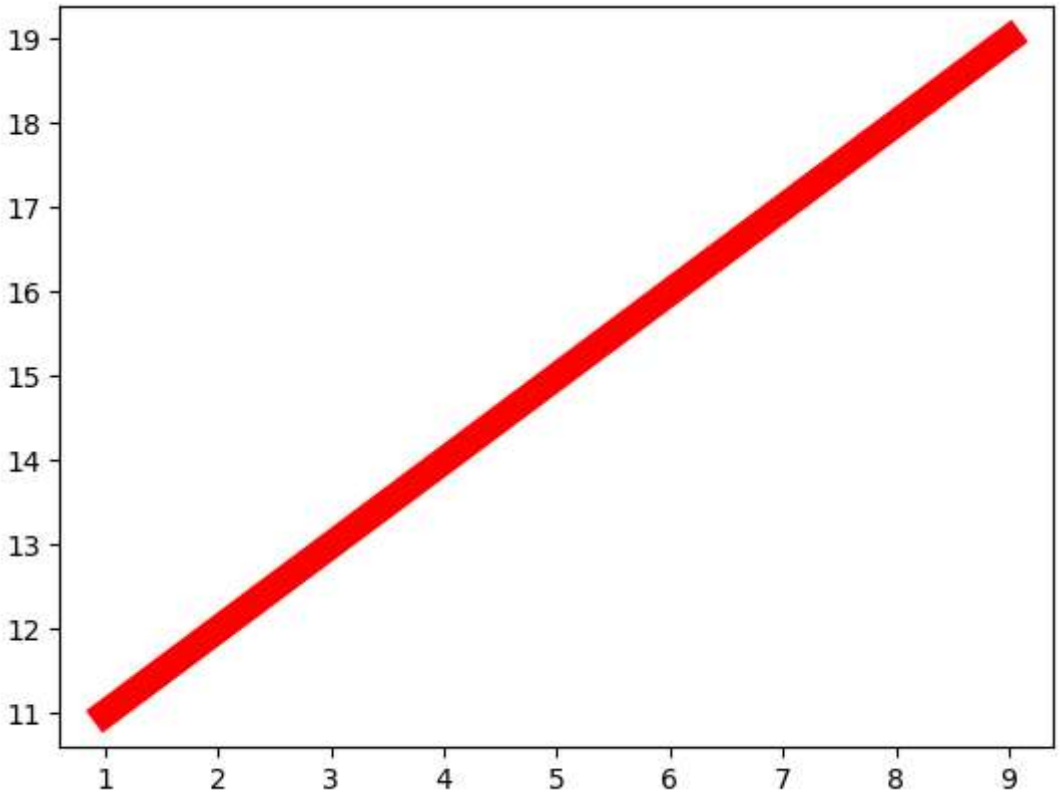


In [29]:
```
# performing variouse operation on line
#here is line color

x = np.arange(1,10,1)
```

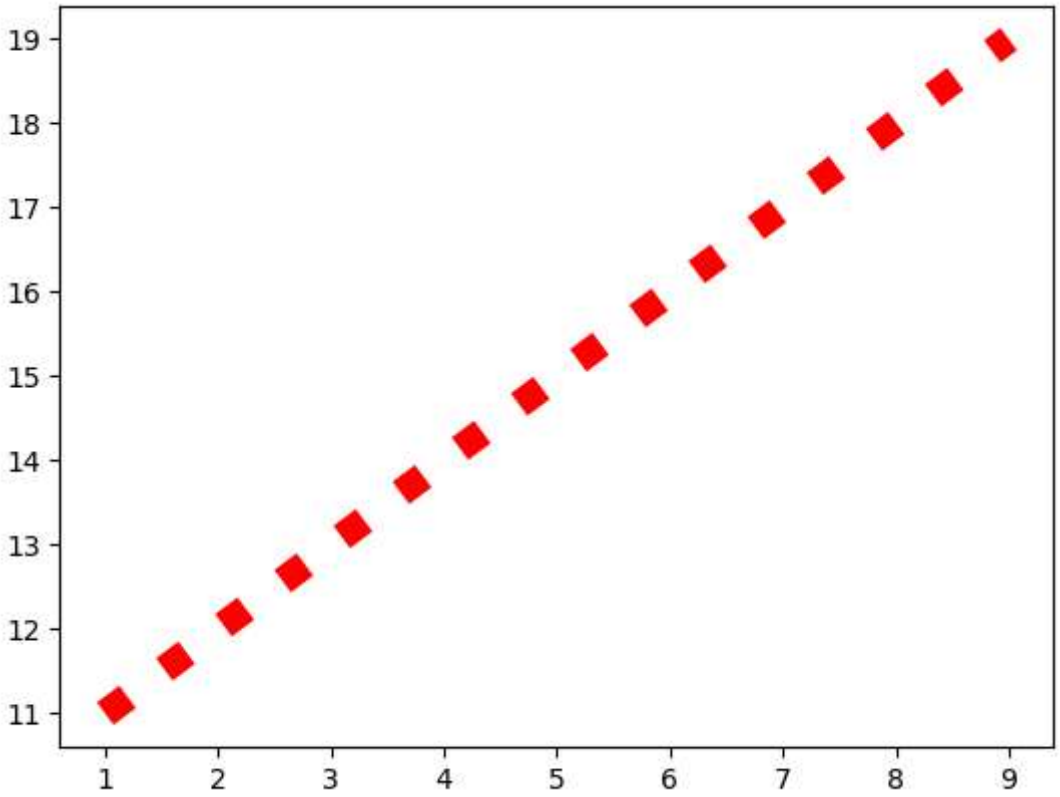
```
y = np.arange(11,20,1)

plt.plot(x,y,"r") # r is shortcode of color red
plt.show()
```



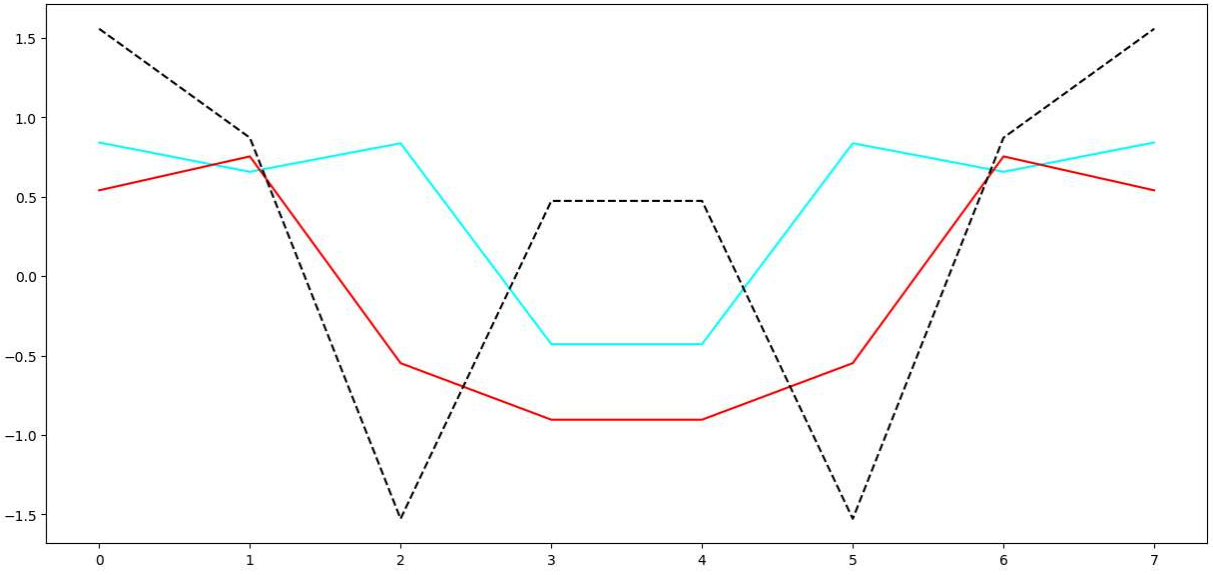
In [33]:
```
# to change the line width by using linewidth = width

x = np.arange(1,10,1)
y = np.arange(11,20,1)

plt.plot(x,y,"r",linewidth = 10) # r is shortcode of color red # linewidth is used
plt.show()
```



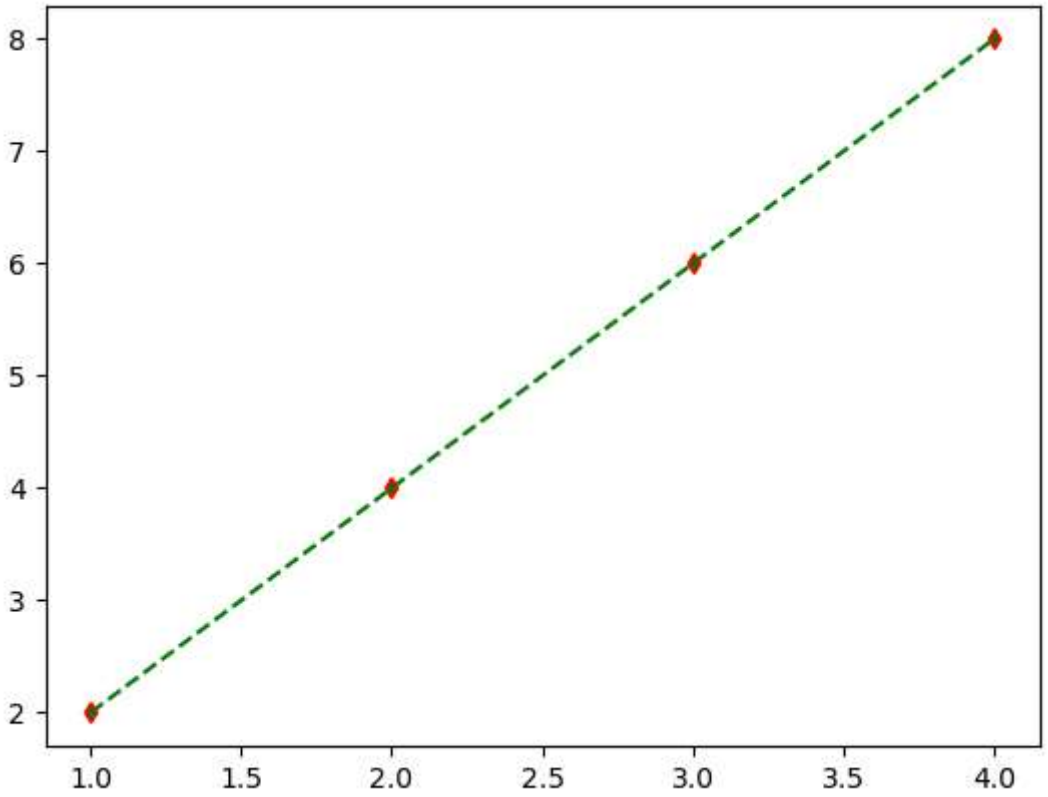
In [37]:
```
# to change the linestyle we can change it by using linestyle = name of the style

x = np.arange(1,10,1)
y = np.arange(11,20,1)

plt.plot(x,y,"r",linewidth = 10,linestyle = "dotted") # r is shortcode of color red
plt.show()
```



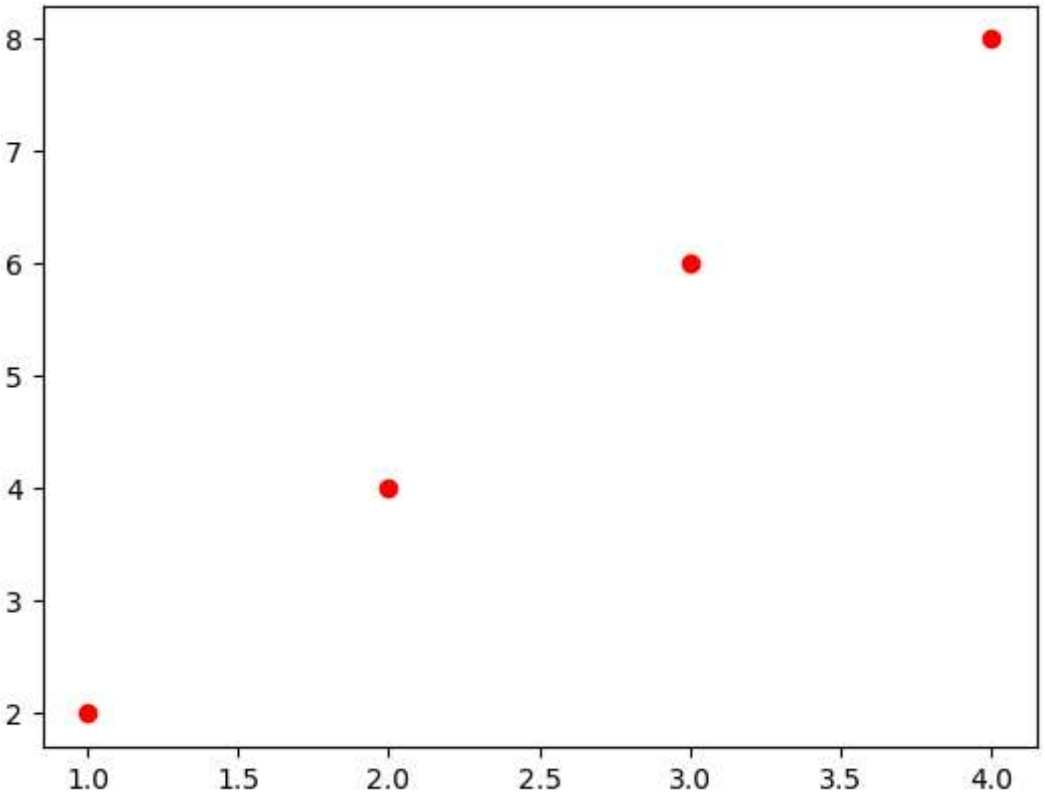
In [45]:
```
ar2 = [1,7,21,35,35,21,7,1]

a = np.sin(ar2)
b = np.cos(ar2)
c = np.tan(ar2)

plt.figure(figsize = (15,7)) # figure of the size by using figsize = length and wid
plt.plot(a,"cyan")
plt.plot(b,"red")
plt.plot(c,"black",linestyle = "dashed")
plt.show()
```



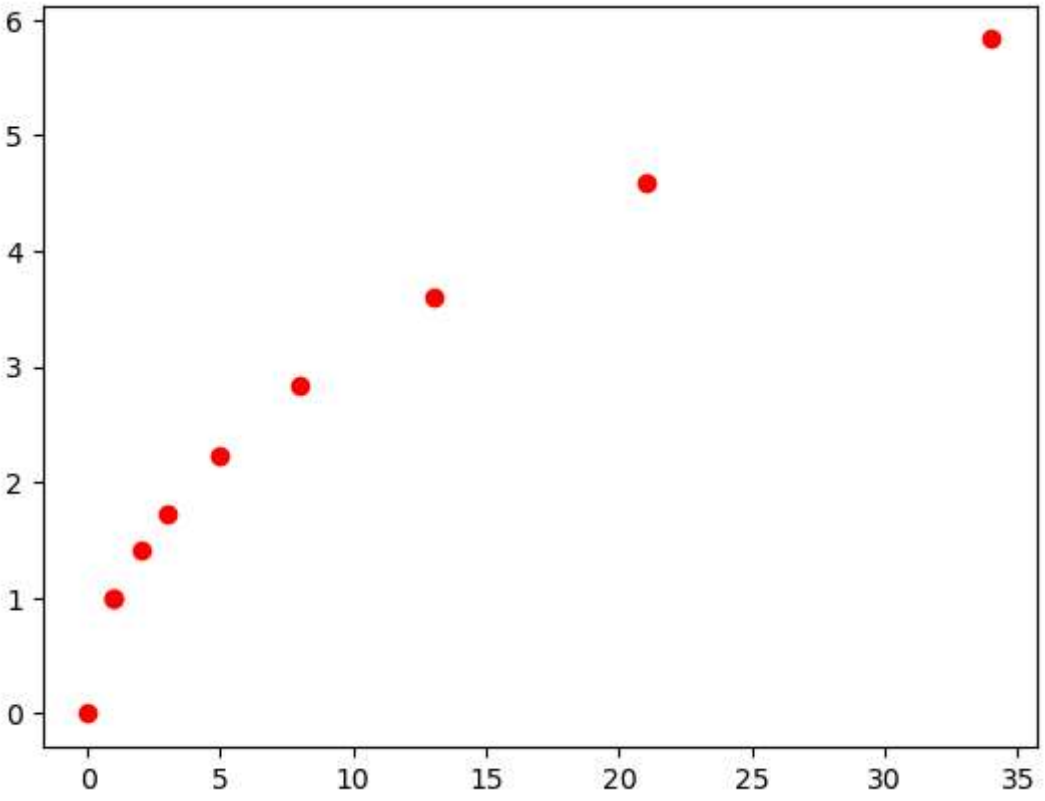
In [51]:
```
# changing the marker type,size and color

p = [1,2,3,4]
q = [2,4,6,8]

plt.plot(p,q,"g",linestyle = "dashed",marker = "d",markersize = 5,markeredgecolor =
plt.show()
```



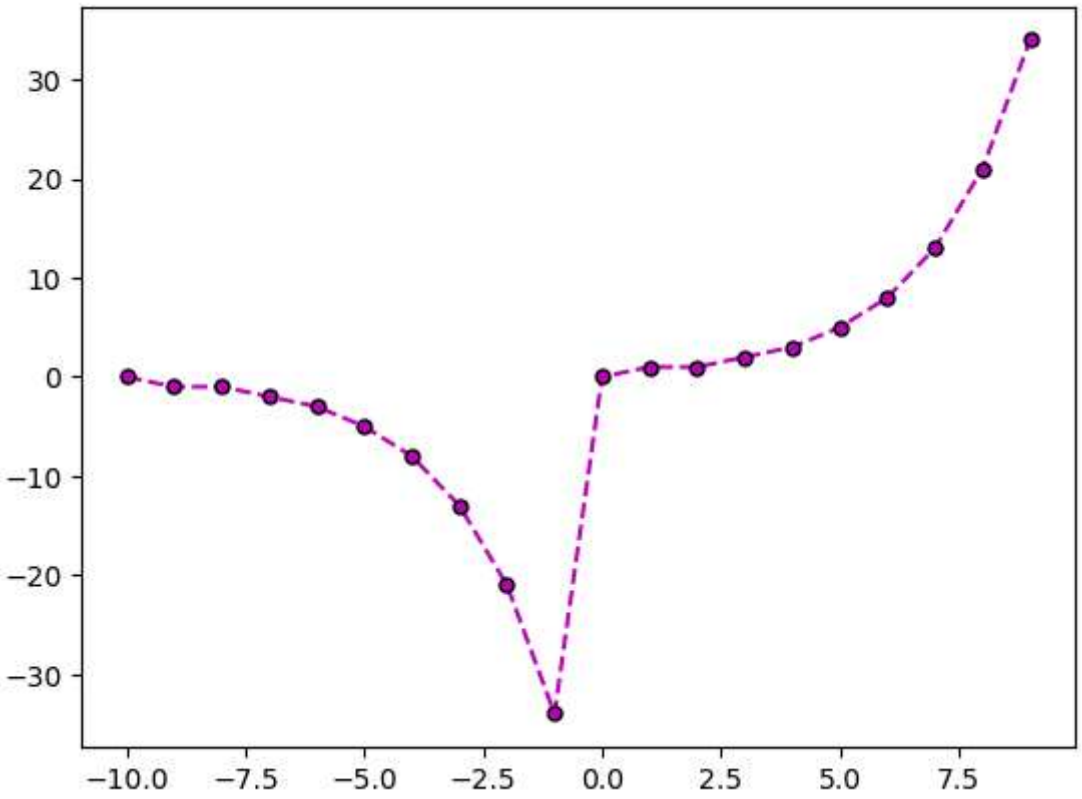
In [55]:
```
# if you forget to give the line property and just give the marker property then th

p = [1,2,3,4]
q = [2,4,6,8]

plt.plot(p,q,"ro") # here you give only marker property like the redcolor and fille
plt.show()
```



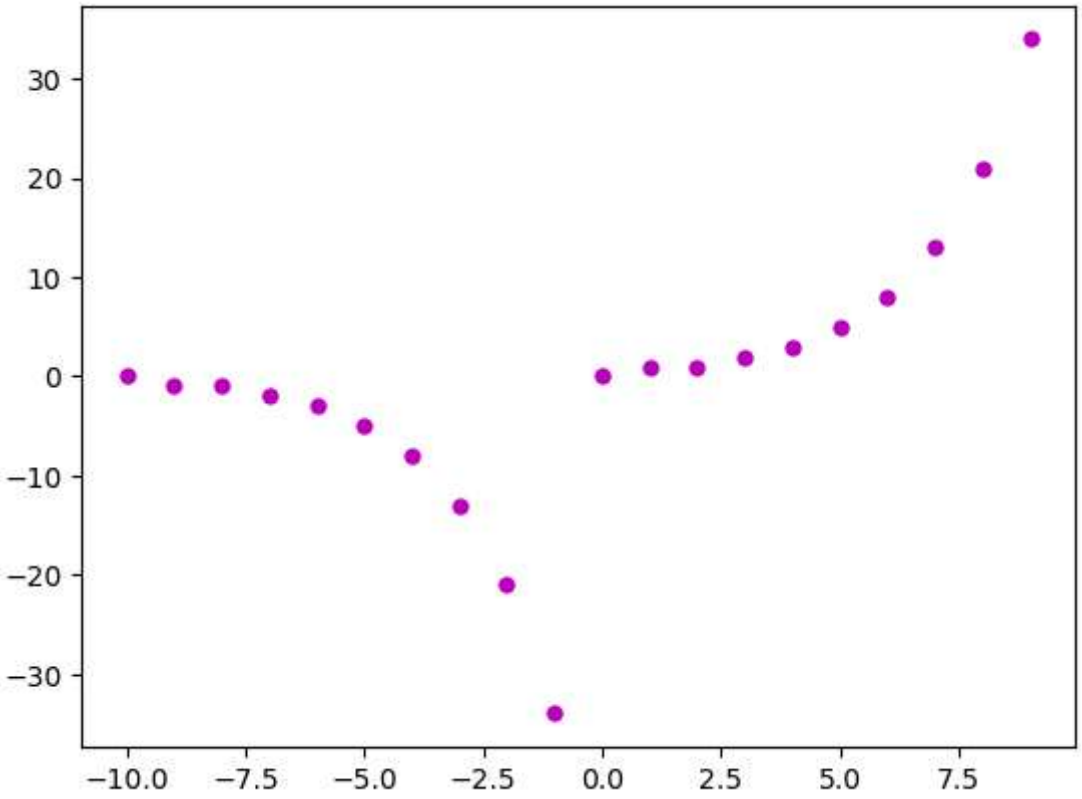
In [57]:
```
fib = [0,1,1,2,3,5,8,13,21,34]
sqrfib = np.sqrt(fib)
plt.plot(fib,sqrfib,"ro")
plt.show()
```



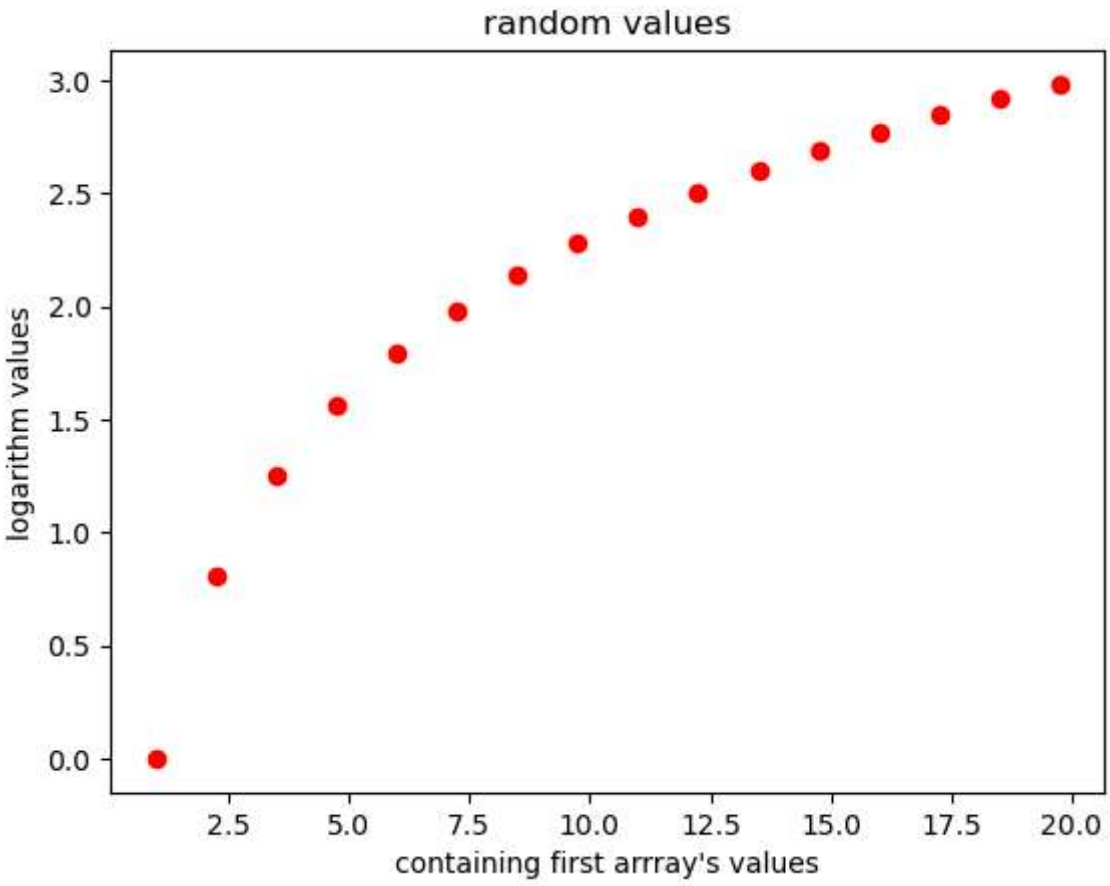
In [61]:
```
reversfib = [0,-1,-1,-2,-3,-5,-8,-13,-21,-34,0,1,1,2,3,5,8,13,21,34]
plt.plot(range(-10,10),reversfib,"mo",markersize = 5,markeredgecolor = "k",linestyl
plt.show()
```



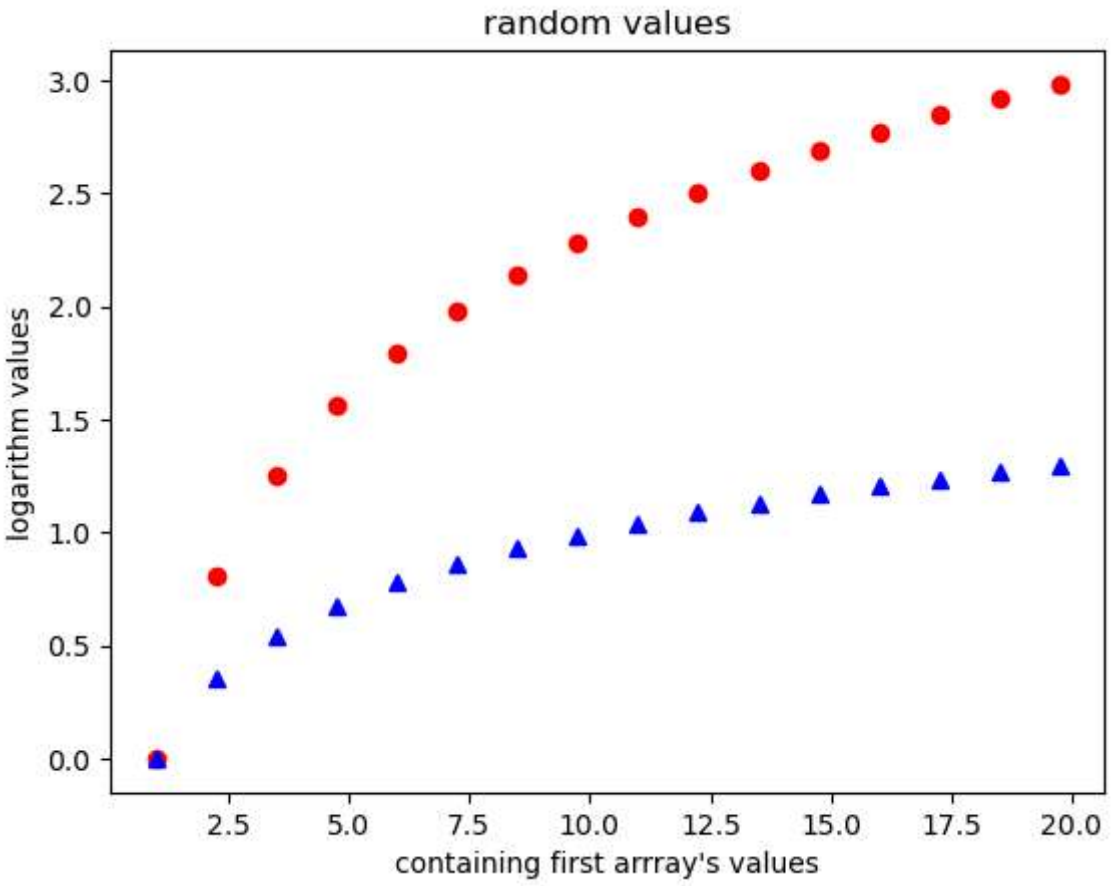
In [69]:
```
# creating scatter plots
# if you forget to specify the linestyle argument then plot will creat the scatter

reversfib = [0,-1,-1,-2,-3,-5,-8,-13,-21,-34,0,1,1,2,3,5,8,13,21,34]
plt.plot(range(-10,10),reversfib,"mo",markersize = 5)
plt.show()
```



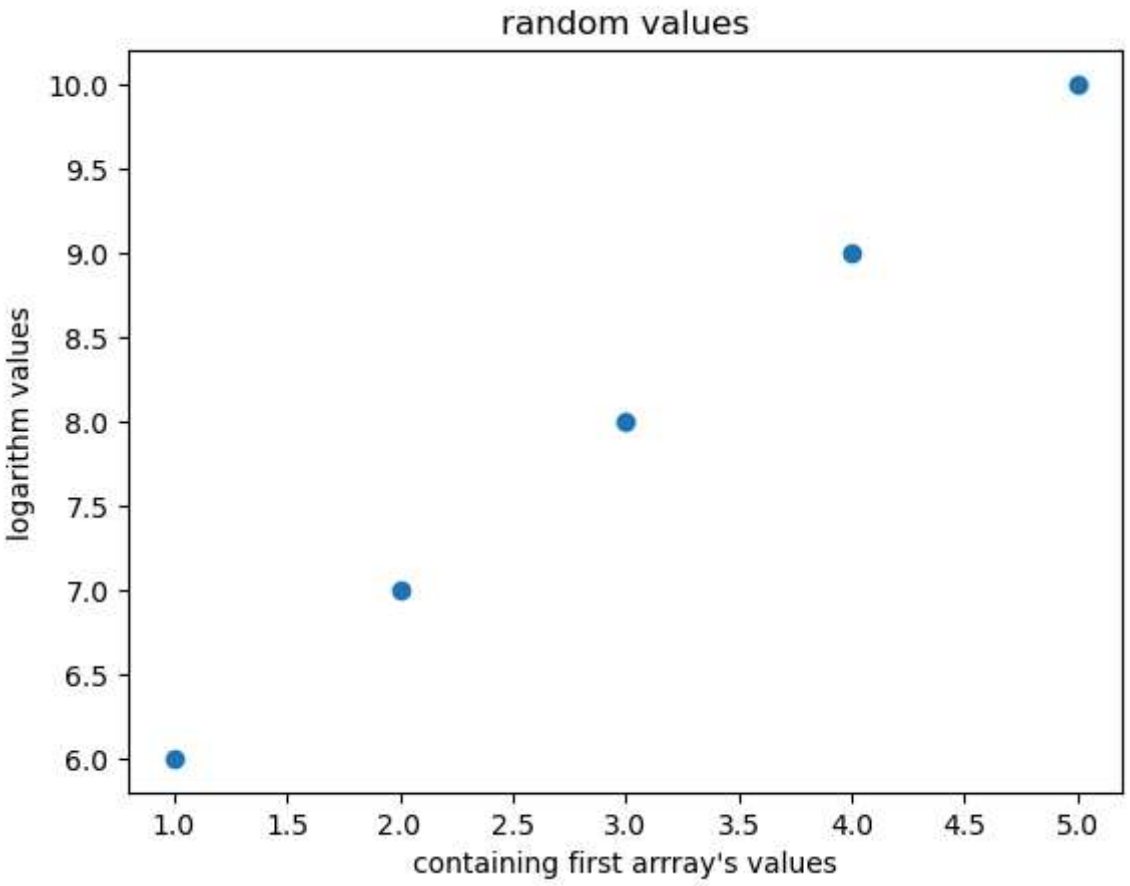
In [77]:
```
a = np.arange(1,20,1.25)
b = np.log(a)

plt.plot(a,b,"ro")
plt.title("random values") # for giving title
plt.xlabel("containing first arrray's values")
plt.ylabel("logarithm values")
plt.show()
```



In [81]:
```
# now we perform the a vs c scatter plots

a = np.arange(1,20,1.25)
b = np.log(a)
c = np.log10(a)

plt.plot(a,b,"ro")
plt.plot(a,c,"b^")
plt.title("random values") # for giving title
plt.xlabel("containing first arrray's values")
plt.ylabel("logarithm values")
plt.show()
```

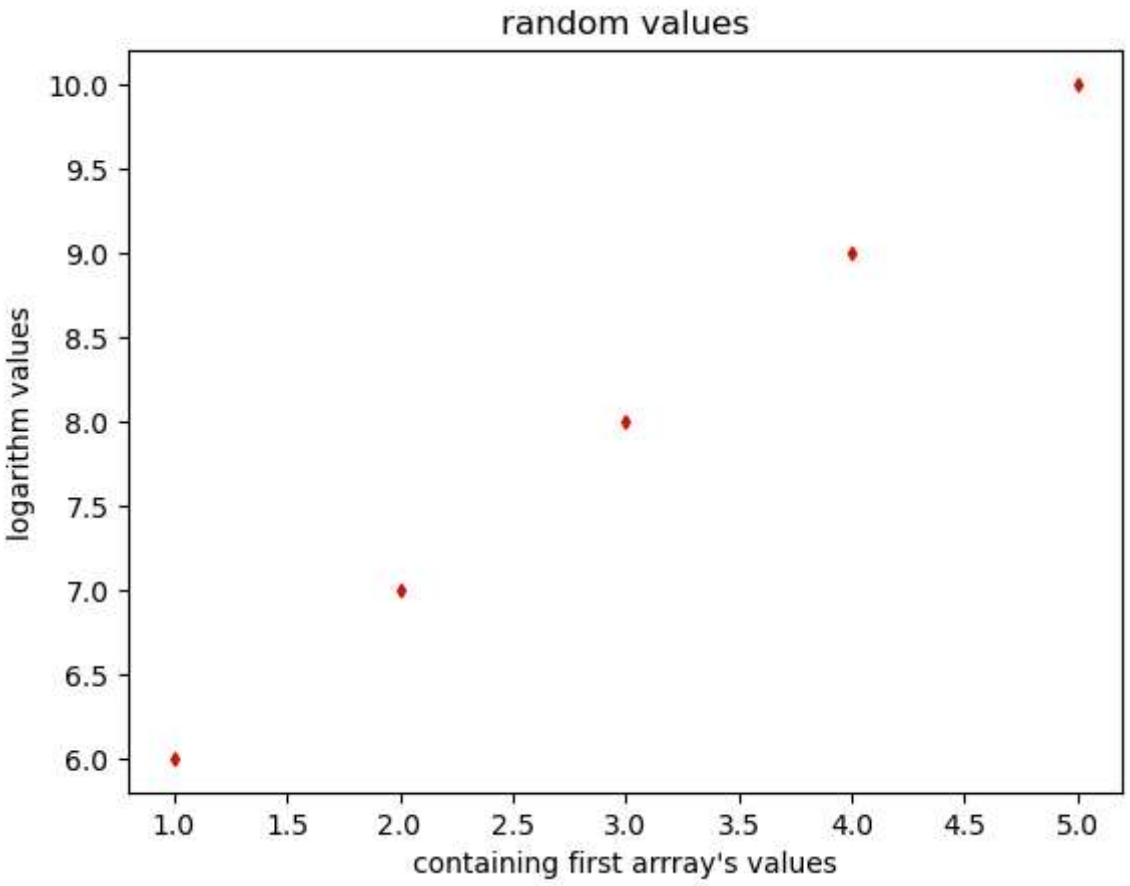


In [97]:
```
# we can create a scatter-plot by using scatter-function

a1 = [1,2,3,4,5]
b1 = [6,7,8,9,10]

plt.scatter(a1,b1) # makesure x and y size is same for creating scatter-plot
plt.title("random values") # for giving title
plt.xlabel("containing first arrray's values")
plt.ylabel("logarithm values")
plt.show()
```



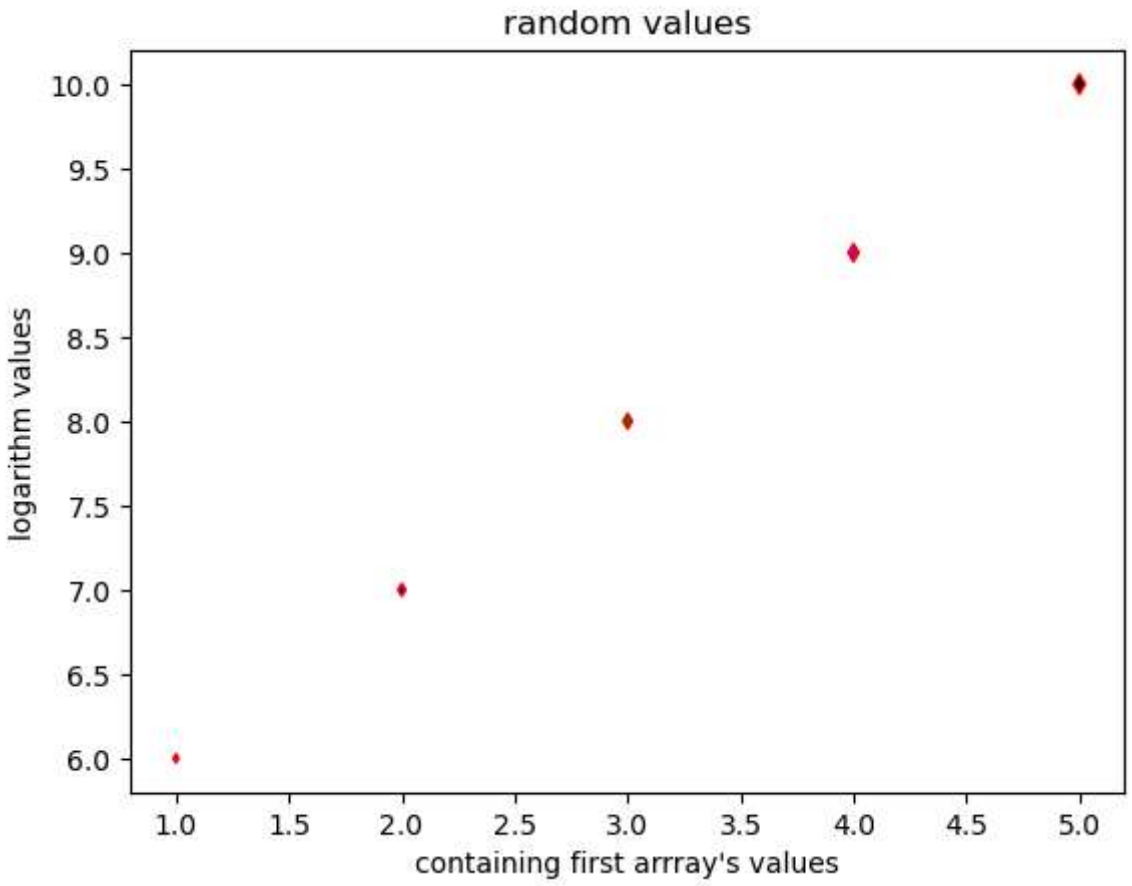
In [110…

```
# for changing the marker property we can not use the name of markercolor or full n
# here we need to use the shortcode for markers like c = color,s=size,marker = styl

a1 = [1,2,3,4,5]
b1 = [6,7,8,9,10]

plt.scatter(a1,b1,marker = "d",edgecolor = "red",s = 10,c = "g") # makesure x and y
plt.title("random values") # for giving title
plt.xlabel("containing first arrray's values")
plt.ylabel("logarithm values")
plt.show()
```

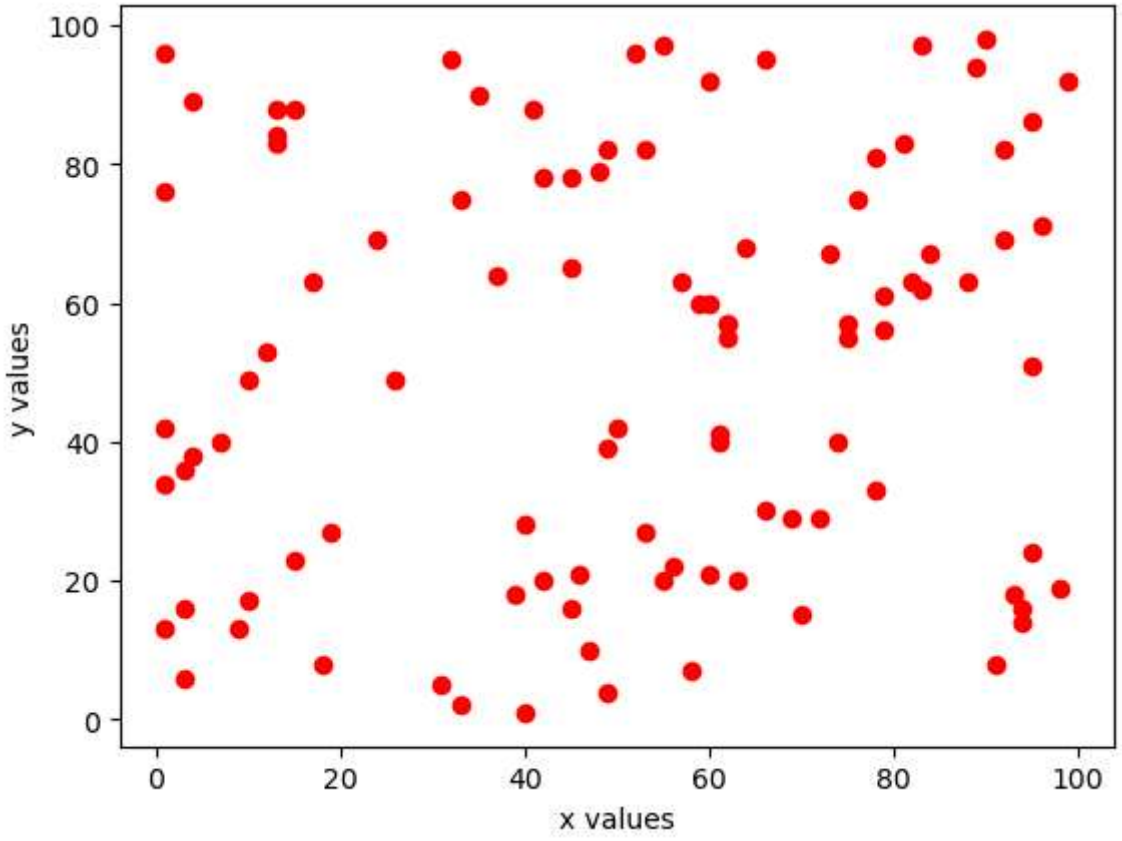


In [116…

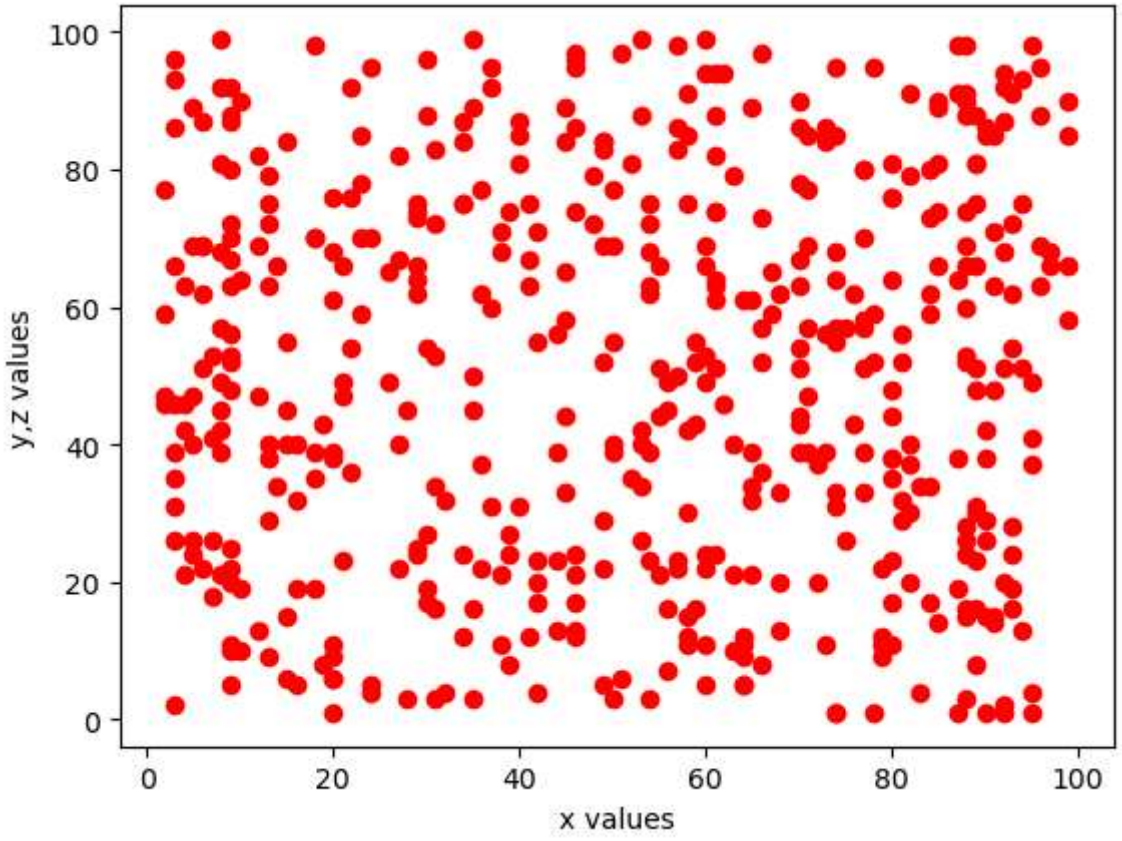
```
# specify the various color and size of the data-points

a1 = [1,2,3,4,5]
b1 = [6,7,8,9,10]

sarr = [5,10,15,20,25]
carr = ['r','b','g','m','k']

# makesure x and y size is same for creating scatter-plot
plt.scatter(a1,b1,marker = "d",edgecolor = "red",s = sarr,c = carr)  #we can declar
plt.title("random values") # for giving title
plt.xlabel("containing first arrray's values")
plt.ylabel("logarithm values")
plt.show()
```



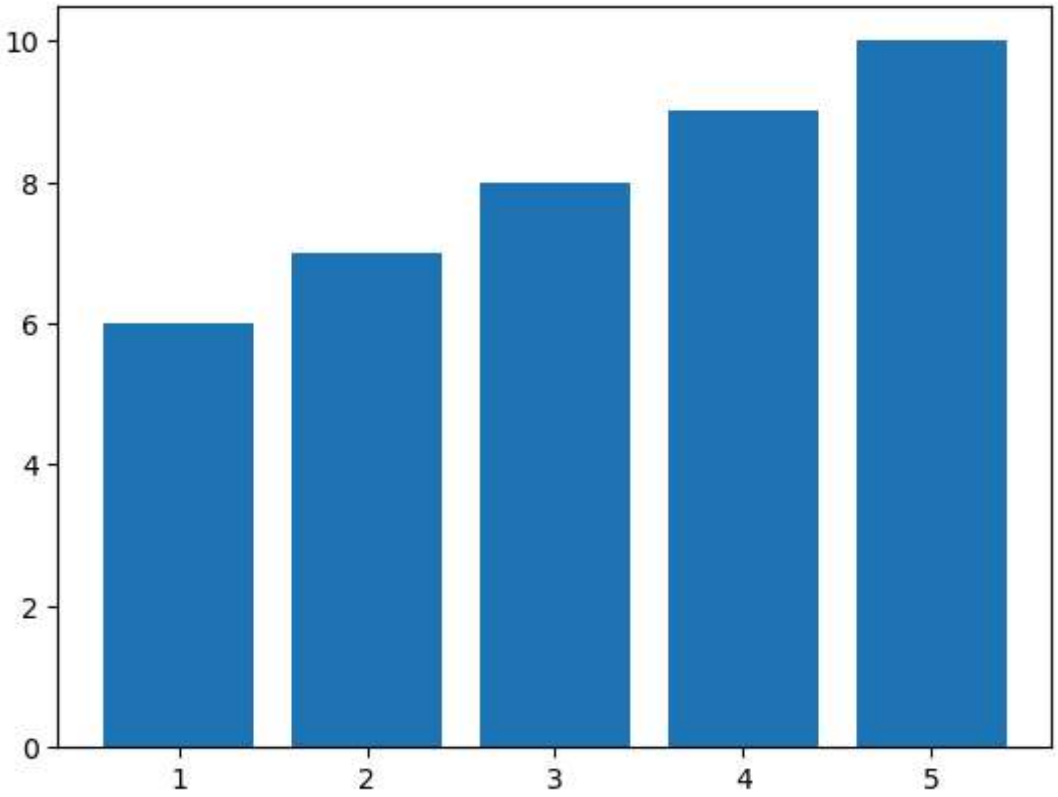
In [17]:
```
x = np.random.randint(1,100,size = (100,))
y = np.random.randint(1,100,size = (100,))

plt.scatter(x,y,color = "red")
plt.xlabel("x values")
plt.ylabel("y values")
plt.show()
```



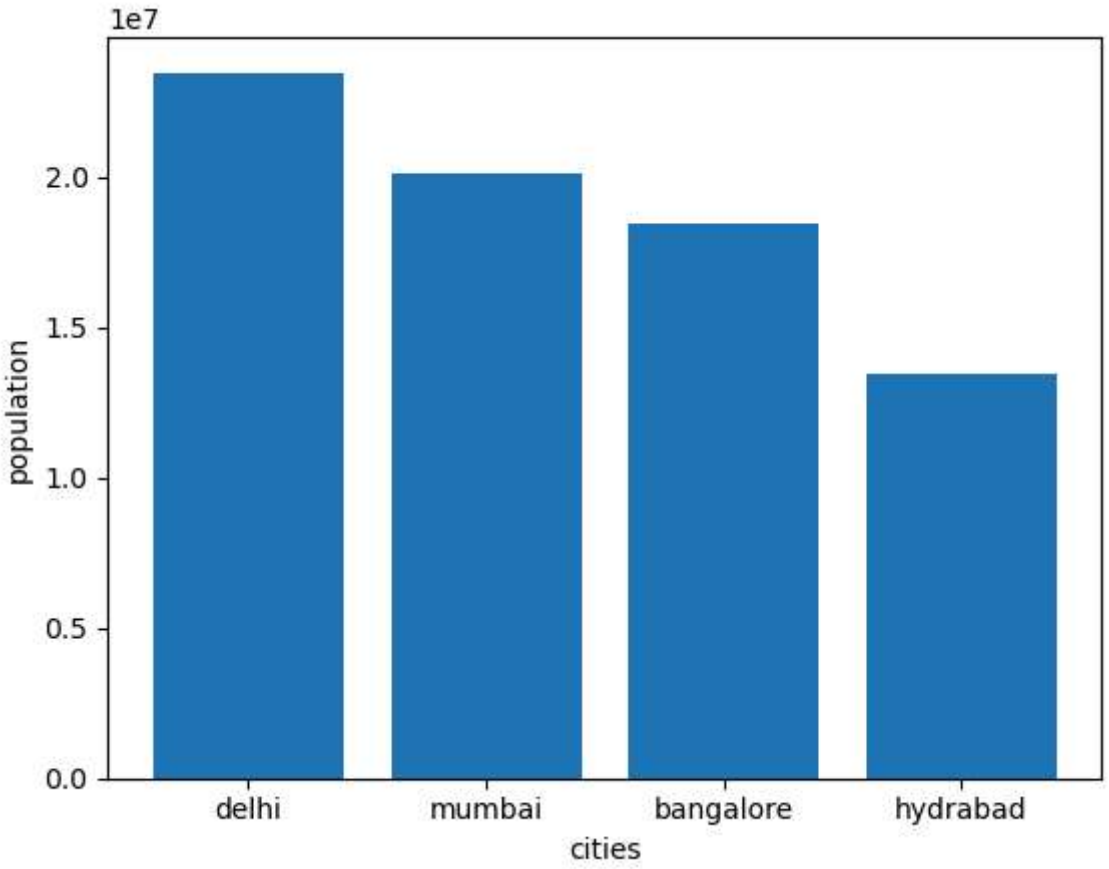
In [32]:
```
x = np.random.randint(1,100,size = (250,))
y = np.random.randint(1,100,size = (250,))
z = np.random.randint(1,100,size = (250,))

size = range(1,60,5)
plt.scatter(x,y,color = "red")
plt.scatter(x,z,color = "red")
plt.xlabel("x values")
plt.ylabel("y,z values")
plt.show()
```



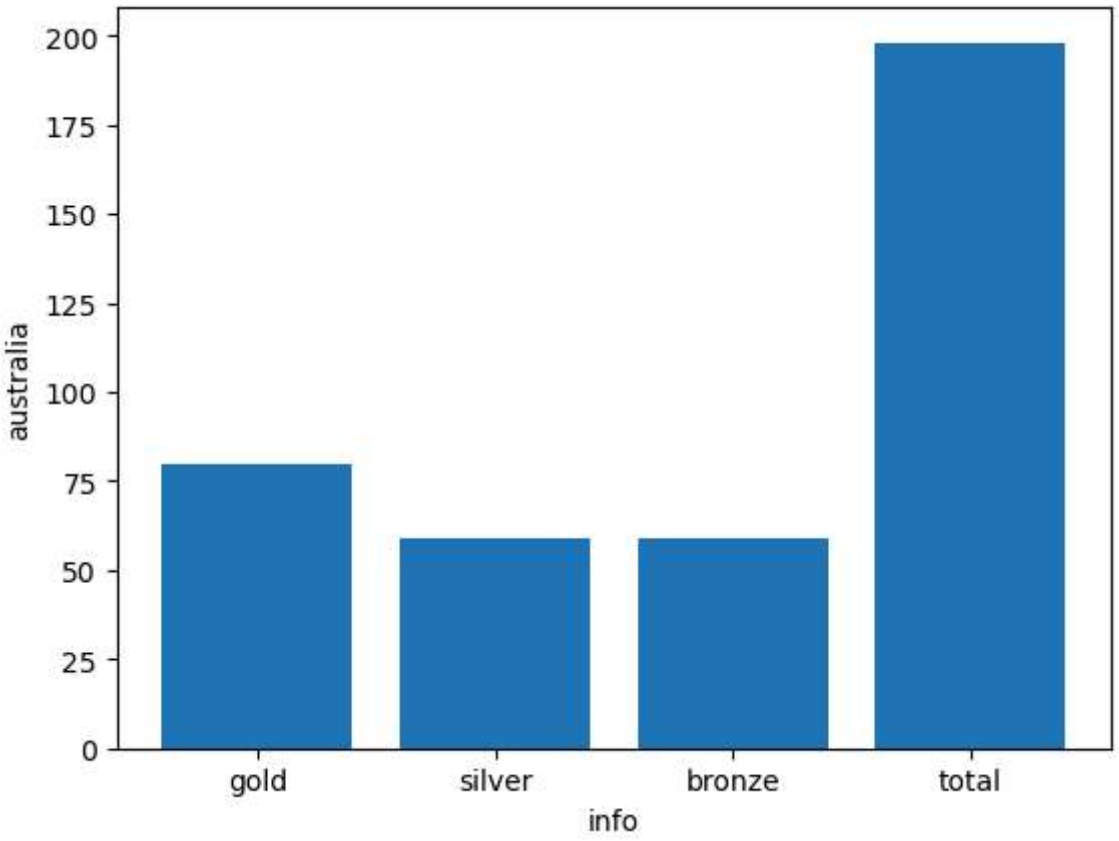
In [44]:
```
# now we create bar charts using bar() function

a = [1,2,3,4,5] # bar on this numbers
b = [6,7,8,9,10] # bar height till this numbers

plt.bar(a,b)
plt.show()
```



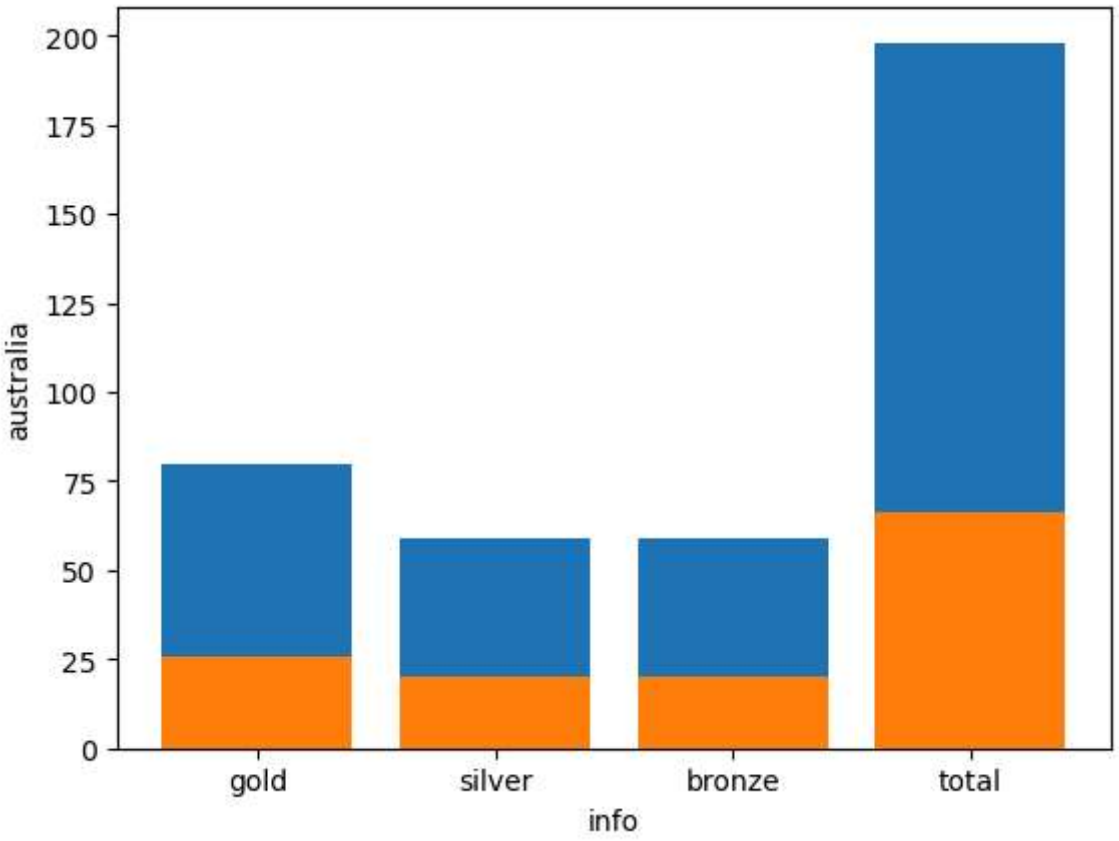
In [46]:
```
cities = ['delhi','mumbai','bangalore','hydrabad']
population = [23456123,20083104,18456123,13411093]

plt.bar(cities,population)
plt.xlabel("cities")
plt.ylabel("population")
plt.show()
```



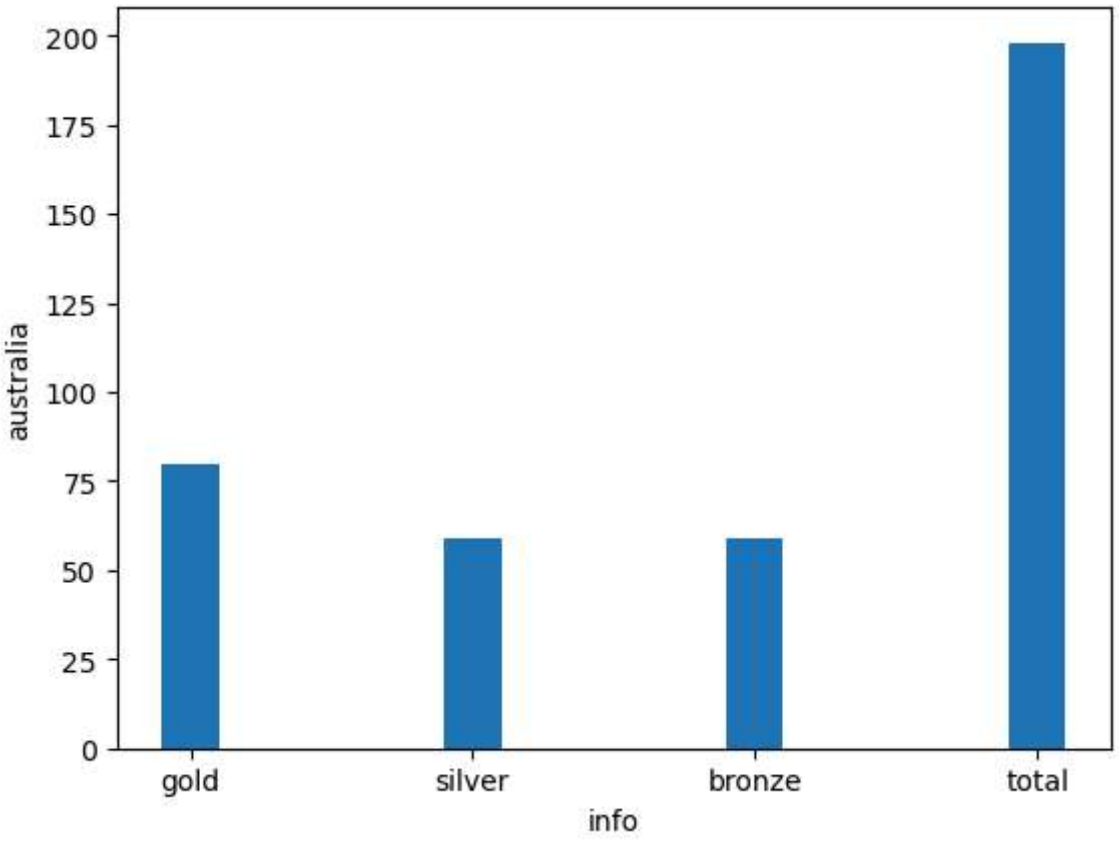
In [94]:
```
info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia)
plt.show()
```



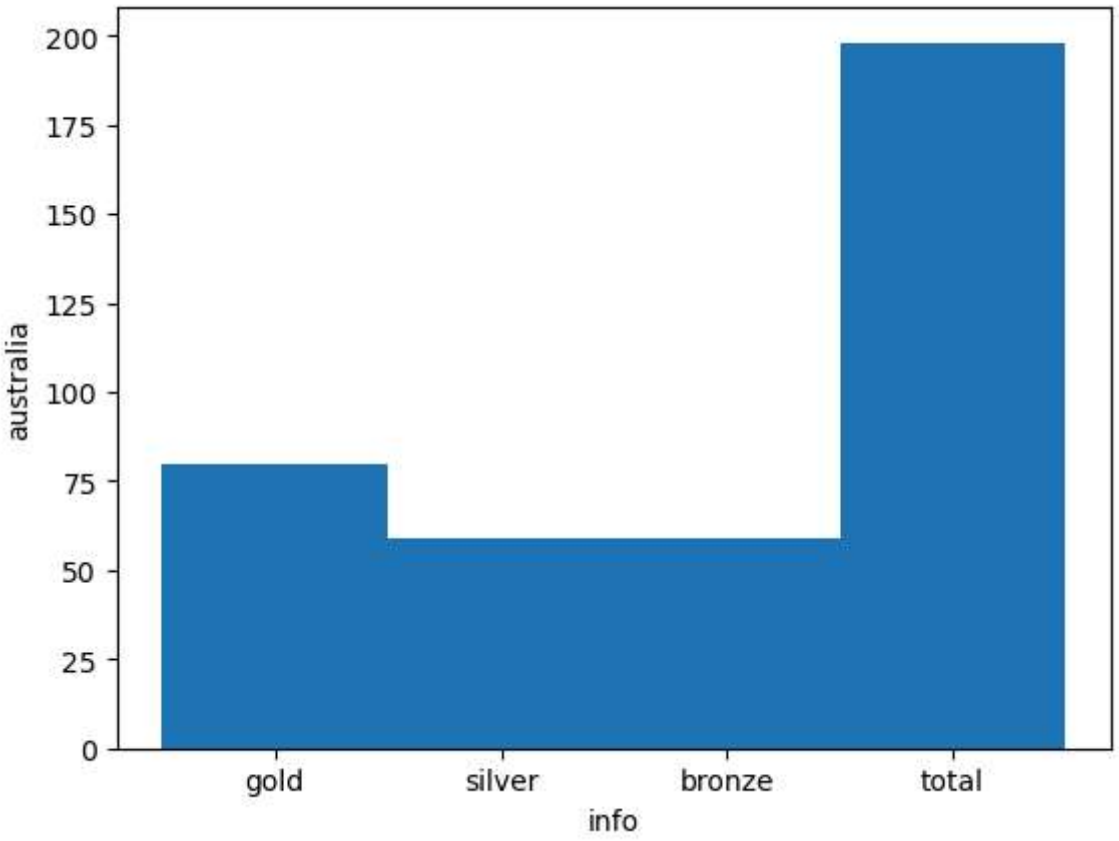
In [54]:
```
# color conflicts due to different data on the same axis data we can say collaps of

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
india = [26,20,20,66]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia)
plt.bar(info,india)
plt.show()
```



In [70]:
```
# changing width of the bars by default bars width is equals to 0.8 units we can us

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,width = 0.2) # if you specify single or scaler value then th
plt.show()
```

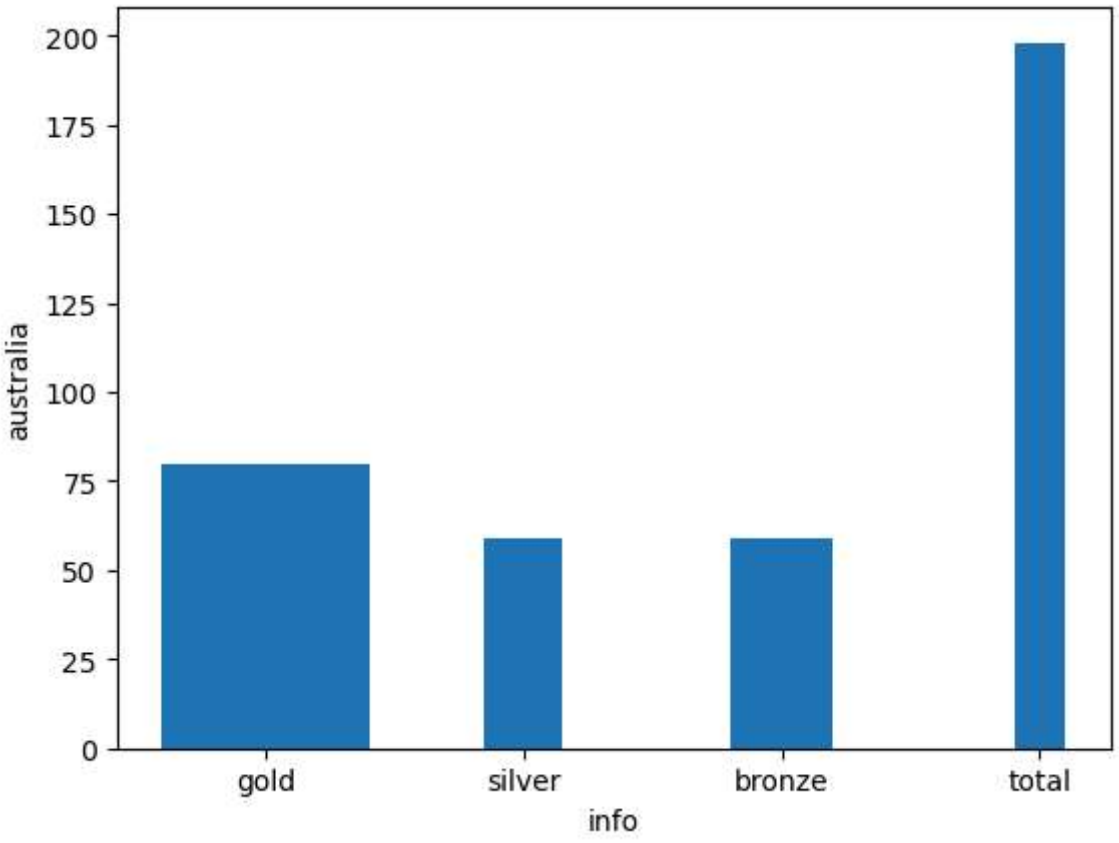


In [72]:
```
# changing width of the bars by default bars width is equals to 0.8 units we can us

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,width = 1) # if you specify single or scaler value then that
plt.show()
```

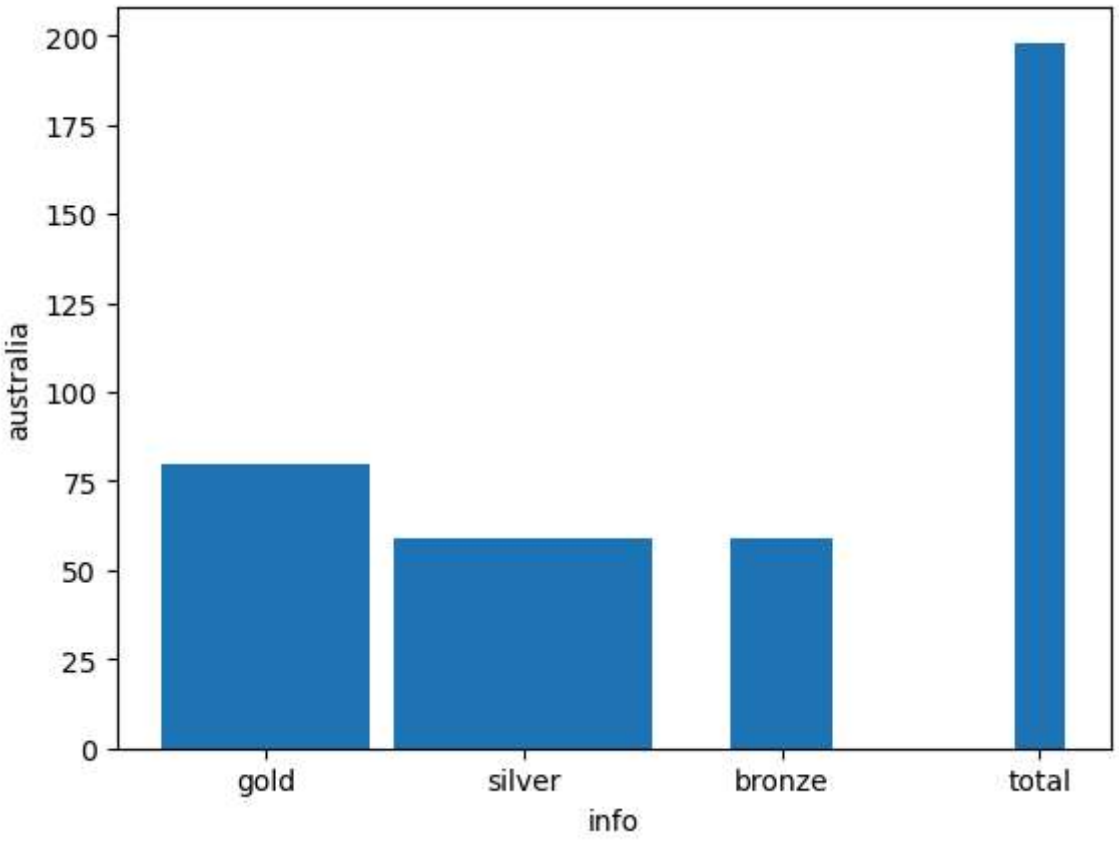


In [100…

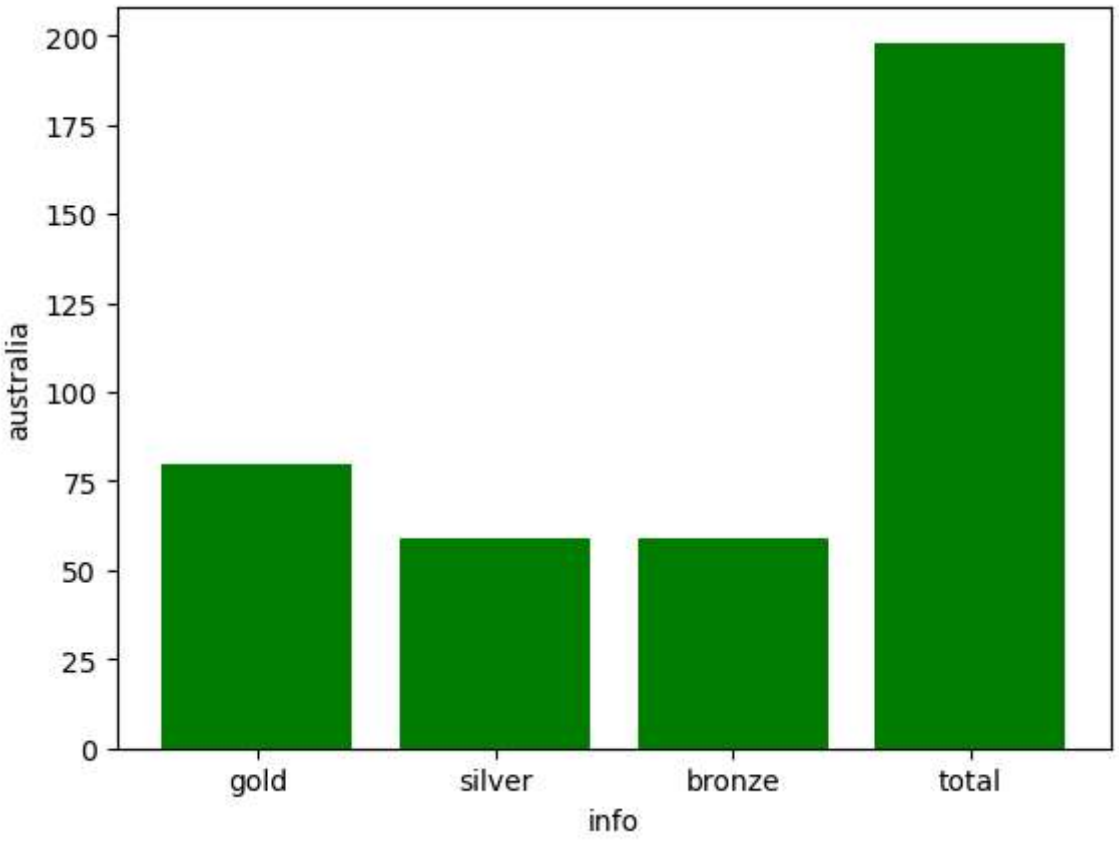
```
# for different bars we can give diffrent width values by using list or tuple

bars = (0.8,0.3,0.4,0.2) # here we use tuple you can use list also

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,width = bars) # if you specify single or scaler value then t
plt.show()
```



In [98]:
```
# for different bars we can give diffrent width values by using list or tuple

bars = [0.8,0.3,0.4,0.2] # here we use list
bars[1] = 1 # here is the benifit of using list we can modify the list values

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,width = bars) # if you specify single or scaler value then t
plt.show()
```

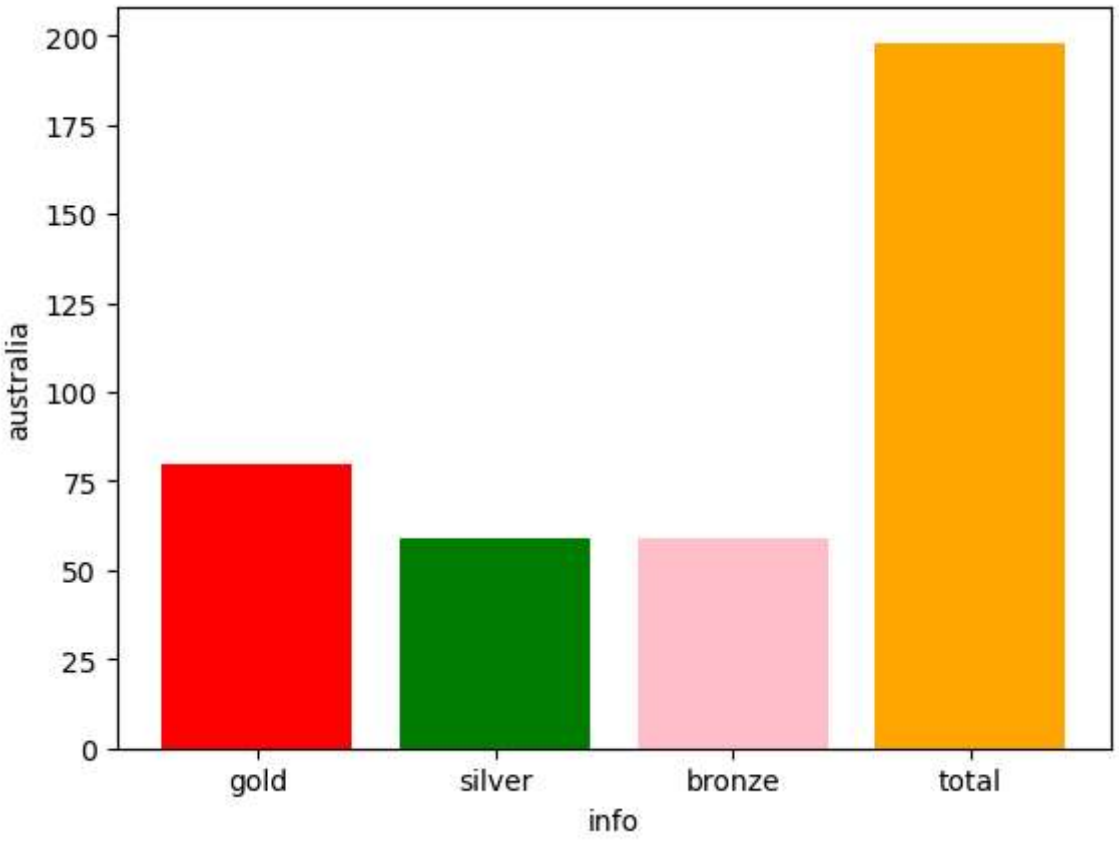


In [96]:
```
# we can change the color of a bar chart by using color = your favourite color

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,color = "green")
plt.show()
```



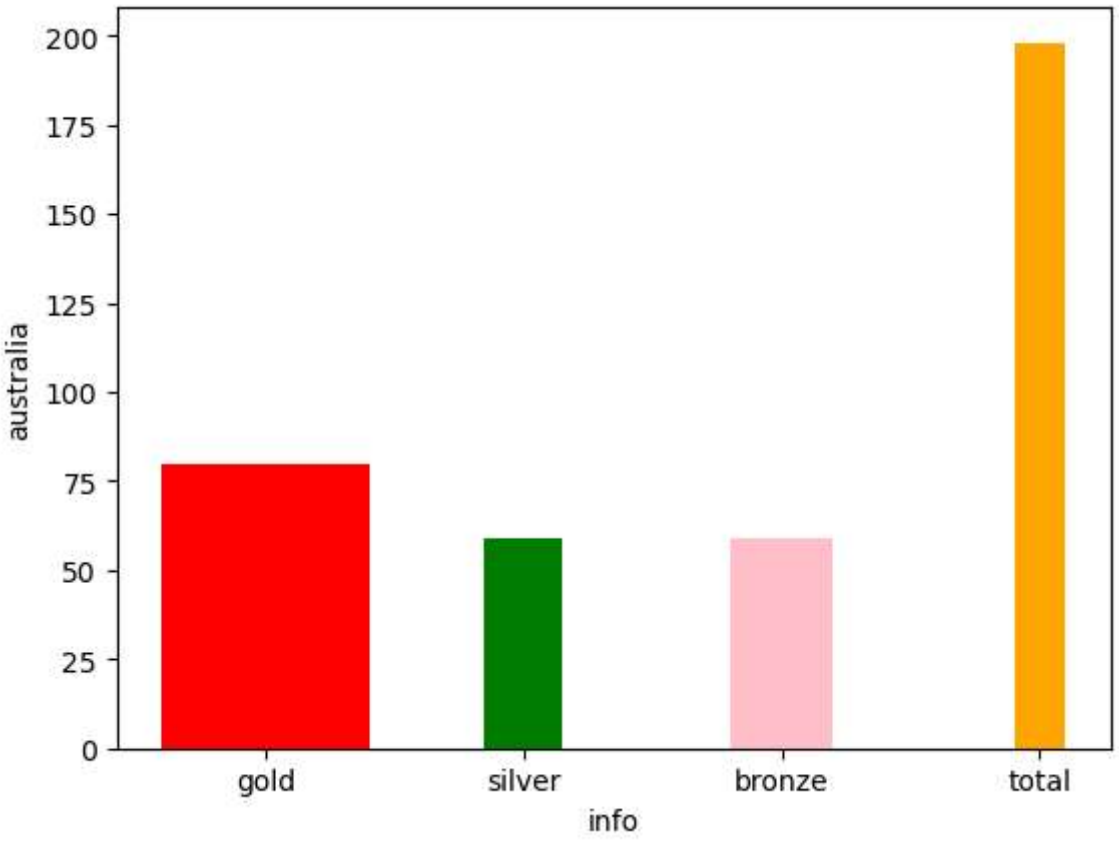
In [102…

```
# for the diffrent bars we can pass multiple color by using list or tuple.

color = ['red','green','pink','orange']

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,color = color)
plt.show()
```



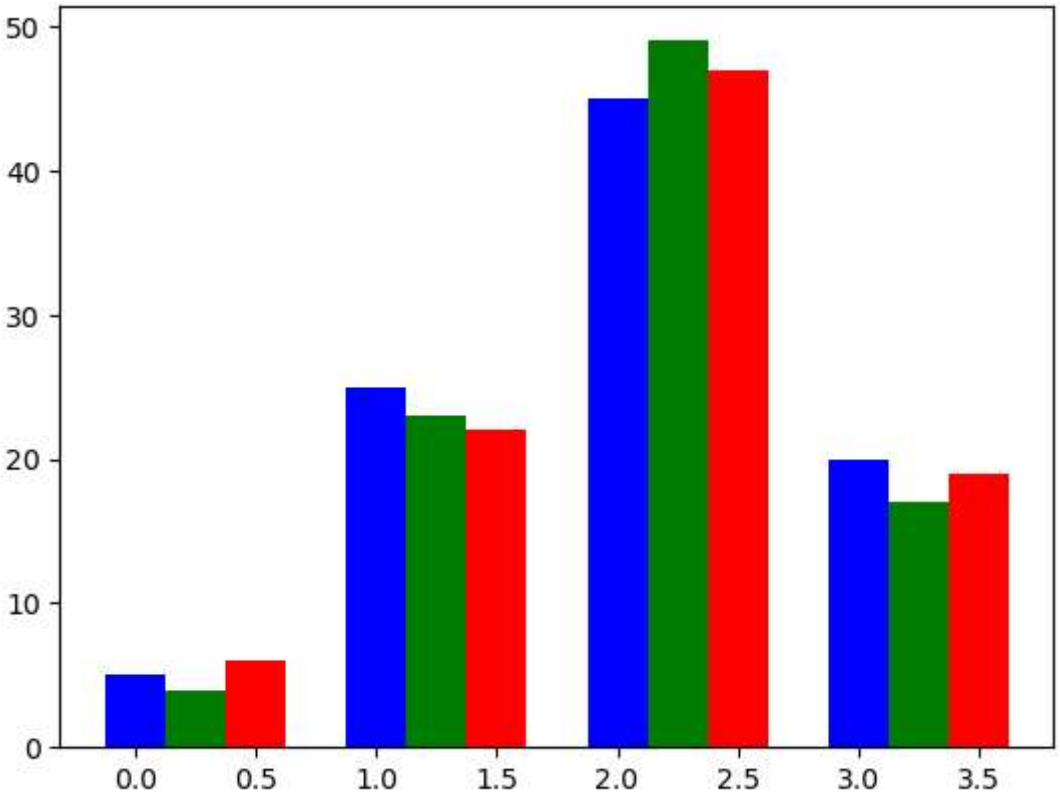
In [104…

```
# we can combine both width and color argument in bar-plot
# for the diffrent bars we can pass multiple color by using list or tuple.

color = ['red','green','pink','orange']
bars = [0.8,0.3,0.4,0.2]

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,color = color,width = bars)
plt.show()
```



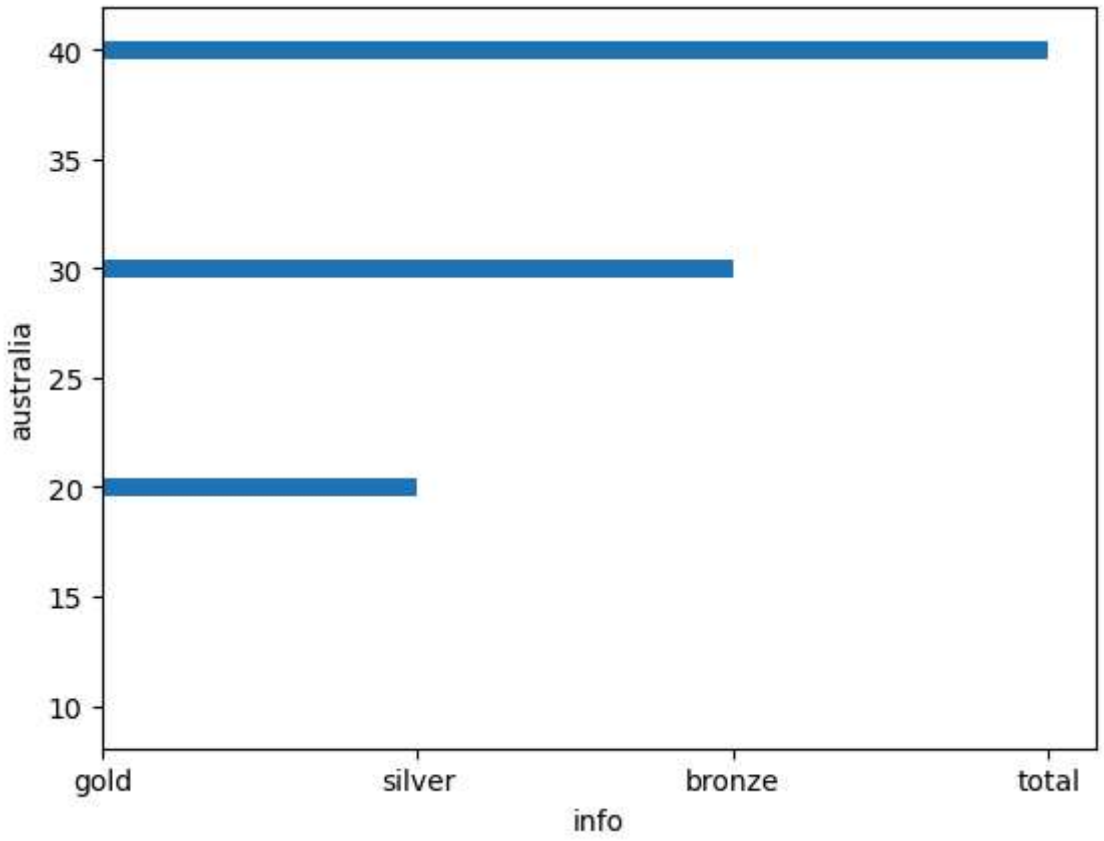
In [106…

```
# creating multiple bars on a chart

val = [[5,25,45,20],[4,23,49,17,],[6,22,47,19]]
x = np.arange(4)
plt.bar(x+0.00,val[0] , color = 'b',width = 0.25)
plt.bar(x+0.25,val[1] , color = 'g',width = 0.25)
plt.bar(x+0.50,val[2] , color = 'r',width = 0.25)
plt.show()
```



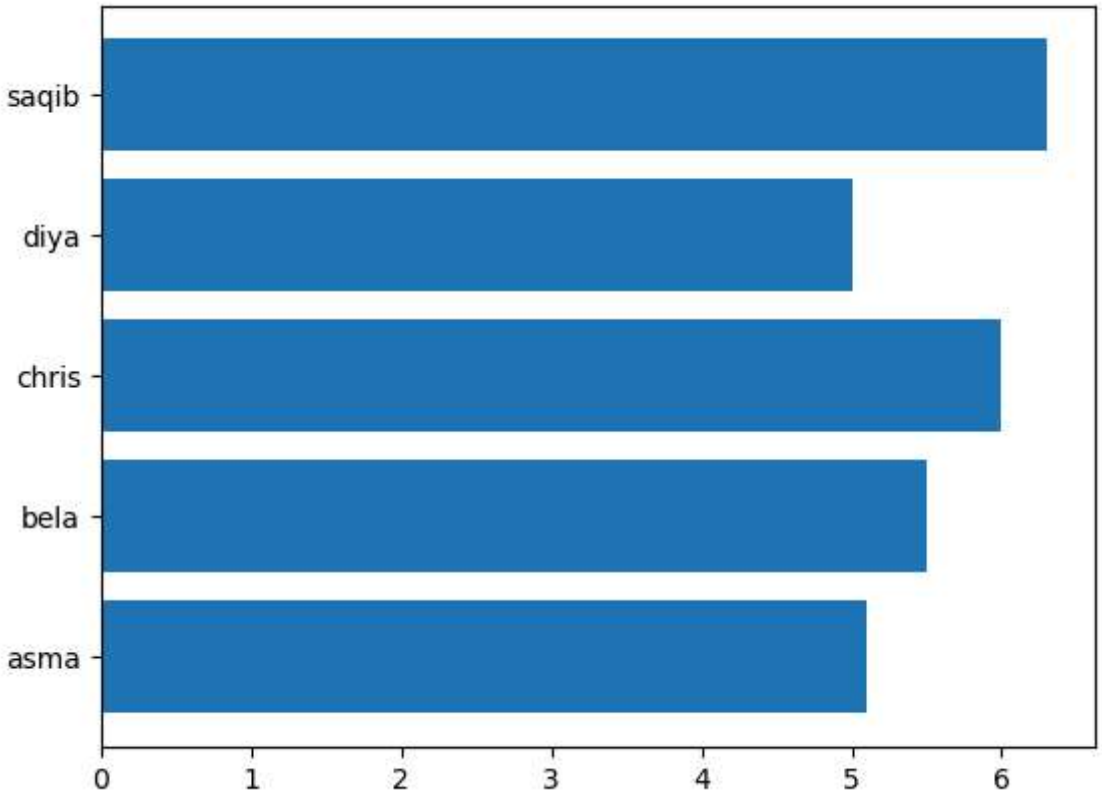
In [114…

```
# creating the horizontal bar chart by using barh function

info = ['gold','silver','bronze','total']
australia = [10,20,30,40]
plt.xlabel("info")
plt.ylabel("australia")
plt.barh(australia,info)
plt.show()
```



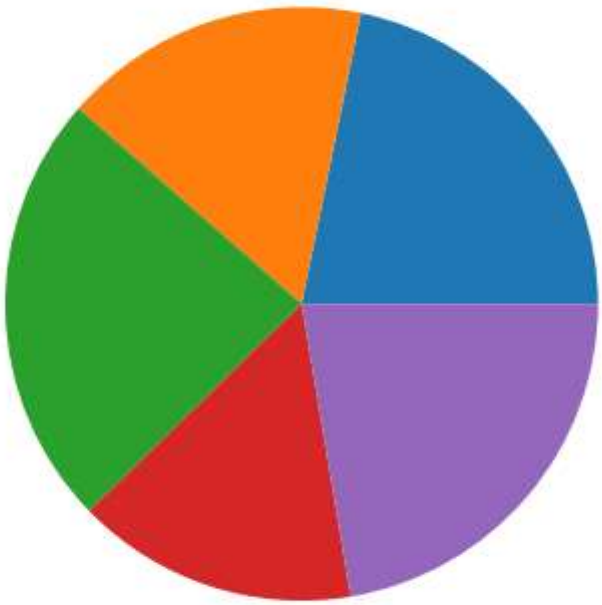
In [119…

```
height = [5.1,5.5,6,5,6.3]
names = ['asma','bela','chris','diya','saqib']

plt.barh(names,height)
plt.show()
```



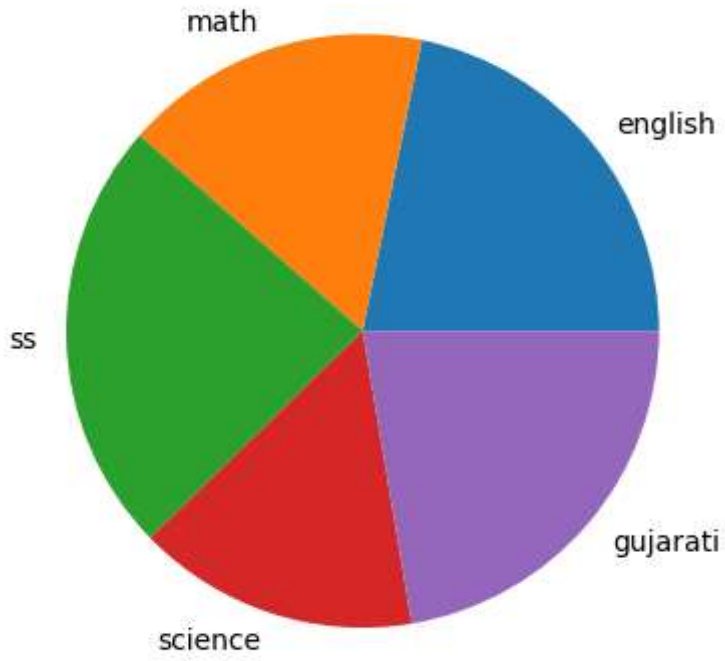
In [121…
```
# creating the pie charts using pie() function

marks = [78,60,85,55,80]
plt.pie(marks)
```

Out[121…
```
([<matplotlib.patches.Wedge at 0x29db8dcfa40>,
  <matplotlib.patches.Wedge at 0x29db7ed2a80>,
  <matplotlib.patches.Wedge at 0x29db90205f0>,
  <matplotlib.patches.Wedge at 0x29db7ecc6b0>,
  <matplotlib.patches.Wedge at 0x29db7ecd040>],
 [Text(0.852221984691252, 0.6954981587386868, ''),
  Text(-0.35091621602580375, 1.0425247284022243, ''),
  Text(-1.0996188340089827, -0.02895548122074864, ''),
  Text(-0.3417542484541052, -1.0455639787519317, ''),
  Text(0.839884807753301, -0.7103474570273344, '')])
```



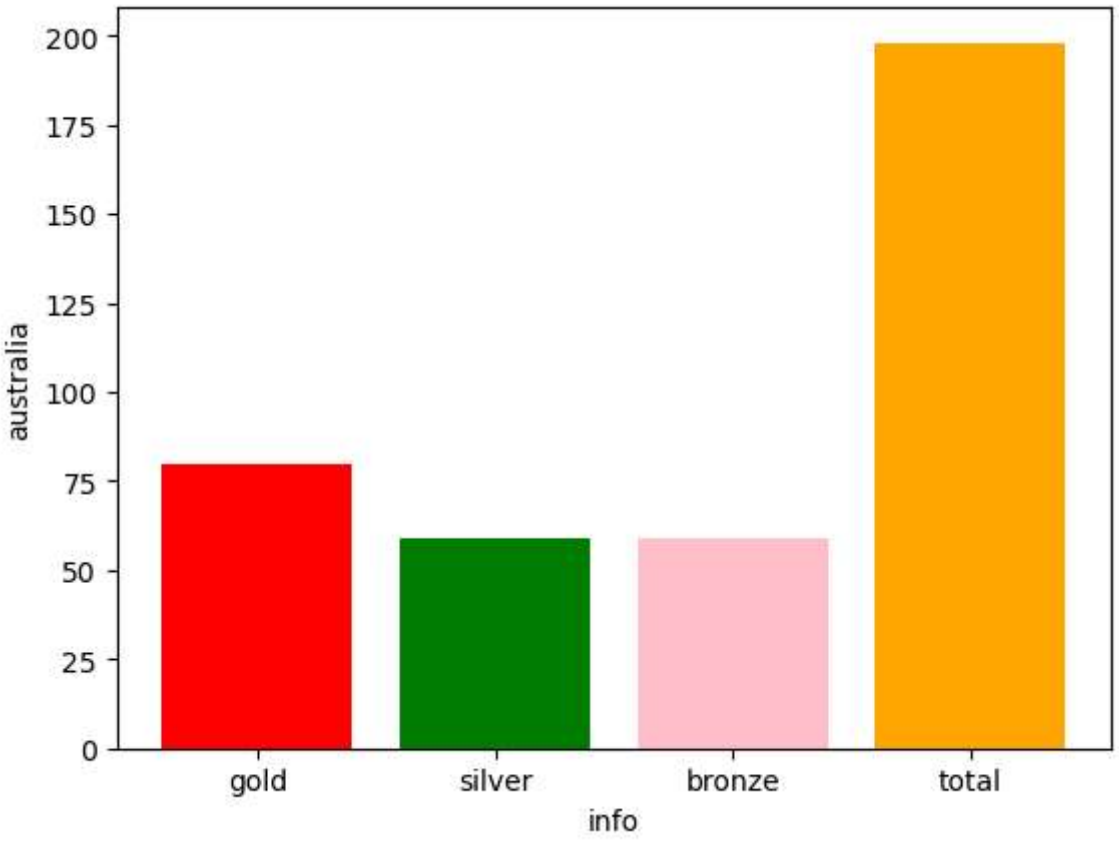
In [129…
```
# we can give the labels for each area by using labels = list or tuple

subjects = ['english','math','ss','science','gujarati']
marks = [78,60,85,55,80]
plt.pie(marks,labels = subjects) # makesure there is lables not the label otherwise
```

Out[129…
```
([<matplotlib.patches.Wedge at 0x29db8f82210>,
  <matplotlib.patches.Wedge at 0x29db8f83e60>,
  <matplotlib.patches.Wedge at 0x29db8f81b20>,
  <matplotlib.patches.Wedge at 0x29db8f825a0>,
  <matplotlib.patches.Wedge at 0x29db8f82900>],
 [Text(0.852221984691252, 0.6954981587386868, 'english'),
  Text(-0.35091621602580375, 1.0425247284022243, 'math'),
  Text(-1.0996188340089827, -0.02895548122074864, 'ss'),
  Text(-0.3417542484541052, -1.0455639787519317, 'science'),
  Text(0.839884807753301, -0.7103474570273344, 'gujarati')])
```



In [131…

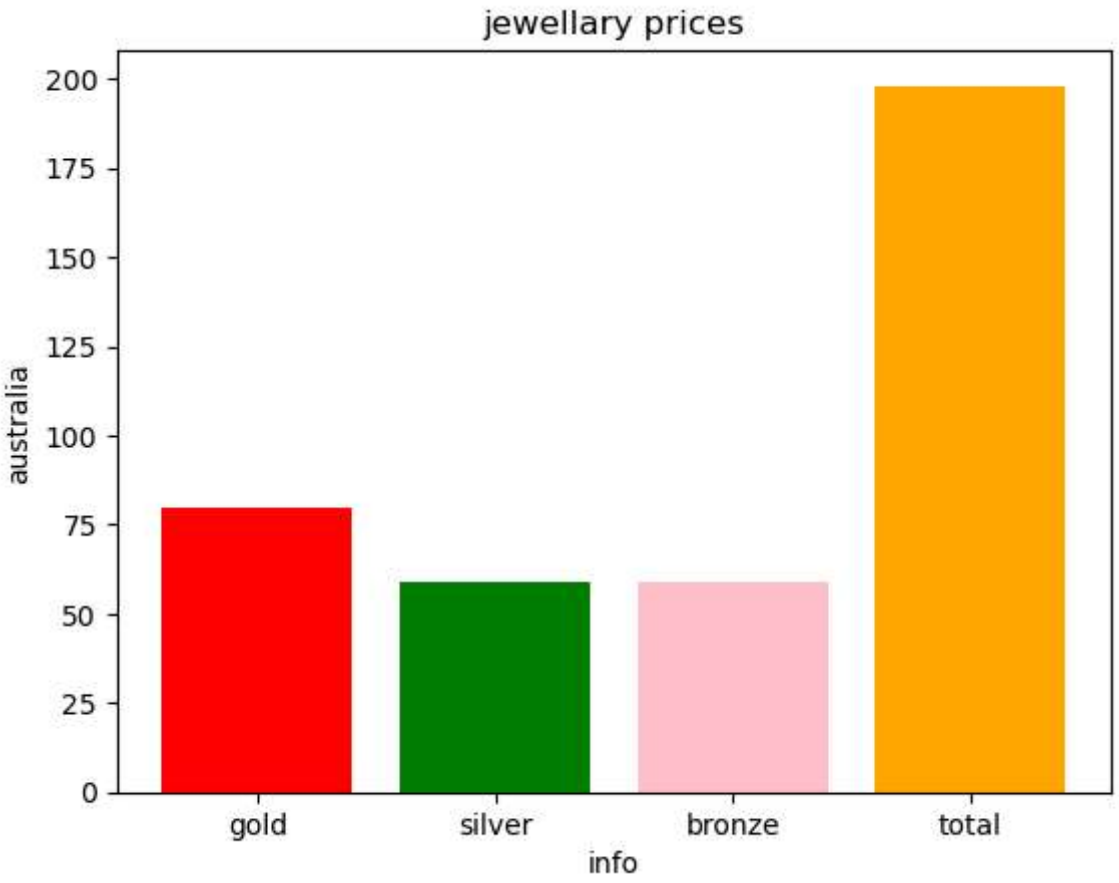
```
# anatomy of the chart in the simple words we can extracts the details of the chart
# for the diffrent bars we can pass multiple color by using list or tuple.

color = ['red','green','pink','orange']

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,color = color)
plt.figure(figsize = (50,50)) # you can set your own figure size
plt.show()
```




```
<Figure size 5000x5000 with 0 Axes>
```

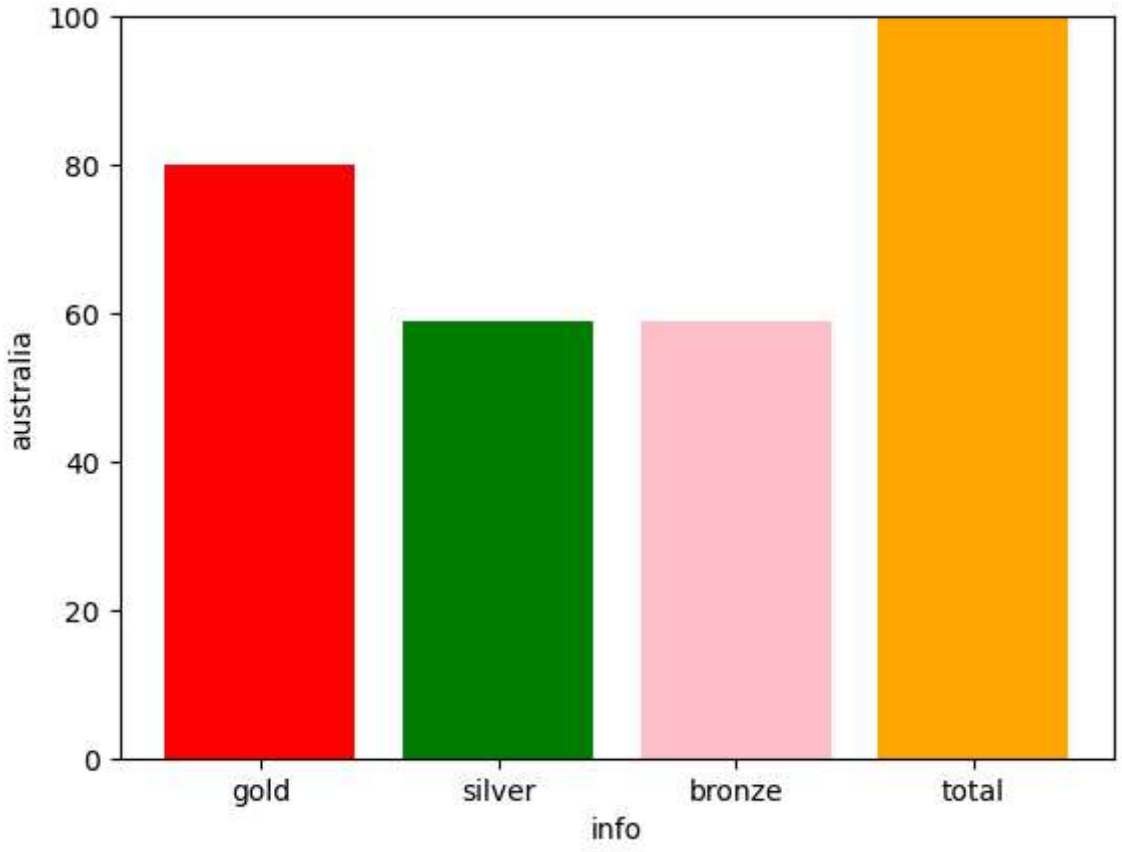
In [135…

```
# we can add the chart title by using title function

# for the diffrent bars we can pass multiple color by using list or tuple.

color = ['red','green','pink','orange']

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.title("jewellary prices") # here we defined the whole chart title
plt.xlabel("info")
plt.ylabel("australia")
plt.bar(info,australia,color = color)
plt.show()
```



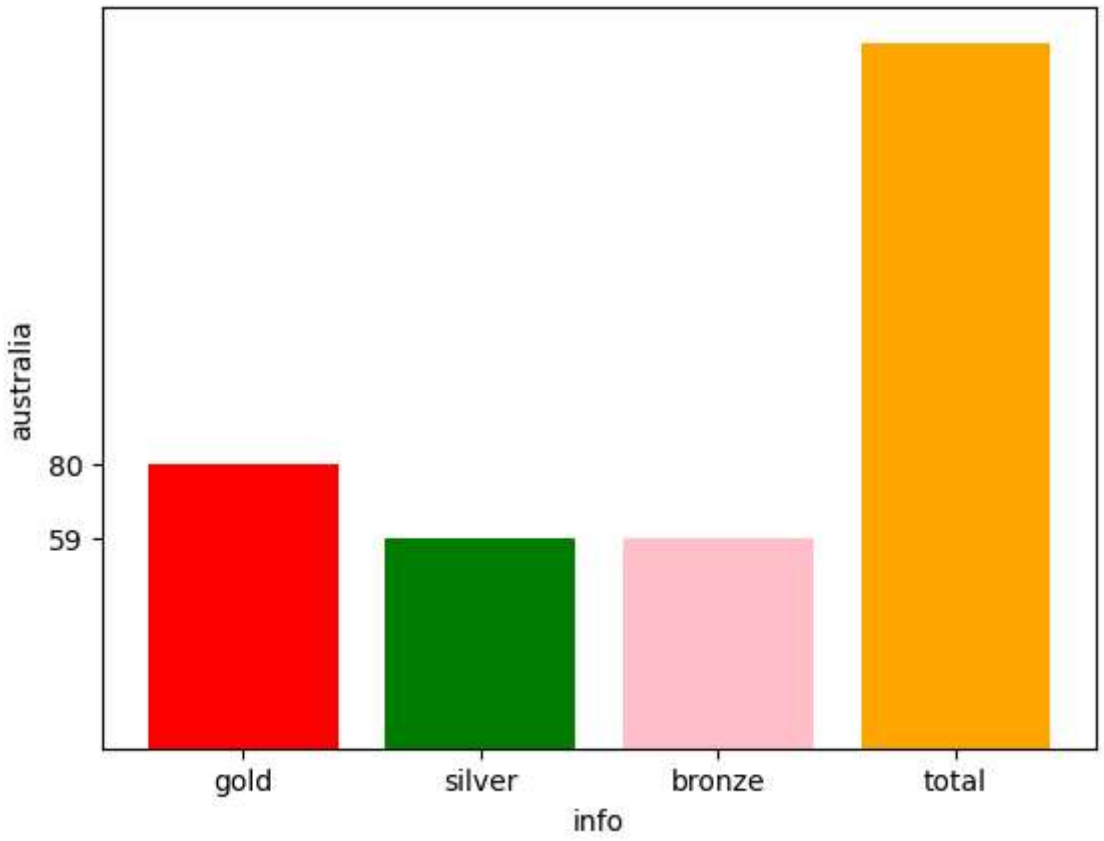
In [145…

```
# we can set the limits by using limits function
# for the diffrent bars we can pass multiple color by using list or tuple.

color = ['red','green','pink','orange']

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylim(0,100) # we can set the limits on x and y axis but makesure the the axis h
plt.ylabel("australia")
plt.bar(info,australia,color = color)
plt.show()
```



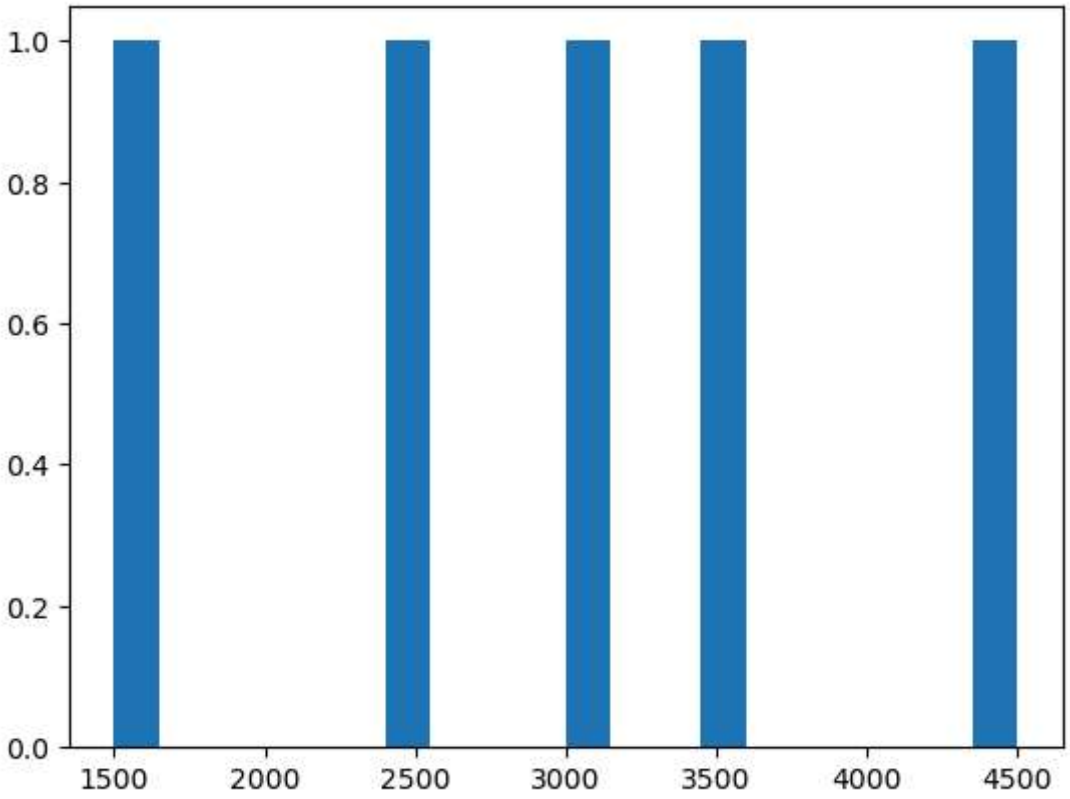
In [161…

```
# we can use ticks for each plots individual values by using this xticks or yticks
# for the diffrent bars we can pass multiple color by using list or tuple.

color = ['red','green','pink','orange']

info = ['gold','silver','bronze','total']
australia = [80,59,59,198]
plt.xlabel("info")
plt.ylabel("australia")
plt.yticks([80,59]) # that data is only visible on yticks makesure the given ticks
plt.bar(info,australia,color = color)
plt.savefig("multibar.pdf") # save the figure in different format
plt.show()
```



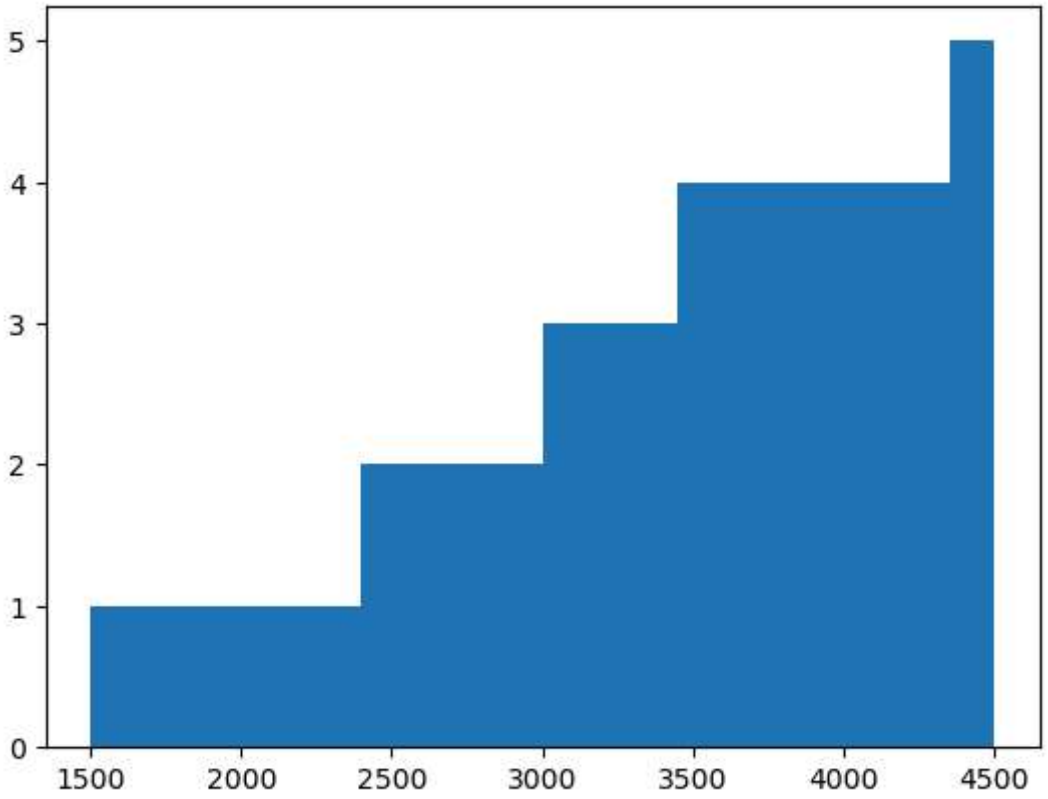
In [166…

```
# now we create a histogram by using hist() function

a = [2500,3000,1500,4500,3500]

plt.hist(a,bins = 20) # bins will be created on the basis on the a list of data fro
plt.show()
```



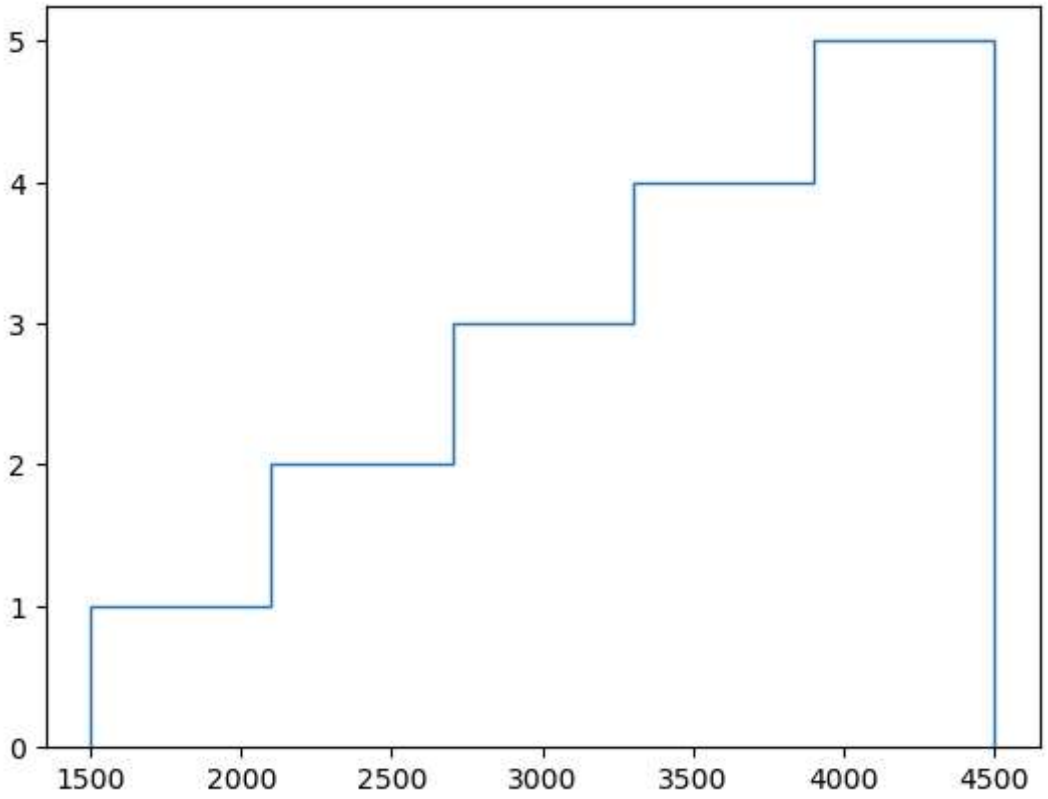
In [170…

```
# now we create a histogram by using hist() function
# cumulative histogram by using cumulative = True
a = [2500,3000,1500,4500,3500]

plt.hist(a,bins = 20,cumulative = True) # bins will be created on the basis on the
plt.show()
```



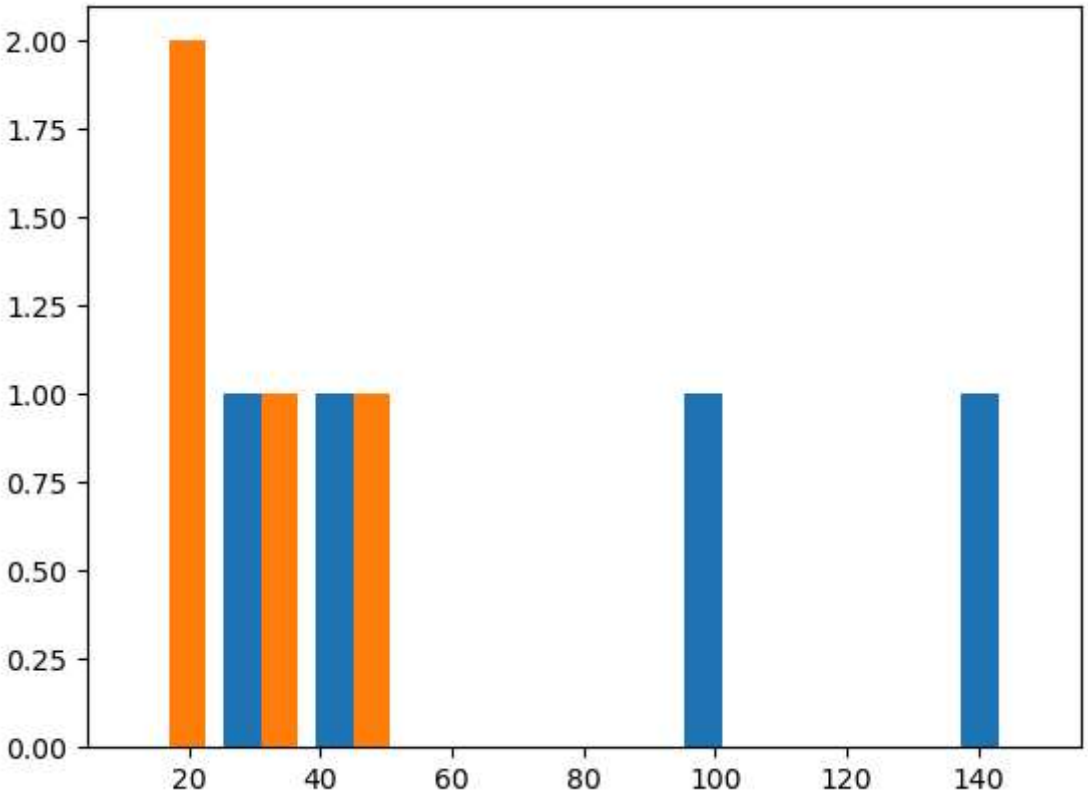
In [176…

```
# now we create a histogram by using hist() function
# cumulative histogram by using cumulative = True
#histogram type is histtype = step then inner color will be removed
a = [2500,3000,1500,4500,3500]

plt.hist(a,bins = 5,cumulative = True,histtype = "step") # bins will be created on
plt.show()
```



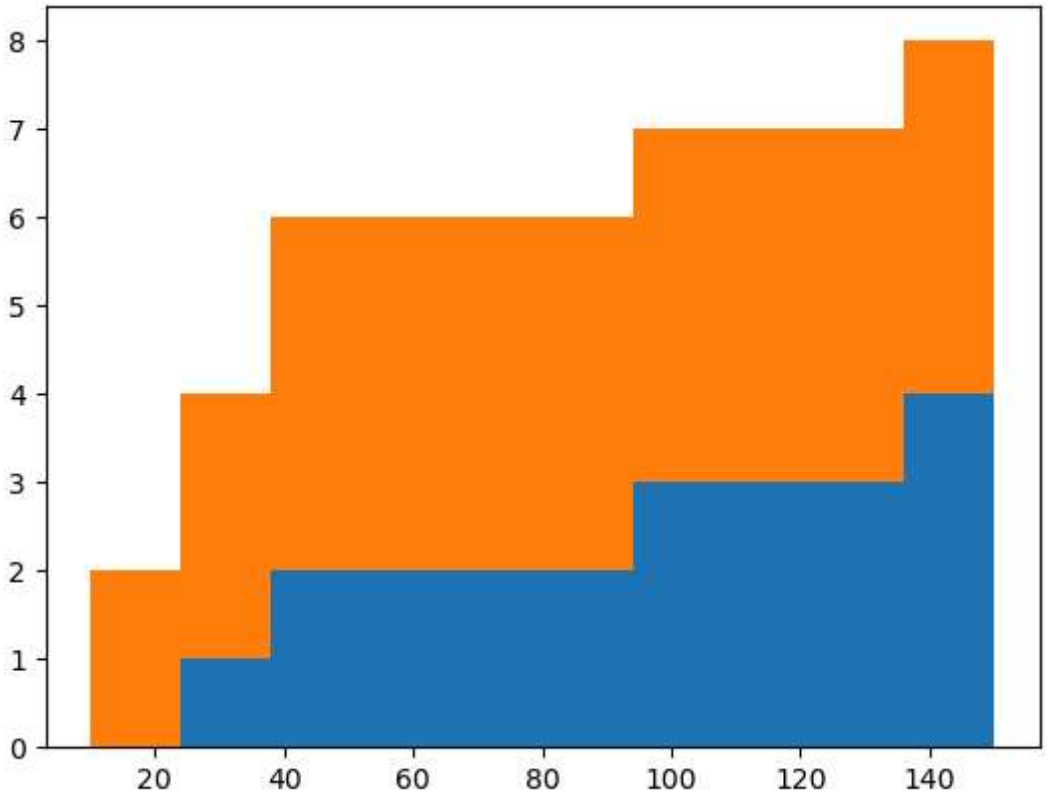
In [182…
```
# for ploting we can use ndarray

x = [25,50,100,150]
y = [10,20,30,40]

plt.hist([x,y]) # if you forgot to specify the bins then hist will auto decide the
```

Out[182…
```
(array([[0., 1., 1., 0., 0., 0., 1., 0., 0., 1.],
        [2., 1., 1., 0., 0., 0., 0., 0., 0., 0.]]),
 array([ 10.,  24.,  38.,  52.,  66.,  80.,  94., 108., 122., 136., 150.]),
 <a list of 2 BarContainer objects>)
```



In [188…
```
# for ploting we can use ndarray

x = [25,50,100,150]
y = [10,20,30,40]

plt.hist([x,y],histtype = "barstacked",cumulative = True) # if you forgot to specif
```

Out[188…
```
(array([[0., 1., 2., 2., 2., 2., 3., 3., 3., 4.],
        [2., 4., 6., 6., 6., 6., 7., 7., 7., 8.]]),
 array([ 10.,  24.,  38.,  52.,  66.,  80.,  94., 108., 122., 136., 150.]),
 <a list of 2 BarContainer objects>)
```

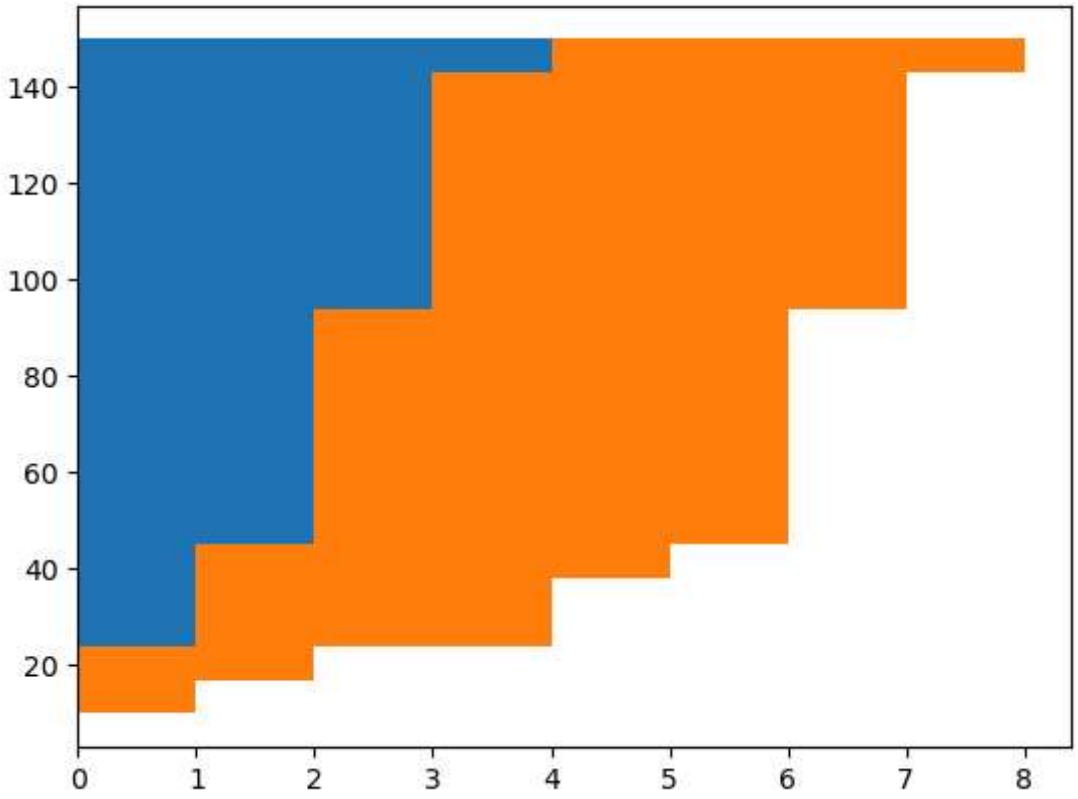


In [200…

```
# for ploting we can use ndarray
# for horizontal histogram use the orientation keyword like orientation = horizonta

x = [25,50,100,150]
y = [10,20,30,40]

plt.hist([x,y],bins = 20,histtype = "barstacked",cumulative = True,orientation = 'h
```

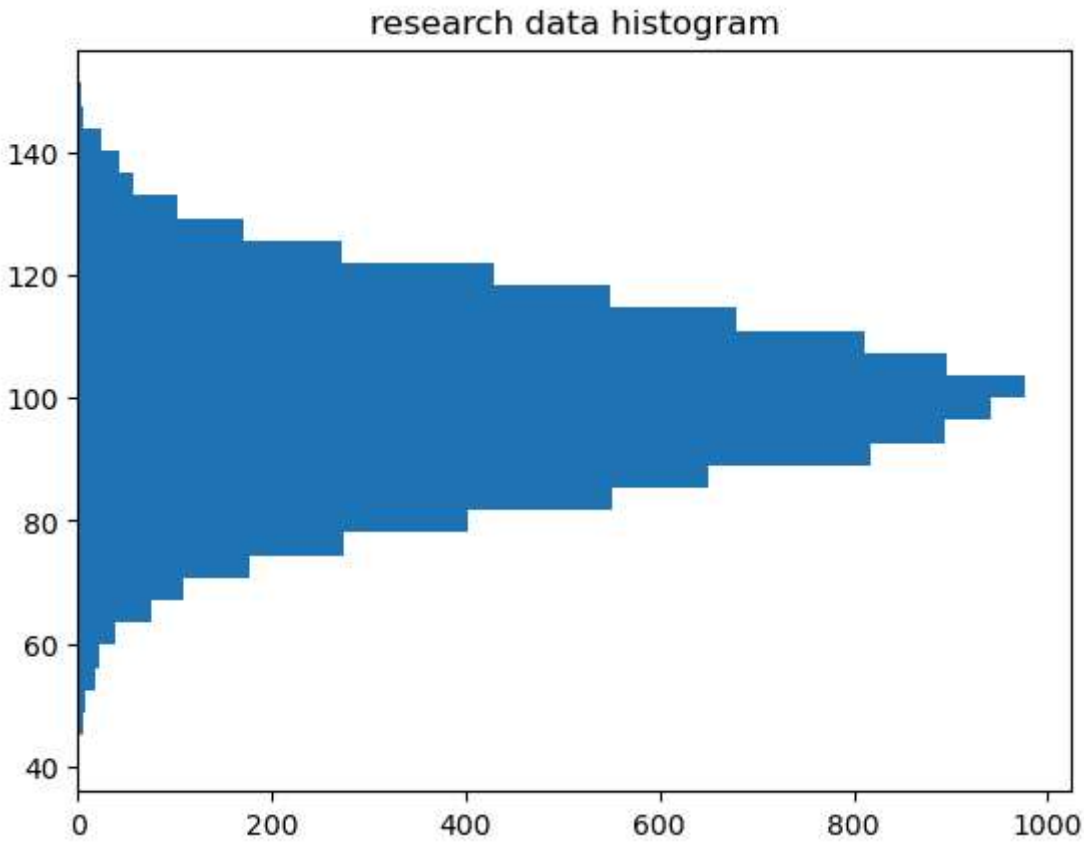
Out[200…

```
(array([[0., 0., 1., 1., 1., 2., 2., 2., 2., 2., 2., 2., 3., 3., 3., 3.,
         3., 3., 3., 4.],
        [1., 2., 4., 4., 5., 6., 6., 6., 6., 6., 6., 6., 7., 7., 7., 7.,
         7., 7., 7., 8.]]),
 array([ 10.,  17.,  24.,  31.,  38.,  45.,  52.,  59.,  66.,  73.,  80.,
         87.,  94., 101., 108., 115., 122., 129., 136., 143., 150.]),
 <a list of 2 BarContainer objects>)
```



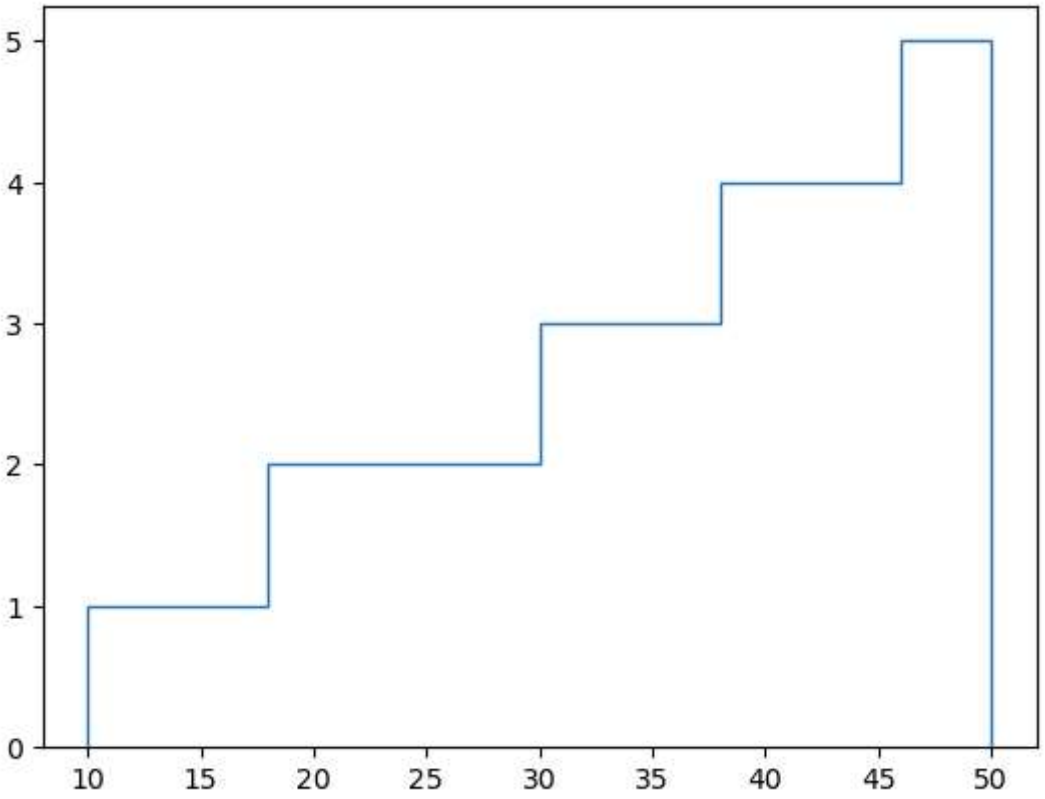
In [208…

```
mu = 100
sigma = 15
x = mu + sigma * np.random.randn(10000)

plt.hist(x,bins = 30,orientation = 'horizontal')
plt.title("research data histogram")
plt.show()
```



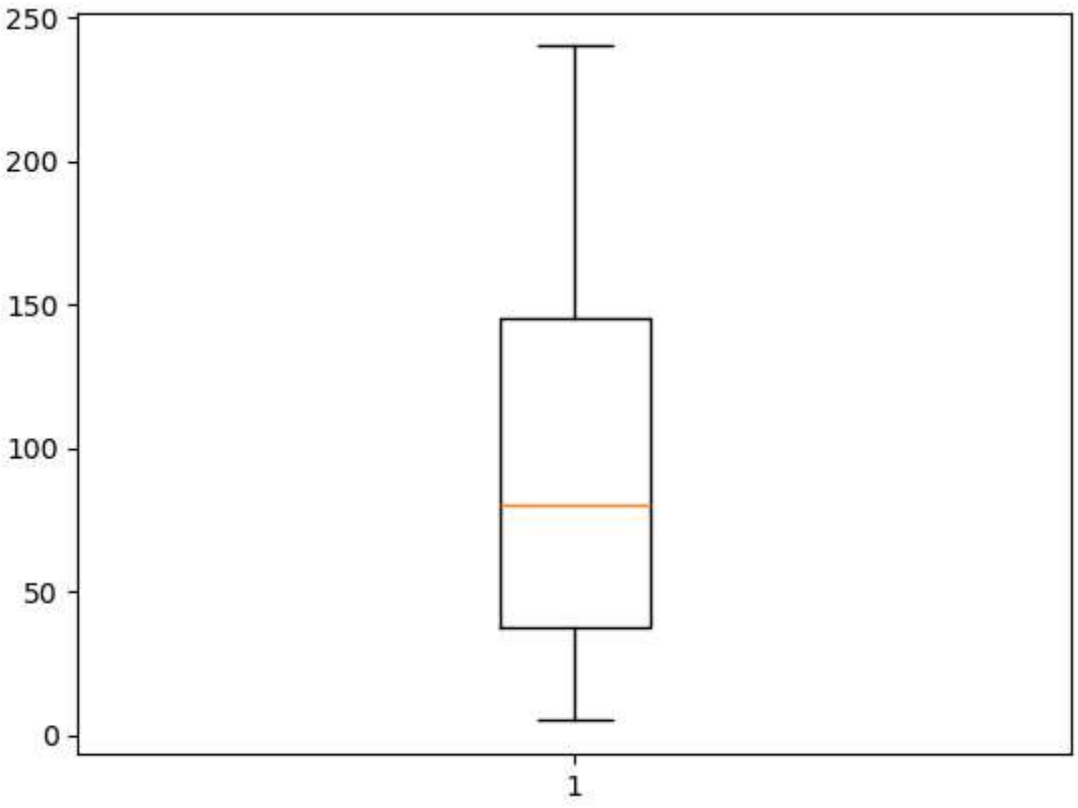
In [212…

```
# frequency polygon there is no function to create the frequency polygon in python
# use can create manualy or by using line chart

a = [10,20,30,40,50]

plt.hist(a,bins = 10,histtype = "step",cumulative = True)
plt.show()
```



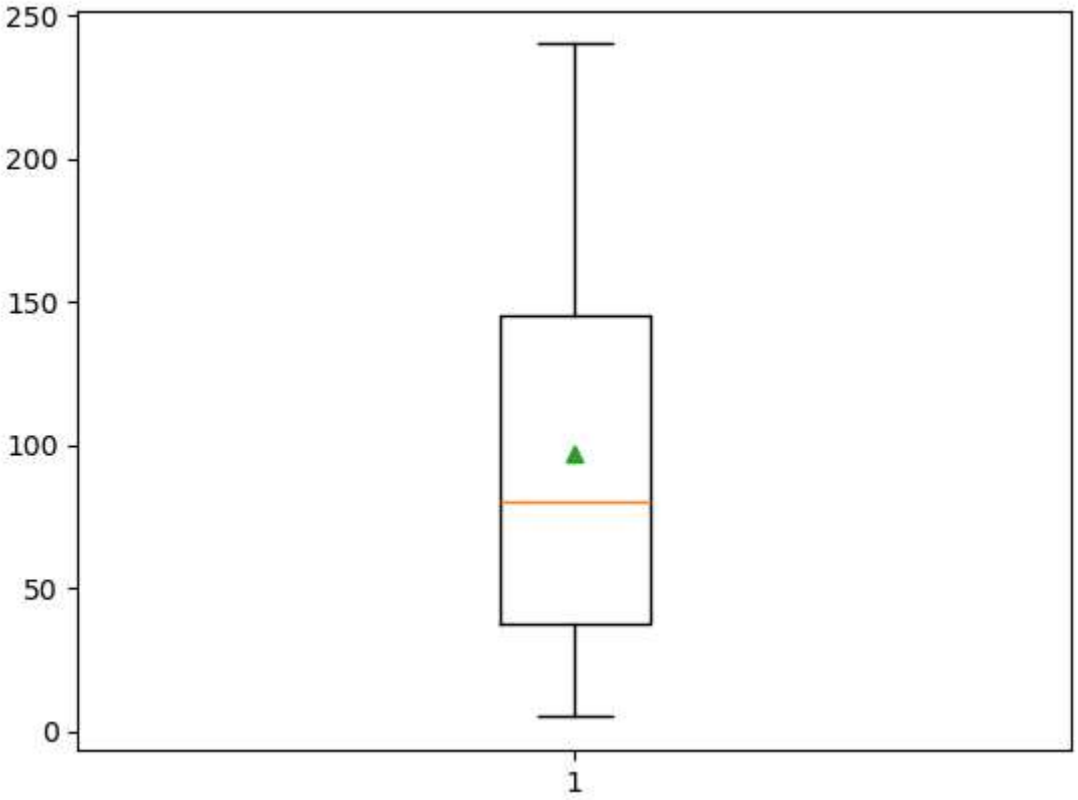
In [214…

```
# creating box plots by using boxplot() function

a = [5,20,30,45,60,80,100,140,150,200,240]
plt.boxplot(a)
plt.show()
```



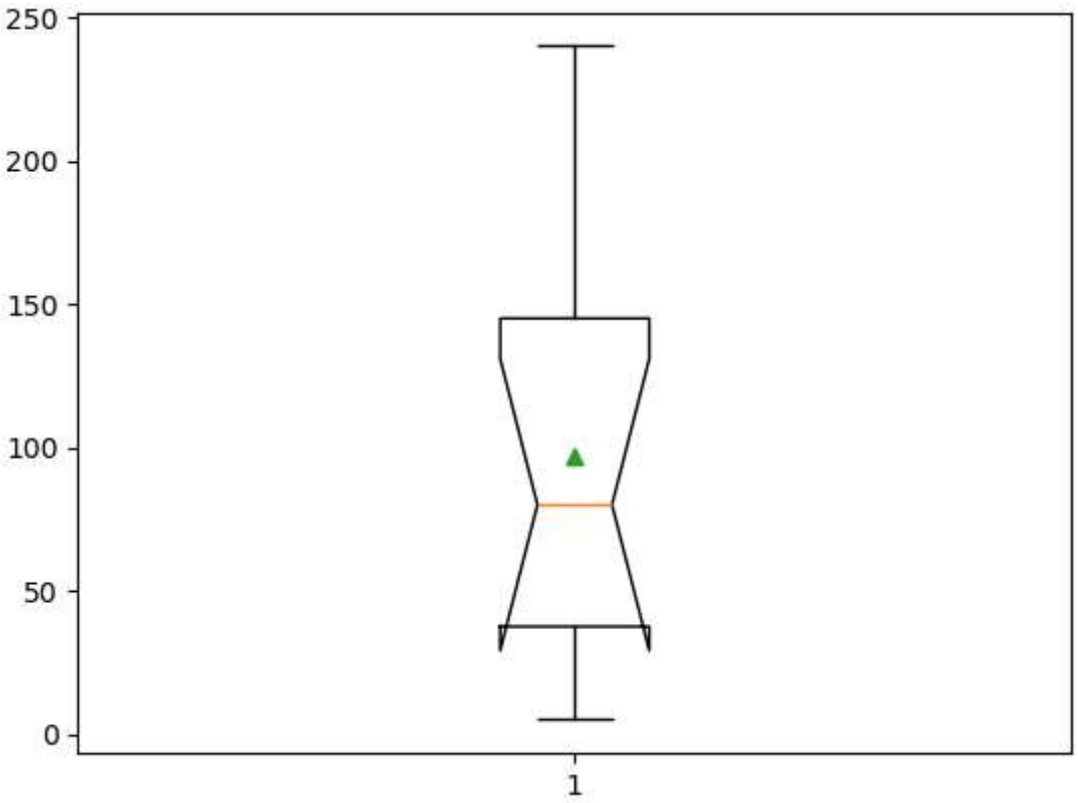
In [216…

```
# if you want to see the mean value just type showmeans = True
# creating box plots by using boxplot() function

a = [5,20,30,45,60,80,100,140,150,200,240]
plt.boxplot(a,showmeans = True)
plt.show()
```



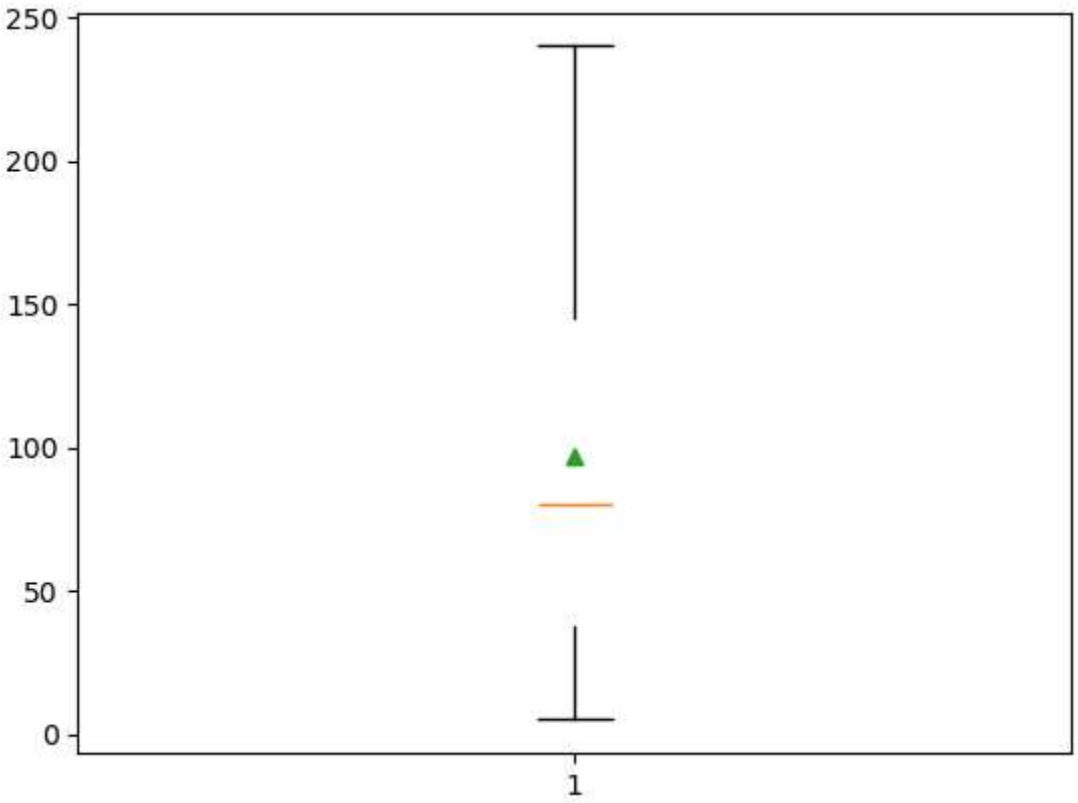
In [218…

```
# draw notched boxplot for the same notch = True
# if you want to see the mean value just type showmeans = True
# creating box plots by using boxplot() function

a = [5,20,30,45,60,80,100,140,150,200,240]
plt.boxplot(a,showmeans = True,notch = True)
plt.show()
```



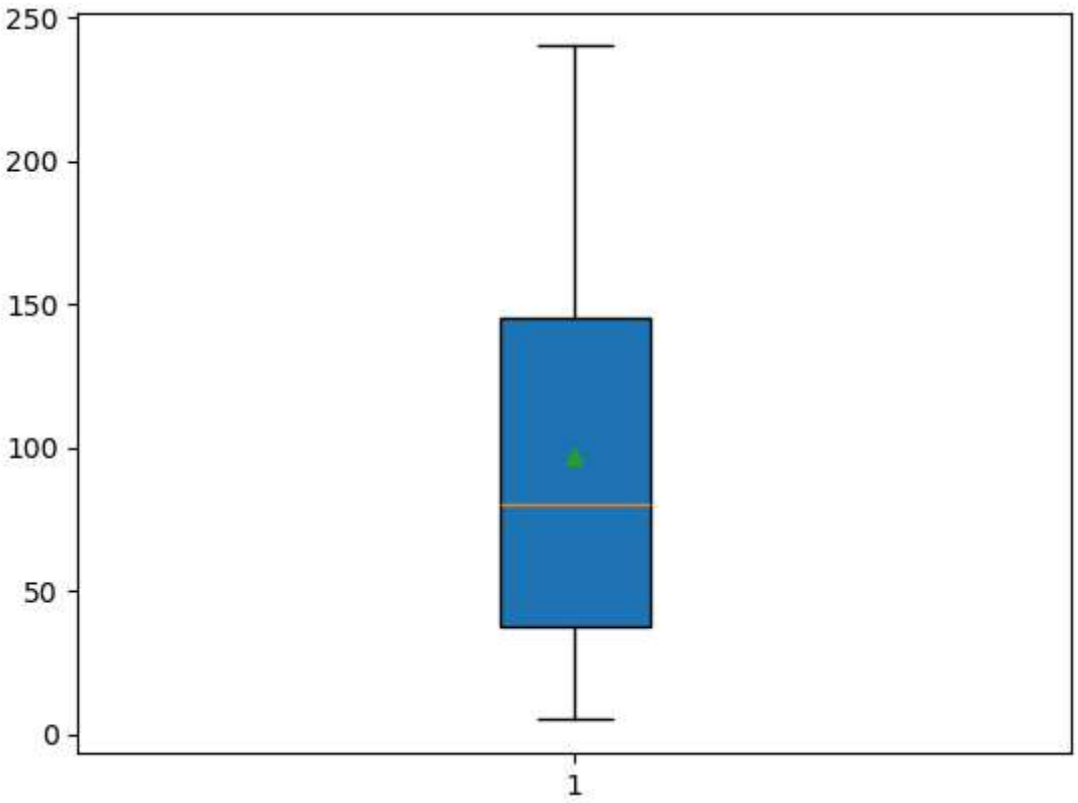
In [224…

```
# draw the boxplot data without the box
# draw notched boxplot for the same notch = True
# if you want to see the mean value just type showmeans = True
# creating box plots by using boxplot() function

a = [5,20,30,45,60,80,100,140,150,200,240]
plt.boxplot(a,showmeans = True,notch = True,showbox = False)
plt.show()
```



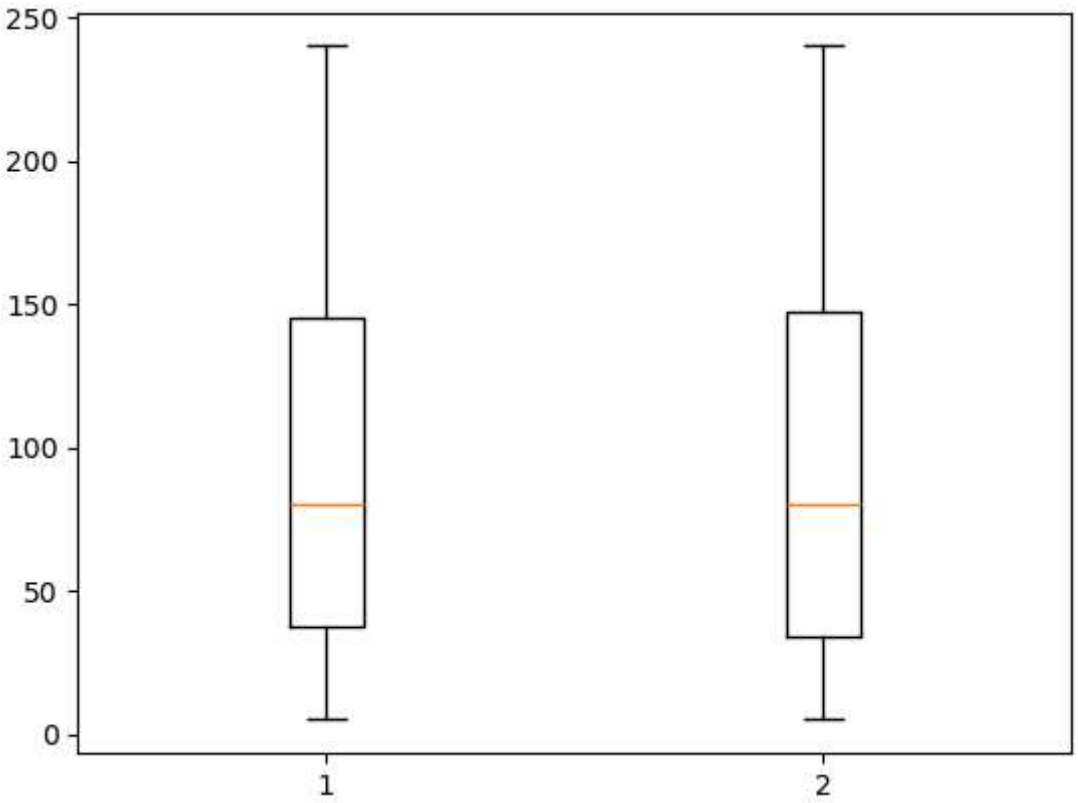
In [226…

```
# if you want to fill some color inside the box then use patch_artist = True
# if you want to see the mean value just type showmeans = True
# creating box plots by using boxplot() function

a = [5,20,30,45,60,80,100,140,150,200,240]
plt.boxplot(a,showmeans = True,patch_artist = True)
plt.show()
```



In [232…

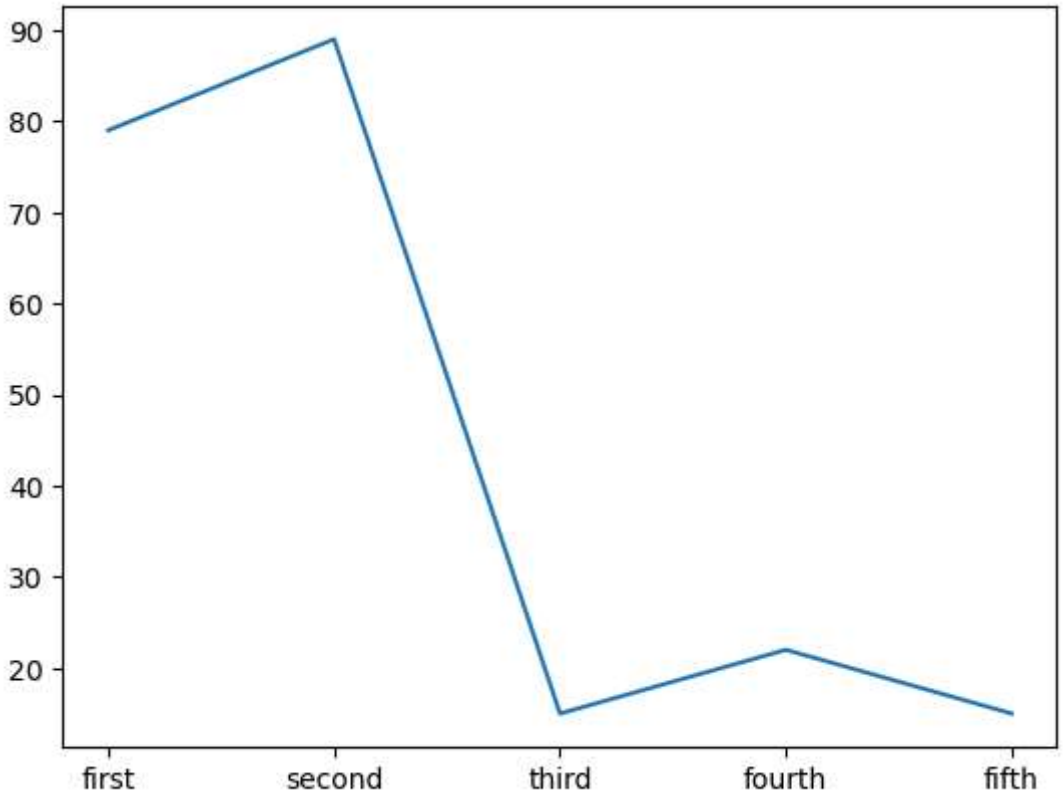
```
# for creating multiple box plot for the multiple values
# if you want to fill some color inside the box then use patch_artist = True
# if you want to see the mean value just type showmeans = True
# creating box plots by using boxplot() function

a = [5,20,30,45,60,80,100,140,150,200,240]
b = a * 2
plt.boxplot([a,b])
plt.show()
```



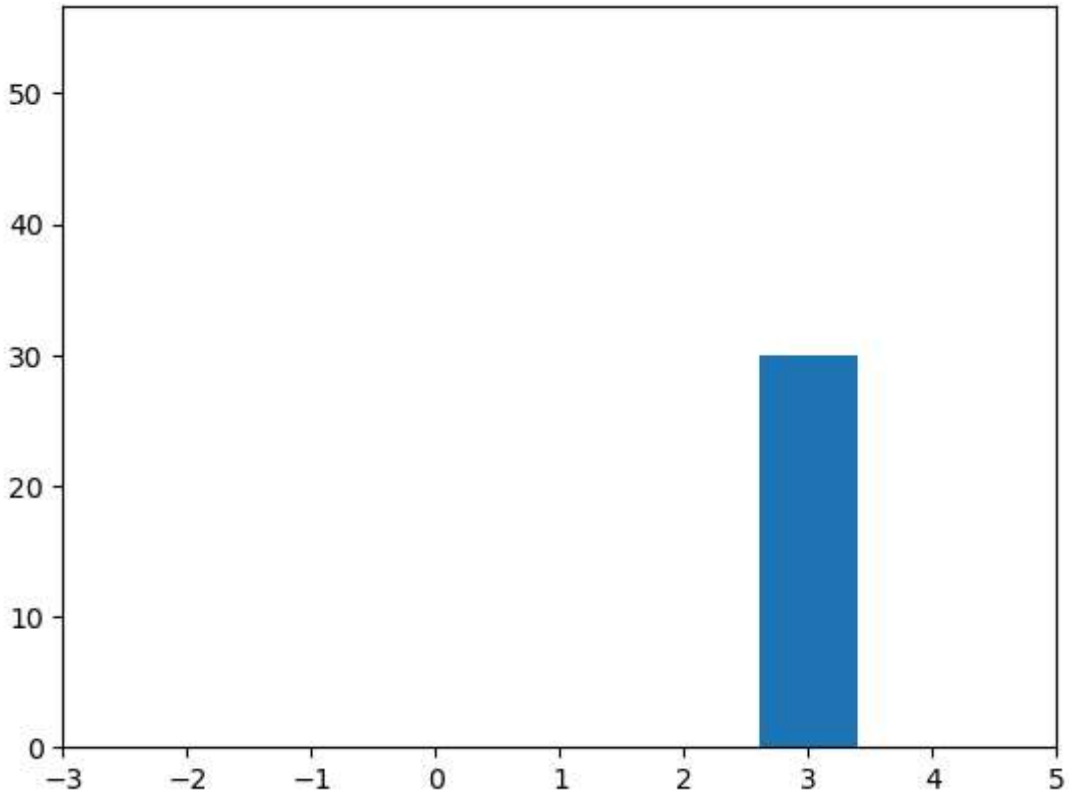
In [234…

```
# ploting data from dataframe
# we can give our own indexing for our easy understanding.

dic1 = {
    'student' : ['ruchika','neha','mark','gurjyot','jamal'],
    'marks' : [79,89,15,22,15]
}
dtf2 = pd.DataFrame(dic1,index = ['first','second','third','fourth','fifth'])
print(dtf2)
plt.plot(dtf2.marks)
plt.show()
```

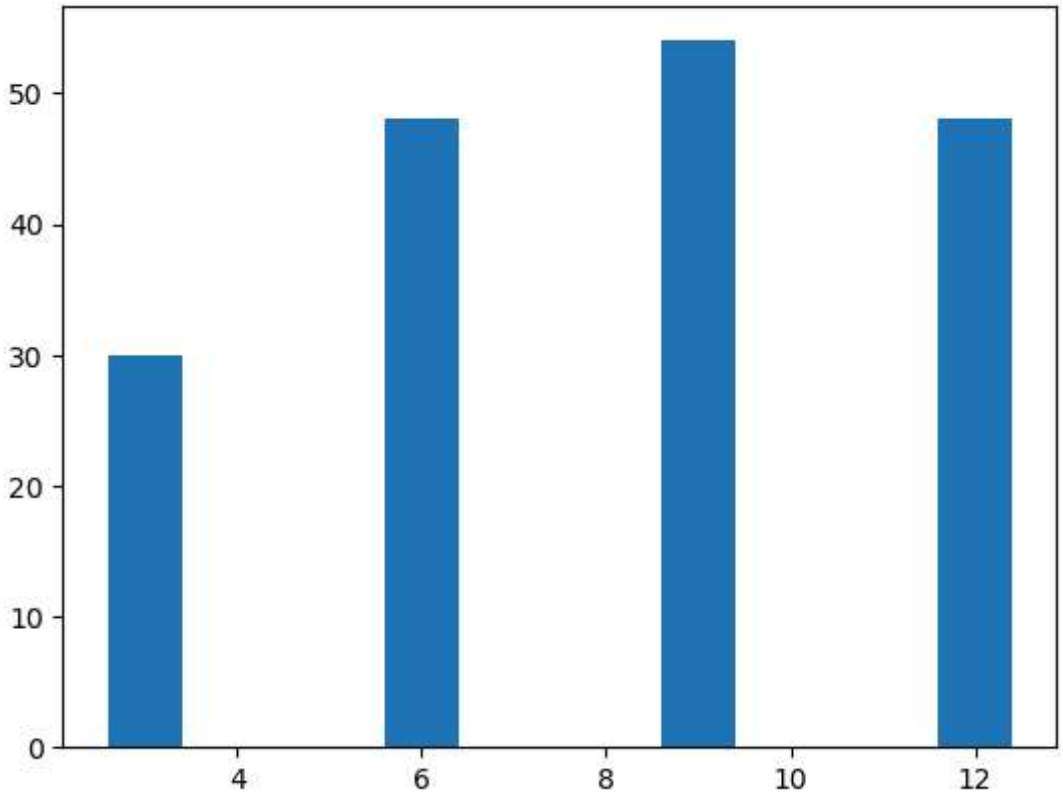
```
        student  marks
first   ruchika     79
second     neha     89
third      mark     15
fourth  gurjyot     22
fifth     jamal     15
```



In [5]:
```
#q8

a = [3,6,9,12]
b = [30,48,54,48]
plt.xlim(-3,5)
plt.bar(a,b)
plt.show() # 6,9,12 are remove during the ploting cause the x-limit is -3 to 5 only
```



In [7]:
```
#q9
# we remove the x-limit so that all the bars are visible on x-axis
a = [3,6,9,12]
b = [30,48,54,48]
plt.bar(a,b)
plt.show()
```



In [ ]: