In [1]:

```
##                                ------WEEK3-------

import pandas as pd
import os

#task1 visualizing the entire dataframe from csv file

path="Dataset.csv"

data = pd.read_csv(path, encoding='utf-8')
print(data)
```

```
    longitude  latitude  housing_median_age  total_rooms  population  \
0      -122.23     37.88             41.0000          880         322
1      -122.23     37.88             41.0000          880         322
2      -122.22     37.86             21.0000         7099        2401
3      -122.25     37.84             52.0001         3104        1157
4      -122.26     37.85             52.0000         3503        1504
5      -121.65     39.32             40.0000          812         374
6      -121.69     39.36             29.0000         2220        1170
7      -121.70     39.37             32.0000         1852         911
8      -121.70     39.36             46.0000         1210         523
9      -121.70     39.36             37.0000         2330        1505
10     -121.69     39.36             34.0000          842         635
11     -121.74     39.38             27.0000         2596        1100
12     -121.80     39.33             30.0000         1019         501
13     -120.46     38.15             16.0000         4221        1516
14     -120.55     38.12             10.0000         1566         785
15     -120.56     38.09             34.0000         2745        1150
16     -124.23     41.75             11.0000         3159        1343
17     -124.21     41.77             17.0000         3461        1947
18     -124.19     41.78             15.0000         3140        1645
19     -124.16     41.74             15.0000         2715        1532
20     -124.14     41.95             21.0000         2696        1208
21     -124.16     41.92             19.0000         1668         841
22     -118.32     33.35             27.0000         1675         744
23     -118.33     33.34             52.0000         2359        1100
24     -118.32     33.33             52.0000         2127         733
25     -118.32     33.34             52.0000          996         341
26     -118.48     33.43             29.0000          716         422
27     -118.48     33.43             29.0000          716         422

    median_income  median_house_value ocean_proximity
0          8.3252              452600        NEAR BAY
1          8.3252              452600        NEAR BAY
2          8.3014              358500        NEAR BAY
3          3.1200              241400        NEAR BAY
4          3.2705              241800        NEAR BAY
5          2.7891               73500          INLAND
6          2.3224               56200          INLAND
7          1.7885               57000          INLAND
8          1.9100               63900          INLAND
9          2.0474               56000          INLAND
10         1.8355               63000          INLAND
11         2.3243               85500             NaN
12         2.5259               81300          INLAND
13         2.3816              116000          INLAND
14         2.5000              116100          INLAND
15         2.3654               94900          INLAND
16         2.4805               73200      NEAR OCEAN
17         2.5795               68400          NEAR O
18         1.6654               74600          NEAR O
19         2.1829               69500      NEAR OCEAN
20            NaN              122400      NEAR OCEAN
21         2.1336               75000      NEAR OCEAN
22         2.1579              450000          ISLAND
```

```
23          2.8333              414700          ISLAND
24          3.3906              300000          ISLAND
25          2.7361              450000          ISLAND
26          2.6042              287500          ISLAND
27          2.6042              287500          ISLAND
```

In [2]:
```
# 2 detect and handle missing values
# There are 2 rows with one NaN values.
```

In [3]:
```
data.isna().sum()
```

Out[3]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
population            0
median_income         1
median_house_value    0
ocean_proximity       1
dtype: int64
```

In [4]:
```
#First we need to find the mean and replacing the value with Nan

mean = data['median_income'].mean()
print(mean)
data['median_income']=data['median_income'].fillna(mean)
```

```
3.092614814814815
```

In [5]:
```
# I will use forward fill to replace the string value of ocean proximity.
data.fillna(axis=0 , method='ffill',limit=1)
data['ocean_proximity']= data.fillna(axis=0 , method='ffill',limit=1)
```

In [6]:
```
#3 Unnecessary duplicates
duplicates=data.duplicated()
```

In [7]:
```
data[duplicates]
```

Out[7]:

	longitude	latitude	housing_median_age	total_rooms	population	median_income	median_hou
1	-122.23	37.88	41.0	880	322	8.3252	
27	-118.48	33.43	29.0	716	422	2.6042	

In [8]:
```
# And we need to drop them
data.drop_duplicates(inplace=True)
```

In [9]:
```
#I did not want it to return but let's check if those rows are dropped.

print(data)
```

```
    longitude  latitude  housing_median_age  total_rooms  population  \
0     -122.23     37.88             41.0000          880         322
2     -122.22     37.86             21.0000         7099        2401
3     -122.25     37.84             52.0001         3104        1157
```

```
4     -122.26     37.85             52.0000         3503         1504
5     -121.65     39.32             40.0000          812          374
6     -121.69     39.36             29.0000         2220         1170
7     -121.70     39.37             32.0000         1852          911
8     -121.70     39.36             46.0000         1210          523
9     -121.70     39.36             37.0000         2330         1505
10    -121.69     39.36             34.0000          842          635
11    -121.74     39.38             27.0000         2596         1100
12    -121.80     39.33             30.0000         1019          501
13    -120.46     38.15             16.0000         4221         1516
14    -120.55     38.12             10.0000         1566          785
15    -120.56     38.09             34.0000         2745         1150
16    -124.23     41.75             11.0000         3159         1343
17    -124.21     41.77             17.0000         3461         1947
18    -124.19     41.78             15.0000         3140         1645
19    -124.16     41.74             15.0000         2715         1532
20    -124.14     41.95             21.0000         2696         1208
21    -124.16     41.92             19.0000         1668          841
22    -118.32     33.35             27.0000         1675          744
23    -118.33     33.34             52.0000         2359         1100
24    -118.32     33.33             52.0000         2127          733
25    -118.32     33.34             52.0000          996          341
26    -118.48     33.43             29.0000          716          422

    median_income  median_house_value ocean_proximity
0        8.325200              452600         -122.23
2        8.301400              358500         -122.22
3        3.120000              241400         -122.25
4        3.270500              241800         -122.26
5        2.789100               73500         -121.65
6        2.322400               56200         -121.69
7        1.788500               57000          -121.7
8        1.910000               63900          -121.7
9        2.047400               56000          -121.7
10       1.835500               63000         -121.69
11       2.324300               85500         -121.74
12       2.525900               81300          -121.8
13       2.381600              116000         -120.46
14       2.500000              116100         -120.55
15       2.365400               94900         -120.56
16       2.480500               73200         -124.23
17       2.579500               68400         -124.21
18       1.665400               74600         -124.19
19       2.182900               69500         -124.16
20       3.092615              122400         -124.14
21       2.133600               75000         -124.16
22       2.157900              450000         -118.32
23       2.833300              414700         -118.33
24       3.390600              300000         -118.32
25       2.736100              450000         -118.32
26       2.604200              287500         -118.48
```

In [10]:

```
# And they are dropped.

#5 Wrong values -> we need to change the median age to int to get rid of from float
data['housing_median_age']=data['housing_median_age'].astype('int')
```

In [11]:

```
data['median_income']=data['median_income'].astype('float')
data['median_income']=data['median_income'].map('{:.4f}'.format)
## made all similar to each other so there is no valu like 2.5
```

In [12]:

```
# 6 we will save the dataframe to csv
data.to_csv("output.csv",index=False,encoding='utf8')
```

In [13]:
```
new_path="output.csv"
output_data = pd.read_csv(new_path, encoding='utf-8')
print(output_data)
```

```
    longitude  latitude  housing_median_age  total_rooms  population  \
0      -122.23     37.88                  41          880         322
1      -122.22     37.86                  21         7099        2401
2      -122.25     37.84                  52         3104        1157
3      -122.26     37.85                  52         3503        1504
4      -121.65     39.32                  40          812         374
5      -121.69     39.36                  29         2220        1170
6      -121.70     39.37                  32         1852         911
7      -121.70     39.36                  46         1210         523
8      -121.70     39.36                  37         2330        1505
9      -121.69     39.36                  34          842         635
10     -121.74     39.38                  27         2596        1100
11     -121.80     39.33                  30         1019         501
12     -120.46     38.15                  16         4221        1516
13     -120.55     38.12                  10         1566         785
14     -120.56     38.09                  34         2745        1150
15     -124.23     41.75                  11         3159        1343
16     -124.21     41.77                  17         3461        1947
17     -124.19     41.78                  15         3140        1645
18     -124.16     41.74                  15         2715        1532
19     -124.14     41.95                  21         2696        1208
20     -124.16     41.92                  19         1668         841
21     -118.32     33.35                  27         1675         744
22     -118.33     33.34                  52         2359        1100
23     -118.32     33.33                  52         2127         733
24     -118.32     33.34                  52          996         341
25     -118.48     33.43                  29          716         422

    median_income  median_house_value  ocean_proximity
0          8.3252              452600          -122.23
1          8.3014              358500          -122.22
2          3.1200              241400          -122.25
3          3.2705              241800          -122.26
4          2.7891               73500          -121.65
5          2.3224               56200          -121.69
6          1.7885               57000          -121.70
7          1.9100               63900          -121.70
8          2.0474               56000          -121.70
9          1.8355               63000          -121.69
10         2.3243               85500          -121.74
11         2.5259               81300          -121.80
12         2.3816              116000          -120.46
13         2.5000              116100          -120.55
14         2.3654               94900          -120.56
15         2.4805               73200          -124.23
16         2.5795               68400          -124.21
17         1.6654               74600          -124.19
18         2.1829               69500          -124.16
19         3.0926              122400          -124.14
20         2.1336               75000          -124.16
21         2.1579              450000          -118.32
22         2.8333              414700          -118.33
23         3.3906              300000          -118.32
24         2.7361              450000          -118.32
25         2.6042              287500          -118.48
```

In [14]:
```
# I am going to find the mean value of 'median_house_value'using mean() method
Mean_value=data['median_house_value'].mean()
print(int(Mean_value))
```

```
174730
```

In [15]:
```
# I am going to find the median value of 'median_house_value'using median() method
```

```
Median_value=data['median_house_value'].median()
print(int(Median_value))
```

```
90200
```

In [16]:

```
# I am going to find the max value of 'median_house_value'using max() method
max_value=data['median_house_value'].max()
print(int(max_value))
```

```
452600
```

In [17]:

```
# I am going to find the min value of 'median_house_value'using min() method
min_value=data['median_house_value'].min()
print(int(min_value))
```

```
56000
```

In [18]:

```
# Finding the range value using min and max
range1 = "Range is between "+str(min_value)+", "+str(max_value)
print(range1)
```

```
Range is between 56000, 452600
```

In [19]:

```
# $ sign added to median_income values using .map() method.
output_data['median_income'] = output_data['median_income'].map('${:,.5f}'.format)
print(output_data)
```

```
    longitude  latitude  housing_median_age  total_rooms  population  \
0      -122.23     37.88                  41          880         322
1      -122.22     37.86                  21         7099        2401
2      -122.25     37.84                  52         3104        1157
3      -122.26     37.85                  52         3503        1504
4      -121.65     39.32                  40          812         374
5      -121.69     39.36                  29         2220        1170
6      -121.70     39.37                  32         1852         911
7      -121.70     39.36                  46         1210         523
8      -121.70     39.36                  37         2330        1505
9      -121.69     39.36                  34          842         635
10     -121.74     39.38                  27         2596        1100
11     -121.80     39.33                  30         1019         501
12     -120.46     38.15                  16         4221        1516
13     -120.55     38.12                  10         1566         785
14     -120.56     38.09                  34         2745        1150
15     -124.23     41.75                  11         3159        1343
16     -124.21     41.77                  17         3461        1947
17     -124.19     41.78                  15         3140        1645
18     -124.16     41.74                  15         2715        1532
19     -124.14     41.95                  21         2696        1208
20     -124.16     41.92                  19         1668         841
21     -118.32     33.35                  27         1675         744
22     -118.33     33.34                  52         2359        1100
23     -118.32     33.33                  52         2127         733
24     -118.32     33.34                  52          996         341
25     -118.48     33.43                  29          716         422

   median_income  median_house_value  ocean_proximity
0       $8.32520              452600          -122.23
1       $8.30140              358500          -122.22
2       $3.12000              241400          -122.25
3       $3.27050              241800          -122.26
4       $2.78910               73500          -121.65
5       $2.32240               56200          -121.69
6       $1.78850               57000          -121.70
```

```
7       $1.91000              63900          -121.70
8       $2.04740              56000          -121.70
9       $1.83550              63000          -121.69
10      $2.32430              85500          -121.74
11      $2.52590              81300          -121.80
12      $2.38160             116000          -120.46
13      $2.50000             116100          -120.55
14      $2.36540              94900          -120.56
15      $2.48050              73200          -124.23
16      $2.57950              68400          -124.21
17      $1.66540              74600          -124.19
18      $2.18290              69500          -124.16
19      $3.09260             122400          -124.14
20      $2.13360              75000          -124.16
21      $2.15790             450000          -118.32
22      $2.83330             414700          -118.33
23      $3.39060             300000          -118.32
24      $2.73610             450000          -118.32
25      $2.60420             287500          -118.48
```

In [20]:
```
######         --------------WEEK 4 ----------------------
```

In [21]:
```
## importing necessary pandas,numpy and mtploblib libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [22]:
```
# Task1 -- Initialise a two-dimensional array consisting of 5 rows and 10 columns of
#data points of integer values from the interval [0..9]. random.randint(low, high=No
# creating a uniformly distributed data points using a loop which is in range 5 and
# which has values between 0-9 and because of this reason I created an array which h
array = np.arange(50).reshape(5,10)
for i in range(5):
    array[i] = np.random.randint(0, 10, 10, dtype='int')
```

In [23]:
```
# printing the array
print(array)
```

```
[[5 2 4 6 2 9 3 3 7 2]
 [2 3 4 8 7 9 6 7 5 3]
 [0 8 2 1 4 1 8 8 5 4]
 [0 2 6 5 2 1 6 8 6 2]
 [0 0 8 7 2 7 2 7 0 2]]
```

In [24]:
```
#checking the dimensions of array.
array.shape
```

Out[24]: (5, 10)

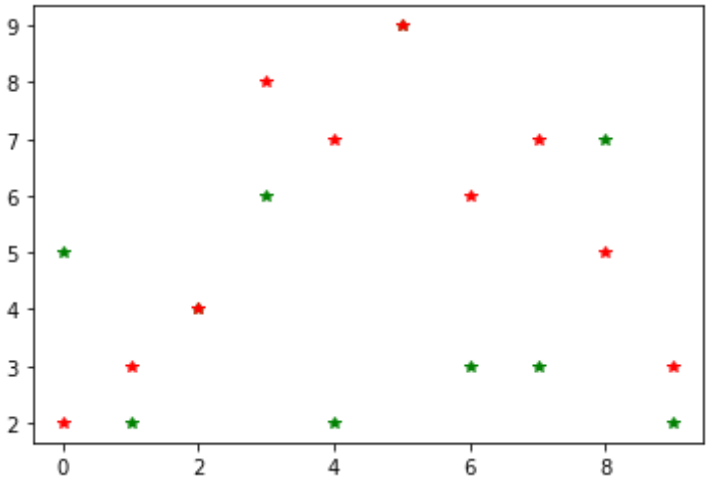
In [25]:
```
#Consider each row of the two-dimensional array as an independent dataset.
#Display the values on the screen as a table.
# I displayed every row individually using a loop and printing them seperately.
for i in range(5):
    print(array[i])
```

```
[5 2 4 6 2 9 3 3 7 2]
[2 3 4 8 7 9 6 7 5 3]
[0 8 2 1 4 1 8 8 5 4]
```

```
[0 2 6 5 2 1 6 8 6 2]
[0 0 8 7 2 7 2 7 0 2]
```

In [26]:
```
# Plot the first two rows on a single diagram with different colours
# I took the values of first two rows using array[0] and array[1] and I put those va
# variables .I plotted them using the plot() method. I gave them color green using '
s1= array[0]
s2= array[1]
plt.plot(s1,'g*',s2,'r*')
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x1550a1e4dc0>,
 <matplotlib.lines.Line2D at 0x1550a1e4eb0>]
```



In [ ]:
```
```

In [27]:
```
## Provide the following information about each individual row:
## Mean, Median, Standard deviation
# I used the loop in range 5 to visit every row and find the mean median and standar
# using mean(), median(),std() built in methods for mean,medium and standart deviati
for i in range(5):
    print("row "+ str(i)+" Mean value : "+ str(array[i].mean())+ ", Median value : "
          ", Standart deviation value : "+ str(np.std(array)))
```

```
row 0 Mean value : 4.3, Median value : 3.5, Standart deviation value : 2.74437606752
42745
row 1 Mean value : 5.4, Median value : 5.5, Standart deviation value : 2.74437606752
42745
row 2 Mean value : 4.1, Median value : 4.0, Standart deviation value : 2.74437606752
42745
row 3 Mean value : 3.8, Median value : 3.5, Standart deviation value : 2.74437606752
42745
row 4 Mean value : 3.5, Median value : 2.0, Standart deviation value : 2.74437606752
42745
```

In [28]:
```
## Initialise a one-dimensional array representing a normal distribution
## of 1000 data points with mean value 17 and standard deviation 0.2.
# I created a normal distribution array using .normal() method and used mean value a
# i chosed the size of 1000 depends on the needs in the question.
mean_value = 17
standart_deviation_val = 0.2
x = np.random.normal(loc=mean_value, scale=standart_deviation_val, size=1000)
```

In [29]:
```
## Find the maximum and the minimum values of the dataset and calculate the range.
# I found the min and max values using .max()and .min() methods and find the range u
maximum = x.max()
```

```
minimum = x.min()
range2 = maximum - minimum ;
print("Max value of dataset : "+str(maximum)+" , Minimum value of dataset : "+str(mi
```

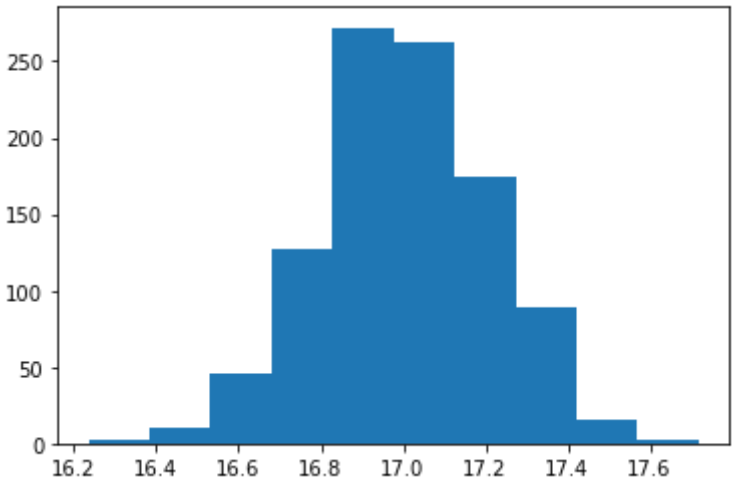
```
Max value of dataset : 17.718835228153885 , Minimum value of dataset : 16.2344770584
0858, range of dataset : 1.484358169745306
```

In [30]:

```
## the dataset by using a histogram with 10 bins. Visualise the probability density
## function
## I used dataset x and plotted it using .hist() method to create histogram with 10
plt.hist(x, bins = 10)
```

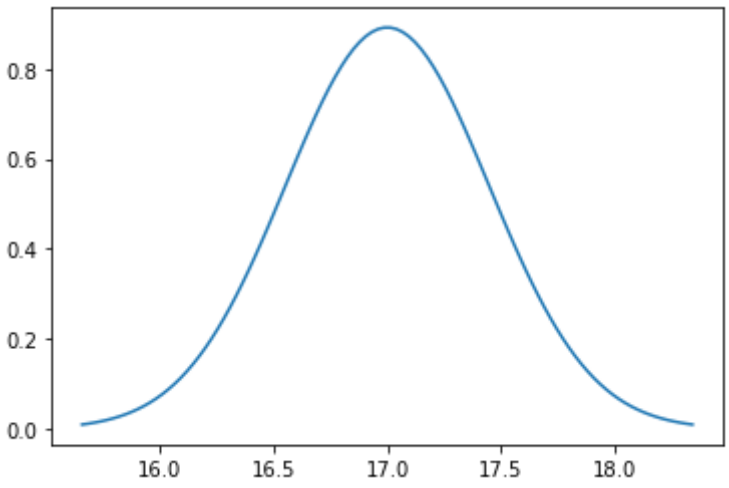
Out[30]:

```
(array([  3.,  10.,  46., 127., 272., 262., 174.,  89.,  15.,   2.]),
 array([16.23447706, 16.38291288, 16.53134869, 16.67978451, 16.82822033,
        16.97665614, 17.12509196, 17.27352778, 17.42196359, 17.57039941,
        17.71883523]),
 <BarContainer object of 10 artists>)
```



In [31]:

```
## Probability density functio. i found that algorithm from internet which is easier
## used the mu value as mean value and vriance as standart deviation because it is a
## plotted the result based on algorithm.
import math
from scipy import stats
mu = 17
variance = 0.2
sigma = math.sqrt(variance)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
plt.plot(x, stats.norm.pdf(x, mu, sigma))
plt.show()
```



In [ ]:

In [32]:
```
##                              ---------- WEEK 5 ----------

## Q.1 Find the determinant, the trace and the inverse of matrix A.
import numpy as np
```

In [33]:
```
## Initializing the matrix - Array initiliazed
A = np.array([[2, 5, 1], [4, 3, 7], [1, 3, 2]])
print(A)
```

```
[[2 5 1]
 [4 3 7]
 [1 3 2]]
```

In [34]:
```
## Trace of matrix A - I used built - in trace() function to find trace of matrix
T = np.trace(A)
print(T)
```

```
7
```

In [35]:
```
##  inverse of matrix A.Using linalg bultn-in for function .det()
inv = np.linalg.inv(A)
print(inv)
```

```
[[ 0.57692308  0.26923077 -1.23076923]
 [ 0.03846154 -0.11538462  0.38461538]
 [-0.34615385  0.03846154  0.53846154]]
```

In [36]:
```
## Determinant of matrice A - Using linalg bultn-in for function .det()
D = np.linalg.det(A)
print(D)
```

```
-26.000000000000014
```

In [37]:
```
## Q.2 Initialise the following square matrices B and C:
## Find the product P of the matrices B and C by using the Python function for matri
## multiplication. Display the result on the screen.

## Initiliazing the matrix B
B = np.array([[4, 7, 2], [3, 2, 5], [6, 4, 3]])
print(B)
## Initiliazing the matrix C
C = np.array([[3, 1, 9], [7, 5, 8], [2, 1, 1]])
print(C)
## i used built-in matmul() function to calculate P = B * C
P = np.matmul(B, C)
print(P)
```

```
[[4 7 2]
 [3 2 5]
 [6 4 3]]
[[3 1 9]
 [7 5 8]
 [2 1 1]]
[[65 41 94]
 [33 18 48]
 [52 29 89]]
```

In [38]:

```
## Q.3 Consider the following system of linear equations
## 3X + 2Y - Z = 25
## 2X - Y + 4Z = 19
## 4X - 2Y + 3Z = 18

## i will expres them as the number as matrix M and (x,y,z) as X[1,3] matrix and C=(
## so i can express them as MX=C

## initiliazing matrix M
M = np.array([[3, 2, -1], [2, -1, 4], [4, -2, 3]])
print(M)

## Initialise matrix C

C = np.array([25, 19, 18])
print(C)
#X will be fined as X= inverse of M * C from the equation in the next question
```

```
[[ 3  2 -1]
 [ 2 -1  4]
 [ 4 -2  3]]
[25 19 18]
```

In [39]:

```
# Q.4 Provide the algebraic steps for solving the system of linear equations from Ta
## using matrix notation
## X= inverse of M * C from the equation so first i will find the inverse of M and m
## inverse of M
invM = np.linalg.inv(M)
print(M)
print()
## and multiplying it with C will give us the values of x,y,z
print("value of x , y ,z")
X = np.matmul(invM, C)
print(X)
```

```
[[ 3  2 -1]
 [ 2 -1  4]
 [ 4 -2  3]]

value of x , y ,z
[5. 7. 4.]
```

In [40]:

```
## when i put the values to variables it shows the correct result. (3*5)+ (2*7) - 4
```

In [41]:

```
## Q.5 Solve the system of linear equations from Task 3 by using Python script utili
## multiplication and inverse matrix.
## I will use the np.linalg.solve(M,C)
## So I will not need to take inverse of M with that function.

print("value of X matrix is: ", np.linalg.solve(M,C))
```

```
value of X matrix is:  [5. 7. 4.]
```

In [42]:

```
## It exactly gave me the same result so calculation is double checked.
```

In [43]:

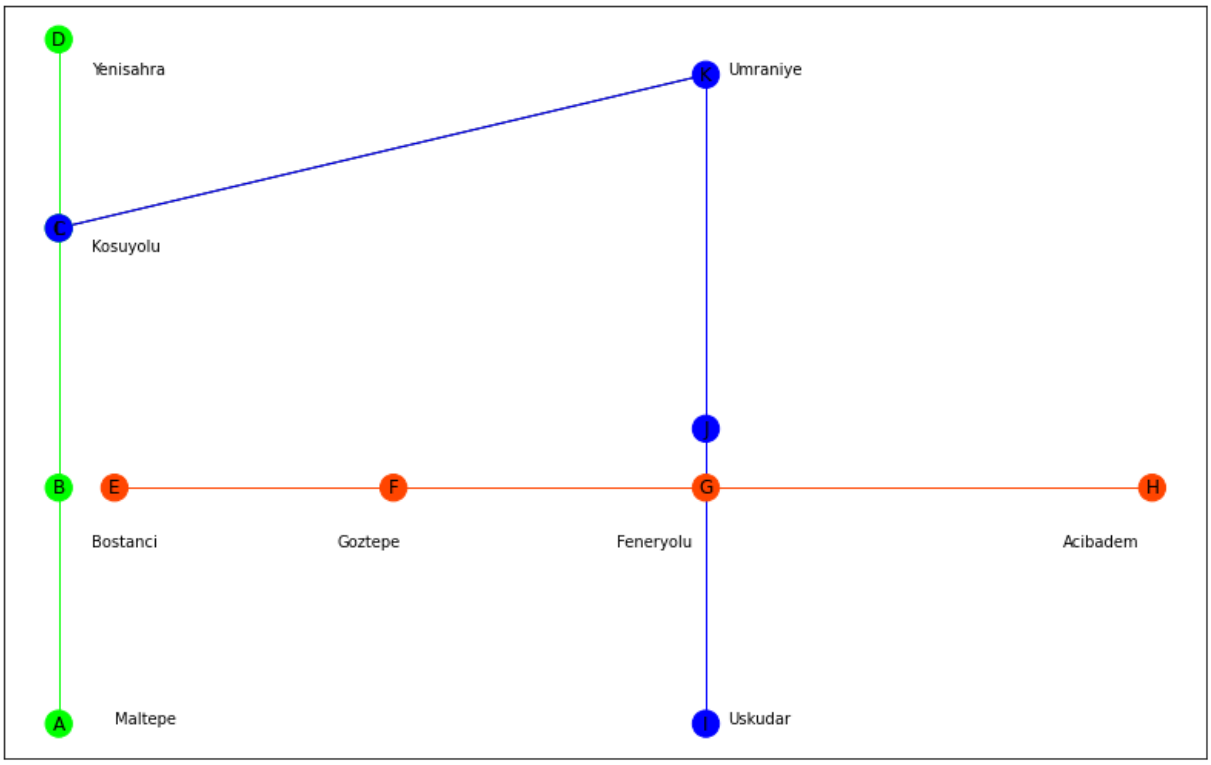
```
##          ----------------- WEEK 6 -----------------
```

In [44]:

```
import networkx as nx
import matplotlib.pyplot as plt
```

```
# I created a blank graph object
ist_graph = nx.Graph()

# In this section I added nodes and gave position values to nodes depending of their

#  First line is Green Line and has four stations maltepe,bostanci,kosuyolu and yeni

# feneryolu,kosuyolu and bostanci are stations that you can change your line.

ist_graph.add_node('A', npos=(10, 10), ccn='#00FF00')
## 900m between maltepe and bostanci
ist_graph.add_node('B', npos=(10, 30), ccn='#00FF00')
##1000 m between bostanci kosuyolu
ist_graph.add_node('C', npos=(10, 52), ccn='#00FF00')
## 750m between kosuyolu and yenisahra
ist_graph.add_node('D', npos=(10, 68), ccn='#00FF00')

#  Second line is Orange Line and has four stations bostanci , goztepe , feneryolu a

ist_graph.add_node('E', npos=(15, 30), ccn='#FF4500')
## distance bostanci and goztepe is 800m
ist_graph.add_node('F', npos=(40, 30), ccn='#FF4500')
## distance goztepe and feneryolu is 870m
ist_graph.add_node('G', npos=(68, 30), ccn='#FF4500')
## distance feneryolu and acibadem is 1000m
ist_graph.add_node('H', npos=(108, 30), ccn='#FF4500')


# Third line is Blue line and has also four stations feneryolu,uskudar,umraniye,kosu
ist_graph.add_node('I', npos=(68, 10), ccn='#0000FF')
## distance feneryolu and uskudar is 770m
ist_graph.add_node('J', npos=(68, 35), ccn='#0000FF')
## distance feneryolu and umraniye is 850m
ist_graph.add_node('K', npos=(68, 65), ccn='#0000FF')
## distance umraniye and kosuyolu is 1200m
ist_graph.add_node('L', npos=(10, 52), ccn='#0000FF')

# I connected the edges using the .add_edge() method.
# Green Line
ist_graph.add_edge('A', 'B', cce='#00FF00')
ist_graph.add_edge('B', 'C', cce='#00FF00')
ist_graph.add_edge('C', 'D', cce='#00FF00')
# Orange Line
ist_graph.add_edge('E', 'F', cce='#FF4500')
ist_graph.add_edge('F', 'G', cce='#FF4500')
ist_graph.add_edge('G', 'H', cce='#FF4500')
# Blue Line
ist_graph.add_edge('I', 'J', cce='#0000FF')
ist_graph.add_edge('J', 'K', cce='#0000FF')
ist_graph.add_edge('K', 'L', cce='#0000FF')

# transferred position values , color values and edgecolor values to variables to us
pos = nx.get_node_attributes(ist_graph, 'npos')
nodecolour = nx.get_node_attributes(ist_graph, 'ccn')
edgecolour = nx.get_edge_attributes(ist_graph, 'cce')

# Putting those color dictionary values to list
NodeList = list(nodecolour.values())
EdgeList = list(edgecolour.values())

# Setting the size
plt.figure(figsize=(14, 9))
```

```
# displaying the station names next to their nodes.
plt.text(15, 10, s='Maltepe')
plt.text(13, 25, s='Bostanci')
plt.text(13, 50, s='Kosuyolu')
plt.text(13, 65, s='Yenisahra')

plt.text(35, 25, s='Goztepe')
plt.text(60, 25, s='Feneryolu')
plt.text(100, 25, s='Acibadem')
plt.text(70, 10, s='Uskudar')

plt.text(70, 65, s='Umraniye')

# drawing the nodes and edges using stored values in pos,edge_color,node color value
nx.draw_networkx(ist_graph, pos, node_color=NodeList)
nx.draw_networkx_edges(ist_graph, pos, edge_color=EdgeList)

# displaying the graph.
plt.show()
```



In [45]:

```
## Find the average monthly temperatures of three cities of your choice. Represent t
## by using a heat map. Provide a colour scale for guidance. Allow the user to speci
## threshold for the heat map. Based on this threshold value, use different base col
## representing the data points
```

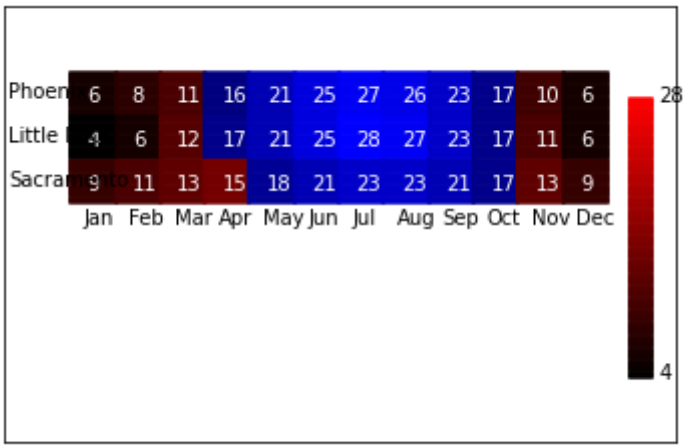
In [49]:

```
## importing the necessary pandas and matplotlib to manipulate data and visualizing
import pandas as pd
import matplotlib.pyplot as plt


# This subroutine encapsulates the 'plot' method, as the most suitable for raster re
## this pat is checking the variables boundaries for faster rending and checking the
def DrawBox(x, y, size, r, g, b):
    if r < 0:
        r = int(0)
    if g < 0:
        g = int(0)
```

```
    if b < 0:
        b = int(0)
    if r > 255:
        r = int(255)
    if g > 255:
        g = int(255)
    if b > 255:
        b = int(255)
    for i in range(0, int(size)):
        plt.plot([x, x + size], [y + i, y + i], '#{:02x}{:02x}{:02x}'.format(r, g, b


# Store the dataset into a data frame
# I choosed the first 3 cities. I used the HeatMap.csv which is provided and I used
## my local directory next to my python code.
df = pd.read_csv('HeatMap.csv',nrows=3)
## i read the csv file but I gave limit which is 3 that it reads only the first 3 ro
# Printing the Phoenix,Sacremento and little rock montly heat values.
print(df.head(3))

# Seting the plot sizes
plt.axis([0, 600, 0, 400])
plt.xticks([])
plt.yticks([])
# Finding the minimum and maximum values to assign color values to those temperature
Min = int(min(df.min(numeric_only=True)))
Max = int(max(df.max(numeric_only=True)))

BoxSize = int(40)
OffsetX = int(15)
OffsetY = int(12)
# Threshold is added due to customer's choice. Colors will be assigned depending to
th = input('Enter the threshold value :')
Threshold = int(th)
# Generate the heat map
for i in range(0, df.shape[0]):
    for j in range(1, df.shape[1]):
        ColourCode = int(((df.values[i, j]-Min)/(Max-Min))*255)
        # if the heat value is bigger than threshold it will change its color brown
        if df.values[i, j] > Threshold:
            DrawBox(20+BoxSize*j, 300-BoxSize*i, BoxSize, 0, 0, ColourCode)
            # if the heat value is bigger than threshold it will change its color br
        if df.values[i, j] <= Threshold:
            DrawBox(20+BoxSize*j, 300-BoxSize*i, BoxSize, ColourCode, 0, 0)
        plt.text(OffsetX+20+BoxSize*j, OffsetY+300-BoxSize*i, str(df.values[i, j]),

# Generating the scale of draw
for i in range(0, 256):
    plt.plot([560, 580], [i + 60, i + 60], '#{:02x}{:02x}{:02x}'.format(int(i), 0, 0
plt.text(585, 58, Min)
plt.text(585, 312, Max)
# placing the months to correct place in the drawing.
plt.text(72, 200, 'Jan')
plt.text(112, 200, 'Feb')
plt.text(152, 200, 'Mar')
plt.text(192, 200, 'Apr')
plt.text(232, 200, 'May')
plt.text(272, 200, 'Jun')
plt.text(312, 200, 'Jul')
plt.text(352, 200, 'Aug')
plt.text(392, 200, 'Sep')
plt.text(432, 200, 'Oct')
plt.text(472, 200, 'Nov')
plt.text(512, 200, 'Dec')
```

```
# putting the city names to correct place in the drawing.
plt.text(5, 315, str(df.values[0, 0]))
plt.text(5, 275, str(df.values[1, 0]))
plt.text(5, 235, str(df.values[2, 0]))

plt.show()
```

```
          City  1   2   3   4   5   6   7   8   9  10  11  12
0      Phoenix  6   8  11  16  21  25  27  26  23  17  10   6
1  Little Rock  4   6  12  17  21  25  28  27  23  17  11   6
2   Sacramento  9  11  13  15  18  21  23  23  21  17  13   9
Enter the threshold value :15
```



In [ ]: