python-assignment / assignment 1-230901023.ipynb

854 lines (854 loc) · 295 KB

In [ ]:

```
#QUESTION-1A

import pandas as pd
scores=[90,80,54,77,40,49,60,99,44,77,67,87,48,93,100,59,78,98,30,84]
grade=[]
result=[]
for i in scores:
  if i>=90:
    grade.append('A')
  elif i>=80:
    grade.append('B')
  elif i>=70:
    grade.append('C')
  elif i>=50:
    grade.append('D')
  else:
    grade.append('F')
for i in scores:
  if i>=50:
    result.append('Pass')
  else:
    result.append('Fail')
df=pd.DataFrame({'Student':[f"Student {i}" for i in range(1,21)],'Scores':scc
print(df)
print("Overall performance:")
print("Average:",df['Scores'].mean())
print("Highest mark:",df['Scores'].max())
print("Lowest score:",df['Scores'].min())
```

```
       Student  Scores Grade Result
0    Student 1      90     A   Pass
1    Student 2      80     B   Pass
2    Student 3      54     D   Pass
3    Student 4      77     C   Pass
4    Student 5      40     F   Fail
5    Student 6      49     F   Fail
6    Student 7      60     D   Pass
7    Student 8      99     A   Pass
8    Student 9      44     F   Fail
9   Student 10      77     C   Pass
10  Student 11      67     D   Pass
11  Student 12      87     B   Pass
12  Student 13      48     F   Fail
13  Student 14      93     A   Pass
14  Student 15     100     A   Pass
15  Student 16      59     D   Pass
16  Student 17      78     C   Pass
17  Student 18      98     A   Pass
18  Student 19      30     F   Fail
19  Student 20      84     B   Pass
Overall performance:
Average: 70.7
Highest mark: 100
Lowest score: 30
```

In [ ]:

```
#QUESTION-1B

import numpy as np
A=np.array([[1,2],[3,4]])
b=np.array([5,6])
x=np.linalg.solve(A,b)
print(x)
```

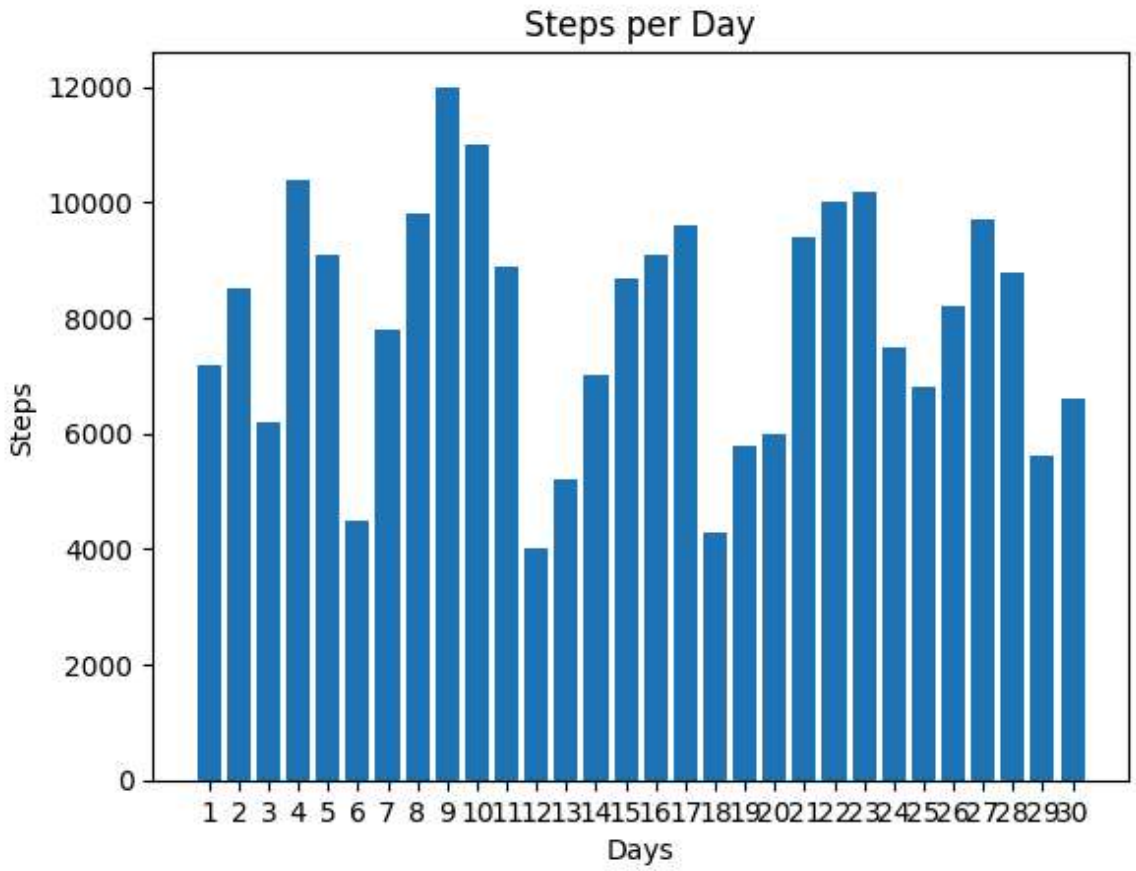
```
[-4.   4.5]
```

In [75]:

```
#QUESTION-2A

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
steps=[ 7200, 8500, 6200, 10400, 9100, 4500, 7800,9800, 12000, 11000, 8900, 4
days=[f"{i}"for i in range(1,31)]
df=pd.DataFrame({'Days':days,'Steps':steps})
print(df)
print("Average steps per day:",df['Steps'].mean())
print("Maximum steps:",df['Steps'].max())
print("Minimum steps:",df['Steps'].min())
HA=[]
for i in range(len(steps)):
  if steps[i]>9000:
    HA.append(i+1)
print("High Activity Days:",HA)
plt.bar(days,steps)
plt.xlabel('Days')
plt.ylabel('Steps')
plt.title('Steps per Day')
plt.show()
```

```
   Days  Steps
0     1   7200
1     2   8500
2     3   6200
3     4  10400
4     5   9100
5     6   4500
6     7   7800
7     8   9800
8     9  12000
9    10  11000
10   11   8900
11   12   4000
12   13   5200
13   14   7000
14   15   8700
15   16   9100
16   17   9600
17   18   4300
18   19   5800
19   20   6000
20   21   9400
21   22  10000
22   23  10200
23   24   7500
```

```
24   25   6800
25   26   8200
26   27   9700
27   28   8800
28   29   5600
29   30   6600
Average steps per day: 7930.0
Maximum steps: 12000
Minimum steps: 4000
High Activity Days: [4, 5, 8, 9, 10, 16, 17, 21, 22, 23, 27]
```



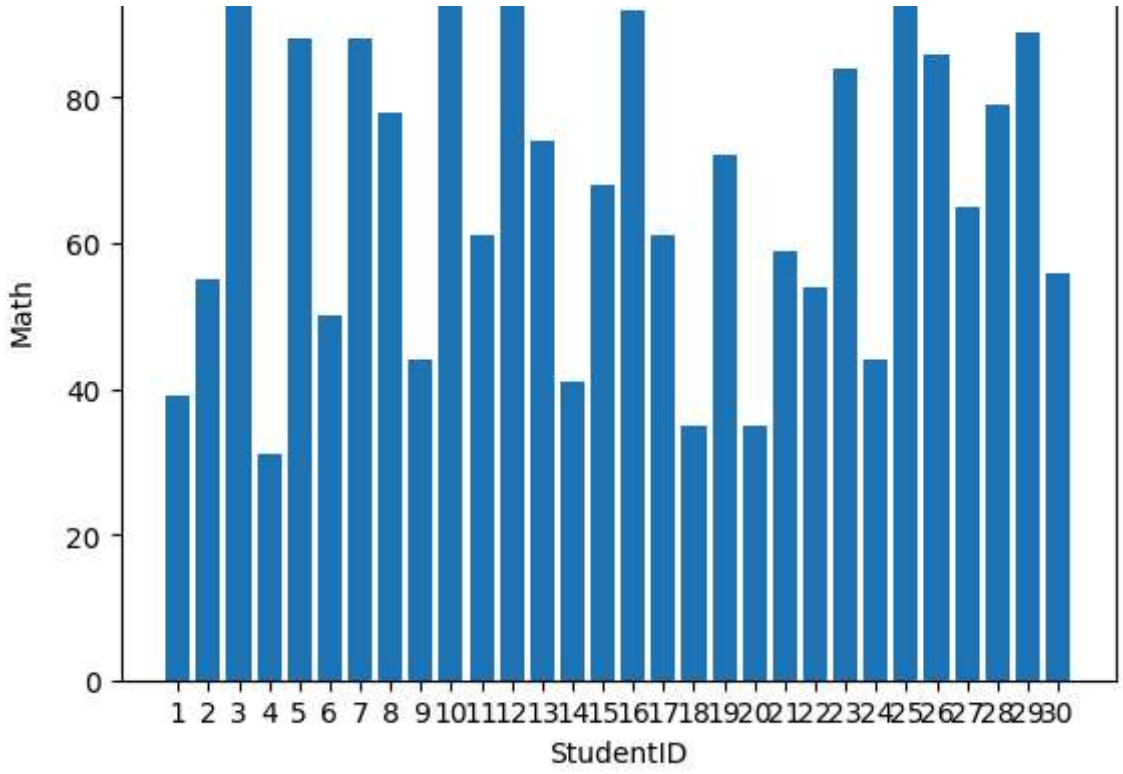
In [72]:

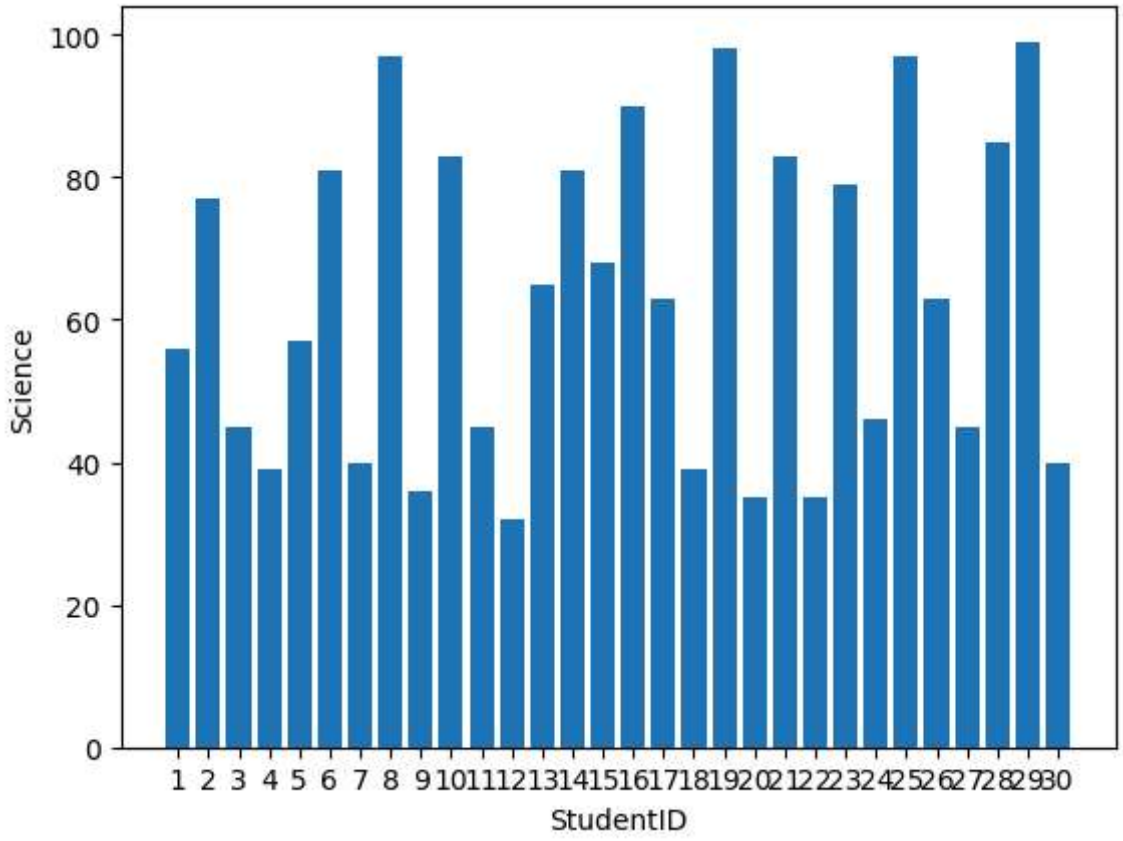
```
#QUESTION-2B

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
scores=np.random.randint(30,100,size=(30,3))
subjects=['Math','Science','English']
math=scores[:,0]
science=scores[:,1]
english=scores[:,2]
df=pd.DataFrame({'StudentID':[f"{i}" for i in range(1,31)],'Math':math,'Scier
total=[]
for i in range(len(math)):
  total.append(math[i]+science[i]+english[i])
df['Total']=total
print(df)
print("Overall performance:")
print("Average:",df['Total'].mean())
print("Highest mark:",df['Total'].max())
print("Lowest score:",df['Total'].min())
```

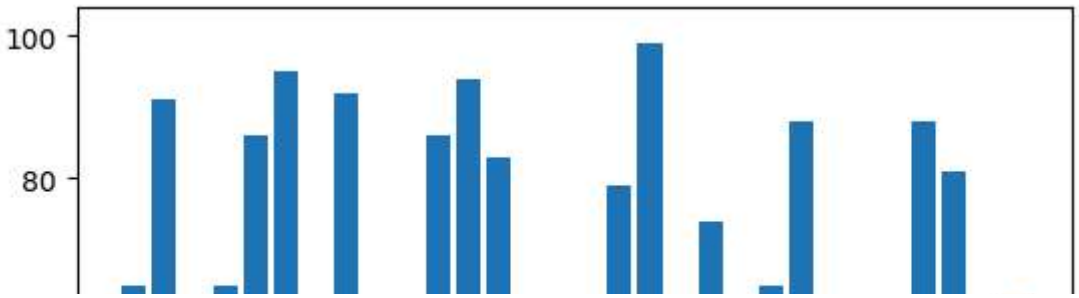
```
max=0
TS=0
for i in range(len(total)):
  if total[i]>max:
    max=total[i]
    TS=i
print(f"Top performing student:Student {TS}")
plt.bar(df['StudentID'],df['Math'])
plt.xlabel('StudentID')
plt.ylabel('Math')
plt.show()
plt.bar(df['StudentID'],df['Science'])
plt.xlabel('StudentID')
plt.ylabel('Science')
plt.show()
plt.bar(df['StudentID'],df['English'])
plt.xlabel('StudentID')
plt.ylabel('English')
plt.show()
```

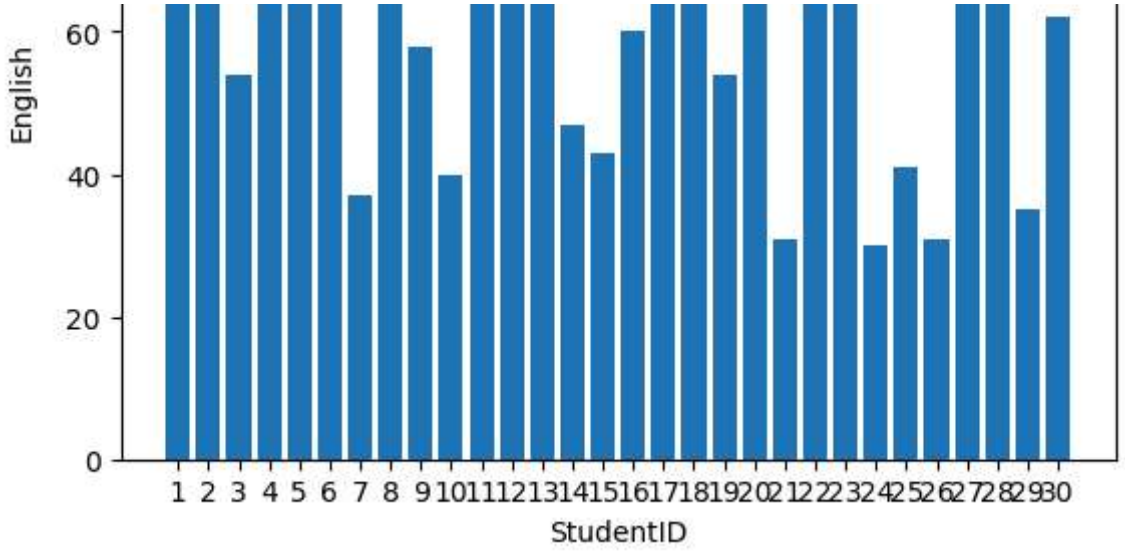
	StudentID	Math	Science	English	Total
0	1	39	56	65	160
1	2	55	77	91	223
2	3	95	45	54	194
3	4	31	39	65	135
4	5	88	57	86	231
5	6	50	81	95	226
6	7	88	40	37	165
7	8	78	97	92	267
8	9	44	36	58	138
9	10	97	83	40	220
10	11	61	45	86	192
11	12	94	32	94	220
12	13	74	65	83	222
13	14	41	81	47	169
14	15	68	68	43	179
15	16	92	90	60	242
16	17	61	63	79	203
17	18	35	39	99	173
18	19	72	98	54	224
19	20	35	35	74	144
20	21	59	83	31	173
21	22	54	35	65	154
22	23	84	79	88	251
23	24	44	46	30	120
24	25	95	97	41	233
25	26	86	63	31	180
26	27	65	45	88	198
27	28	79	85	81	245
28	29	89	99	35	223
29	30	56	40	62	158

```
Overall performance:
Average: 195.4
Highest mark: 267
Lowest score: 120
Top performing student:Student 7
```


Math
0
20
40
60
80
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
StudentID


Science
0
20
40
60
80
100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
StudentID


80
100



In [ ]:
```
#QUESTION-3A

import cmath
import pandas as pd
def imp(c,v,f):
  w=2*cmath.pi*f
  if c=='R':
    return complex(v,0)
  elif c=='L':
    return complex(0,w*v)
  elif c=='C':
    return complex(0,-1/(w*v))
  else:
    return 'Invalid component'
def tot(df,config):
  if config=='Series':
    return df['Impedance'].sum()
  else:
    return 1 / sum(1 / z for z in df['Impedance'])
f=float(input("Enter Frequency:"))
config=input("Enter the configuration(Series/Parallel):")
n=int(input('No of components:'))
d=[]
for i in range(n):
  c=input("Enter the component(R/L/C):").upper()
  v=float(input("Enter the value:"))
  z=imp(c,v,f)
  d.append({'Component':f'{c}{i+1}','Type':c,'Value': v, 'Impedance': z})
df=pd.DataFrame(d)
pd.set_option("display.precision",2)
print(df)
t=tot(df,config)
print(f'Total Impedance:{t:3.2f}')
```

```
Enter Frequency:50
Enter the configuration(Series/Parallel):Series
No of components:2
Enter the component(R/L/C):R
Enter the value:3
Enter the component(R/L/C):L
```

```
Enter the value:3
  Component Type  Value     Impedance
0        R1    R    3.0  3.00+  0.00j
1        L2    L    3.0  0.00+942.48j
Total Impedance:3.00+942.48j
```

In [ ]:

```
#QUESTION-3B

import numpy as np
stock_a = np.array([
    [100, 105, 95],
    [102, 108, 98],
    [101, 106, 97],
    [103, 107, 99],
    [104, 109, 100]
])
stock_b = np.array([
    [98, 104, 94],
    [99, 103, 95],
    [100, 102, 96],
    [101, 105, 97],
    [102, 106, 98]
])
sum=stock_a + stock_b
diff=stock_a - stock_b
print("Stock A:\n",stock_a)
print("\nStock B (5x3):\n",stock_b)
print("\nElement-wise Sum:\n",sum)
print("\nElement-wise Difference\n:",diff)
```

```
Stock A:
 [[100 105  95]
 [102 108  98]
 [101 106  97]
 [103 107  99]
 [104 109 100]]

Stock B (5x3):
 [[ 98 104  94]
 [ 99 103  95]
 [100 102  96]
 [101 105  97]
 [102 106  98]]

Element-wise Sum:
 [[198 209 189]
 [201 211 193]
 [201 208 193]
 [204 212 196]
 [206 215 198]]

Element-wise Difference
: [[2 1 1]
 [3 5 3]
 [1 4 1]
 [2 2 2]
 [2 3 2]]
```

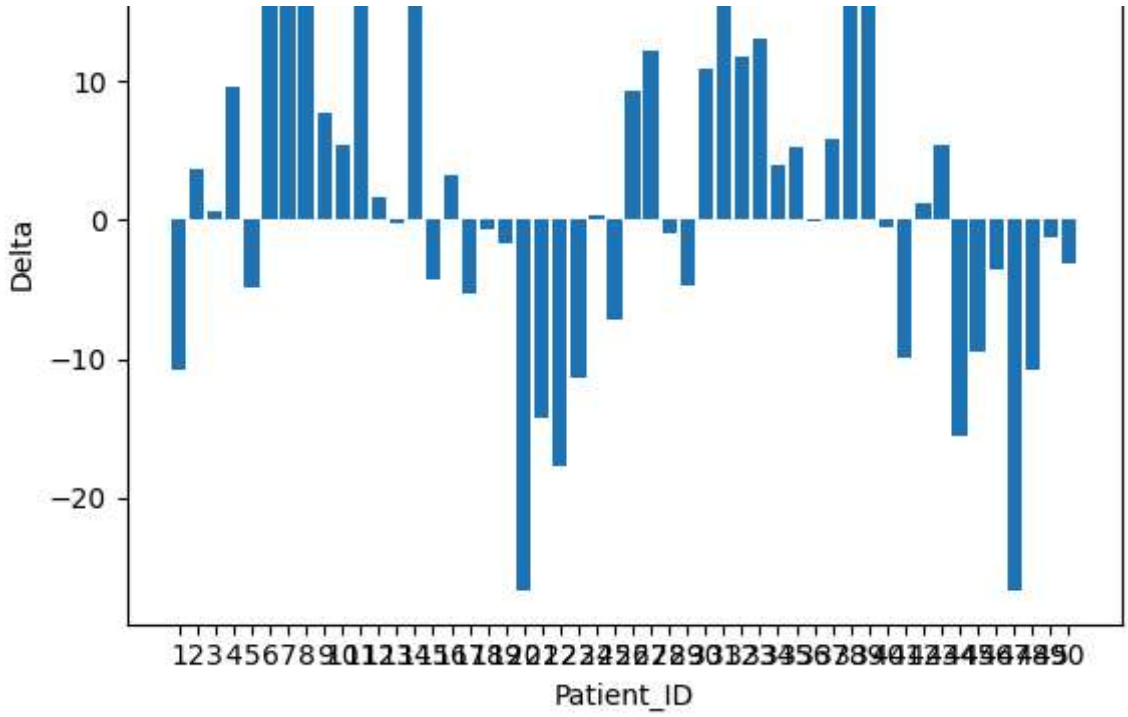
In [74]:

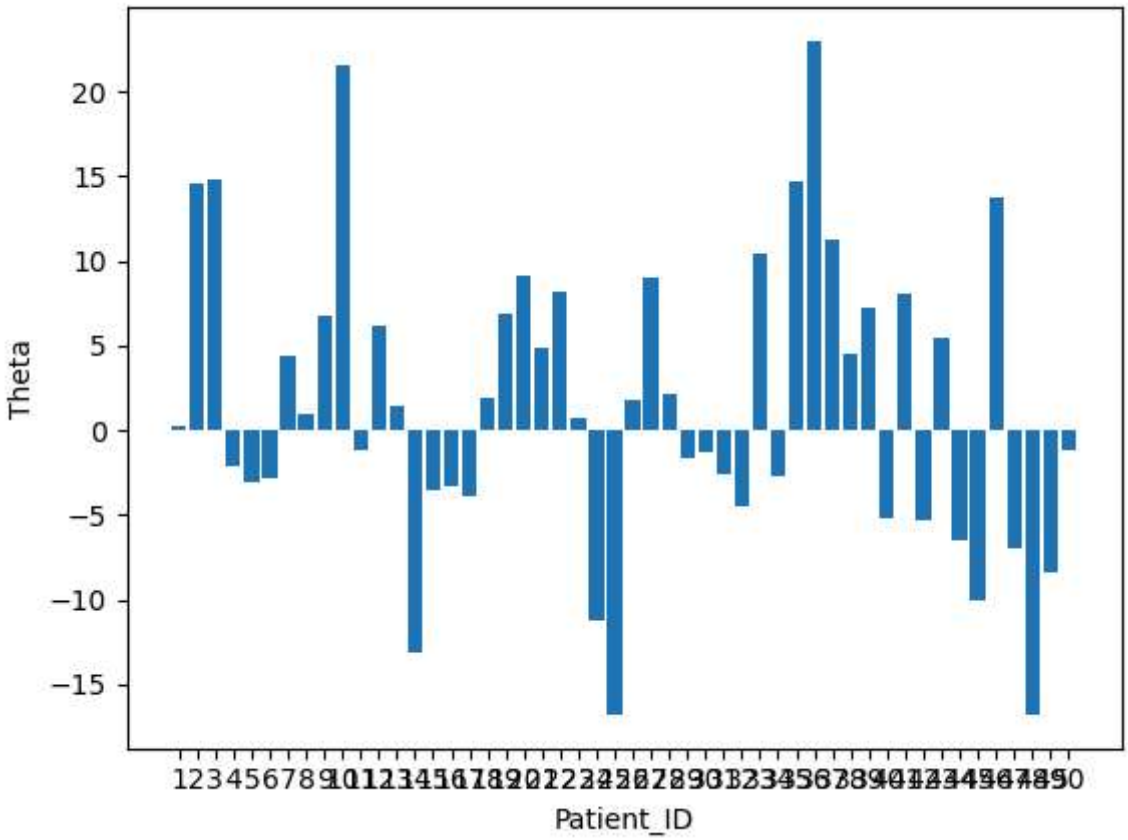
```
#QUESTION-4A
```

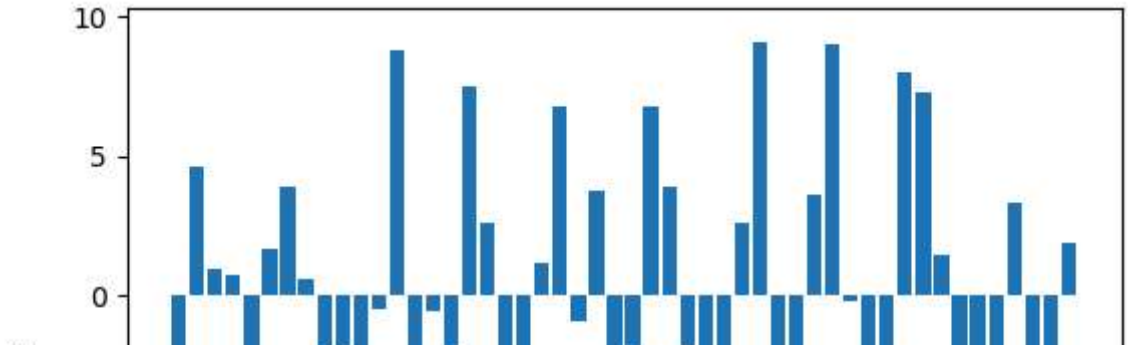
```
#QUESTION 4A

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
sampling_rate = 50  # Hz
duration = 1  # in seconds
t = np.linspace(0, duration, int(sampling_rate * duration), endpoint=False)
def generate_wave(freq, amp, noise_level):
    signal = amp * np.sin(2 * np.pi * freq * t)
    noise = np.random.normal(0, noise_level, t.shape)
    return signal + noise
data={
    'Patient_ID':[f"{i}" for i in range(1,51)],
    'Time(s)':t,
    'Delta': generate_wave(2, 10, 10),
    'Theta': generate_wave(6, 6, 8),
    'Alpha': generate_wave(10, 4, 5),
    'Beta':  generate_wave(20, 5, 3)
}
df = pd.DataFrame(data)
df.to_csv(r"C:\Users\DELL\Downloads\EEG-Sheet1.csv",index=False)
print(df)
print("\nStatistics (Mean, Variance, Max):")
print('Mean:\n',df[['Delta', 'Theta', 'Alpha', 'Beta']].mean())
print('Variance:\n',df[['Delta', 'Theta', 'Alpha', 'Beta']].var())
print('Max:\n',df[['Delta', 'Theta', 'Alpha', 'Beta']].max())
plt.bar(df['Patient_ID'],df['Delta'])
plt.xlabel('Patient_ID')
plt.ylabel('Delta')
plt.show()
plt.bar(df['Patient_ID'],df['Theta'])
plt.xlabel('Patient_ID')
plt.ylabel('Theta')
plt.show()
plt.bar(df['Patient_ID'],df['Alpha'])
plt.xlabel('Patient_ID')
plt.ylabel('Alpha')
plt.show()
plt.bar(df['Patient_ID'],df['Beta'])
plt.xlabel('Patient_ID')
plt.ylabel('Beta')
plt.show()
```

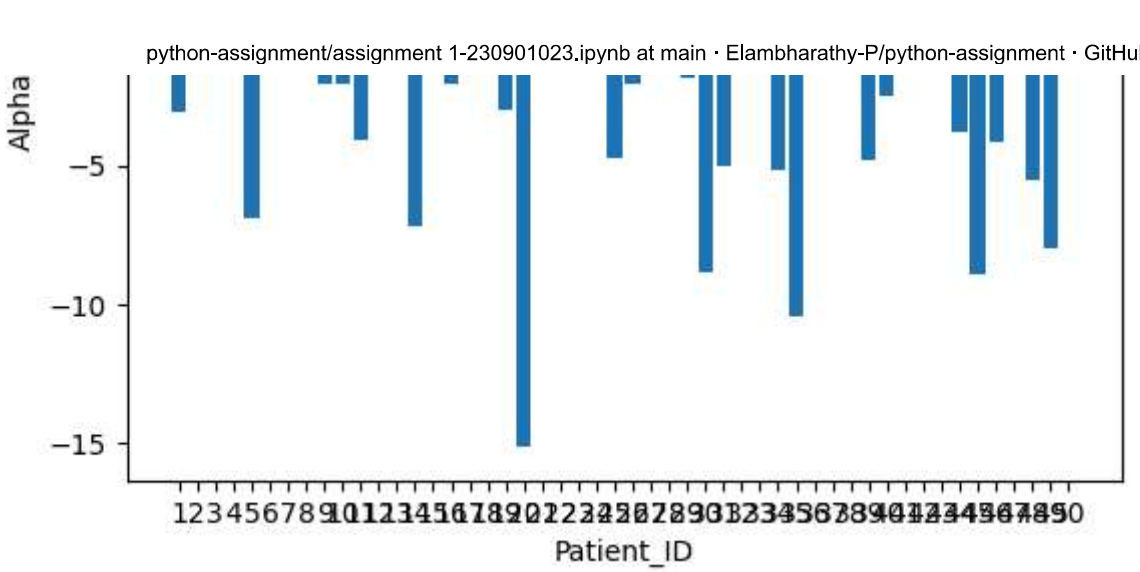
```
   Patient_ID  Time(s)  Delta  Theta  Alpha   Beta
0           1     0.00 -10.83   0.31  -3.06   0.49
1           2     0.02   3.65  14.55   4.63   3.65
2           3     0.04   0.65  14.76   0.97  -2.73
3           4     0.06   9.68  -2.07   0.74   4.88
4           5     0.08  -4.80  -3.06  -6.90  -0.90
5           6     0.10  20.84  -2.79   1.66   1.41
6           7     0.12  15.61   4.45   3.91   3.06
7           8     0.14  18.36   0.99   0.61  -6.18
8           9     0.16   7.69   6.72  -2.09   0.70
9          10     0.18   5.46  21.53  -2.07  -2.26
10         11     0.20  21.96  -1.20  -4.09  -1.84
11         12     0.22   1.59   6.13  -0.50   4.30
12         13     0.24  -0.22   1.47   8.75   1.02
13         14     0.26  15.77 -13.12  -7.23   1.20
```

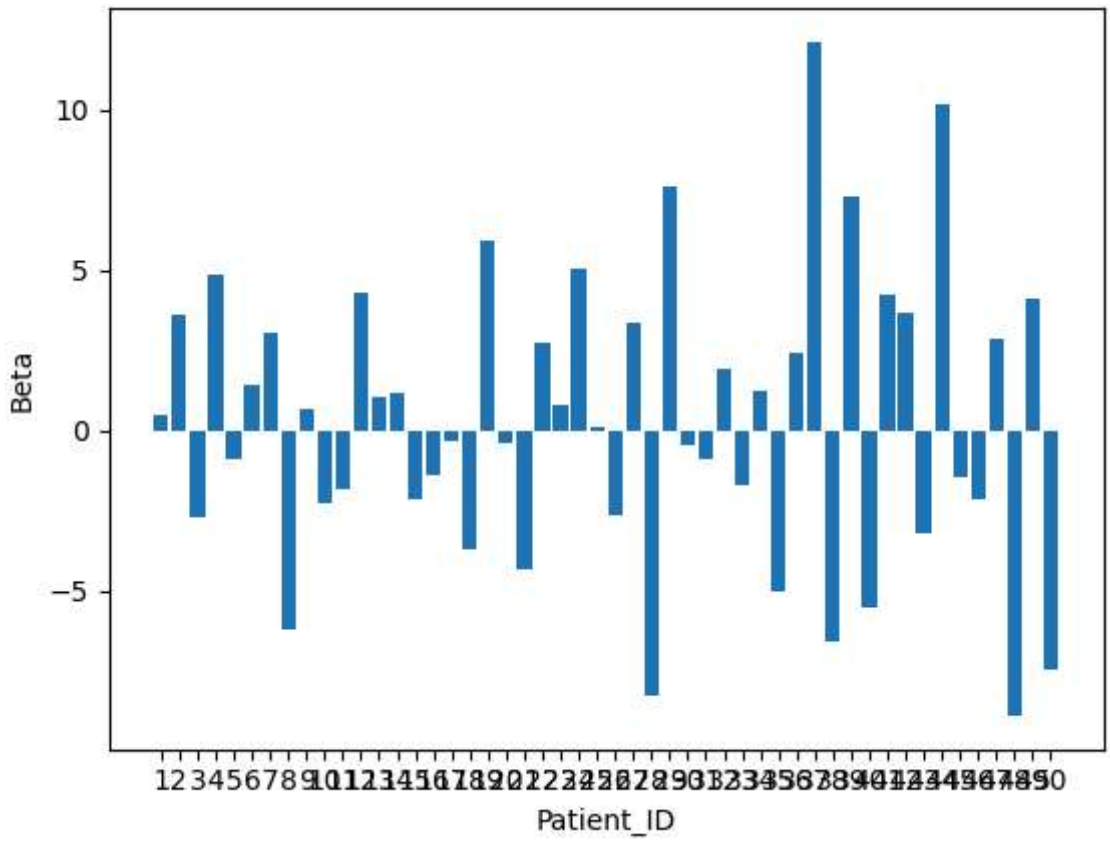
```
14          15     0.28  -4.21  -3.49  -0.56  -2.11
15          16     0.30   3.22  -3.30  -2.10  -1.38
16          17     0.32  -5.35  -3.93   7.47  -0.35
17          18     0.34  -0.72   1.97   2.57  -3.73
18          19     0.36  -1.68   6.92  -3.05   5.93
19          20     0.38 -26.67   9.15 -15.10  -0.39
20          21     0.40 -14.29   4.86   1.16  -4.31
21          22     0.42 -17.74   8.23   6.79   2.72
22          23     0.44 -11.32   0.72  -0.91   0.81
23          24     0.46   0.32 -11.26   3.75   5.06
24          25     0.48  -7.14 -16.75  -4.76   0.13
25          26     0.50   9.35   1.82  -2.07  -2.65
26          27     0.52  12.20   9.01   6.74   3.39
27          28     0.54  -0.94   2.10   3.90  -8.24
28          29     0.56  -4.70  -1.60  -1.89   7.62
29          30     0.58  10.95  -1.27  -8.83  -0.47
30          31     0.60  19.88  -2.59  -5.05  -0.86
31          32     0.62  11.75  -4.48   2.62   1.92
32          33     0.64  13.16  10.49   9.09  -1.70
33          34     0.66   3.91  -2.72  -5.17   1.22
34          35     0.68   5.32  14.70 -10.45  -5.03
35          36     0.70  -0.11  23.00   3.59   2.42
36          37     0.72   5.91  11.27   8.99  12.13
37          38     0.74  21.00   4.54  -0.19  -6.56
38          39     0.76  15.94   7.22  -4.80   7.32
39          40     0.78  -0.48  -5.12  -2.53  -5.51
40          41     0.80  -9.93   8.08   8.01   4.26
41          42     0.82   1.29  -5.33   7.27   3.66
42          43     0.84   5.45   5.42   1.41  -3.18
43          44     0.86 -15.55  -6.48  -3.79  10.18
44          45     0.88  -9.53 -10.05  -8.93  -1.47
45          46     0.90  -3.49  13.77  -4.16  -2.14
46          47     0.92 -26.69  -6.95   3.31   2.86
47          48     0.94 -10.73 -16.73  -5.51  -8.88
48          49     0.96  -1.20  -8.34  -7.96   4.11
49          50     0.98  -3.17  -1.12   1.86  -7.43

Statistics (Mean, Variance, Max):
Mean:
 Delta    1.39
Theta    1.61
Alpha   -0.48
Beta     0.32
dtype: float64
Variance:
 Delta    135.34
Theta     77.57
Alpha     30.41
Beta      20.36
dtype: float64
Max:
 Delta    21.96
Theta    23.00
Alpha     9.09
Beta     12.13
dtype: float64
```


Delta
10
0
−10
−20
Patient_ID


Theta
20
15
10
5
0
−5
−10
−15
Patient_ID


10
5
0



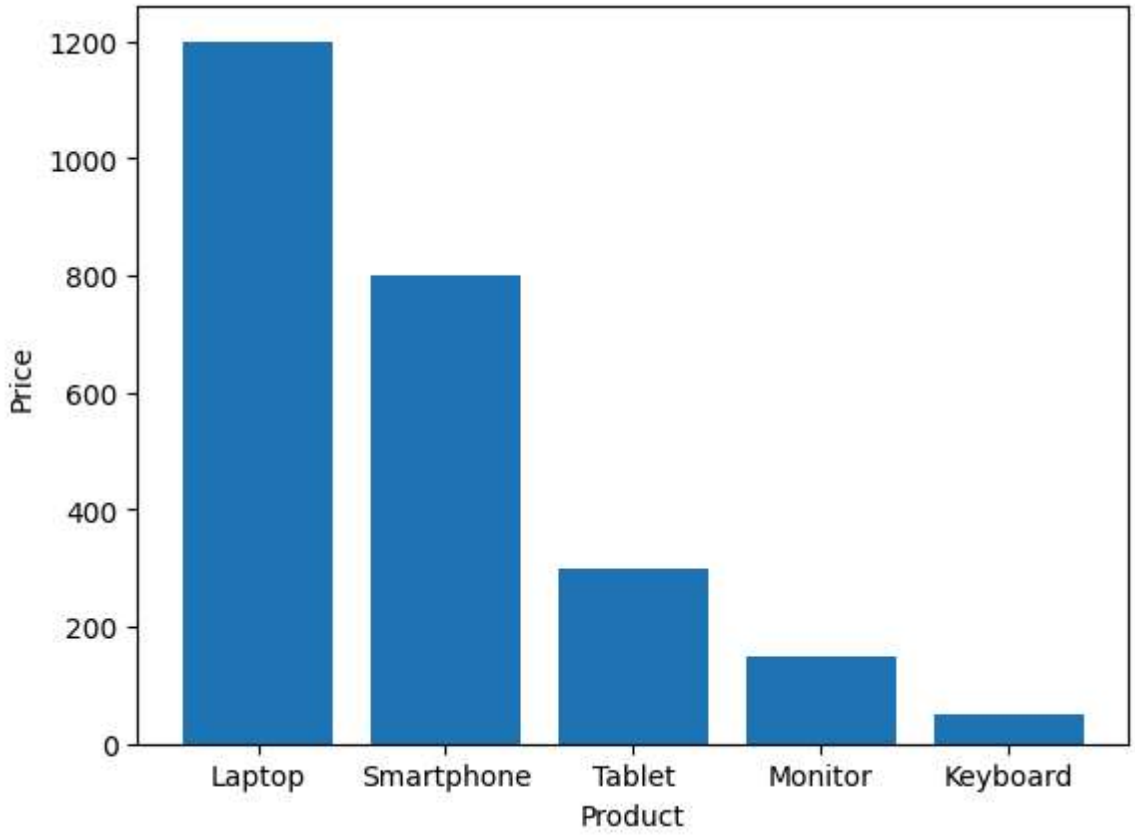


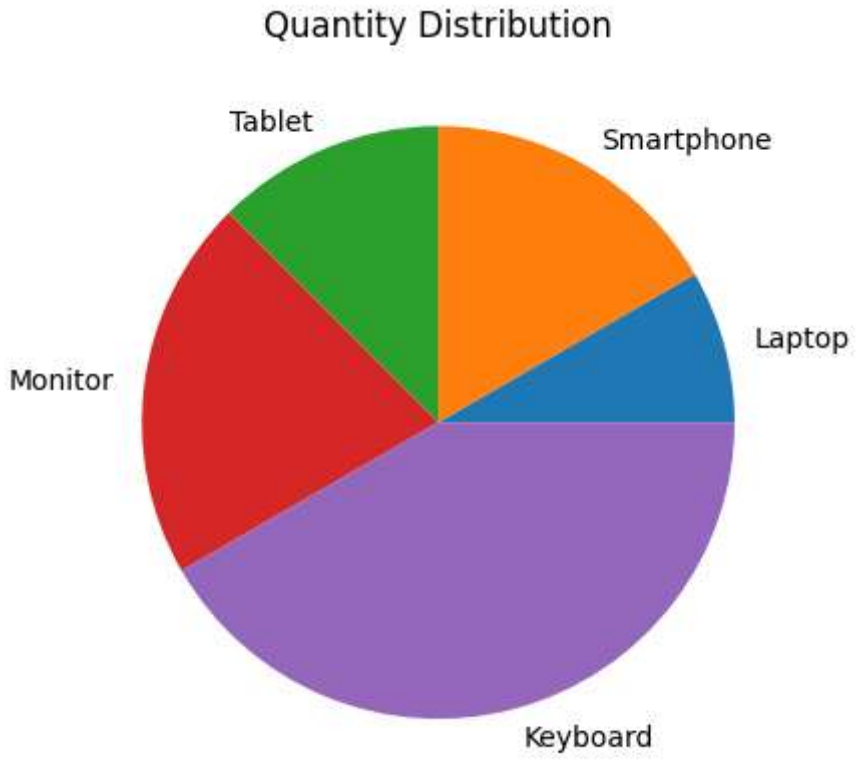
In [40]:

```
#QUESTION-4B

import matplotlib.pyplot as plt
d={'Product':['Laptop','Smartphone','Tablet','Monitor','Keyboard'],'Price':[1
df=pd.DataFrame(d)
df['Total']=df['Price']*df['Quantity']
print("Average:",df['Total'].mean())
print("Highest Price:",df['Price'].max())
print("Lowest Price:",df['Price'].min())
plt.bar(df['Product'],df['Price'])
plt.xlabel('Product')
plt.ylabel('Price')
plt.show()
plt.pie(df['Quantity'],labels=df['Product'])
plt.title('Quantity Distribution')
```

```
plt.show()
```

```
Average: 7750.0
Highest Price: 1200
Lowest Price: 50
```

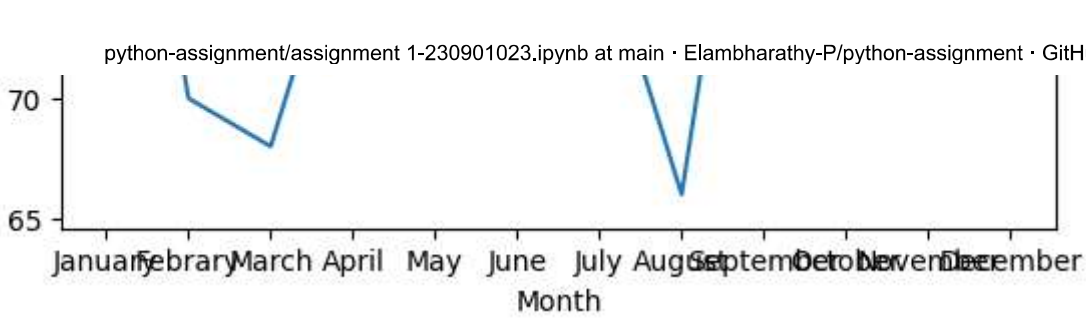




In [62]:
```
#QUESTION-5A
```

```
import pandas as pd
sheet_id = '1jVHGkKb1P_iwZAbbQ98aUUWVIJAfPUeksIwifQeY1ao'
url = f"https://docs.google.com/spreadsheets/d/{sheet_id}/export?format=csv"
df = pd.read_csv(url)
print(df)
k=df.ffill()
print(k)
x=k['Month']
y=k['Sales(%)']
plt.xlabel('Month')
plt.ylabel('Sales(%)')
plt.plot(x,y)
plt.show()
```

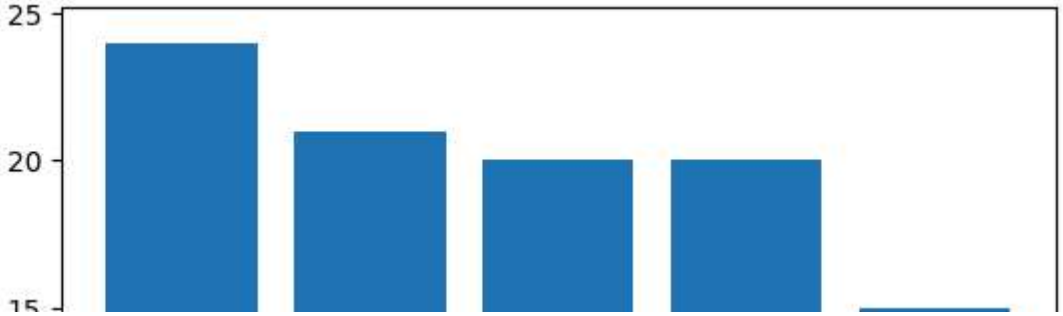
```
        Month  Sales(%)
0     January      90.0
1     Febrary      70.0
2       March      68.0
3       April      79.0
4         May       NaN
5        June      88.0
6        July      77.0
7      August      66.0
8   September      88.0
9     October      76.0
10   November      77.0
11   December      94.0
        Month  Sales(%)
0     January      90.0
1     Febrary      70.0
2       March      68.0
3       April      79.0
4         May      79.0
5        June      88.0
6        July      77.0
7      August      66.0
8   September      88.0
9     October      76.0
10   November      77.0
11   December      94.0
```

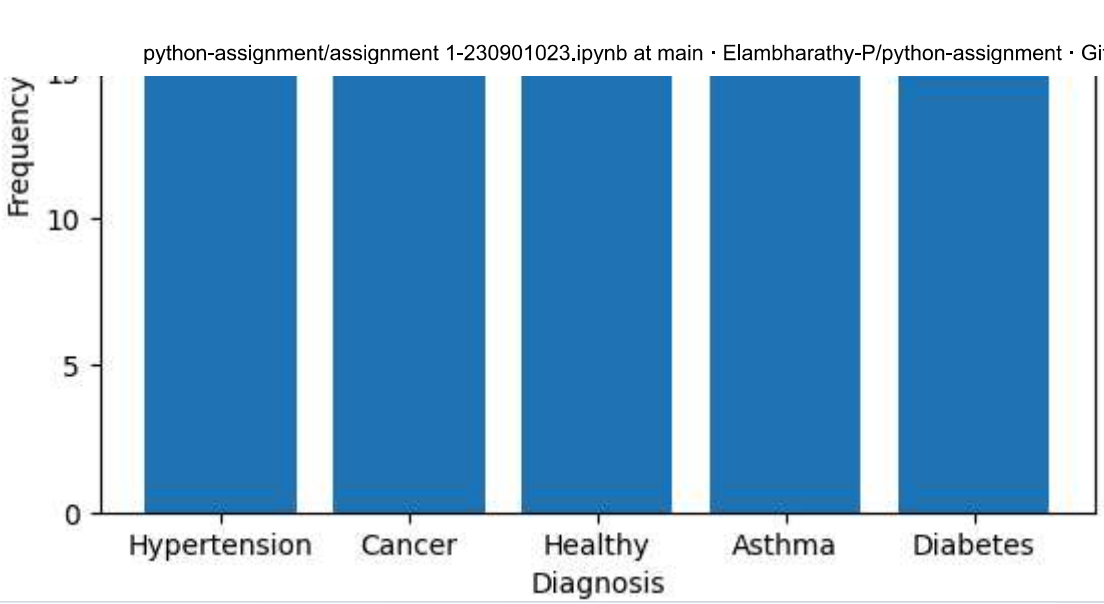


In [71]:

```
#QUESTION-5B

import pandas as pd
sheet_id ='1y1sLJE-CYg-MhhctnLj9rC8cKw7f3ZUh5-sVSoTebfY'
url=f"https://docs.google.com/spreadsheets/d/{sheet_id}/export?format=csv"
df=pd.read_csv(url)
print(df)
print("Column names and data types:")
print(df.dtypes)
avg=df['Age'].mean()
print("\nAverage Age of the Patients:",avg)
y=df['Diagnosis'].value_counts()
plt.bar(y.index,y.values)
plt.xlabel('Diagnosis')
plt.ylabel('Frequency')
plt.show()
```

```
    PatientID  Age  Gender     Diagnosis             Medical History
0           1   34  Female       Healthy                     Obesity
1           2   30  Female  Hypertension  Family history of diabetes
2           3   43    Male        Asthma                     Obesity
3           4   64  Female        Asthma                      Smoker
4           5   73    Male        Cancer                         NaN
..        ...  ...     ...           ...                         ...
95         96   55  Female      Diabetes                         NaN
96         97   39    Male  Hypertension                      Smoker
97         98   31  Female        Cancer                         NaN
98         99   50    Male        Asthma                         NaN
99        100   76  Female        Cancer               Chronic pain

[100 rows x 5 columns]
Column names and data types:
PatientID           int64
Age                 int64
Gender             object
Diagnosis          object
Medical History    object
dtype: object

Average Age of the Patients: 50.39
```




Frequency
10
5
0
Hypertension
Cancer
Healthy
Asthma
Diabetes
Diagnosis