In [7]:

```
# Example1:
import numpy as np
demo = np.array([[30,75,70],[80,90,20],[50,95,60]])
print(demo)
print()
print(np.mean(demo))
print()
print(np.median(demo))
print()
```

```
[[30 75 70]
 [80 90 20]
 [50 95 60]]

63.333333333333336

70.0
```

In [3]:

```
#  Q1. Write a Python program to find the maximum and minimum value of a give
# array.
import numpy as np
data = np.array([[0,1],[2,3]])
print(data)
print('Maximum value of above flattend array:')
print(data.max())
print('Minimum value of above flattend array:')
print(data.min())
```

```
[[0 1]
 [2 3]]
Maximum value of above flattend array:
3
Minimum value of above flattend array:
0
```

In [5]:

```
#  Q2. Write a python program to compute Euclidian Distance between two data
# dataset. [Hint: Use linalgo.norm function from NumPy]
import pandas as pd
import numpy as np
x = pd.Series([1, 2, 3, 4, 5])
y = pd.Series([6, 7, 8, 9, 10])
dist = (np.linalg.norm(x-y))
print("Series 1:")
print(x)
print("Series 2:")
print(y)
print("Euclidean distance between two series is:", dist.round(2))
```

```
Series 1:
0    1
1    2
2    3
3    4
4    5
dtype: int64
Series 2:
0     6
1     7
2     8
3     9
4    10
dtype: int64
Euclidean distance between two series is: 11.18
```

In [6]:

```
#  Q3. Create one dataframe of data values. Find out mean, range and IQR for
import pandas as pd
import numpy as np
data = np.array([24, 29, 20, 22, 24, 26, 27, 30, 20, 31, 26, 38, 44,
47])
print(data)
print('Mean:',np.mean(data)) #average
print('Median :',np.median(data)) #middle value
print('Range :',np.max(data)-np.min(data))
q3,q1 = np.percentile(data,[75,25])
iqr = q3 - q1
print('iqr:',iqr)
```

```
[24 29 20 22 24 26 27 30 20 31 26 38 44 47]
Mean: 29.142857142857142
Median : 26.5
Range : 27
iqr: 6.75
```

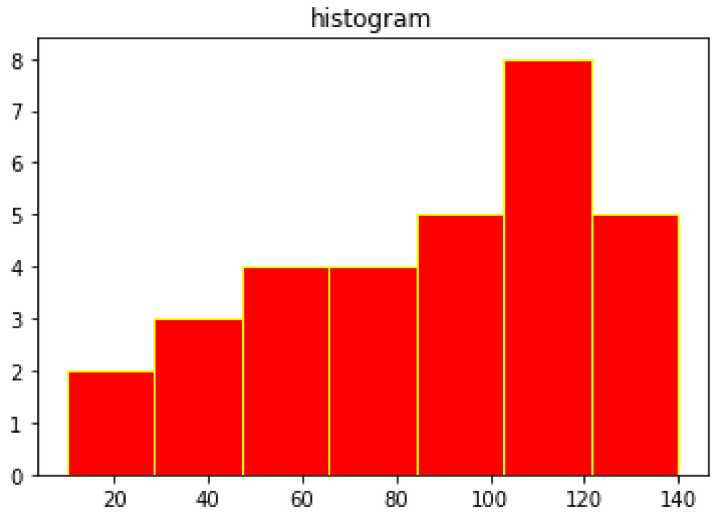
In [10]:

```
#  Q4. Write a python program to compute sum of Manhattan distance between al
# points.
def distancesum (x, y, n):
    sum = 0
    for i in range(n):
        for j in range(i+1,n):
            sum += (abs(x[i] - x[j]) +  abs(y[i] - y[j]))
    return sum
x = [ -2, 2, 3, 2 ]
y = [ 4, 7, 5, 3 ]
n = len(x)
print(distancesum(x, y, n))
```

```
28
```

In [11]:

```
# Q5.  Write a NumPy program to compute the histogram of nums against the bin
# width of bin = (max-min)/bin
# bin is number of towers in histogram
import matplotlib.pyplot as plt
data = [72, 33, 46, 61, 57, 68, 72,
120, 124, 61, 92, 117, 123,
109, 126, 104, 11, 111, 89,
102, 91, 80, 121, 110, 90, 107,
10, 29, 140, 139, 51]
plt.title("histogram")
plt.hist(data, bins=7, edgecolor="yellow", color="red")
plt.show()
```



In [14]:

```
# Q6. Create a dataframe for students' information such name, graduation perc
# Display average age of students, average of graduation percentage. And, als
# all basic statistics of data. (Hint: use describe()).
import pandas as pd
df = pd.DataFrame(columns=['Name','Age','Percentage'])
df.loc[0] = ['Aishwarya',20,99.00]
df.loc[1] = ['Mona',20,80.00]
df.loc[2] = ['Sayali',17,90.00]
df.loc[3] = ['Sita',18,95.00]
df.loc[4] = ['Sona',21,80.00]
df.loc[5] = ['Rita',15,85.00]
df.loc[6] = ['Rutuja',16,77.00]
df.loc[7] = ['Siddhi',17,78.00]
df.loc[8] = ['Riya',20,91.00]
df.loc[9] = ['Pinky',21,75.00]
df.loc[10] = ['Soniya',17,68.00]
print('\nDataFrame:\n')
print(df)
import scipy.stats as s
print('Mean of Age:',s.tmean(df['Age']).round(2))
print('Mean of Percentage:',s.tmean(df['Percentage']).round(2))
print(df.describe())
```

```
DataFrame:

         Name Age  Percentage
0   Aishwarya  20        99.0
1        Mona  20        80.0
2      Sayali  17        90.0
3        Sita  18        95.0
4        Sona  21        80.0
5        Rita  15        85.0
6      Rutuja  16        77.0
7      Siddhi  17        78.0
8        Riya  20        91.0
9       Pinky  21        75.0
10     Soniya  17        68.0
Mean of Age: 18.36
Mean of Percentage: 83.45
       Percentage
count   11.000000
mean    83.454545
std      9.395357
min     68.000000
25%     77.500000
50%     80.000000
75%     90.500000
max     99.000000
```

In [1]:

```
# Q1. Download iris dataset file. Read this csv file using read_csv() functio
# from entire dataset. Display maximum and minimum values of all numeric attr
import pandas as pd
df=pd.read_csv('iris.data')
print('Reading random 20 rows: ')
print(df.sample(20))
print('Maximum sepal length: ',df["sepal length"].max())
print('Minimum sepal length: ',df["sepal length"].min())
print('Maximum sepal width: ',df["sepal width"].max())
print('Minimum sepal width: ',df["sepal width"].min())
print('Maximum petal length: ',df["petal length"].max())
print('Minimum petal length: ',df["petal length"].min())
print('Maximum petal width: ',df["petal width"].max())
print('Minimum petal width: ',df["petal width"].min())
```

```
Reading random 20 rows:
     sepal length  sepal width  petal length  petal width            class
82            5.8          2.7           3.9          1.2  Iris-versicolor
126           6.2          2.8           4.8          1.8   Iris-virginica
107           7.3          2.9           6.3          1.8   Iris-virginica
65            6.7          3.1           4.4          1.4  Iris-versicolor
24            4.8          3.4           1.9          0.2      Iris-setosa
92            5.8          2.6           4.0          1.2  Iris-versicolor
57            4.9          2.4           3.3          1.0  Iris-versicolor
87            6.3          2.3           4.4          1.3  Iris-versicolor
55            5.7          2.8           4.5          1.3  Iris-versicolor
79            5.7          2.6           3.5          1.0  Iris-versicolor
29            4.7          3.2           1.6          0.2      Iris-setosa
105           7.6          3.0           6.6          2.1   Iris-virginica
145           6.7          3.0           5.2          2.3   Iris-virginica
18            5.7          3.8           1.7          0.3      Iris-setosa
121           5.6          2.8           4.9          2.0   Iris-virginica
53            5.5          2.3           4.0          1.3  Iris-versicolor
21            5.1          3.7           1.5          0.4      Iris-setosa
37            4.9          3.1           1.5          0.1      Iris-setosa
11            4.8          3.4           1.6          0.2      Iris-setosa
101           5.8          2.7           5.1          1.9   Iris-virginica
Maximum sepal length:  7.9
Minimum sepal length:  4.3
Maximum sepal width:  4.4
Minimum sepal width:  2.0
Maximum petal length:  6.9
Minimum petal length:  1.0
Maximum petal width:  2.5
Minimum petal width:  0.1
```

In [2]:
```
# Q2. Continue with above dataset, find number of records for each distinct v
# attribute. Consider entire dataset and not the samples.
df.drop_duplicates(inplace = True)
print(df["class"].value_counts())
```

```
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: class, dtype: int64
```

In [3]:
```
# Q3. Display column-wise mean, and median for iris dataset from Q.4 (Hint: U
# median() functions of pandas dataframe.
import numpy as np
print('Mean:',np.mean(df['sepal length']))
print('Median:',np.median(df['sepal length']))
print('Mean:',np.mean(df['sepal width']))
print('Median:',np.median(df['sepal width']))
print('Mean:',np.mean(df['petal length']))
print('Median:',np.median(df['petal length']))
print('Mean:',np.mean(df['petal width']))
print('Median:',np.median(df['petal width']))
```

```
Mean: 5.8564625850340155
Median: 5.8
Mean: 3.055782312925171
Median: 3.0
Mean: 3.78027210884354
Median: 4.4
Mean: 1.2088435374149666
Median: 1.3
```

In [10]:
```
# Q1. Write a python program to find Minkowskii Distance between two points.
# Minkowskii Distance is generalization of both the Euclidean distance and th
from math import *
from decimal import Decimal
def p_root(value, root):
    root_value = 1 / float(root)
    return round (Decimal(value) **
             Decimal(root_value), 3)

def minkowski_distance(x, y, p_value):
    return (p_root(sum(pow(abs(a-b), p_value)
            for a, b in zip(x, y)), p_value))
vector1 = [0, 2, 3, 4]
vector2 = [2, 4, 3, 7]
p = 3
print(minkowski_distance(vector1, vector2, p))
```

```
3.503
```

In [21]:

```
# Q2. Write a Python NumPy program to compute the weighted average along the
# axis of a given flattened array.
import numpy as np
a = np.array([[0,1,2],[3,4,5],[6,7,8]])
print("Original flattened array:")
print(a)
print("Weighted average along the specified axis of the above flattened array
print(np.average(a, axis=1, weights=[1./4, 2./4, 2./4]))
```

```
Original flattened array:
[[0 1 2]
 [3 4 5]
 [6 7 8]]
Weighted average along the specified axis of the above flattened array:
[1.2 4.2 7.2]
```

In [19]:

```
# Q3. Write a NumPy program to compute cross-correlation of two given arrays.
import numpy as np
x = np.array([0, 1, 3])
y = np.array([2, 4, 5])
print("\nOriginal array1:")
print(x)
print("\nOriginal array1:")
print(y)
print("\nCross-correlation of the said arrays:\n",np.cov(x, y))
```

```

Original array1:
[0 1 3]

Original array1:
[2 4 5]

Cross-correlation of the said arrays:
 [[2.33333333 2.16666667]
 [2.16666667 2.33333333]]
```

In [10]:

```
# Q4. Download any dataset from UCI (do not repeat it from set B). Read this
# read_csv() function. Describe the dataset using appropriate function. Displ
# value of numeric attribute. Check any data values are missing or not.
import pandas as pd
df = pd.read_csv('nursery.data')
print(df)
print("Describe dataset: \n")
print(df.describe())
# print("Mean: ",df.mean())
print("Total no. of missing values: ",df.isnull().sum().sum())
```

```
              parents  has_nurs  form    children     housing        financ
e  \
usual          proper  complete     1  convenient  convenient        nonpro
b
usual          proper  complete     1  convenient  convenient        nonpro
b
usual          proper  complete     1  convenient  convenient        nonpro
b
usual          proper  complete     1  convenient  convenient  slightly_pro
b
usual          proper  complete     1  convenient  convenient  slightly_pro
b
...               ...       ...   ...         ...         ...
...
great_pret  very_crit    foster  more    critical      inconv  slightly_pro
b
great_pret  very_crit    foster  more    critical      inconv  slightly_pro
b
great_pret  very_crit    foster  more    critical      inconv    problemati
c
great_pret  very_crit    foster  more    critical      inconv    problemati
c
great_pret  very_crit    foster  more    critical      inconv    problemati
c

                social      health
usual       recommended   recommend
usual          priority    priority
usual         not_recom   not_recom
usual       recommended   recommend
usual          priority    priority
...                 ...         ...
great_pret     priority  spec_prior
great_pret    not_recom   not_recom
great_pret  recommended  spec_prior
great_pret     priority  spec_prior
great_pret    not_recom   not_recom

[12960 rows x 8 columns]
Describe dataset:

         parents    has_nurs   form  children housing        finance  \
count      12960       12960  12960     12960   12960          12960
unique         5           4      4         3       2              3
top     critical  incomplete   more  critical  inconv  slightly_prob
```

```
freq        2592        3240   3240      4320    6480          4320

           social     health
count       12960      12960
unique          3          5
top     not_recom  not_recom
freq         4320       4320
Total no. of missing values:  0
```

In [11]:

```
# Q5. Download nursery dataset from UCI. Split dataset on any one categorical
# Compare the means of each split. (Use groupby)
df.groupby(['parents'])
df
```

Out[11]:

	parents	has_nurs	form	children	housing	finance	social	he
usual	proper	complete	1	convenient	convenient	nonprob	recommended	recomn
usual	proper	complete	1	convenient	convenient	nonprob	priority	pr
usual	proper	complete	1	convenient	convenient	nonprob	not_recom	not_re
usual	proper	complete	1	convenient	convenient	slightly_prob	recommended	recomn
usual	proper	complete	1	convenient	convenient	slightly_prob	priority	pr
...	...	...	...	...	...	...	...	
great_pret	very_crit	foster	more	critical	inconv	slightly_prob	priority	spec_
great_pret	very_crit	foster	more	critical	inconv	slightly_prob	not_recom	not_re
great_pret	very_crit	foster	more	critical	inconv	problematic	recommended	spec_
great_pret	very_crit	foster	more	critical	inconv	problematic	priority	spec_
great_pret	very_crit	foster	more	critical	inconv	problematic	not_recom	not_re

12960 rows × 8 columns

In [37]:

```
# Q6. Create one dataframe with 5 subjects and marks of 10 students for each
# arithmetic mean, geometric mean, and harmonic mean.
import pandas as pd
df = pd.DataFrame(columns=['name','java','blockchain','php','python','c'])
df.loc[0] = ['Aishwarya',97,99,88,86,90.00]
df.loc[1] = ['Mona',97,99,88,86,90.00]
df.loc[2] = ['Sayali',97,89,88,86,90.00]
df.loc[3] = ['Sita',67,89,88,86,80.00]
df.loc[4] = ['Sona',77,69,88,86,60.00]
df.loc[5] = ['Rita',57,79,78,86,70.00]
df.loc[6] = ['Rutuja',57,98,88,86,85.00]
df.loc[7] = ['Siddhi',77,95,88,86,70.00]
df.loc[8] = ['Riya',37,99,48,36,78.00]
df.loc[9] = ['Pinky',57,99,88,56,90.00]
df.loc[10] = ['Soniya',67,99,58,86,90.00]
print(df)
import scipy.stats as s
print("Arithmetic Mean: ",s.tmean(df['java']))
print("Harmonic Mean: ",s.hmean(df['java']))
print("Gemotric mean: ",s.gmean(df['c']))
```

```
         name java blockchain php python     c
0   Aishwarya   97         99  88     86  90.0
1        Mona   97         99  88     86  90.0
2      Sayali   97         89  88     86  90.0
3        Sita   67         89  88     86  80.0
4        Sona   77         69  88     86  60.0
5        Rita   57         79  78     86  70.0
6      Rutuja   57         98  88     86  85.0
7      Siddhi   77         95  88     86  70.0
8        Riya   37         99  48     36  78.0
9       Pinky   57         99  88     56  90.0
10     Soniya   67         99  58     86  90.0
Arithmetic Mean:  71.54545454545455
Harmonic Mean:  66.10131483293476
Gemotric mean:  80.50688734152958
```

In [23]:

```
# Q7. Download any csv file of your choice and display details about data usi
# profiling. Show stats in HTML form.
import pandas as pd
df = pd.read_csv('iris.data')
data = df.describe()
result = data.to_html()
print(result)
```

```
<table border="1" class="dataframe">
  <thead>
    <tr style="text-align: right;">
      <th></th>
      <th>sepal length</th>
      <th>sepal width</th>
      <th>petal length</th>
      <th>petal width</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>count</th>
      <td>150.000000</td>
      <td>150.000000</td>
      <td>150.000000</td>
      <td>150.000000</td>
    </tr>
    <tr>
      <th>mean</th>
      <td>5.843333</td>
      <td>3.054000</td>
      <td>3.758667</td>
      <td>1.198667</td>
    </tr>
    <tr>
      <th>std</th>
      <td>0.828066</td>
      <td>0.433594</td>
      <td>1.764420</td>
      <td>0.763161</td>
    </tr>
    <tr>
      <th>min</th>
      <td>4.300000</td>
      <td>2.000000</td>
      <td>1.000000</td>
      <td>0.100000</td>
    </tr>
    <tr>
      <th>25%</th>
      <td>5.100000</td>
      <td>2.800000</td>
      <td>1.600000</td>
      <td>0.300000</td>
    </tr>
    <tr>
      <th>50%</th>
```

```
      <td>5.800000</td>
      <td>3.000000</td>
      <td>4.350000</td>
      <td>1.300000</td>
    </tr>
    <tr>
      <th>75%</th>
      <td>6.400000</td>
      <td>3.300000</td>
      <td>5.100000</td>
      <td>1.800000</td>
    </tr>
    <tr>
      <th>max</th>
      <td>7.900000</td>
      <td>4.400000</td>
      <td>6.900000</td>
      <td>2.500000</td>
    </tr>
  </tbody>
</table>
```

Out[23]:

	Unnamed: 0	sepal length	sepal width	petal length	petal width
0	count	150.000000	150.000000	150.000000	150.000000
1	mean	5.843333	3.054000	3.758667	1.198667
2	std	0.828066	0.433594	1.764420	0.763161
3	min	4.300000	2.000000	1.000000	0.100000
4	25%	5.100000	2.800000	1.600000	0.300000
5	50%	5.800000	3.000000	4.350000	1.300000
6	75%	6.400000	3.300000	5.100000	1.800000
7	max	7.900000	4.400000	6.900000	2.500000

In [ ]: