In [1]:
```
#Q1.Create the evenly spaced array using arange ?
import numpy as np
a = np.arange(1, 10)
print(a)
x = range(1, 10)
print(x)    # x is an iterator
print(list(x))

# further arange examples:
x = np.arange(10.4)
print(x)
x = np.arange(0.5, 10.4, 0.8)
print(x)
```

```
[1 2 3 4 5 6 7 8 9]
range(1, 10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[ 0.5  1.3  2.1  2.9  3.7  4.5  5.3  6.1  6.9  7.7  8.5  9.3 10.1]
```

In [2]:
```
#Q2.Create the evenly spaced array using linespace ?
import numpy as np
# 50 values between 1 and 10:
print("50 values between 1 to 10",np.linspace(1, 10))
# 7 values between 1 and 10:
print("7 values between 1 to 10 ",np.linspace(1, 10, 7))
# excluding the endpoint:
print("excluding end points",np.linspace(1, 10, 7, endpoint=False))
```

```
50 values between 1 to 10 [ 1.          1.18367347  1.36734694  1.55102041  1.73469388  1.91836735
  2.10204082  2.28571429  2.46938776  2.65306122  2.83673469  3.02040816
  3.20408163  3.3877551   3.57142857  3.75510204  3.93877551  4.12244898
  4.30612245  4.48979592  4.67346939  4.85714286  5.04081633  5.2244898
  5.40816327  5.59183673  5.7755102   5.95918367  6.14285714  6.32653061
  6.51020408  6.69387755  6.87755102  7.06122449  7.24489796  7.42857143
  7.6122449   7.79591837  7.97959184  8.16326531  8.34693878  8.53061224
  8.71428571  8.89795918  9.08163265  9.26530612  9.44897959  9.63265306
  9.81632653 10.        ]
7 values between 1 to 10  [ 1.   2.5  4.   5.5  7.   8.5 10. ]
excluding end points [1.         2.28571429 3.57142857 4.85714286 6.14285714 7.42857143
 8.71428571]
```

In [3]:
```
#Q3.Create a Zero dimension array in Numpy also print its type and dimension?
import numpy as np
x = np.array(42)
print("x: ", x)
print("The type of x: ", type(x))
print("The dimension of x:", np.ndim(x))
```

```
x:  42
The type of x:  <class 'numpy.ndarray'>
The dimension of x: 0
```

In [4]:
```
#Q4. Create two One dimension array in Numpy also print its type and dimension?
F = np.array([1, 1, 2, 3, 5, 8, 13, 21])
V = np.array([3.4, 6.9, 99.8, 12.8])
print("F: ", F)
print("V: ", V)
print("Type of F: ", F.dtype)
print("Type of V: ", V.dtype)
print("Dimension of F: ", np.ndim(F))
print("Dimension of V: ", np.ndim(V))
```

```
F:  [ 1  1  2  3  5  8 13 21]
V:  [ 3.4  6.9 99.8 12.8]
Type of F:  int32
Type of V:  float64
Dimension of F:  1
Dimension of V:  1
```

In [5]:
```
#Q5. Write a code to create a two dimension array in Numpy and print its dimension?
A = np.array([ [3.4, 8.7, 9.9],
               [1.1, -7.8, -0.7],
               [4.1, 12.3, 4.8]])
print(A)
print(A.ndim)
```

```
[[ 3.4  8.7  9.9]
 [ 1.1 -7.8 -0.7]
 [ 4.1 12.3  4.8]]
2
```

In [6]:
```
#Q6. Create a multi dimension array in Numpy and print its dimension?

B = np.array([ [[111, 112], [121, 122]],
               [[211, 212], [221, 222]],
               [[311, 312], [321, 322]] ])
print(B)
```

```
print(B.ndim)
```

```
[[[111 112]
  [121 122]]

 [[211 212]
  [221 222]]

 [[311 312]
  [321 322]]]
3
```

In [8]:
```
#Q7. Write a code to return the shape of an array
x = np.array([ [67, 63, 87],
               [77, 69, 59],
               [85, 87, 99],
               [79, 72, 71],
               [63, 89, 93],
               [68, 92, 78]])

print(np.shape(x))
```

```
(6, 3)
```

In [9]:
```
#Q8. Write a code  to change the shape of the array
x.shape=(3, 6)
print(x)
```

```
[[67 63 87 77 69 59]
 [85 87 99 79 72 71]
 [63 89 93 68 92 78]]
```

In [10]:
```
#Q9. Write a code  to print the numbers in an array using its index value?

F = np.array([1, 1, 2, 3, 5, 8, 13, 21])
# print the first element of F
print(F[0])
# print the last element of F
print(F[-1])
```

```
1
21
```

In [11]:
```
#Q10. Write a code  to print the number using index value in multidimensional array?
A = np.array([ [3.4, 8.7, 9.9],
               [1.1, -7.8, -0.7],
               [4.1, 12.3, 4.8]])

print(A[1][0])
```

```
1.1
```

In [12]:
```
#Q11.Print the index 1 of array A and return the 0th index in the 1st index of A?
tmp = A[1]
print(tmp)
print(tmp[0])
```

```
[ 1.1 -7.8 -0.7]
1.1
```

In [13]:
```
#Q12.Perform slicing in single dimensional array?
S = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
print(S[2:5])
print(S[:4])
print(S[6:])
print(S[:])
```

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

In [14]:
```
#Q13. Write a code  to Perform slicing in multi dimensional array?
A = np.array([
[11, 12, 13, 14, 15],
[21, 22, 23, 24, 25],
[31, 32, 33, 34, 35],
[41, 42, 43, 44, 45],
[51, 52, 53, 54, 55]])

print(A[:3, 2:])
print(A[3:, :])
print(A[:, 4:])
```

```
[[13 14 15]
 [23 24 25]
 [33 34 35]]
[[41 42 43 44 45]
 [51 52 53 54 55]]
[[15]
 [25]
 [35]
 [45]
 [55]]
```

In [18]:
```
#Q14.Given A= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] where the values are taken from
#index 2 to index 5 is stored in S and change the index 0 to 22 and index 1 to 23 in S and print A.
A = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
S = A[2:6]
S[0] = 22
S[1] = 23
print(A)
```

```
[ 0  1 22 23  4  5  6  7  8  9]
```

In [19]:
```
# Q15.Write a code to check whether the two array A and S share the same memory space?
np.may_share_memory(A, S)
```

Out[19]:
```
True
```

In [20]:
```
#Q16.Create an array of 3 rows and 4 columns using numpy.arange function and change its 0 index value to be 42
A = np.arange(12)
B = A.reshape(3, 4)
A[0] = 42
print(B)
```

```
[[42  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

In [21]:
```
#Q17.A list is given  lst = [2,3, 7.9, 3.3, 6.9, 0.11, 10.3, 12.9] , add 2 to each
#element , multiply each element with 2.2 and subtract each element with
# 1.38 in the given list .

import numpy as np
lst = [2,3, 7.9, 3.3, 6.9, 0.11, 10.3, 12.9]
v = np.array(lst)
v = v + 2
print(v)
print(v * 2.2)
print(v - 1.38)
```

```
[ 4.    5.    9.9   5.3   8.9   2.11 12.3  14.9 ]
[ 8.8   11.    21.78  11.66  19.58   4.642 27.06  32.78 ]
[ 2.62  3.62  8.52  3.92  7.52  0.73 10.92 13.52]
```

In [22]:
```
#Q18.Create two arrays named A and B and add them, add 1 with each
#element in B and find the product of two arrays

import numpy as np
A = np.array([ [11, 12, 13], [21, 22, 23], [31, 32, 33] ])
B = np.ones((3,3))
print("Adding to arrays: ")
print(A + B)
print("\nMultiplying two arrays: ")
print(A * (B + 1))
```

```
Adding to arrays:
[[12. 13. 14.]
 [22. 23. 24.]
 [32. 33. 34.]]

Multiplying two arrays:
[[22. 24. 26.]
 [42. 44. 46.]
 [62. 64. 66.]]
```

In [1]:
```
#Q19. Write a code to perform multiplication of two arrays
import numpy as np
A = np.array([ [1, 2, 3], [2, 2, 2], [3, 3, 3] ])
B = np.array([ [3, 2, 1], [1, 2, 3], [-1, -2, -3] ])
R = A * B
print(R)
```

```
[[ 3  4  3]
 [ 2  4  6]
 [-3 -6 -9]]
```

In [3]:
```
#Q20.Check whether two arrays are equal using comparison operator '==',
#where A=[ [11, 12, 13], [21, 22, 23], [31, 32, 33] ] and B=[ [11, 102, 13], [201, 22, 203], [31, 32, 303] ]

import numpy as np
A = np.array([ [11, 12, 13], [21, 22, 23], [31, 32, 33] ])
```

```
B = np.array([ [11, 102, 13], [201, 22, 203], [31, 32, 303] ])
A == B
```

Out[3]:
```
array([[ True, False,  True],
       [False,  True, False],
       [ True,  True, False]])
```

In [4]:
```
#Q21. Check whether two arrays are equal using
#numpy.array_equal(),using the previously created array “A” and “B”
print(np.array_equal(A, B))
print(np.array_equal(A, A))
```

```
False
True
```

In [5]:
```
#Q22.Given a=([ [True, True], [False, False]]) and b=([ [True, False], [True, False]]) check the arrays using l
a = np.array([ [True, True], [False, False]])
b = np.array([ [True, False], [True, False]])
print(np.logical_or(a, b))
print(np.logical_and(a, b))
```

```
[[ True  True]
 [ True False]]
[[ True False]
 [False False]]
```

In [6]:
```
#Q23.Write a code to create multidimensional array A and a single
#dimensional array B .Multiply A and B ,Add A and B
import numpy as np
A = np.array([ [11, 12, 13], [21, 22, 23], [31, 32, 33] ])
B = np.array([1, 2, 3])
print("Multiplication with broadcasting: ")
print(A * B)
print("... and now addition with broadcasting: ")
print(A + B)
```

```
Multiplication with broadcasting:
[[11 24 39]
 [21 44 69]
 [31 64 99]]
... and now addition with broadcasting:
[[12 14 16]
 [22 24 26]
 [32 34 36]]
```

In [7]:
```
#Q24.Given B = [1, 2, 3] , print array B with 3 rows and 3 columns
B = np.array([[1, 2, 3],] * 3)
print(B)
```

```
[[1 2 3]
 [1 2 3]
 [1 2 3]]
```

In [8]:
```
#Q25.Consider the output of previously given array B and print the transpose of B
np.array([[1, 2, 3],] * 3).transpose()
```

Out[8]:
```
array([[1, 1, 1],
       [2, 2, 2],
       [3, 3, 3]])
```

In [11]:
```
#Q26. Given B = [1, 2, 3],consider the rows of B as columns and print the output.
B = np.array([1, 2, 3])
B[:, np.newaxis]
```

Out[11]:
```
array([[1],
       [2],
       [3]])
```

In [14]:
```
#Q27.Write a code for flattening using the values "C", "F" and "A".
import numpy as np
A = np.array([[[ 0,  1],
               [ 2,  3],
               [ 4,  5],
               [ 6,  7]],
              [[ 8,  9],
               [10, 11],
               [12, 13],
               [14, 15]],
              [[16, 17],
               [18, 19],
               [20, 21],
               [22, 23]]])
Flattened_X = A.flatten()
print(Flattened_X)
print(A.flatten(order="C"))
print(A.flatten(order="F"))
print(A.flatten(order="A"))
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
[ 0  8 16  2 10 18  4 12 20  6 14 22  1  9 17  3 11 19  5 13 21  7 15 23]
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
```

In [15]:
```
#Q28.Write a code to create an array named X using reshape ()
X = np.array(range(24))
Y = X.reshape((3,4,2))
Y
```

Out[15]:
```
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11],
        [12, 13],
        [14, 15]],

       [[16, 17],
        [18, 19],
        [20, 21],
        [22, 23]]])
```

In [16]:
```
#Q29.Write a code to print a random number between 1 and 6 using random.randint function
import random
outcome = random.randint(1,6)
print(outcome)
```

```
4
```

In [17]:
```
#Q30.Write a code to print 10 random numbers between 1 and 6 using random.randint function
import random
[ random.randint(1, 6) for _ in range(10) ]
```

Out[17]:
```
[6, 3, 2, 4, 3, 4, 4, 6, 4, 1]
```

In [18]:
```
#Q31.Using random.randint()
#Random number between 1 to 7
#Random number between 1 to 7 with size=1
#Random number between 1 to 7 with size =10
#Random number between 1 to 7 with 5 rows and 4 columns
import numpy as np
print(np.random.randint(1, 7))
print(np.random.randint(1, 7, size=1))
print(np.random.randint(1, 7, size=10))
print(np.random.randint(1, 7, size=(5, 4)))
```

```
1
[5]
[1 5 1 4 1 6 4 1 5 2]
[[4 3 5 5]
 [1 6 5 3]
 [5 4 6 2]
 [6 6 6 3]
 [4 6 1 4]]
```

In [19]:
```
#Q32.Create a list using string values and by using random choice function print the random string value from t

from random import choice
possible_destinations = ["Berlin", "Hamburg", "Munich",
                         "Amsterdam", "London", "Paris",
                         "Zurich", "Heidelberg", "Strasbourg",
                         "Augsburg", "Milan", "Rome"]

print(choice(possible_destinations))
```

```
Rome
```

In [20]:
```
#Q33.Using random.random_sample function print the random float values with 3 rows and 4 columns
import numpy as np
x = np.random.random_sample((3, 4))
print(x)
```

```
[[0.56646202 0.53035344 0.04610968 0.25416027]
 [0.71732738 0.12598241 0.88148168 0.19267001]
 [0.91921971 0.03633379 0.81953204 0.48244875]]
```

In [21]:
```
#Q34.Given B = np.array([[42,56,89,65], [99,88,42,12],[55,42,17,18]])
#print False for the values which are  greater than 42 using Boolean indexing
B = np.array([[42,56,89,65],
              [99,88,42,12],
              [55,42,17,18]])

print(B>=42)
```

```
[[ True  True  True  True]
 [ True  True  True False]
 [ True  True False False]]
```

In [22]:
```
#Q35.Write a program of one array to select the corresponding index of
#another array A contains are of C where the corresponding value of (A<=5) is true.C= np.array([123,188,190,99,

C = np.array([123,188,190,99,77,88,100])
A = np.array([4,7,2,8,6,9,5])
R = C[A<=5]
print(R)
```

```
[123 190 100]
```

In [23]:
```
#Q36.Extract from the array np.array([3,4,6,10,24,89,45,43,46,99,100]) with Boolean masking all the number
#which are not divisible by 3
#which are divisible by 5
#which are divisible by 3 and 5
#which are divisible by 3 and set them to 42
import numpy as np
A = np.array([3,4,6,10,24,89,45,43,46,99,100])
div3 = A[A%3!=0]
print("Elements of A not divisible by 3:")
print(div3)
div5 = A[A%5==0]
print("Elements of A divisible by 5:")
print(div5)
print("Elements of A, which are divisible by 3 and 5:")
print(A[(A%3==0) & (A%5==0)])
print("------------------")
A[A%3==0] = 42
print("""New values of A after setting the elements of A,
which are divisible by 3, to 42:""")
print(A)
```

```
Elements of A not divisible by 3:
[  4  10  89  43  46 100]
Elements of A divisible by 5:
[ 10  45 100]
Elements of A, which are divisible by 3 and 5:
[45]
------------------
New values of A after setting the elements of A,
which are divisible by 3, to 42:
[ 42   4  42  10  42  89  42  43  46  42 100]
```

In [24]:
```
                    #DATA MANIPULATION WITH PANDAS
```

In [27]:
```
#Q1.   Write the program to  define a simple Series object in the following example by instantiating a Pandas S
import pandas as pd
S = pd.Series([11, 28, 72, 3, 5, 8])
S
```

Out[27]:
```
0    11
1    28
2    72
3     3
4     5
5     8
dtype: int64
```

In [28]:
```
#Q2.  What is the code that can directly access the index and the values of our Series S ?
print(S.index)
print(S.values)
```

```
RangeIndex(start=0, stop=6, step=1)
[11 28 72  3  5  8]
```

In [29]:
```
#Q3.  If two series are taken S and S2 then write the code for addition of these two series with printing the s

fruits = ['apples', 'oranges', 'cherries', 'pears']
S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits)
print(S + S2)
print("sum of S: ", sum(S))
```

```
apples      37
oranges     46
cherries    83
pears       42
dtype: int64
sum of S:  115
```

In [30]:
```
#Q4.  What will be the program code for the above question if the indices do not have to be the same for the Se
fruits = ['peaches', 'oranges', 'cherries', 'pears']
fruits2 = ['raspberries', 'oranges', 'cherries', 'pears']
S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits2)
print(S + S2)
```

```
cherries       83.0
oranges        46.0
peaches         NaN
pears          42.0
raspberries     NaN
dtype: float64
```

In [37]:
```
#Q5. Write the code  function to extract some elements from the given series object based on the actual positio

# importing pandas as pd
import pandas as pd

# importing re for regular expressions
import re

# Creating the Series
sr = pd.Series(['New_York', 'Lisbon', 'Tokyo', 'Paris', 'Munich'])

# Creating the index
idx = ['City 1', 'City 2', 'City 3', 'City 4', 'City 5']

# set the index
sr.index = idx

# Print the series
print(sr)
```

```
City 1    New_York
City 2      Lisbon
City 3       Tokyo
City 4       Paris
City 5      Munich
dtype: object
```

In [40]:
```
#Q6. Using indexing write the code to access single values of a Series ?
import pandas as pd
fruits = ['apples', 'oranges', 'cherries', 'pears']
S = pd.Series([20, 33, 52, 10], index=fruits)
print(S['apples'])
```

```
20
```

In [41]:
```
#Q7. Write a code for Manipulating Pandas Data frame using Applying lambda function to a column?

import pandas as pd
values = [['Rohan', 455], ['Elvish', 250], ['Deepak', 495],
          ['Sai', 400], ['Radha', 350], ['Vansh', 450]]

df = pd.DataFrame(values, columns=['Name', 'Univ_Marks'])
df = df.assign(Percentage=lambda x: (x['Univ_Marks'] / 500 * 100))
df
```

Out[41]:

	Name	Univ_Marks	Percentage
0	Rohan	455	91.0
1	Elvish	250	50.0
2	Deepak	495	99.0
3	Sai	400	80.0
4	Radha	350	70.0
5	Vansh	450	90.0

In [42]:
```
#Q8.  How to create a Series object in pandas for the resulting Series to contain the dict's keys as the indice
cities = {"London":    8615246,
          "Berlin":    3562166,
          "Madrid":    3165235,
          "Rome":      2874038,
          "Paris":     2273305,
          "Vienna":    1805681,
          "Bucharest": 1803425,
          "Hamburg":   1760433,
          "Budapest":  1754000,
          "Warsaw":    1740119,
          "Barcelona": 1602386,
          "Munich":    1493900,
          "Milan":     1350680}
city_series = pd.Series(cities)
print(city_series)
```

```
London       8615246
Berlin       3562166
Madrid       3165235
Rome         2874038
Paris        2273305
Vienna       1805681
Bucharest    1803425
Hamburg      1760433
Budapest     1754000
Warsaw       1740119
Barcelona    1602386
Munich       1493900
Milan        1350680
dtype: int64
```

In [44]:
```
#Q9. Three series are defined using pandas write the code to concantenate and show the output display ?
import pandas as pd
years = range(2014, 2018)
shop1 = pd.Series([2409.14, 2941.01, 3496.83, 3119.55], index=years)
shop2 = pd.Series([1203.45, 3441.62, 3007.83, 3619.53], index=years)
shop3 = pd.Series([3412.12, 3491.16, 3457.19, 1963.10], index=years)
pd.concat([shop1, shop2, shop3])
```

Out[44]:
```
2014    2409.14
2015    2941.01
2016    3496.83
2017    3119.55
2014    1203.45
2015    3441.62
2016    3007.83
2017    3619.53
2014    3412.12
2015    3491.16
2016    3457.19
2017    1963.10
dtype: float64
```

In [45]:
```
#Q10. Give an example to derive a dataframe from a dictionary using pandas library function ?

cities = {"name": ["London", "Berlin", "Madrid", "Rome",
                   "Paris", "Vienna", "Bucharest", "Hamburg",
                   "Budapest", "Warsaw", "Barcelona",
                   "Munich", "Milan"],
          "population": [8615246, 3562166, 3165235, 2874038,
                         2273305, 1805681, 1803425, 1760433,
                         1754000, 1740119, 1602386, 1493900,
                         1350680],
          "country": ["England", "Germany", "Spain", "Italy",
                      "France", "Austria", "Romania",
                      "Germany", "Hungary", "Poland", "Spain",
                      "Germany", "Italy"]}
city_frame = pd.DataFrame(cities)
city_frame
```

Out[45]:

	name	population	country
0	London	8615246	England
1	Berlin	3562166	Germany
2	Madrid	3165235	Spain
3	Rome	2874038	Italy
4	Paris	2273305	France
5	Vienna	1805681	Austria
6	Bucharest	1803425	Romania
7	Hamburg	1760433	Germany
8	Budapest	1754000	Hungary
9	Warsaw	1740119	Poland
10	Barcelona	1602386	Spain
11	Munich	1493900	Germany
12	Milan	1350680	Italy

In [46]:
```
#Q11. Give an example to derive a dataframe from a dictionary using pandas function?


import pandas
data = {'Ojaswi': {'Age': 15, 'subject': 'java', 'Address': 'Hyderabad'},
        'Rohith':  {'Age': 9, 'subject': 'python', 'Address': 'Hyderabad'},
        'Gnanesh':  {'Age': 15, 'subject': 'c/c++', 'Address': 'Guntur'},
        'divya':  {'Age': 21, 'subject': 'html', 'Address': 'ponnur'},
        'ramya':  {'Age': 15, 'subject': 'c/c++', 'Address': 'delhi'}}

data = pandas.DataFrame(data)
```

```
data
```

Out[46]:

	Ojaswi	Rohith	Gnanesh	divya	ramya
Age	15	9	15	21	15
subject	java	python	c/c++	html	c/c++
Address	Hyderabad	Hyderabad	Guntur	ponnur	delhi

In [48]:
```
#Q12.  Give the program to change both the column order and the ordering of the index with the function reindex
city_frame.reindex(index=[0, 2, 4, 6,  8, 10, 12, 1, 3, 5, 7, 9, 11],
columns=['country', 'name', 'population'])
```

Out[48]:

	country	name	population
0	England	London	8615246
2	Spain	Madrid	3165235
4	France	Paris	2273305
6	Romania	Bucharest	1803425
8	Hungary	Budapest	1754000
10	Spain	Barcelona	1602386
12	Italy	Milan	1350680
1	Germany	Berlin	3562166
3	Italy	Rome	2874038
5	Austria	Vienna	1805681
7	Germany	Hamburg	1760433
9	Poland	Warsaw	1740119
11	Germany	Munich	1493900

In [49]:
```
#Q13. Write the code to rename a dataframe using pandas library function?
city_frame.rename(columns={"name":"Soyadı",
                           "country":"Ülke",
                           "population":"Nüfus"},
                  inplace=True)
city_frame
```

Out[49]:

	Soyadı	Nüfus	Ülke
0	London	8615246	England
1	Berlin	3562166	Germany
2	Madrid	3165235	Spain
3	Rome	2874038	Italy
4	Paris	2273305	France
5	Vienna	1805681	Austria
6	Bucharest	1803425	Romania
7	Hamburg	1760433	Germany
8	Budapest	1754000	Hungary
9	Warsaw	1740119	Poland
10	Barcelona	1602386	Spain
11	Munich	1493900	Germany
12	Milan	1350680	Italy

In [50]:
```
#Q14.  Write the program for accessing row via indexing value ie select the  German cities in the following exa
city_frame = pd.DataFrame(cities,
                          columns=("name", "population"),
                          index=cities["country"])
print(city_frame.loc["Germany"])
```

```
            name  population
Germany   Berlin     3562166
Germany  Hamburg     1760433
Germany   Munich     1493900
```

In [51]:
```
#Q15 . Write a program to perform a pivot table format by reshaping a dataframe in pandas library function ?
import pandas as pd
d = {'A': ['kırmızı', 'yeşil', 'mavi', 'kırmızı', 'yeşil', 'mavi'],
     'B': ['bir', 'iki', 'bir',  'iki',  'bir', 'iki'],
     'C': [345, 325, 898, 989, 23, 143],
     'D': [1, 2, 3, 4, 5, 6]}
df = pd.DataFrame(d)
df
```

Out[51]:

	A	B	C	D
0	kırmızı	bir	345	1
1	yeşil	iki	325	2
2	mavi	bir	898	3
3	kırmızı	iki	989	4
4	yeşil	bir	23	5
5	mavi	iki	143	6

In [53]:
```
#Q16.  Write a program where a Series object with an index of size nvalues. The index will not be unique, becau

import pandas as pd
import numpy as np
import random
nvalues = 30
values = np.random.randint(1, 20, (nvalues,))
fruits = ["bananas", "oranges", "apples", "clementines", "cherries", "pears"]
fruits_index = np.random.choice(fruits, (nvalues,))
s = pd.Series(values, index=fruits_index)
print(s[:10])
```

```
oranges         1
pears           5
clementines    13
pears           5
oranges         1
apples         16
bananas         4
bananas        11
apples         14
cherries        2
dtype: int32
```

In [54]:
```
#Q17. Write a program to get the given series in sorted label form using groupby function so that the solution 
grouped = s.groupby(s.index)
for fruit, s_obj in grouped:
    print(f"===== {fruit} =====")
    print(s_obj)
```

```
===== apples =====
apples    16
apples    14
apples    10
apples     8
dtype: int32
===== bananas =====
bananas     4
bananas    11
bananas     5
bananas    10
bananas     5
dtype: int32
===== cherries =====
cherries     2
cherries     5
cherries    15
cherries     5
cherries     9
cherries     9
cherries    17
cherries    15
dtype: int32
===== clementines =====
clementines    13
clementines    19
clementines    19
dtype: int32
===== oranges =====
oranges    1
oranges    1
oranges    3
oranges    9
dtype: int32
===== pears =====
pears     5
pears     5
pears    11
pears    16
pears     1
pears     8
dtype: int32
```

In [55]:
```
#Q18.  The DataFrame has two columns one containing names Name and the other one coffee contains integers which
import pandas as pd
beverages = pd.DataFrame({'Name': ['Robert', 'Melinda', 'Brenda',
```

```
                                'Samantha', 'Melinda', 'Robert',
                                'Melinda', 'Brenda', 'Samantha'],
                    'Coffee': [3, 0, 2, 2, 0, 2, 0, 1, 3],
                    'Tea':    [0, 4, 2, 0, 3, 0, 3, 2, 0]})
beverages
```

Out[55]:

	Name	Coffee	Tea
0	Robert	3	0
1	Melinda	0	4
2	Brenda	2	2
3	Samantha	2	0
4	Melinda	0	3
5	Robert	2	0
6	Melinda	0	3
7	Brenda	1	2
8	Samantha	3	0

In [56]:
```
#Q19. Write the program to calculate the average number of coffee and tea cups the persons had using gropby fun
beverages.groupby(['Name']).mean()
```

Out[56]:

Name	Coffee	Tea
Brenda	1.5	2.000000
Melinda	0.0	3.333333
Robert	2.5	0.000000
Samantha	2.5	0.000000

In [67]:
```
#Q20.  Write a program with function 'find bin' with two list or tuple of bins of two elements to find the inde
def create_bins(lower_bound, width, quantity):
    bins = []
    for low in range(lower_bound,
                     lower_bound + quantity*width + 1, width):
        bins.append((low, low+width))
    return bins


bins = create_bins(lower_bound=10,
                   width=10,
                   quantity=5)

bins
def find_bin(value, bins):


    for i in range(0, len(bins)):
        if bins[i][0] <= value < bins[i][1]:
            return i
    return -1
from collections import Counter

bins = create_bins(lower_bound=50,
                   width=4,
                   quantity=10)

print(bins)

weights_of_persons = [73.4, 69.3, 64.9, 75.6, 74.9, 80.3,
                      78.6, 84.1, 88.9, 90.3, 83.4, 69.3,
                      52.4, 58.3, 67.4, 74.0, 89.3, 63.4]

binned_weights = []

for value in weights_of_persons:
    bin_index = find_bin(value, bins)
    print(value, bin_index, bins[bin_index])
    binned_weights.append(bin_index)

frequencies = Counter(binned_weights)
print(frequencies)
```

```
[(50, 54), (54, 58), (58, 62), (62, 66), (66, 70), (70, 74), (74, 78), (78, 82), (82, 86), (86, 90), (90, 94)]
73.4 5 (70, 74)
69.3 4 (66, 70)
64.9 3 (62, 66)
75.6 6 (74, 78)
74.9 6 (74, 78)
80.3 7 (78, 82)
78.6 7 (78, 82)
84.1 8 (82, 86)
88.9 9 (86, 90)
90.3 10 (90, 94)
83.4 8 (82, 86)
69.3 4 (66, 70)
52.4 0 (50, 54)
58.3 2 (58, 62)
67.4 4 (66, 70)
74.0 6 (74, 78)
89.3 9 (86, 90)
63.4 3 (62, 66)
Counter({4: 3, 6: 3, 3: 2, 7: 2, 8: 2, 9: 2, 5: 1, 10: 1, 0: 1, 2: 1})
```

In [68]:
```python
#Q21.  Write the code for multilevel indexing using Pandas data structures. It's an efficient way to store and

import pandas as pd

cities = ["Vienna", "Vienna", "Vienna",
          "Hamburg", "Hamburg", "Hamburg",
          "Berlin", "Berlin", "Berlin",
          "Zürich", "Zürich", "Zürich"]
index = [cities, ["country", "area", "population",
                  "country", "area", "population",
                  "country", "area", "population",
                  "country", "area", "population"]]
print(index)
```

```
[['Vienna', 'Vienna', 'Vienna', 'Hamburg', 'Hamburg', 'Hamburg', 'Berlin', 'Berlin', 'Berlin', 'Zürich', 'Züric
h', 'Zürich'], ['country', 'area', 'population', 'country', 'area', 'population', 'country', 'area', 'populatio
n', 'country', 'area', 'population']]
```

In [71]:
```python
#Q22. Write the program to sort the index using slicing operation for the given series in pandas function ?
city_series = city_series.sort_index()
print("city_series with sorted index:")
print(city_series)
print("\n\nSlicing the city_series:")
city_series["Berlin":"Vienna"]
```

```
city_series with sorted index:
Barcelona    1602386
Berlin       3562166
Bucharest    1803425
Budapest     1754000
Hamburg      1760433
London       8615246
Madrid       3165235
Milan        1350680
Munich       1493900
Paris        2273305
Rome         2874038
Vienna       1805681
Warsaw       1740119
dtype: int64


Slicing the city_series:
```

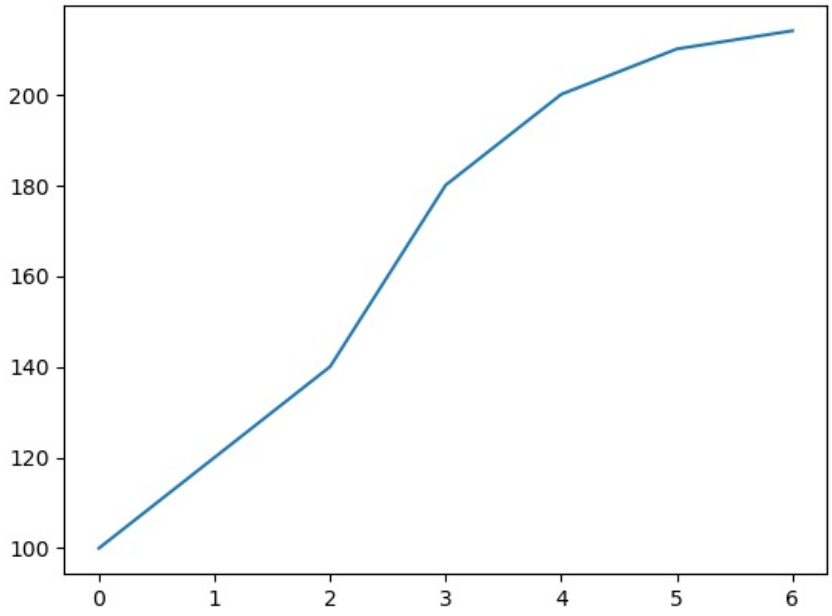
Out[71]:
```
Berlin       3562166
Bucharest    1803425
Budapest     1754000
Hamburg      1760433
London       8615246
Madrid       3165235
Milan        1350680
Munich       1493900
Paris        2273305
Rome         2874038
Vienna       1805681
dtype: int64
```

In [79]:
```python
#Q23.   Write a program to perform swapping multidex levels using pandas library function ?
import pandas as pd

city_series.sort_index(inplace=True)
city_series
```

Out[79]:
```
Barcelona    1602386
Berlin       3562166
Bucharest    1803425
Budapest     1754000
Hamburg      1760433
London       8615246
Madrid       3165235
Milan        1350680
Munich       1493900
Paris        2273305
Rome         2874038
Vienna       1805681
Warsaw       1740119
dtype: int64
```

In [80]:
```
#Q24 . If given a tuple data =[100, 120, 140, 180, 200, 210, 214] ,using pandas   series  function plot a line
import pandas as pd
data = [100, 120, 140, 180, 200, 210, 214]
s = pd.Series(data, index=range(len(data)))
s.plot()
```

Out[80]:
```
<Axes: >
```



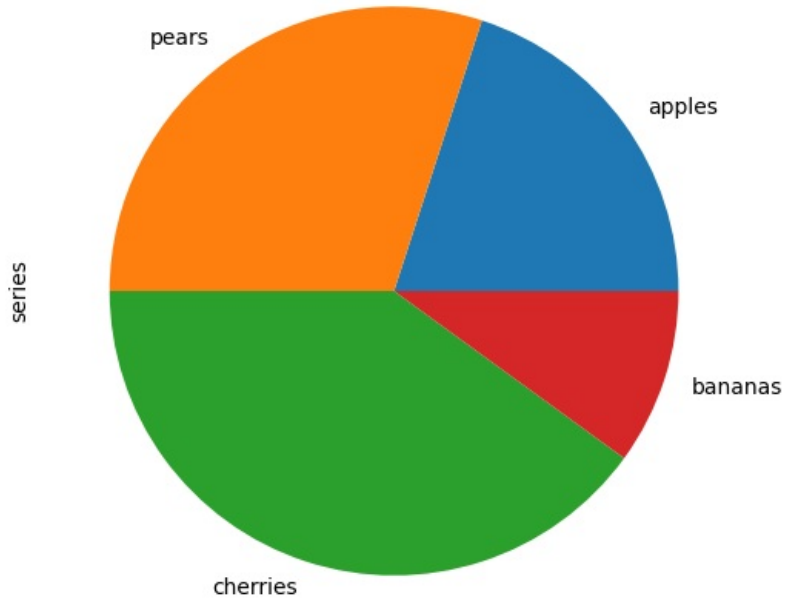
In [82]:
```
#Q25.  For the  defined  dcitionary with the population and area figures. This dictionary can be used to create
import pandas as pd
cities = {"name": ["London", "Berlin", "Madrid", "Rome",
                   "Paris", "Vienna", "Bucharest", "Hamburg",
                   "Budapest", "Warsaw", "Barcelona",
                   "Munich", "Milan"],
          "population": [8615246, 3562166, 3165235, 2874038,
                         2273305, 1805681, 1803425, 1760433,
                         1754000, 1740119, 1602386, 1493900,
                         1350680],
          "area" : [1572, 891.85, 605.77, 1285,
                    105.4, 414.6, 228, 755,
                    525.2, 517, 101.9, 310.4,
                    181.8]
}
city_frame = pd.DataFrame(cities,
                          columns=["population", "area"],
                          index=cities["name"])
print(city_frame)
```

```
           population     area
London        8615246  1572.00
Berlin        3562166   891.85
Madrid        3165235   605.77
Rome          2874038  1285.00
Paris         2273305   105.40
Vienna        1805681   414.60
Bucharest     1803425   228.00
Hamburg       1760433   755.00
Budapest      1754000   525.20
Warsaw        1740119   517.00
Barcelona     1602386   101.90
Munich        1493900   310.40
Milan         1350680   181.80
```

In [83]:
```
#Q26. Wite a program for pie chart diagram in pandas for the given series using plot function ?
import pandas as pd
fruits = ['apples', 'pears', 'cherries', 'bananas']
series = pd.Series([20, 30, 40, 10],
```

```
                  index=fruits,
                  name='series')
series.plot.pie(figsize=(6, 6))
```

Out[83]:
```
<Axes: ylabel='series'>
```



In [85]:
```
#Q27.  Write a program to print the date and time using pandas date-time function?
from datetime import date
x = date(1993, 12, 14)
print(x)
```

```
1993-12-14
```

In [86]:
```
#Q28.  Write a program to instantiate dates in the range from January 1, 1 to December 31, 9999. This can be in
from datetime import date
print(date.min)
print(date.max)
```

```
0001-01-01
9999-12-31
```

In [87]:
```
#Q29. Write a program to show an output a dataframe of data and time ?
import pandas as pd
data = pd.date_range('1/1/2011', periods = 10, freq ='H')

data
```

Out[87]:
```
DatetimeIndex(['2011-01-01 00:00:00', '2011-01-01 01:00:00',
               '2011-01-01 02:00:00', '2011-01-01 03:00:00',
               '2011-01-01 04:00:00', '2011-01-01 05:00:00',
               '2011-01-01 06:00:00', '2011-01-01 07:00:00',
               '2011-01-01 08:00:00', '2011-01-01 09:00:00'],
              dtype='datetime64[ns]', freq='H')
```

In [88]:
```
#Q30. Using time series in pandas write the program to find the index consisting of time stamps?
import numpy as np
import pandas as pd
from datetime import datetime, timedelta as delta
ndays = 10
start = datetime(2017, 3, 31)
dates = [start - delta(days=x) for x in range(0, ndays)]
values = [25, 50, 15, 67, 70, 9, 28, 30, 32, 12]
ts = pd.Series(values, index=dates)
ts
```

Out[88]:
```
2017-03-31    25
2017-03-30    50
2017-03-29    15
2017-03-28    67
2017-03-27    70
2017-03-26     9
2017-03-25    28
2017-03-24    30
2017-03-23    32
2017-03-22    12
dtype: int64
```

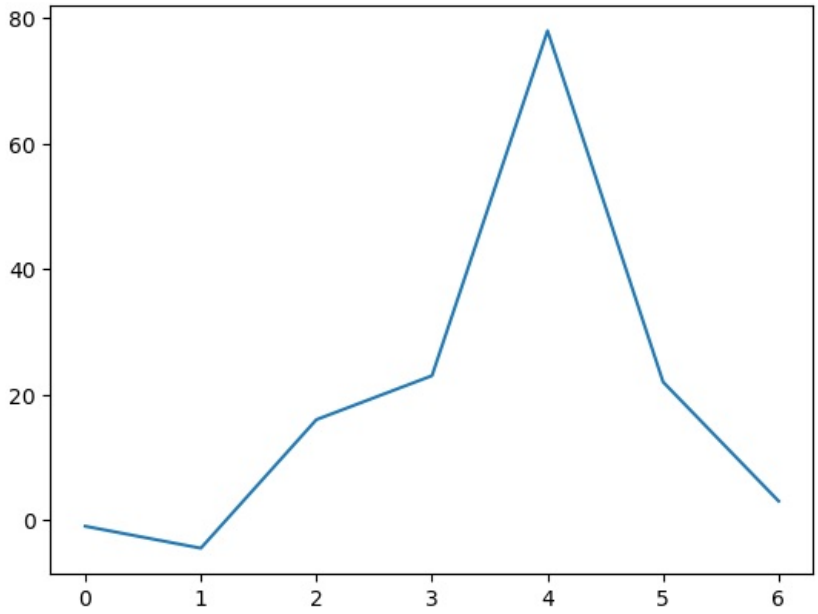
In [89]:
```
#Q31. Write a program to using a function to return the values lying in the given time duration?
```

```
import pandas as pd
sr = pd.Series([11, 21, 8, 18, 65, 18, 32, 10, 5, 32, None])
index_ = pd.date_range('2010-10-09 08:45', periods = 11, freq ='H')
sr.index = index_
print(sr)
```

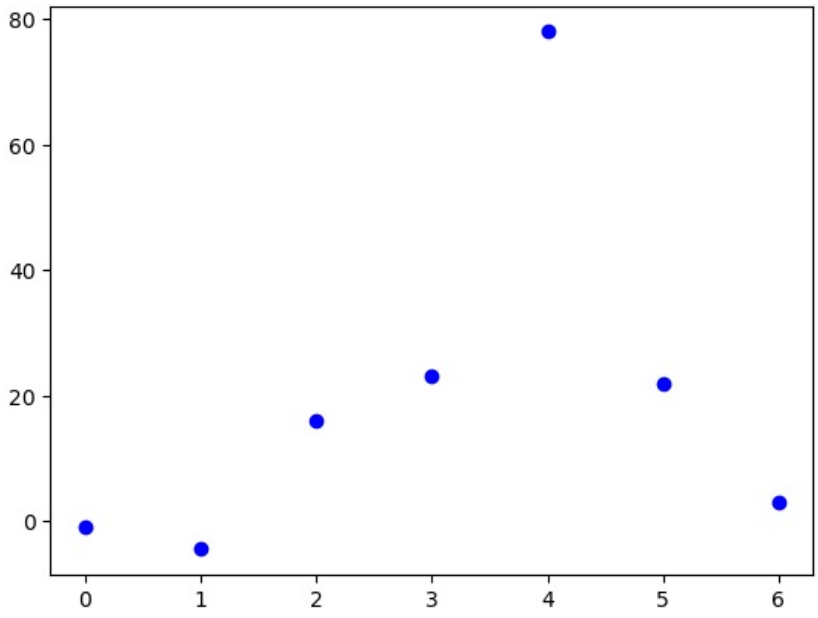
```
2010-10-09 08:45:00    11.0
2010-10-09 09:45:00    21.0
2010-10-09 10:45:00     8.0
2010-10-09 11:45:00    18.0
2010-10-09 12:45:00    65.0
2010-10-09 13:45:00    18.0
2010-10-09 14:45:00    32.0
2010-10-09 15:45:00    10.0
2010-10-09 16:45:00     5.0
2010-10-09 17:45:00    32.0
2010-10-09 18:45:00     NaN
Freq: H, dtype: float64
```

In [90]:
```
#VISUALIZATION WITH MATPLOTLIB
```

In [91]:
```
#Q1.Write a python code for simple line plot .
import matplotlib.pyplot as plt
plt.plot([-1, -4.5, 16, 23, 78, 22, 3])
plt.show()
```



In [92]:
```
#Q2.Write a python code for scatter plot .
import matplotlib.pyplot as plt
plt.plot([-1, -4.5, 16, 23, 78, 22, 3], "ob")
plt.show()
```

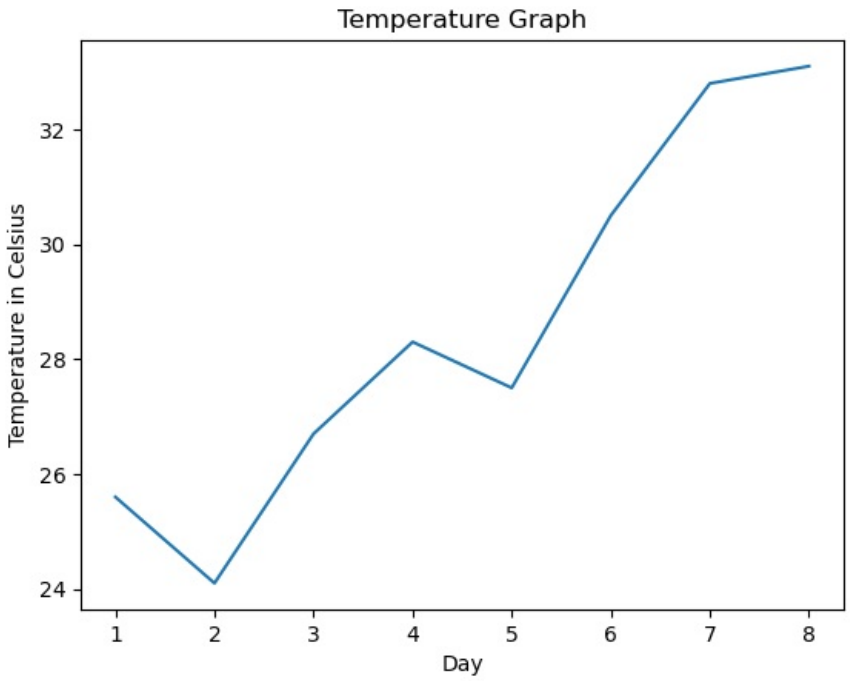


In [93]:
```
#Q3.Write a python code for line plot and display with axis name with title.
#(i)x-axis named as “day”
#(ii)y-axis named as “Temperature in Celsius.”
#(iii) title named as “Temperature Graph.”
```

```
import matplotlib.pyplot as plt
days = range(1, 9)
celsius_values = [25.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
fig, ax = plt.subplots()
ax.plot(days, celsius_values)
ax.set(xlabel='Day',
       ylabel='Temperature in Celsius',
       title='Temperature Graph')
```

Out[93]:

```
[Text(0.5, 0, 'Day'),
 Text(0, 0.5, 'Temperature in Celsius'),
 Text(0.5, 1.0, 'Temperature Graph')]
```

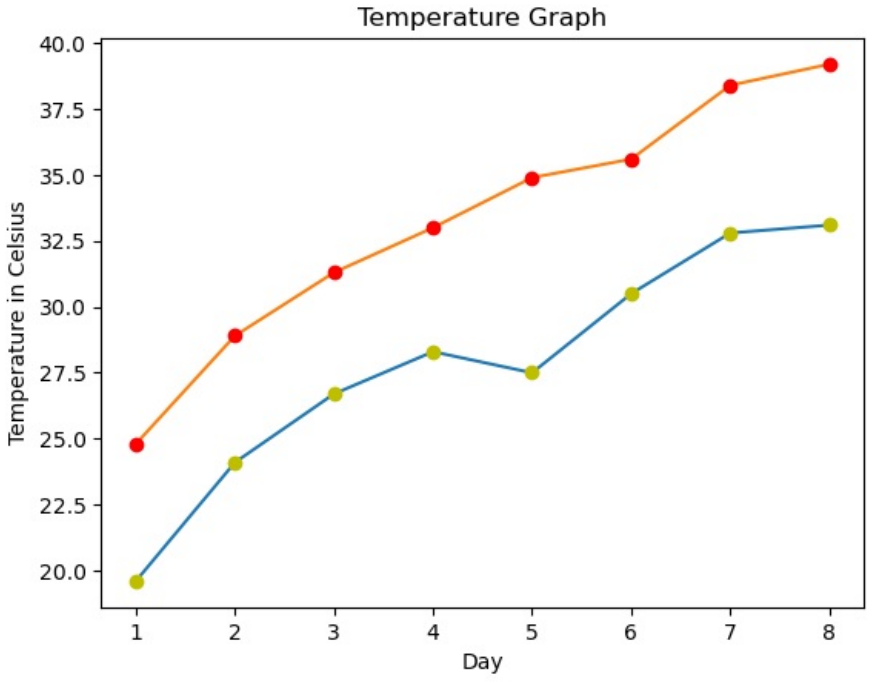


In [94]:

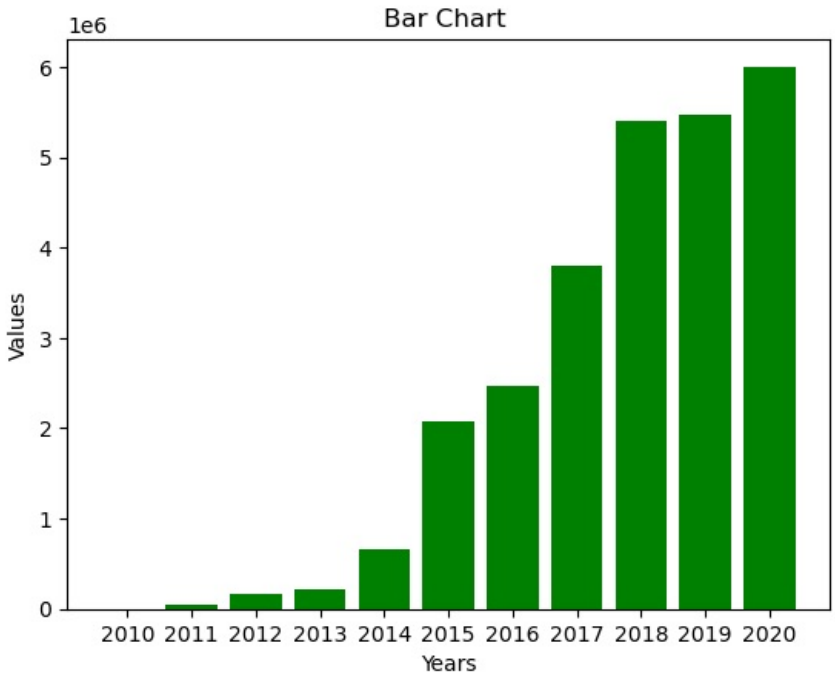
```
#Q4.Write a python code for multiple plot(scatter,line).
import matplotlib.pyplot as plt
days = list(range(1,9))
celsius_min = [19.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
celsius_max = [24.8, 28.9, 31.3, 33.0, 34.9, 35.6, 38.4, 39.2]
fig, ax = plt.subplots()
ax.set(xlabel='Day',
       ylabel='Temperature in Celsius',
       title='Temperature Graph')
ax.plot(days, celsius_min,
        days, celsius_min, "oy",
        days, celsius_max,
        days, celsius_max, "or")
```

Out[94]:

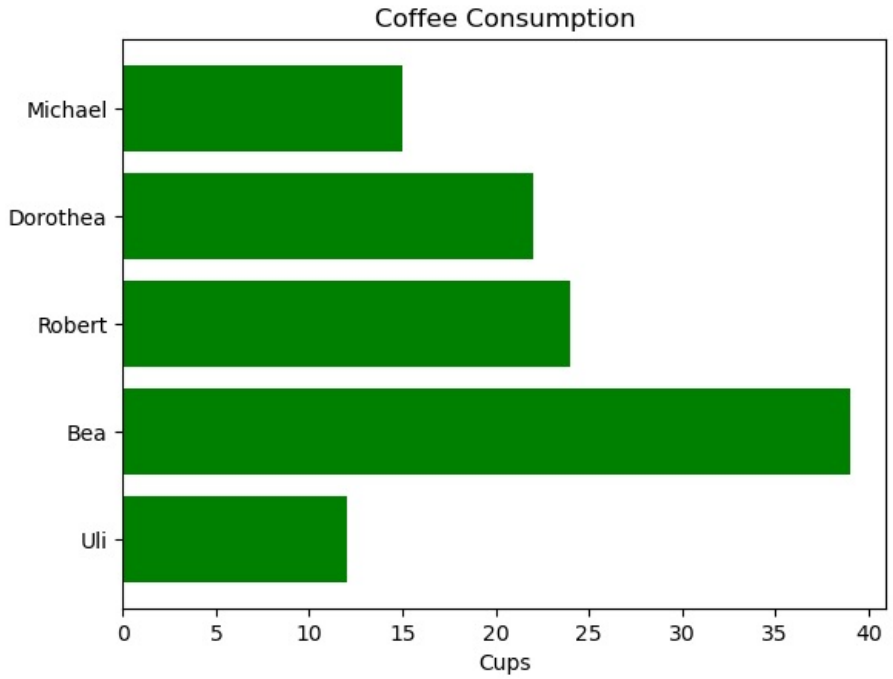
```
[<matplotlib.lines.Line2D at 0x12cb71da2c0>,
 <matplotlib.lines.Line2D at 0x12cb71da290>,
 <matplotlib.lines.Line2D at 0x12cb71da230>,
 <matplotlib.lines.Line2D at 0x12cb71da260>]
```



In [95]:
```
#Q5.Write a python code to display the bar plot.
import matplotlib.pyplot as plt
import numpy as np
years = [str(year) for year in range(2010, 2021)]
visitors = (1241, 50927, 162242, 222093,
            665004, 2071987, 2460407, 3799215,
            5399000, 5474016, 6003672)
plt.bar(years, visitors, color="green")
plt.xlabel("Years")
plt.ylabel("Values")
plt.title("Bar Chart ")
plt.plot()
plt.show()
```

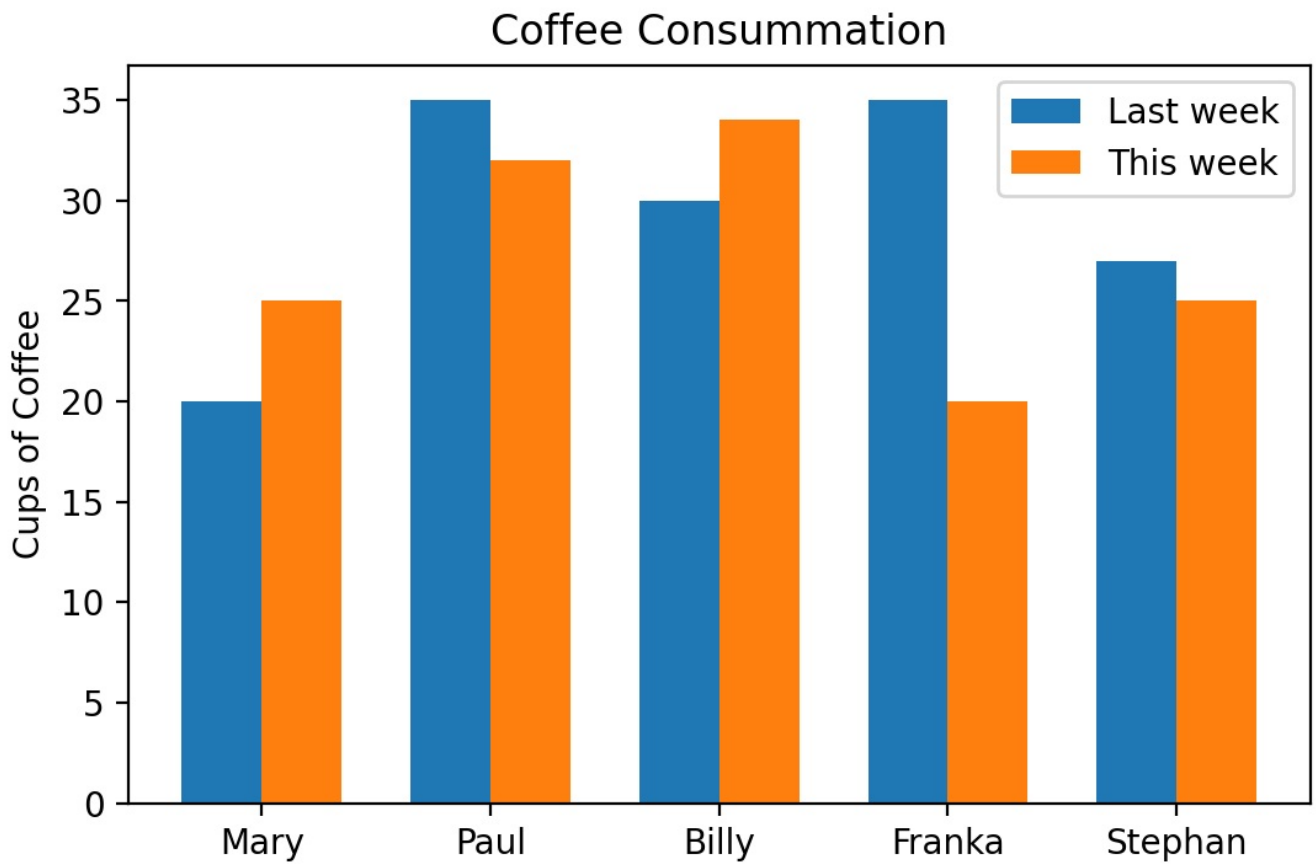


In [97]:
```
#Q7.write a python code to display the bar plot in vertical order .
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
# restore default parameters:
plt.rcdefaults()
fig, ax = plt.subplots()
personen = ('Michael', 'Dorothea', 'Robert', 'Bea', 'Uli')
y_pos = np.arange(len(personen))
cups = (15, 22, 24, 39, 12)
ax.barh(y_pos, cups, align='center',
        color='green', ecolor='black')
ax.set_yticks(y_pos)
ax.set_yticklabels(personen)
ax.invert_yaxis()
ax.set_xlabel('Cups')
ax.set_title('Coffee Consumption')
plt.show()
```



In [98]:

```
#Q8.Write a python code for grouped  bar charts .
import matplotlib.pyplot as plt
import numpy as np

last_week_cups = (20, 35, 30, 35, 27)
this_week_cups = (25, 32, 34, 20, 25)
names = ['Mary', 'Paul', 'Billy', 'Franka', 'Stephan']

fig = plt.figure(figsize=(6,5), dpi=200)
left, bottom, width, height = 0.1, 0.3, 0.8, 0.6
ax = fig.add_axes([left, bottom, width, height])

width = 0.35
ticks = np.arange(len(names))
ax.bar(ticks, last_week_cups, width, label='Last week')
ax.bar(ticks + width, this_week_cups, width, align="center",
    label='This week')

ax.set_ylabel('Cups of Coffee')
ax.set_title('Coffee Consummation')
ax.set_xticks(ticks + width/2)
ax.set_xticklabels(names)

ax.legend(loc='best')
plt.show()
```

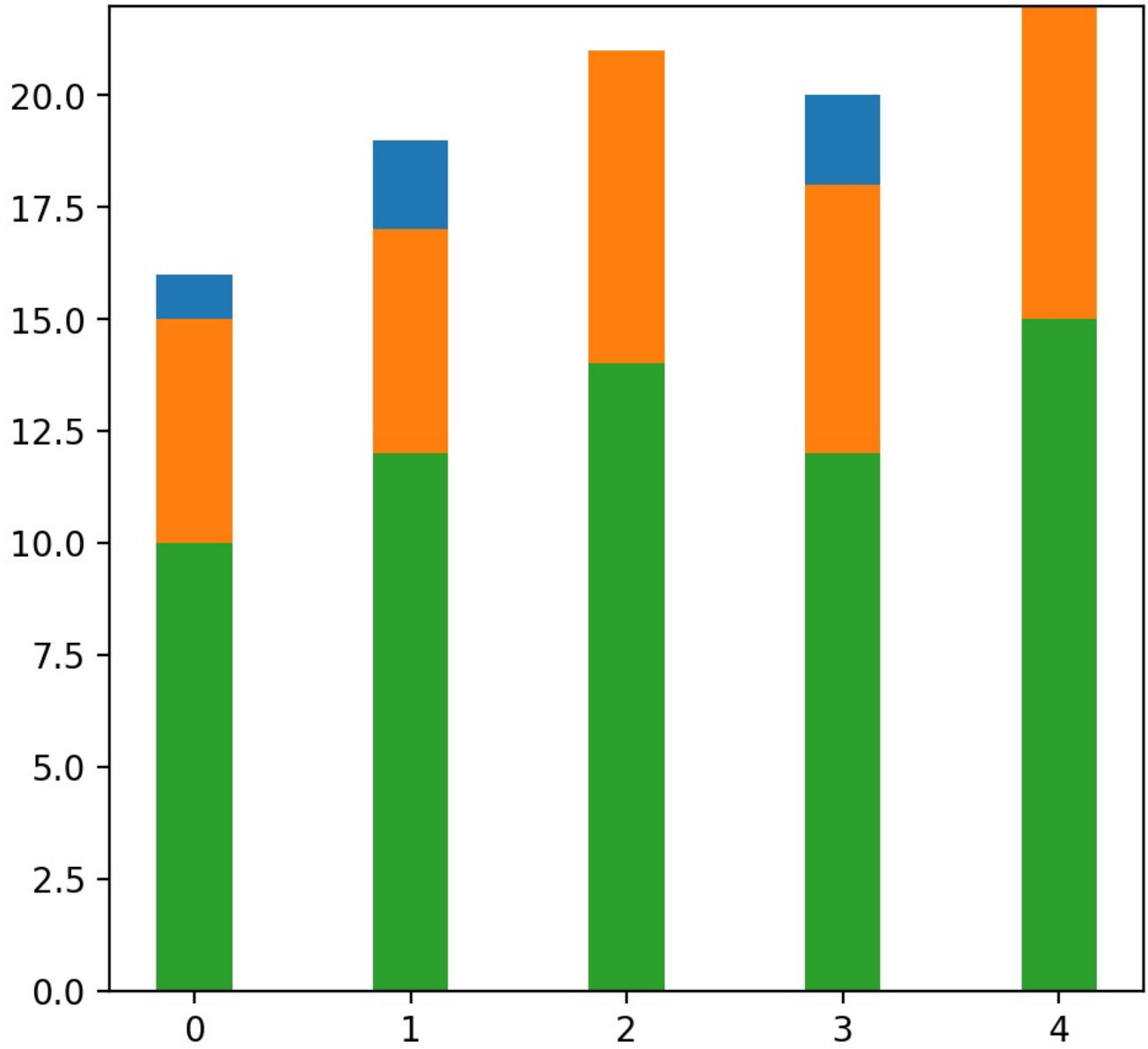


In [101…

```python
#Q9.Write a python code to display the stacked bar  chart .
import matplotlib.pyplot as plt
import numpy as np

coffee = np.array([5, 5, 7, 6, 7])
tea = np.array([1, 2, 0, 2, 0])
water = np.array([10, 12, 14, 12, 15])
names = ['Mary', 'Paul', 'Billy', 'Franka', 'Stephan']

fig = plt.figure(figsize=(6,5), dpi=200)
left, bottom, width, height = 0.2, 0.1, 0.7, 0.8
ax = fig.add_axes([left, bottom, width, height])

width = 0.35
ticks = np.arange(len(names))
ax.bar(ticks, tea, width, label='Coffee', bottom=water+coffee)
ax.bar(ticks, coffee, width, align="center", label='Tea',
       bottom=water)
ax.bar(ticks, water, width, align="center", label='Water')
```

Out[101]:

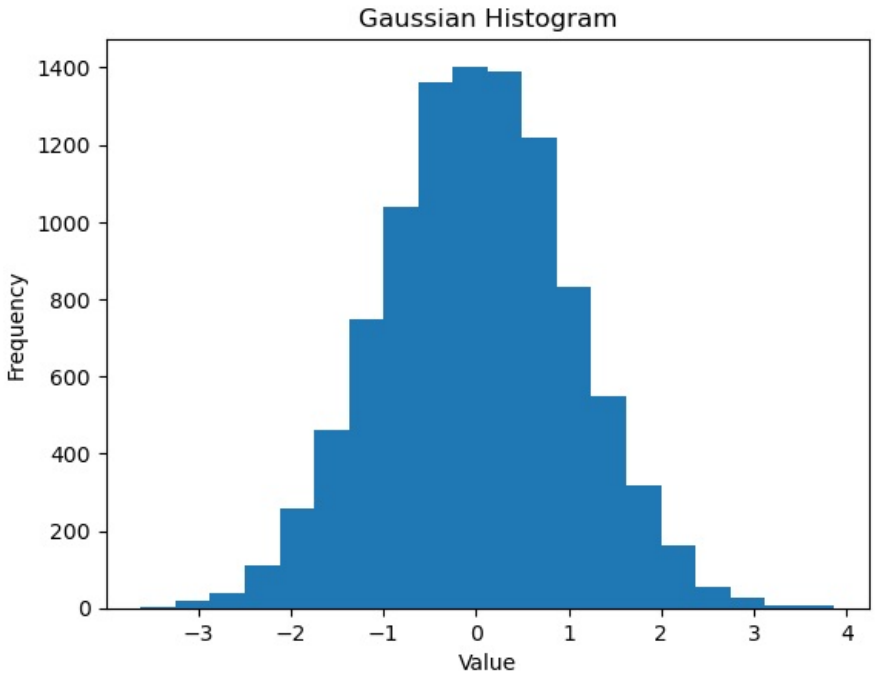
```
<BarContainer object of 5 artists>
```



In [102…

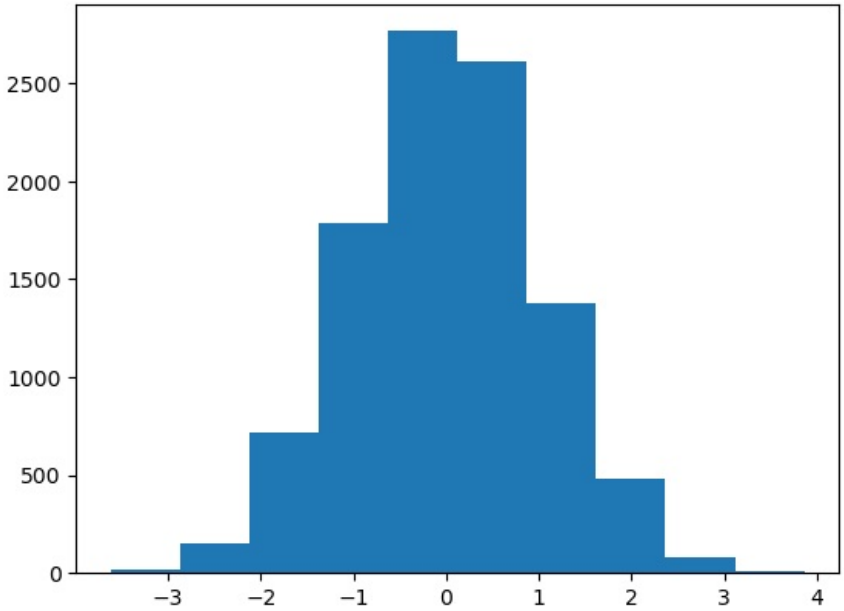
```
#Q10.Write a python code  for histogram .
import matplotlib.pyplot as plt
import numpy as np
gaussian_numbers = np.random.normal(size=10000)
gaussian_numbers
plt.hist(gaussian_numbers, bins=20)
plt.title("Gaussian Histogram")
```

```
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

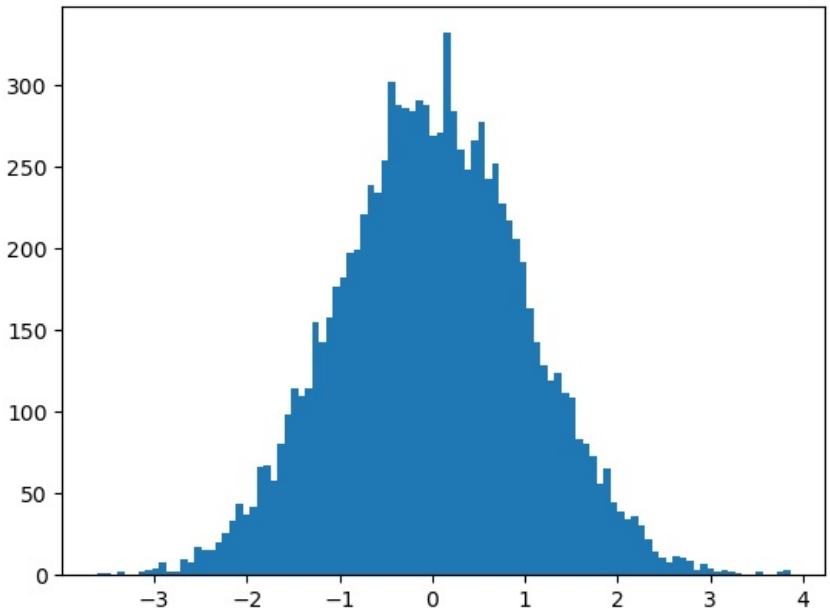


In [103…
```
#Q11.Write a python code to display the histogram using binnings .
n, bins, patches = plt.hist(gaussian_numbers)
print("n: ", n, sum(n))
print("bins: ", bins)
for i in range(len(bins)-1):
    print(bins[i+1] -bins[i])
print("patches: ", patches)
print(patches[1])
print(patches[2])
```

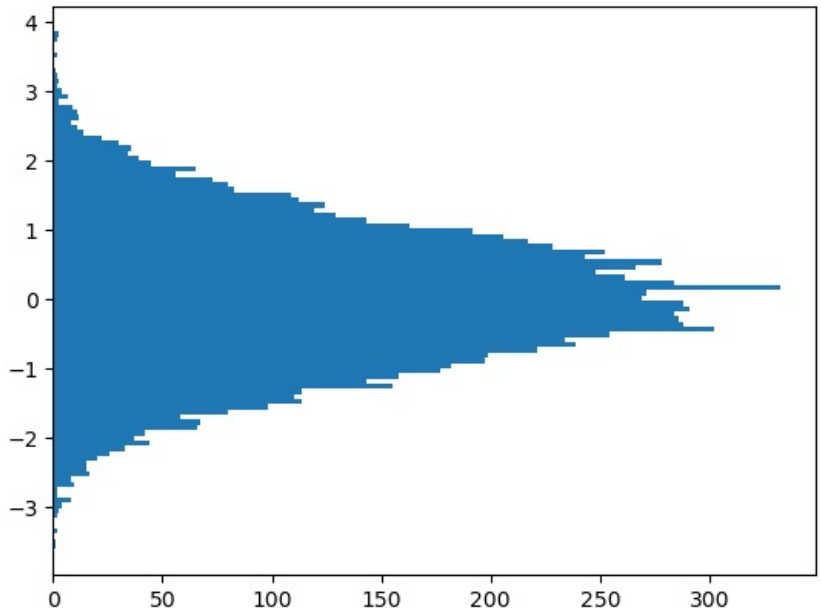
```
n:  [  21.  148.  716. 1785. 2767. 2609. 1380.  480.   81.   13.] 10000.0
bins:  [-3.61820543 -2.86987527 -2.12154511 -1.37321495 -0.62488479  0.12344538
  0.87177554  1.6201057   2.36843586  3.11676602  3.86509618]
0.7483301604501369
0.7483301604501369
0.7483301604501369
0.7483301604501369
0.7483301604501369
0.7483301604501369
0.7483301604501369
0.7483301604501369
0.7483301604501369
0.7483301604501365
patches:  <BarContainer object of 10 artists>
Rectangle(xy=(-2.86988, 0), width=0.74833, height=148, angle=0)
Rectangle(xy=(-2.12155, 0), width=0.74833, height=716, angle=0)
```



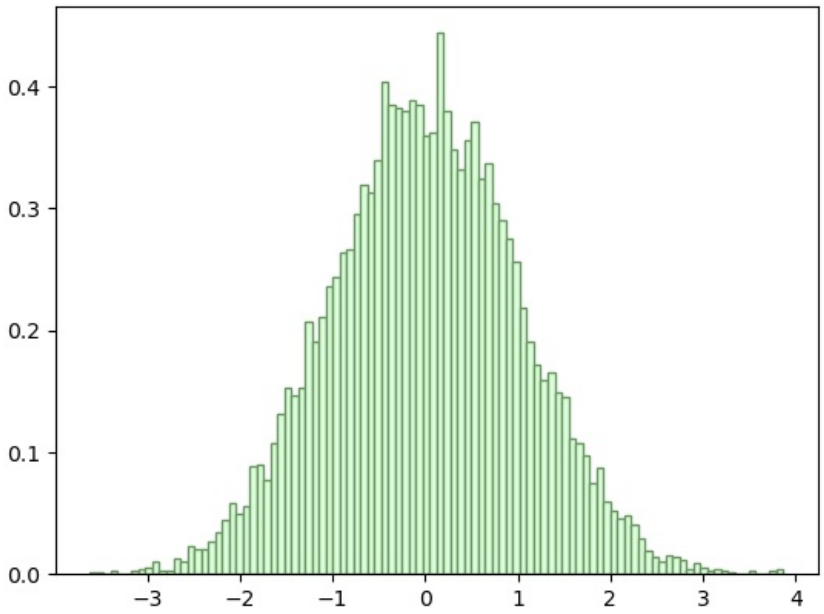
In [104…
```
#Q12.Write a python code to display histogram to increase the no.of.binnings
plt.hist(gaussian_numbers, bins=100)
plt.show()
```



In [105…
```python
#Q13.Write a python code to display the histogram in horizontal orientation.
plt.hist(gaussian_numbers,
         bins=100,
         orientation="horizontal")
plt.show()
```

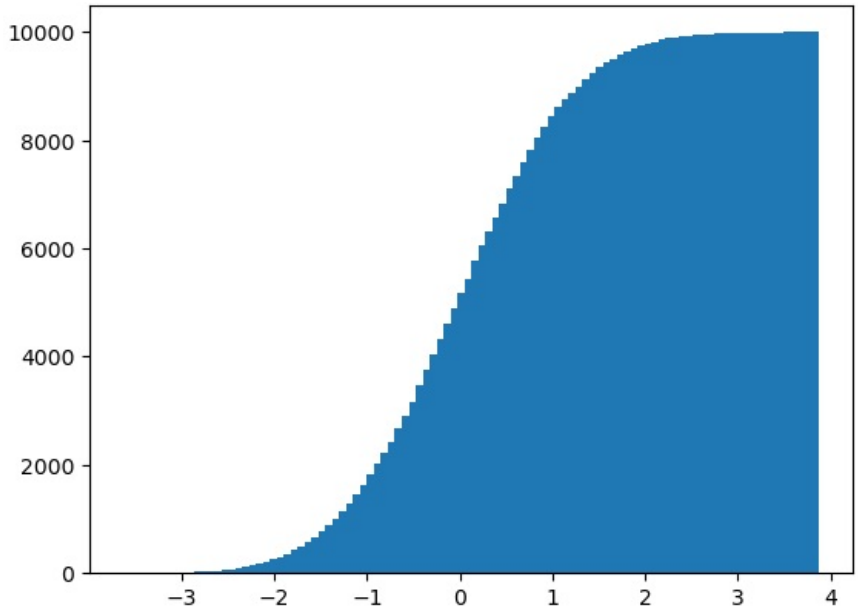


In [106…
```python
#Q14.Write a python code to display the histogram with edge color.
plt.hist(gaussian_numbers,
         bins=100,
         density=True,
         stacked=True,
         edgecolor="#6A9662",
         color="#DDFFDD")
plt.show()
```



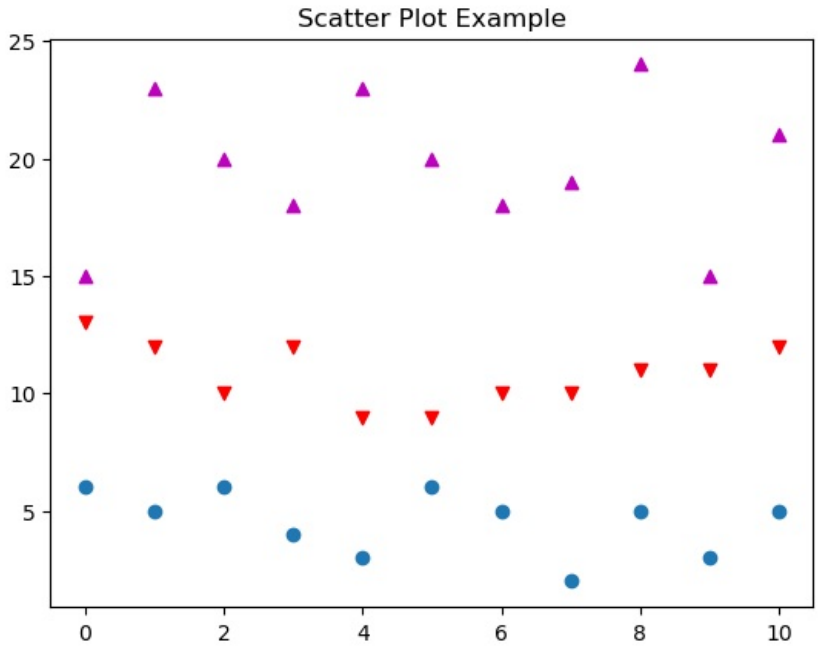
In [107…

```
#Q15.Write a python code to display the histogram in cumulative using bins .
plt.hist(gaussian_numbers,
         bins=100,
         stacked=True,
         cumulative=True)
plt.show()
```



In [108…

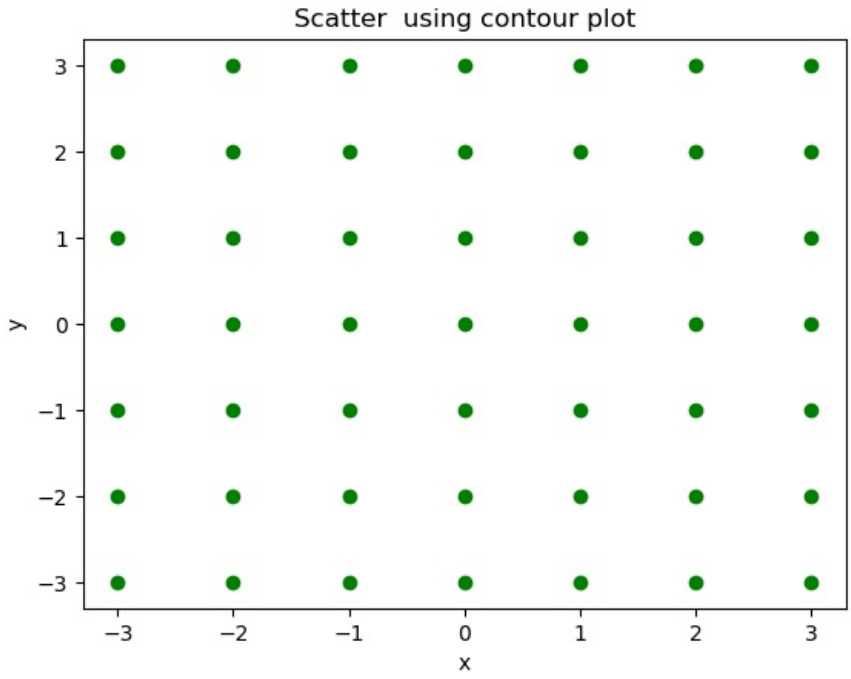
```
#Q16.Write a python code to display scatter plot using  markers.
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 11)
y1 = np.random.randint(2, 7, (11,))
y2 = np.random.randint(9, 14, (11,))
y3 = np.random.randint(15, 25, (11,))
plt.scatter(x, y1)
plt.scatter(x, y2, marker='v', color='r')
plt.scatter(x, y3, marker='^', color='m')
plt.title('Scatter Plot Example')
plt.show()
```



In [109…
```
#Q17.Write  a python code for contour plot without charts  .
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
n, m = 7, 7
start = -3
x_vals = np.arange(start, start+n, 1)
y_vals = np.arange(start, start+m, 1)
X, Y = np.meshgrid(x_vals, y_vals)
print(X)
print(Y)
```

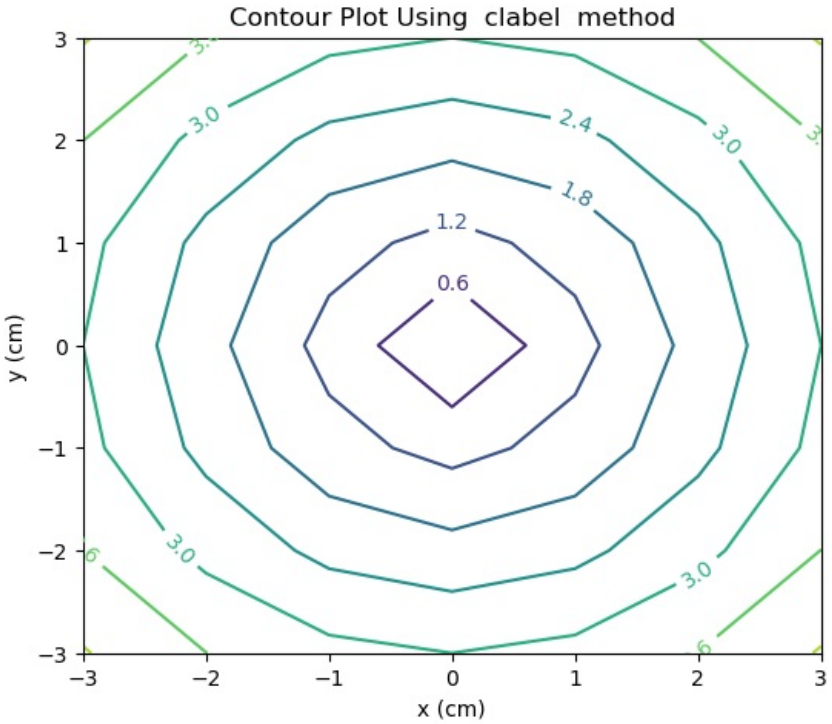
```
[[-3 -2 -1  0  1  2  3]
 [-3 -2 -1  0  1  2  3]
 [-3 -2 -1  0  1  2  3]
 [-3 -2 -1  0  1  2  3]
 [-3 -2 -1  0  1  2  3]
 [-3 -2 -1  0  1  2  3]
 [-3 -2 -1  0  1  2  3]]
[[-3 -3 -3 -3 -3 -3 -3]
 [-2 -2 -2 -2 -2 -2 -2]
 [-1 -1 -1 -1 -1 -1 -1]
 [ 0  0  0  0  0  0  0]
 [ 1  1  1  1  1  1  1]
 [ 2  2  2  2  2  2  2]
 [ 3  3  3  3  3  3  3]]
```

In [110…
```
#Q18.Write a  python code to display the scatter diagram using contour plot.
#(i)    X axis named as “x”
#(ii)   Y-axis named as “y”
#(iii)  Titled named as “Scatter  using contour plot”
fig, ax = plt.subplots()
ax.scatter(X, Y, color="green")
ax.set_title('Scatter  using contour plot')
ax.set_xlabel('x')
ax.set_ylabel('y')
plt.show()
```



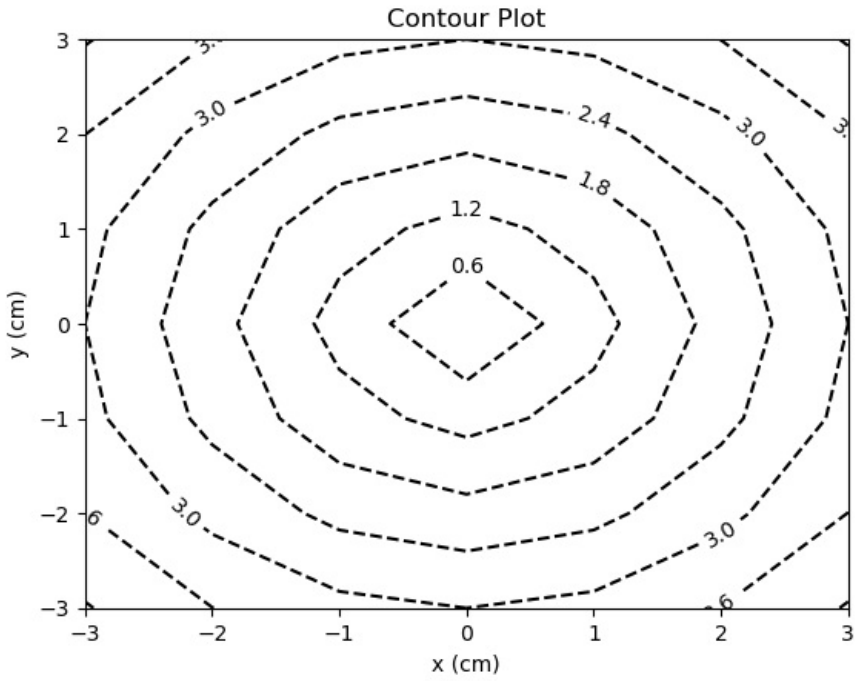
In [111…

```
#Q19.Write a python code to display contour plot using clabel method.
fig = plt.figure(figsize=(6,5))
left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
ax = fig.add_axes([left, bottom, width, height])
Z = np.sqrt(X**2 + Y**2)
cp = ax.contour(X, Y, Z)
ax.clabel(cp, inline=True,
          fontsize=10)
ax.set_title('Contour Plot Using  clabel  method')
ax.set_xlabel('x (cm)')
ax.set_ylabel('y (cm)')
plt.show()
```



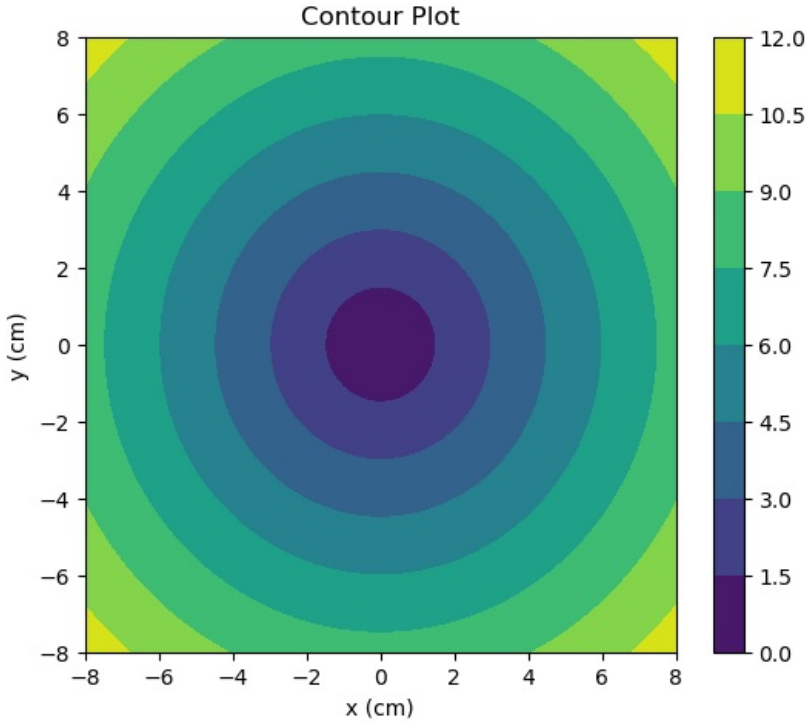
In [112…

```
#Q20.Write a python code  for changing the linestyle= “dashed line” and color=”black” in contour plot.
import matplotlib.pyplot as plt
plt.figure()
cp = plt.contour(X, Y, Z, colors='black', linestyles='dashed')
plt.clabel(cp, inline=True,
          fontsize=10)
plt.title('Contour Plot')
plt.xlabel('x (cm)')
plt.ylabel('y (cm)')
plt.show()
```



In [113…

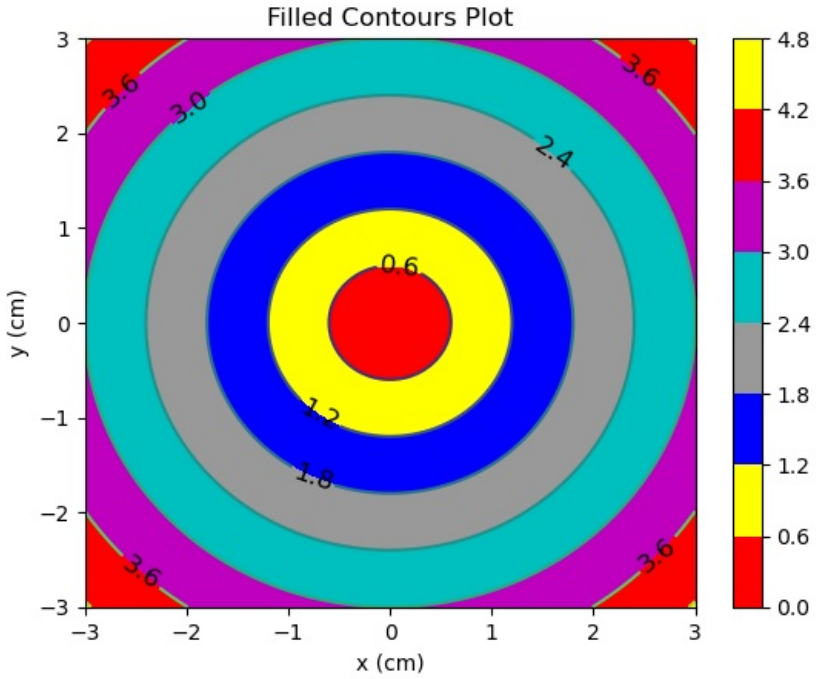
```
#Q21.Write a python code for filled color using contour plot .
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure(figsize=(6,5))
left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
ax = fig.add_axes([left, bottom, width, height])
start, stop, n_values = -8, 8, 800
x_vals = np.linspace(start, stop, n_values)
y_vals = np.linspace(start, stop, n_values)
X, Y = np.meshgrid(x_vals, y_vals)
Z = np.sqrt(X**2 + Y**2)
cp = plt.contourf(X, Y, Z)
plt.colorbar(cp)
ax.set_title('Contour Plot')
ax.set_xlabel('x (cm)')
ax.set_ylabel('y (cm)')
plt.show()
```



In [114…

```
#Q22.Write a python code for to displaty individual color using contour plot.
import numpy as np
import matplotlib.pyplot as plt
xlist = np.linspace(-3.0, 3.0, 100)
ylist = np.linspace(-3.0, 3.0, 100)
X, Y = np.meshgrid(xlist, ylist)
Z = np.sqrt(X**2 + Y**2)
plt.figure()
contour = plt.contour(X, Y, Z)
plt.clabel(contour, colors = 'k', fmt = '%2.1f', fontsize=12)
```

```
c = ('#ff0000', '#ffff00', '#0000FF', '0.6', 'c', 'm')
contour_filled = plt.contourf(X, Y, Z, colors=c)
plt.colorbar(contour_filled)
plt.title('Filled Contours Plot')
plt.xlabel('x (cm)')
plt.ylabel('y (cm)')
plt.savefig('contourplot_own_colours.png', dpi=300)
plt.show()
```



In [ ]: