ai-program-day-2

May 5, 2024

[3]:
```
#claculate the no. of upper case & lower case alphabets in given string.
#String:She sells seashells by the shore
def calc_string(x):
    lower=0
    upper=0
    for i in x:
        if i.isupper()==True:
            upper+=1
        else:
            lower+=1
    return lower, upper
print(calc_string("She sells Seashells by the Seashore"))
```

```
(32, 3)
```

[4]:
```
def add(x,y):
    return(x+y)
```

[5]:
```
add(1,8)
```

[5]:
```
9
```

[6]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[7]:
```
data={"Nmae":["Raji","Teju","Vijju","Anu"],
     "Age":[24,25,27,26],
     "Salary":[23000,24000,25000,26000]}
```

[8]:
```
df=pd.DataFrame(data)
```

[9]:
```
df["Location"]=["Hyderabad","Mumbai","Banglore","Pune"]
```

[10]:
```
df
```

[10]:
```
   Nmae  Age  Salary   Location
0  Raji   24   23000  Hyderabad
1  Teju   25   24000     Mumbai
```

```
      2  Vijju   27   25000   Banglore
      3    Anu   26   26000       Pune
```

```
[11]: #Filtering DataFrame
      df_fil=df[df["Age"]>22]
      df_fil
```

```
[11]:     Nmae  Age  Salary   Location
      0   Raji   24   23000  Hyderabad
      1   Teju   25   24000     Mumbai
      2  Vijju   27   25000   Banglore
      3    Anu   26   26000       Pune
```

```
[12]: df.tail(2)
```

```
[12]:     Nmae  Age  Salary  Location
      2  Vijju   27   25000  Banglore
      3    Anu   26   26000      Pune
```

```
[13]: df.replace(24,22, inplace=True)
      df
```

```
[13]:     Nmae  Age  Salary   Location
      0   Raji   22   23000  Hyderabad
      1   Teju   25   24000     Mumbai
      2  Vijju   27   25000   Banglore
      3    Anu   26   26000       Pune
```

```
[14]: df.replace(24,22)
```

```
[14]:     Nmae  Age  Salary   Location
      0   Raji   22   23000  Hyderabad
      1   Teju   25   24000     Mumbai
      2  Vijju   27   25000   Banglore
      3    Anu   26   26000       Pune
```

```
[15]: df["Salary"].mean()
```

```
[15]: 24500.0
```

```
[16]: df["Age"].max()
```

```
[16]: 27
```

```
[17]: df["Salary"].sum()
```

```
[17]: 98000
```

[18]:

```
df.isna()
```

[18]:

```
    Nmae    Age  Salary  Location
0  False  False   False     False
1  False  False   False     False
2  False  False   False     False
3  False  False   False     False
```

[19]:

```
df.isna().sum()
```

[19]:

```
Nmae        0
Age         0
Salary      0
Location    0
dtype: int64
```

[20]:

```
#axis=1 for columns
#axis=0 for rows
df.drop("Location",axis=1)
```

[20]:

```
    Nmae  Age  Salary
0   Raji   22   23000
1   Teju   25   24000
2  Vijju   27   25000
3    Anu   26   26000
```

[21]:

```
df.drop(index=1)
```

[21]:

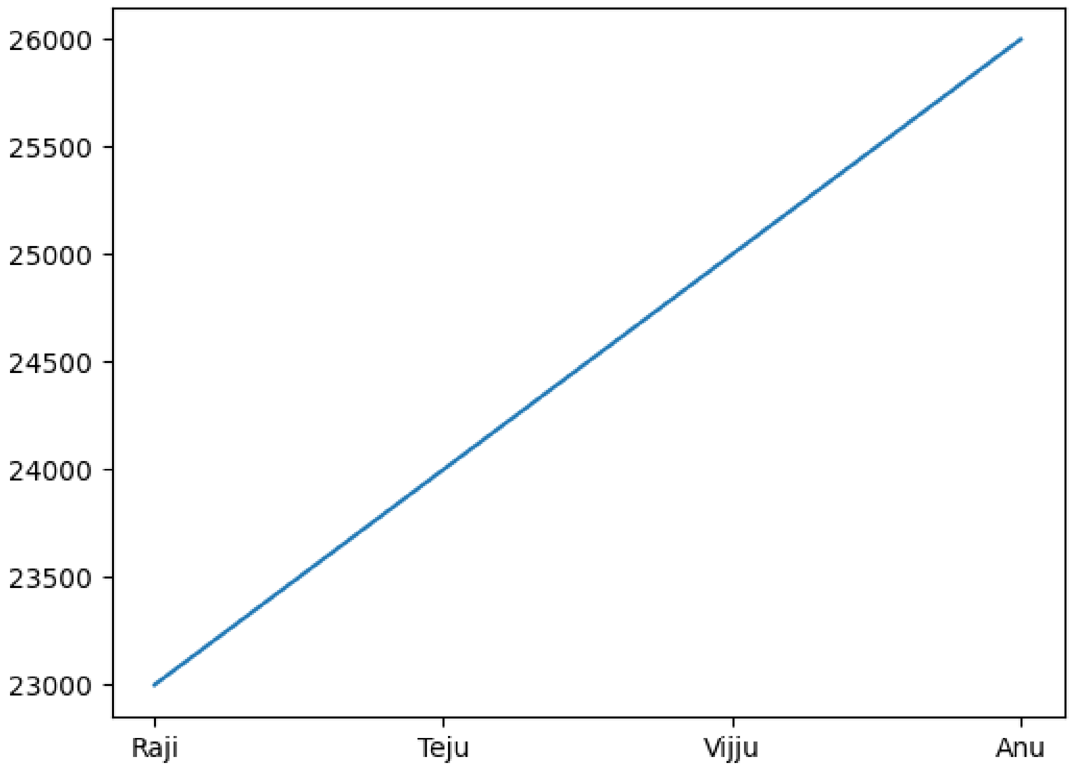
```
    Nmae  Age  Salary   Location
0   Raji   22   23000  Hyderabad
2  Vijju   27   25000   Banglore
3    Anu   26   26000       Pune
```

[22]:

```
import matplotlib.pyplot as plt
```

[23]:

```
plt.plot(df["Nmae"],df["Salary"])
plt.show()
```



[24]:
```
df.drop(index=1).reset_index()
```

[24]:
```
   index   Nmae  Age  Salary   Location
0      0   Raji   22   23000  Hyderabad
1      2  Vijju   27   25000   Banglore
2      3    Anu   26   26000       Pune
```

[25]:
```
df.drop(index=1).reset_index().drop("index",axis=1)
```

[25]:
```
    Nmae  Age  Salary   Location
0   Raji   22   23000  Hyderabad
1  Vijju   27   25000   Banglore
2    Anu   26   26000       Pune
```

[26]:
```
data1={"ID":[1,2,3,4],
      "Name":["Raji","Teju","Vijju","Anu"],
       "Age":[21,22,23,24]}
df1=pd.DataFrame(data1)
data2={"ID":[3,4,5,6],
      "Occupation":["Doc","Eng","Tech","Pol"],
      "City":[8,5,6,7]}
df2=pd.DataFrame(data2)
```

```
concat_df=pd.concat([df1,df2],ignore_index=True)
concat_df
```

[26]:

```
   ID   Name   Age Occupation  City
0   1   Raji  21.0        NaN   NaN
1   2   Teju  22.0        NaN   NaN
2   3  Vijju  23.0        NaN   NaN
3   4    Anu  24.0        NaN   NaN
4   3    NaN   NaN        Doc   8.0
5   4    NaN   NaN        Eng   5.0
6   5    NaN   NaN       Tech   6.0
7   6    NaN   NaN        Pol   7.0
```

[27]:

```
merge_data=pd.merge(df1,df2,on="ID")
merge_data
```

[27]:

```
   ID   Name  Age Occupation  City
0   3  Vijju   23        Doc     8
1   4    Anu   24        Eng     5
```

[28]:

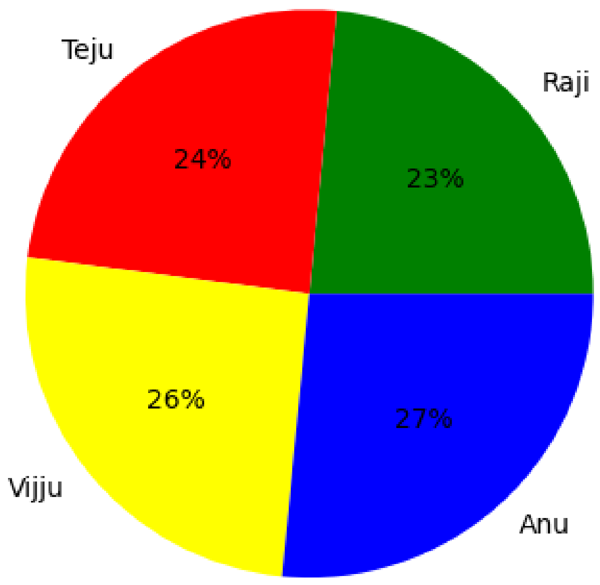
```
pivot_df=df.pivot(index="Nmae",columns="Location",values="Age")
pivot_df
```

[28]:

```
Location  Banglore  Hyderabad  Mumbai  Pune
Nmae
Anu            NaN        NaN     NaN  26.0
Raji           NaN       22.0     NaN   NaN
Teju           NaN        NaN    25.0   NaN
Vijju         27.0        NaN     NaN   NaN
```

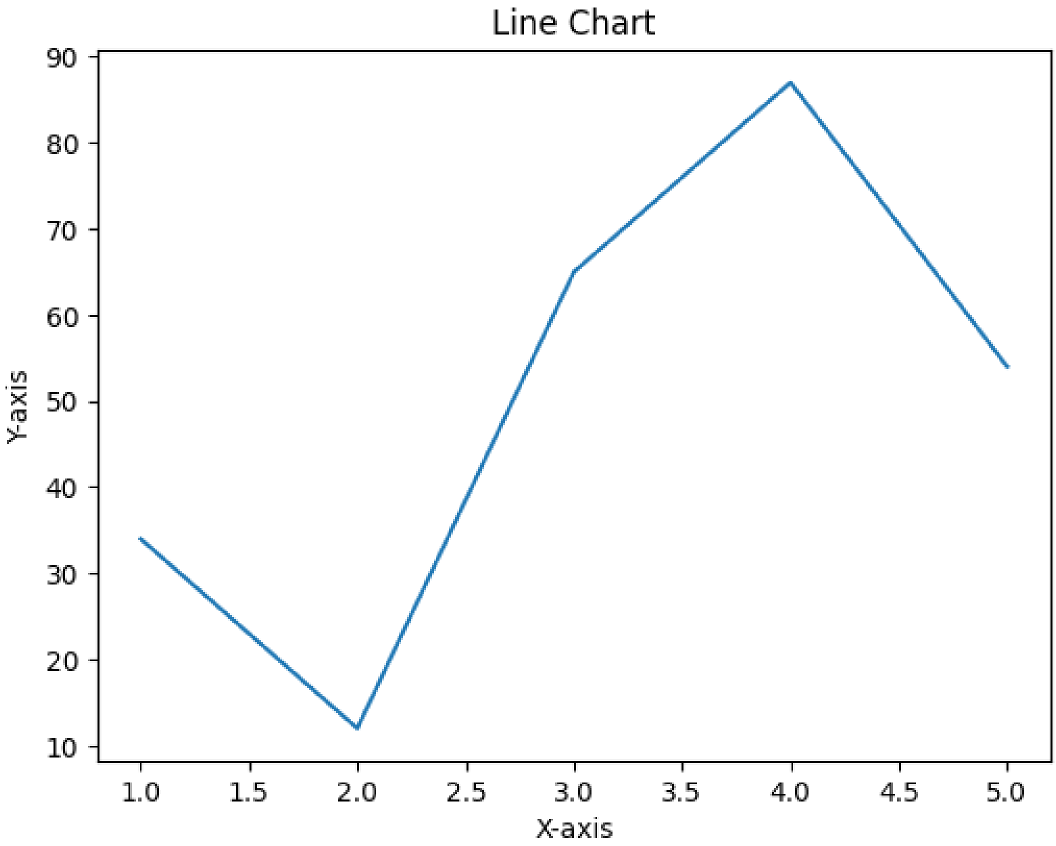
[29]:

```
colour=["green","red","yellow","blue"]
plt.pie(df["Salary"],labels=df["Nmae"],colors=colour,autopct="%1.0f%%")
plt.show()
```



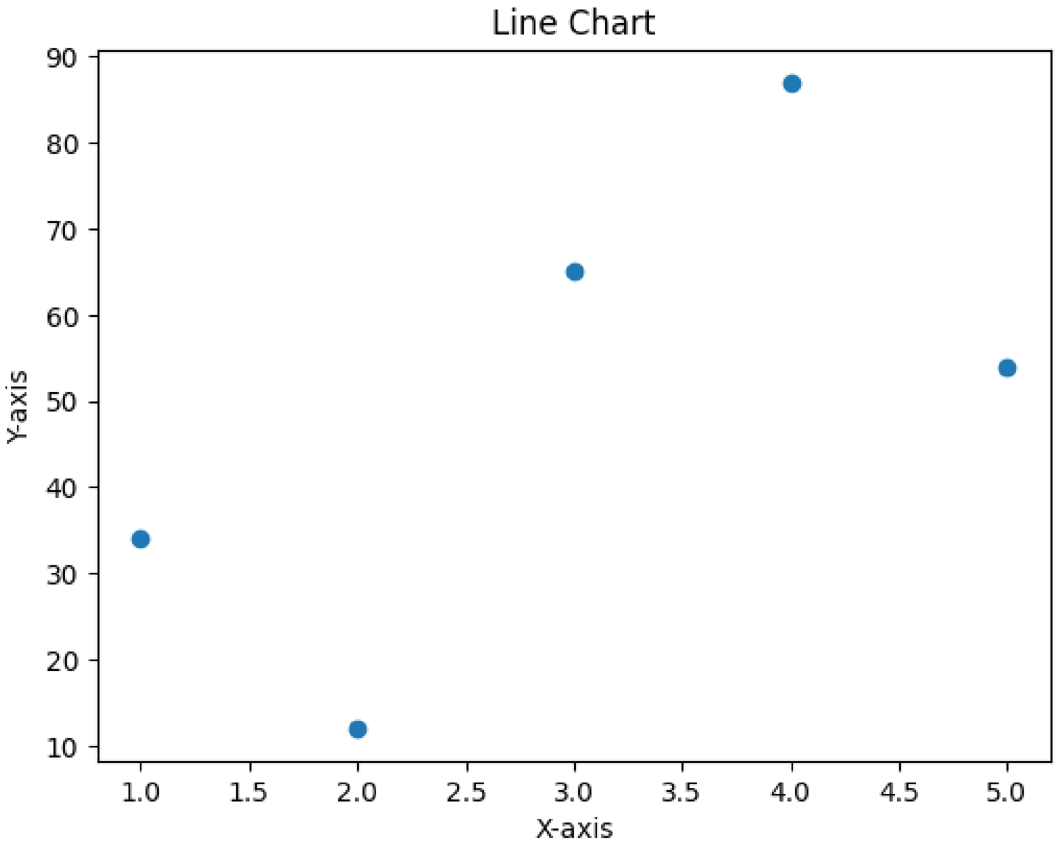
```
[30]: x=[1,2,3,4,5]
      y=[34,12,65,87,54]
      plt.plot(x,y)
      plt.title("Line Chart")
      plt.xlabel("X-axis")
      plt.ylabel("Y-axis")
      plt.show
```

```
[30]: <function matplotlib.pyplot.show(close=None, block=None)>
```



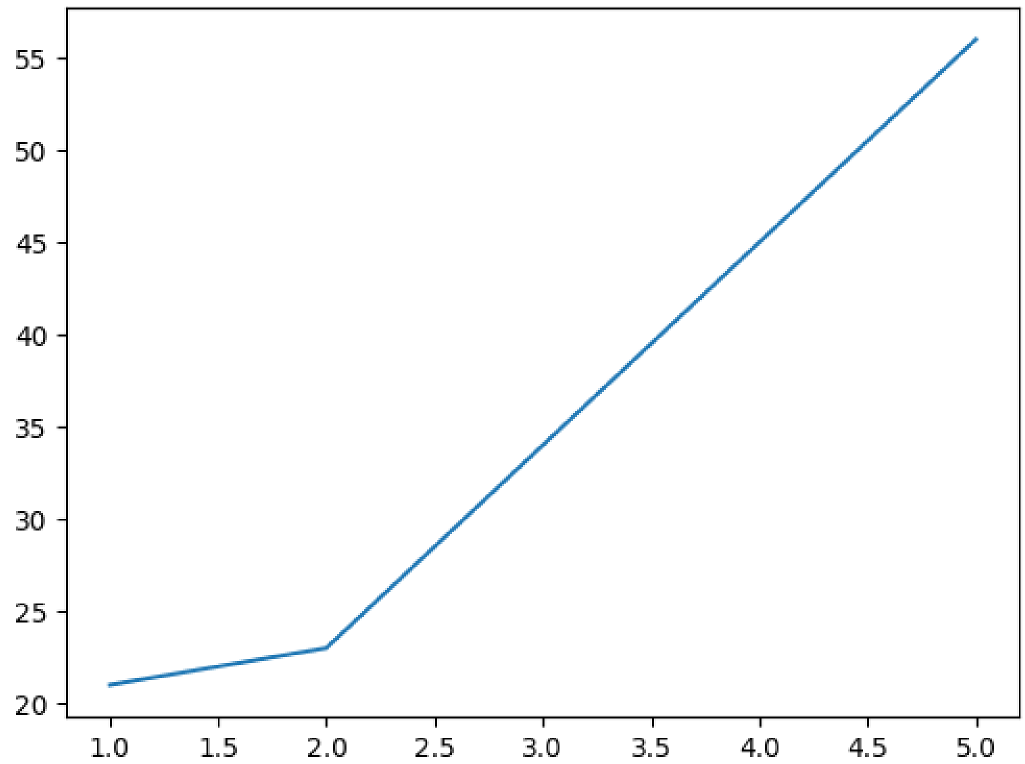
```
[31]: x=[1,2,3,4,5]
      y=[34,12,65,87,54]
      plt.scatter(x,y)
      plt.title("Line Chart")
      plt.xlabel("X-axis")
      plt.ylabel("Y-axis")
      plt.show
```

```
[31]: <function matplotlib.pyplot.show(close=None, block=None)>
```

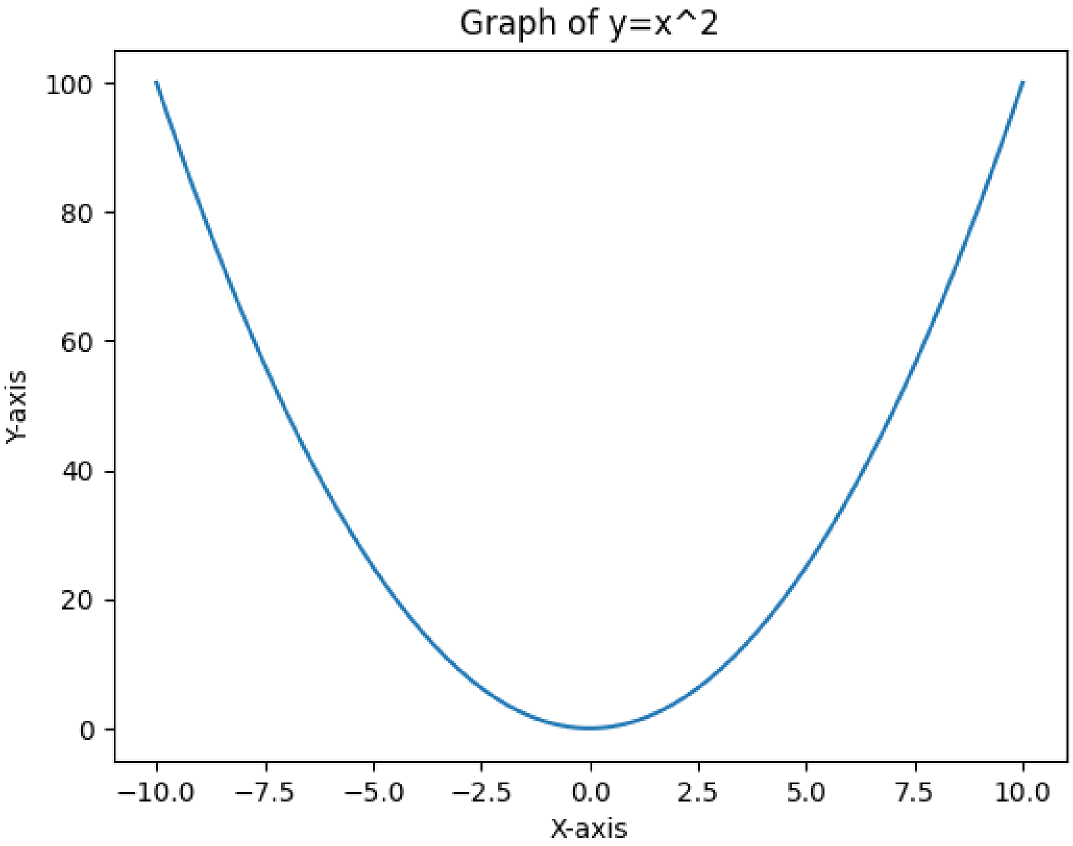


[32]:

```
#plot graph for equation y=x^2
import matplotlib.pyplot as plt
x=[1,2,3,4,5]
y=[21,23,34,45,56]
plt.plot(x,y)
plt.show()
```

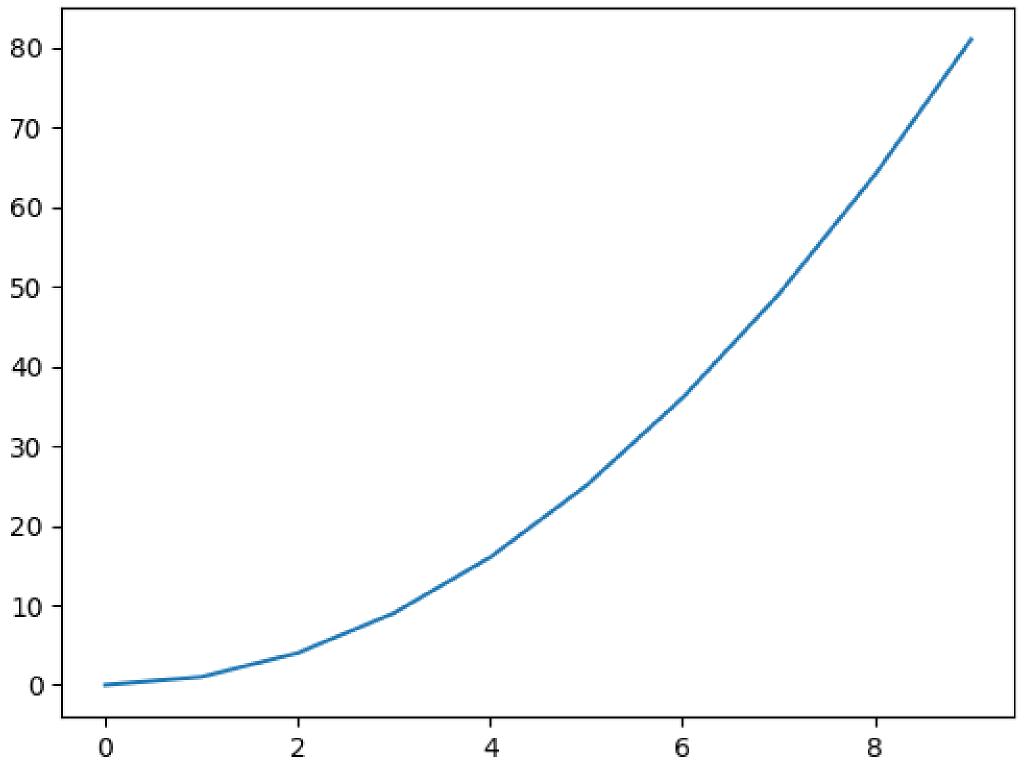


[33]:
```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(-10,10,400)
y=x**2
plt.plot(x,y)
plt.title("Graph of y=x^2")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



[34]:
```
#plot graph for equation y=x^2
import matplotlib.pyplot as plt
x=[x for x in range(10)]#This structure with output as list,known as list␣
  ↪comprehension
y=[i**2 for i in x]
print(x)
print(y)
plt.plot(x,y)
plt.show()
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

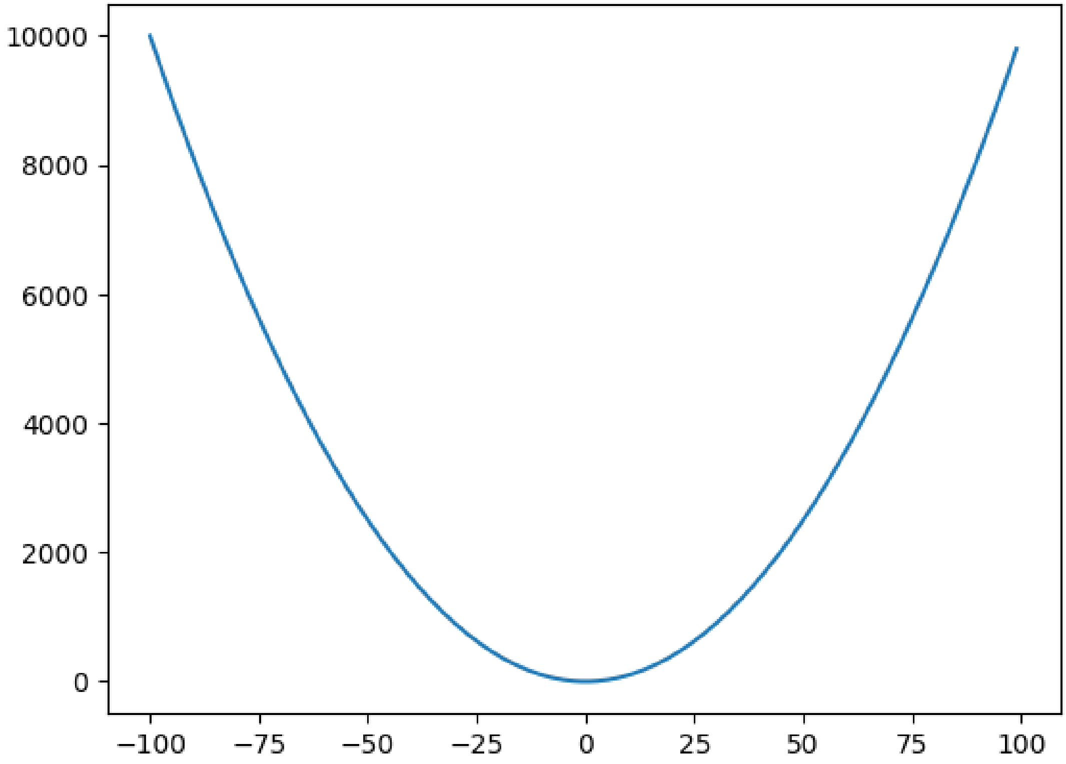


[35]:

```
#plot graph for equation y=x^2
import matplotlib.pyplot as plt
x=[x for x in range(-100,100)]#This structure with output as list,known as list comprehension
y=[i**2 for i in x]
print(x)
print(y)
plt.plot(x,y)
plt.show()
```

```
[-100, -99, -98, -97, -96, -95, -94, -93, -92, -91, -90, -89, -88, -87, -86,
-85, -84, -83, -82, -81, -80, -79, -78, -77, -76, -75, -74, -73, -72, -71, -70,
-69, -68, -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54,
-53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38,
-37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25, -24, -23, -22,
-21, -20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5,
-4, -3, -2, -1, 0, 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, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99]
```

```
[10000, 9801, 9604, 9409, 9216, 9025, 8836, 8649, 8464, 8281, 8100, 7921, 7744,
7569, 7396, 7225, 7056, 6889, 6724, 6561, 6400, 6241, 6084, 5929, 5776, 5625,
5476, 5329, 5184, 5041, 4900, 4761, 4624, 4489, 4356, 4225, 4096, 3969, 3844,
3721, 3600, 3481, 3364, 3249, 3136, 3025, 2916, 2809, 2704, 2601, 2500, 2401,
2304, 2209, 2116, 2025, 1936, 1849, 1764, 1681, 1600, 1521, 1444, 1369, 1296,
1225, 1156, 1089, 1024, 961, 900, 841, 784, 729, 676, 625, 576, 529, 484, 441,
400, 361, 324, 289, 256, 225, 196, 169, 144, 121, 100, 81, 64, 49, 36, 25, 16,
9, 4, 1, 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256,
289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024,
1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025,
2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364,
3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041,
5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889, 7056,
7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409,
9604, 9801]
```

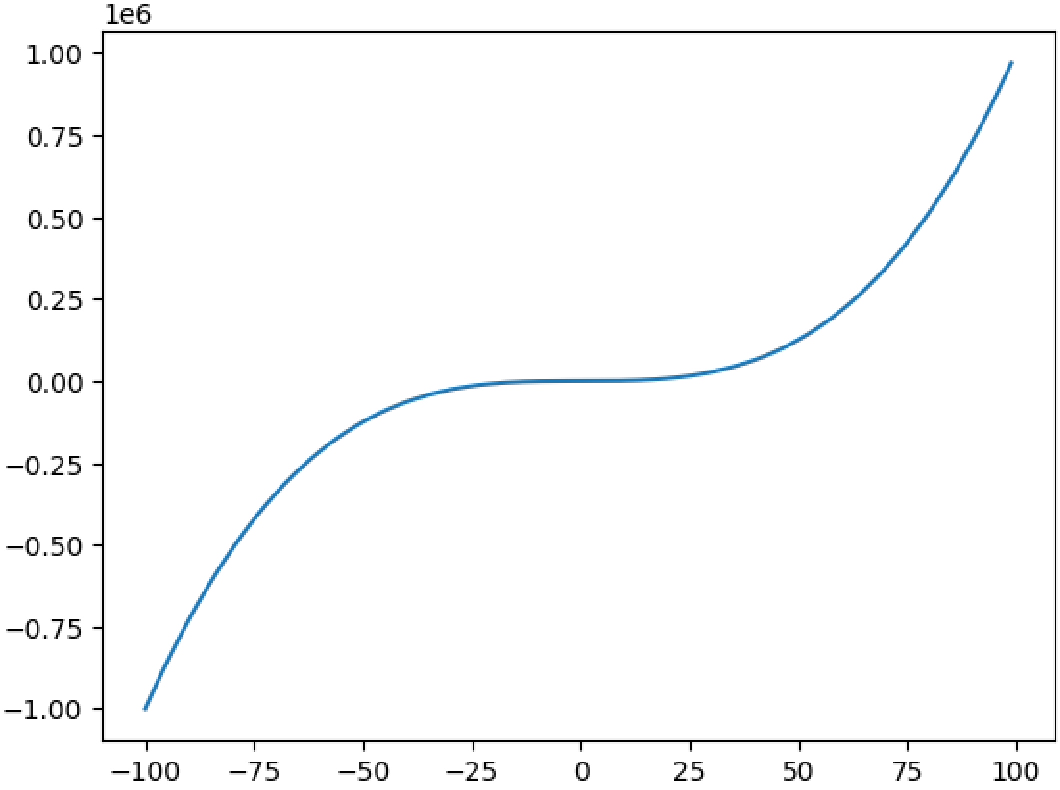


[36]:

```
#plot graph for equation y=x^2
import matplotlib.pyplot as plt
x=[x for x in range(-100,100)]#This structure with output as list,known as list comprehension
y=[i**3 for i in x]
print(x)
```

```python
print(y)
plt.plot(x,y)
plt.show()
```

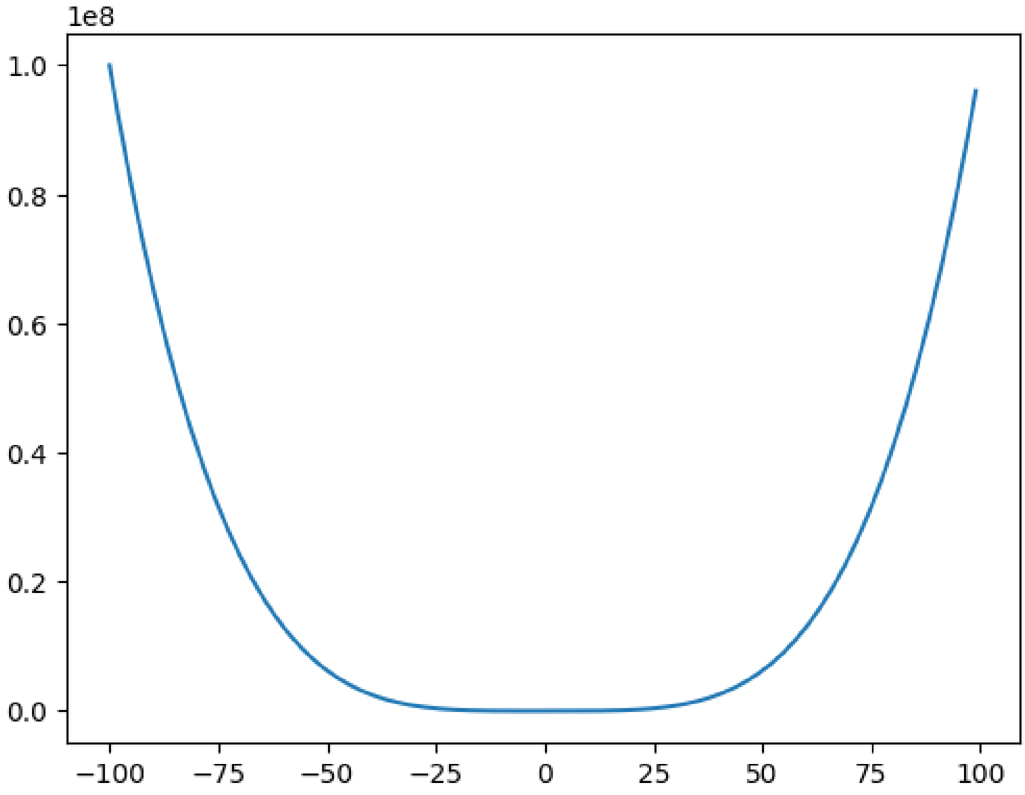
```
[-100, -99, -98, -97, -96, -95, -94, -93, -92, -91, -90, -89, -88, -87, -86,
-85, -84, -83, -82, -81, -80, -79, -78, -77, -76, -75, -74, -73, -72, -71, -70,
-69, -68, -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54,
-53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38,
-37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25, -24, -23, -22,
-21, -20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5,
-4, -3, -2, -1, 0, 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, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99]
[-1000000, -970299, -941192, -912673, -884736, -857375, -830584, -804357,
-778688, -753571, -729000, -704969, -681472, -658503, -636056, -614125, -592704,
-571787, -551368, -531441, -512000, -493039, -474552, -456533, -438976, -421875,
-405224, -389017, -373248, -357911, -343000, -328509, -314432, -300763, -287496,
-274625, -262144, -250047, -238328, -226981, -216000, -205379, -195112, -185193,
-175616, -166375, -157464, -148877, -140608, -132651, -125000, -117649, -110592,
-103823, -97336, -91125, -85184, -79507, -74088, -68921, -64000, -59319, -54872,
-50653, -46656, -42875, -39304, -35937, -32768, -29791, -27000, -24389, -21952,
-19683, -17576, -15625, -13824, -12167, -10648, -9261, -8000, -6859, -5832,
-4913, -4096, -3375, -2744, -2197, -1728, -1331, -1000, -729, -512, -343, -216,
-125, -64, -27, -8, -1, 0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331,
1728, 2197, 2744, 3375, 4096, 4913, 5832, 6859, 8000, 9261, 10648, 12167, 13824,
15625, 17576, 19683, 21952, 24389, 27000, 29791, 32768, 35937, 39304, 42875,
46656, 50653, 54872, 59319, 64000, 68921, 74088, 79507, 85184, 91125, 97336,
103823, 110592, 117649, 125000, 132651, 140608, 148877, 157464, 166375, 175616,
185193, 195112, 205379, 216000, 226981, 238328, 250047, 262144, 274625, 287496,
300763, 314432, 328509, 343000, 357911, 373248, 389017, 405224, 421875, 438976,
456533, 474552, 493039, 512000, 531441, 551368, 571787, 592704, 614125, 636056,
658503, 681472, 704969, 729000, 753571, 778688, 804357, 830584, 857375, 884736,
912673, 941192, 970299]
```



[37]:
```
#plot graph for equation y=x^2
import matplotlib.pyplot as plt
p=int(input("Which index you want?"))
x=[x for x in range(-100,100)]#This structure with output as list,known as list␣
 ↪comprehension
y=[i**p for i in x]
print(x)
print(y)
plt.plot(x,y)
plt.show()
```

```
Which index you want? 4
```

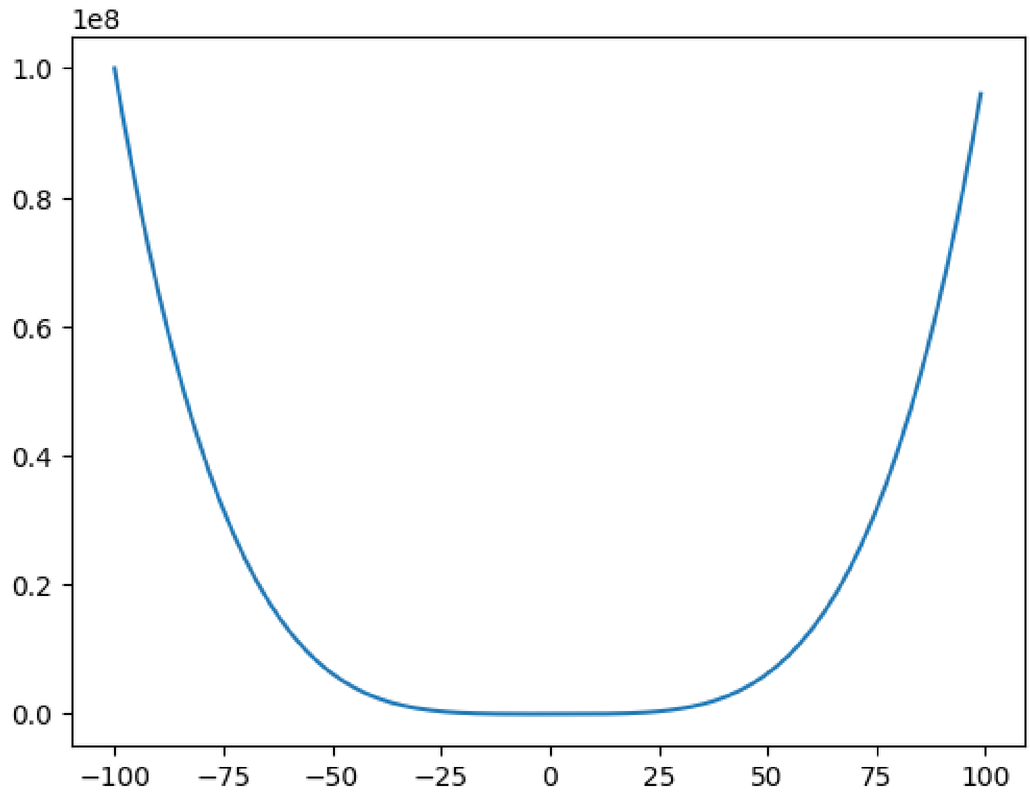
```
[-100, -99, -98, -97, -96, -95, -94, -93, -92, -91, -90, -89, -88, -87, -86,
-85, -84, -83, -82, -81, -80, -79, -78, -77, -76, -75, -74, -73, -72, -71, -70,
-69, -68, -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54,
-53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38,
-37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25, -24, -23, -22,
-21, -20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5,
-4, -3, -2, -1, 0, 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, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99]
[100000000, 96059601, 92236816, 88529281, 84934656, 81450625, 78074896,
74805201, 71639296, 68574961, 65610000, 62742241, 59969536, 57289761, 54700816,
52200625, 49787136, 47458321, 45212176, 43046721, 40960000, 38950081, 37015056,
35153041, 33362176, 31640625, 29986576, 28398241, 26873856, 25411681, 24010000,
22667121, 21381376, 20151121, 18974736, 17850625, 16777216, 15752961, 14776336,
13845841, 12960000, 12117361, 11316496, 10556001, 9834496, 9150625, 8503056,
7890481, 7311616, 6765201, 6250000, 5764801, 5308416, 4879681, 4477456, 4100625,
3748096, 3418801, 3111696, 2825761, 2560000, 2313441, 2085136, 1874161, 1679616,
1500625, 1336336, 1185921, 1048576, 923521, 810000, 707281, 614656, 531441,
456976, 390625, 331776, 279841, 234256, 194481, 160000, 130321, 104976, 83521,
65536, 50625, 38416, 28561, 20736, 14641, 10000, 6561, 4096, 2401, 1296, 625,
256, 81, 16, 1, 0, 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000, 14641,
20736, 28561, 38416, 50625, 65536, 83521, 104976, 130321, 160000, 194481,
234256, 279841, 331776, 390625, 456976, 531441, 614656, 707281, 810000, 923521,
1048576, 1185921, 1336336, 1500625, 1679616, 1874161, 2085136, 2313441, 2560000,
2825761, 3111696, 3418801, 3748096, 4100625, 4477456, 4879681, 5308416, 5764801,
6250000, 6765201, 7311616, 7890481, 8503056, 9150625, 9834496, 10556001,
11316496, 12117361, 12960000, 13845841, 14776336, 15752961, 16777216, 17850625,
18974736, 20151121, 21381376, 22667121, 24010000, 25411681, 26873856, 28398241,
29986576, 31640625, 33362176, 35153041, 37015056, 38950081, 40960000, 43046721,
45212176, 47458321, 49787136, 52200625, 54700816, 57289761, 59969536, 62742241,
65610000, 68574961, 71639296, 74805201, 78074896, 81450625, 84934656, 88529281,
92236816, 96059601]
```



[38]:
```
#plot graph for equation y=x^2
import matplotlib.pyplot as plt
p=int(input("Which index you want?"))
x=[x for x in range(-100,100)]#This structure with output as list,known as list␣
  ↪comprehension
y=[i**p for i in x]
print(x)
print(y)
plt.plot(x,y)
plt.show()
plt.savefig("x3Graph.png")
```

```
Which index you want? 4
```

```
[-100, -99, -98, -97, -96, -95, -94, -93, -92, -91, -90, -89, -88, -87, -86,
-85, -84, -83, -82, -81, -80, -79, -78, -77, -76, -75, -74, -73, -72, -71, -70,
-69, -68, -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54,
-53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38,
-37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25, -24, -23, -22,
-21, -20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5,
-4, -3, -2, -1, 0, 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, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99]
[100000000, 96059601, 92236816, 88529281, 84934656, 81450625, 78074896,
74805201, 71639296, 68574961, 65610000, 62742241, 59969536, 57289761, 54700816,
52200625, 49787136, 47458321, 45212176, 43046721, 40960000, 38950081, 37015056,
35153041, 33362176, 31640625, 29986576, 28398241, 26873856, 25411681, 24010000,
22667121, 21381376, 20151121, 18974736, 17850625, 16777216, 15752961, 14776336,
13845841, 12960000, 12117361, 11316496, 10556001, 9834496, 9150625, 8503056,
7890481, 7311616, 6765201, 6250000, 5764801, 5308416, 4879681, 4477456, 4100625,
3748096, 3418801, 3111696, 2825761, 2560000, 2313441, 2085136, 1874161, 1679616,
1500625, 1336336, 1185921, 1048576, 923521, 810000, 707281, 614656, 531441,
456976, 390625, 331776, 279841, 234256, 194481, 160000, 130321, 104976, 83521,
65536, 50625, 38416, 28561, 20736, 14641, 10000, 6561, 4096, 2401, 1296, 625,
256, 81, 16, 1, 0, 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000, 14641,
20736, 28561, 38416, 50625, 65536, 83521, 104976, 130321, 160000, 194481,
234256, 279841, 331776, 390625, 456976, 531441, 614656, 707281, 810000, 923521,
1048576, 1185921, 1336336, 1500625, 1679616, 1874161, 2085136, 2313441, 2560000,
2825761, 3111696, 3418801, 3748096, 4100625, 4477456, 4879681, 5308416, 5764801,
6250000, 6765201, 7311616, 7890481, 8503056, 9150625, 9834496, 10556001,
11316496, 12117361, 12960000, 13845841, 14776336, 15752961, 16777216, 17850625,
18974736, 20151121, 21381376, 22667121, 24010000, 25411681, 26873856, 28398241,
29986576, 31640625, 33362176, 35153041, 37015056, 38950081, 40960000, 43046721,
45212176, 47458321, 49787136, 52200625, 54700816, 57289761, 59969536, 62742241,
65610000, 68574961, 71639296, 74805201, 78074896, 81450625, 84934656, 88529281,
92236816, 96059601]
```

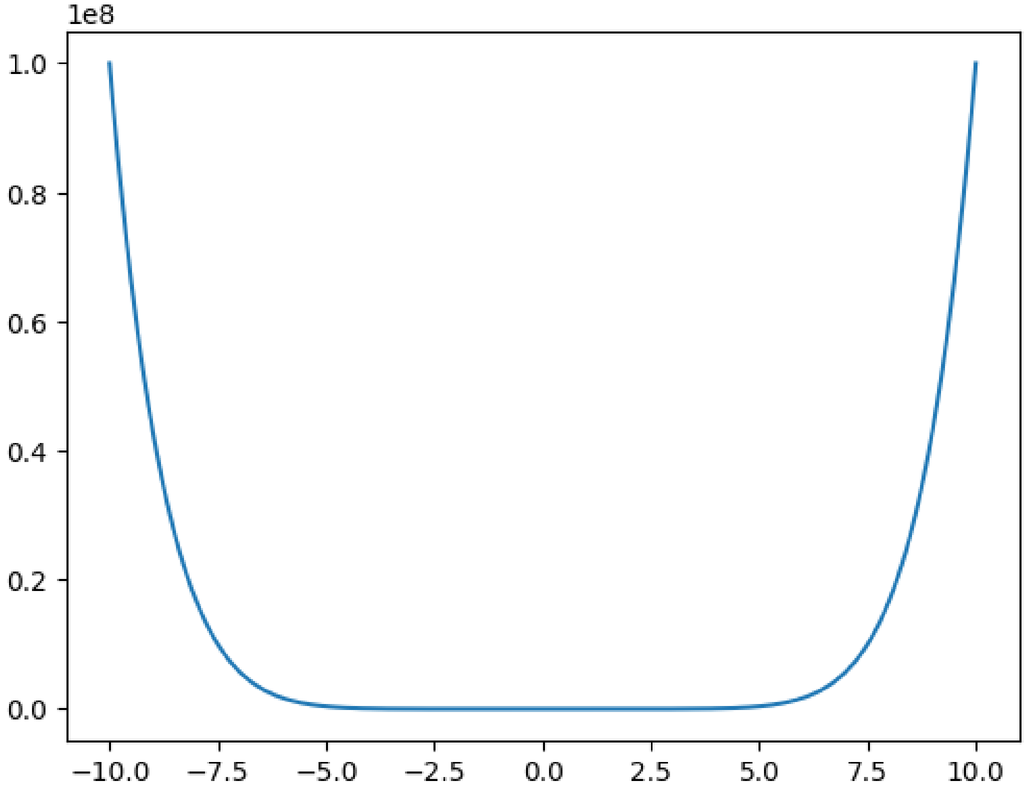


```
<Figure size 640x480 with 0 Axes>
```

```
[39]: def plot_equation():
          import matplotlib.pyplot as plt
          import numpy as np
          p=int(input("which index you want"))
          x=np.linspace(-10,10,200)
          y=[i**p for i in x]
          plt.plot(x,y)
          plt.show()
```

```
[41]: plot_equation()
```

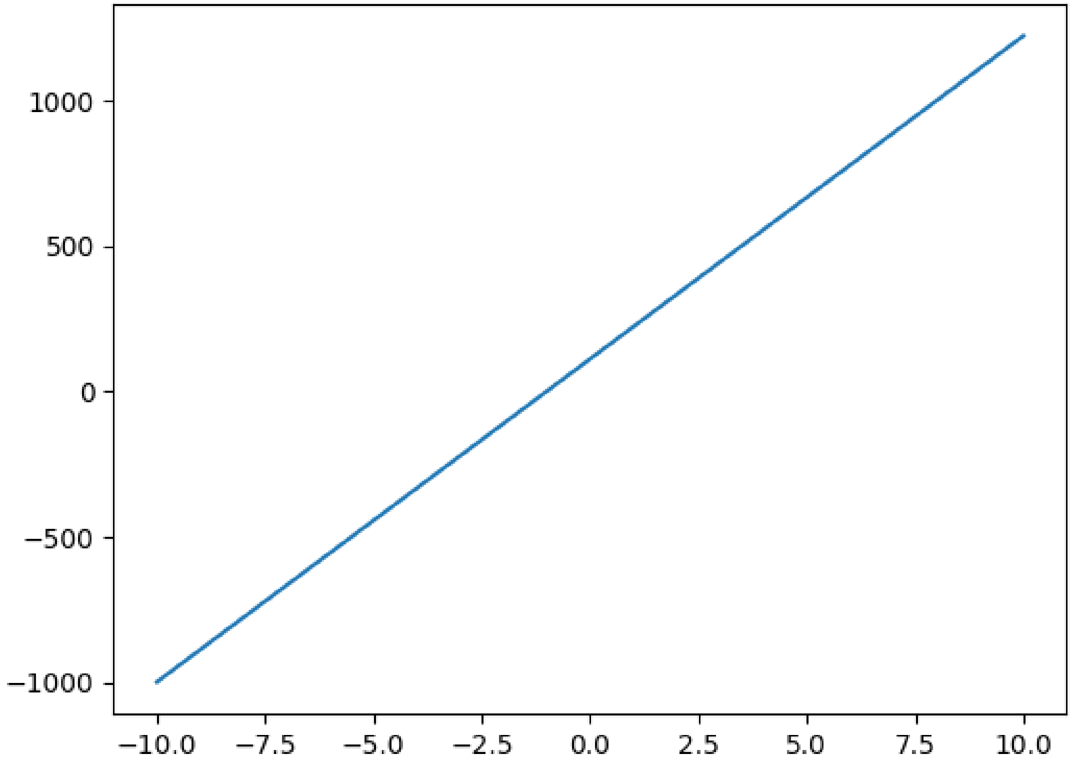
```
which index you want 8
```



[42]:

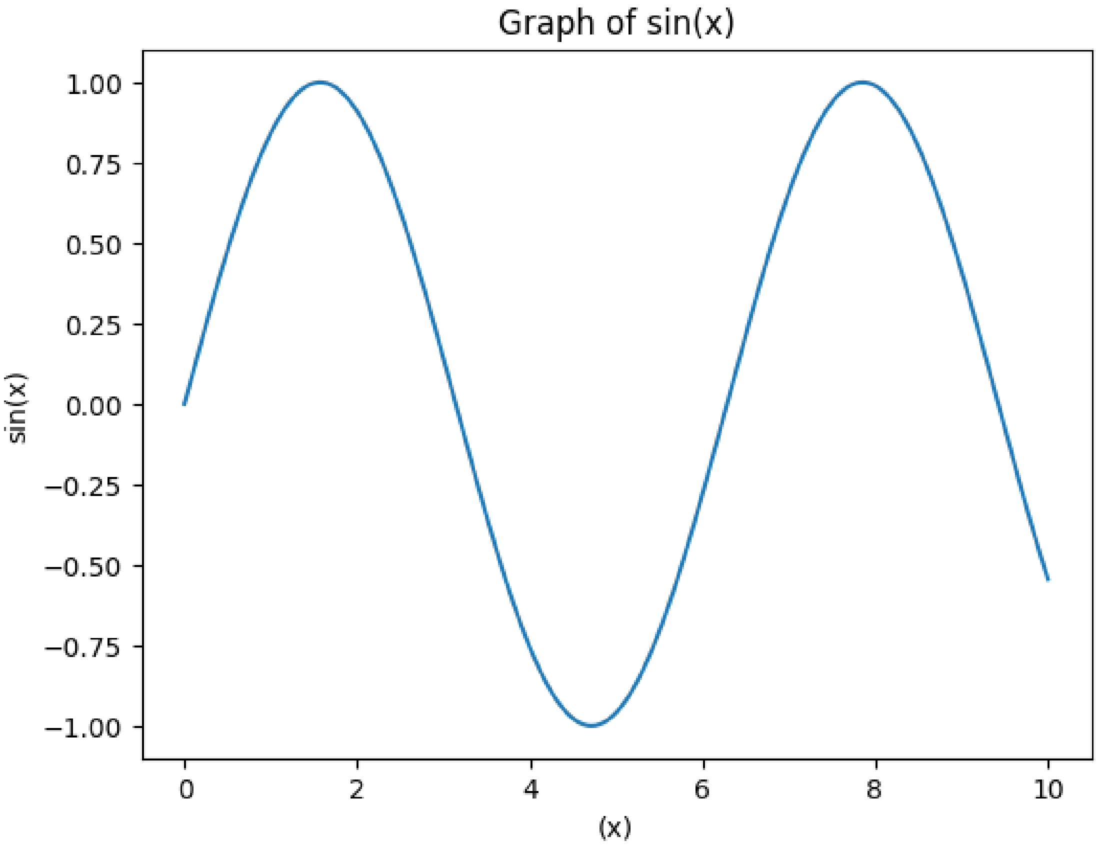
```
#plot graph for equation y=mx+c.
import matplotlib.pyplot as plt
import numpy as np
m=int(input("Slope?"))
c=int(input("c?"))
x=np.linspace(-10,10,200)
y=[(i*m)+c for i in x]
plt.plot(x,y)
plt.show()
```

```
Slope? 111
c? 111
```



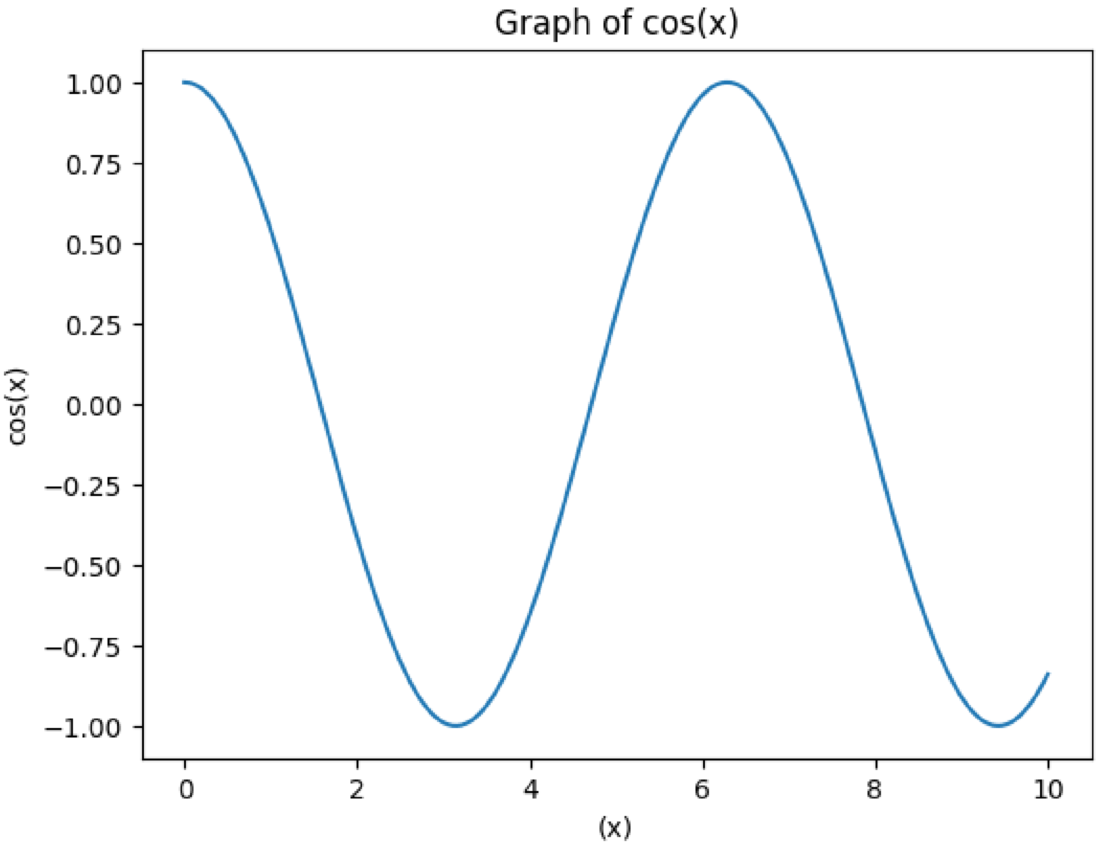
[43]:

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,10,400)
y=np.sin(x)
plt.plot(x,y)
plt.title("Graph of sin(x)")
plt.xlabel("(x)")
plt.ylabel("sin(x)")
plt.show()
```



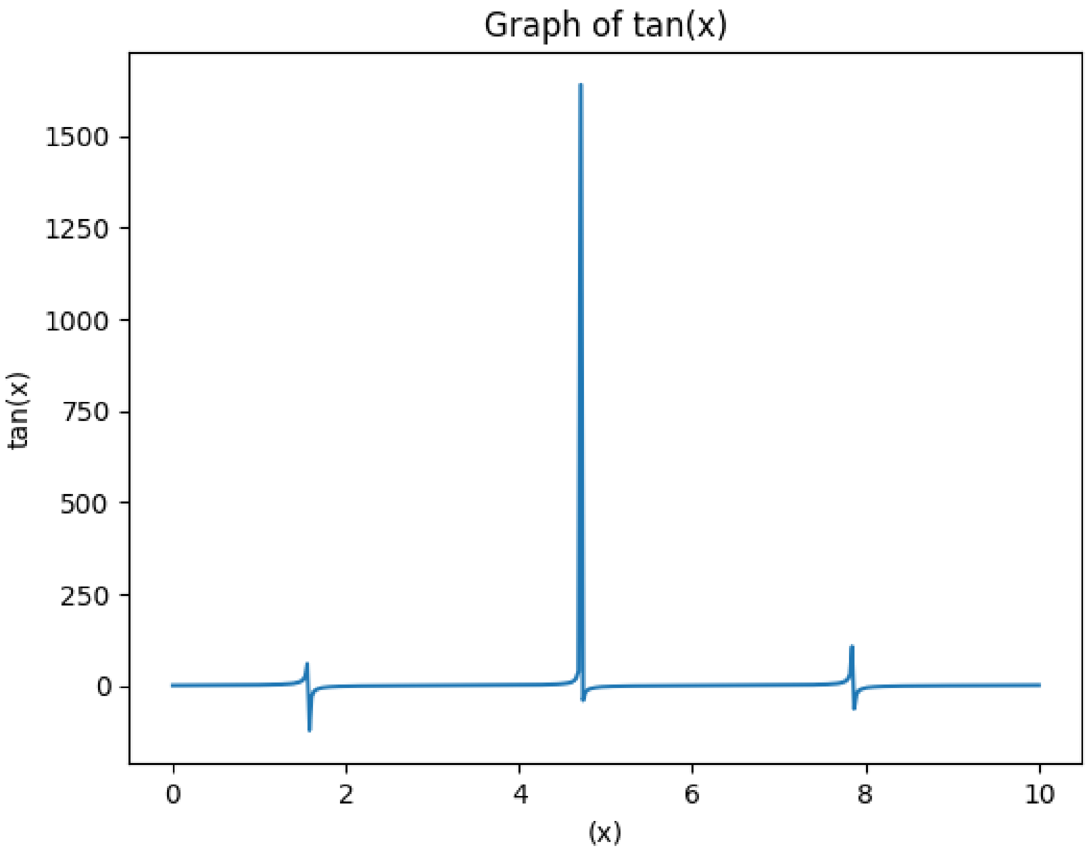
[44]:

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,10,400)
y=np.cos(x)
plt.plot(x,y)
plt.title("Graph of cos(x)")
plt.xlabel("(x)")
plt.ylabel("cos(x)")
plt.show()
```



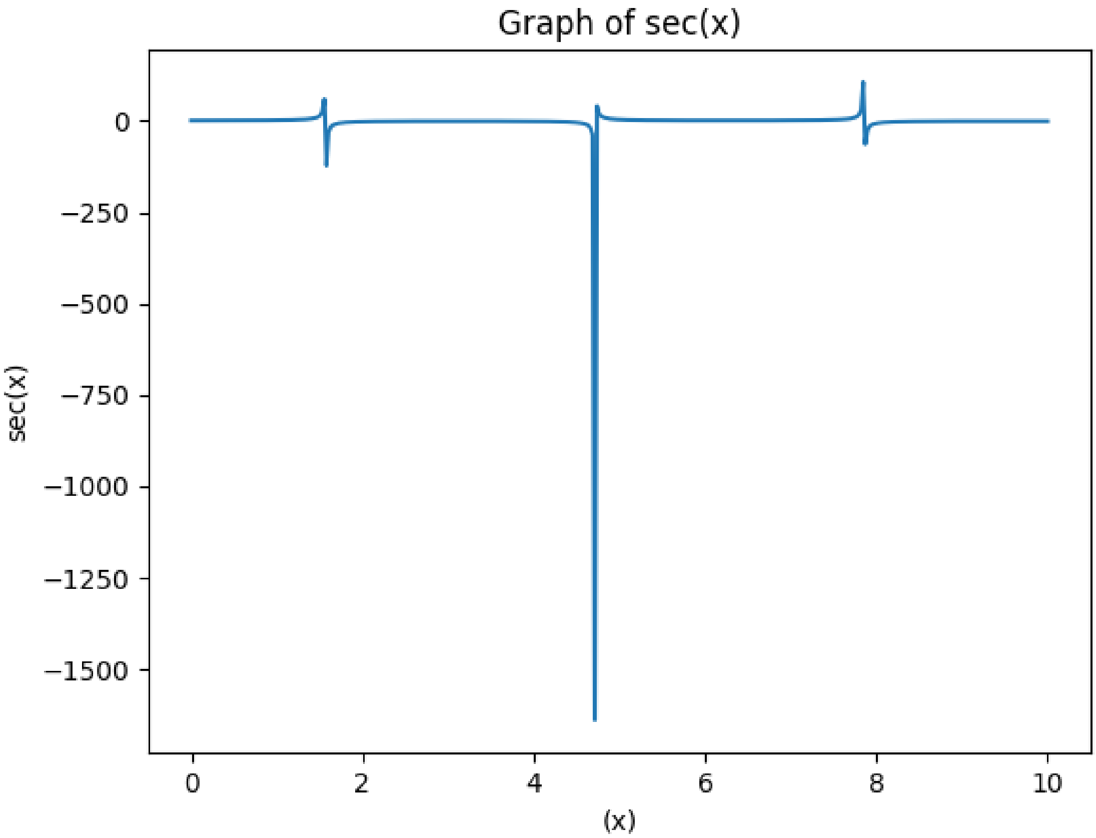
[45]:

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,10,400)
y=np.tan(x)
plt.plot(x,y)
plt.title("Graph of tan(x)")
plt.xlabel("(x)")
plt.ylabel("tan(x)")
plt.show()
```



[46]:

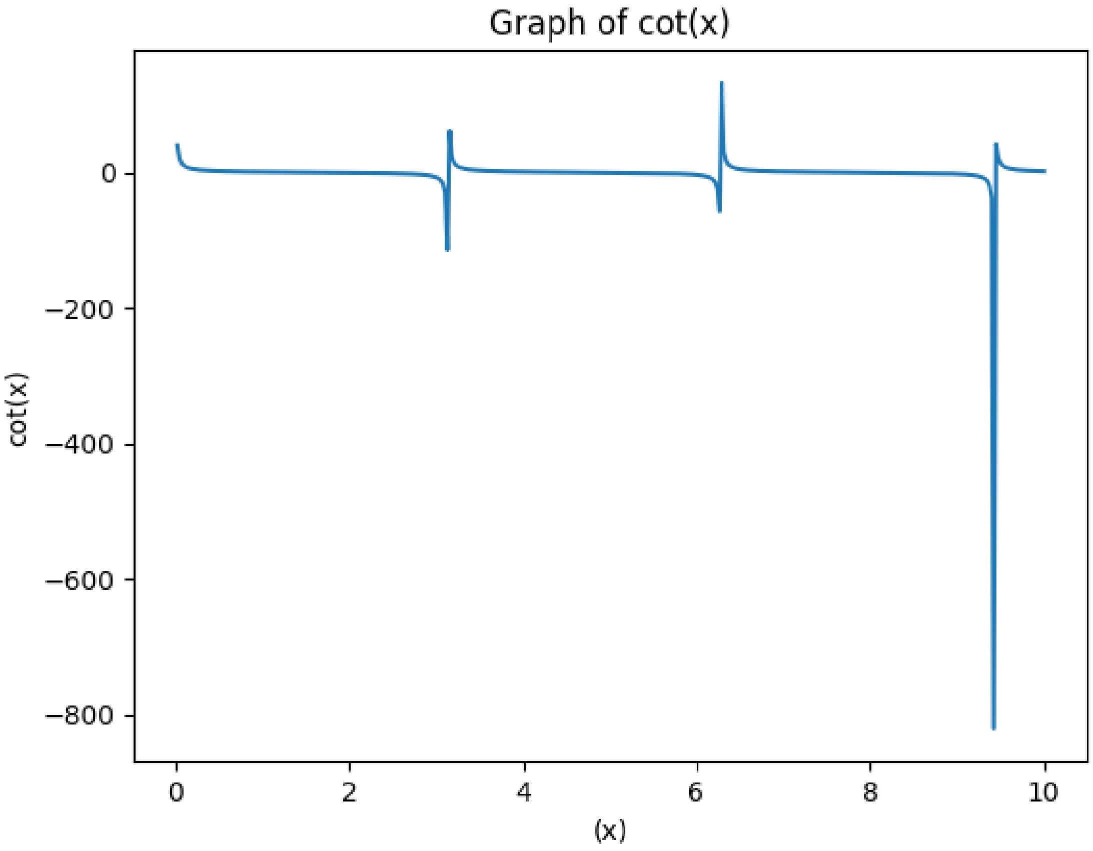
```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,10,400)
y=1/np.cos(x)
plt.plot(x,y)
plt.title("Graph of sec(x)")
plt.xlabel("(x)")
plt.ylabel("sec(x)")
plt.show()
```



[47]:

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,10,400)
y=1/np.tan(x)
plt.plot(x,y)
plt.title("Graph of cot(x)")
plt.xlabel("(x)")
plt.ylabel("cot(x)")
plt.show()
```

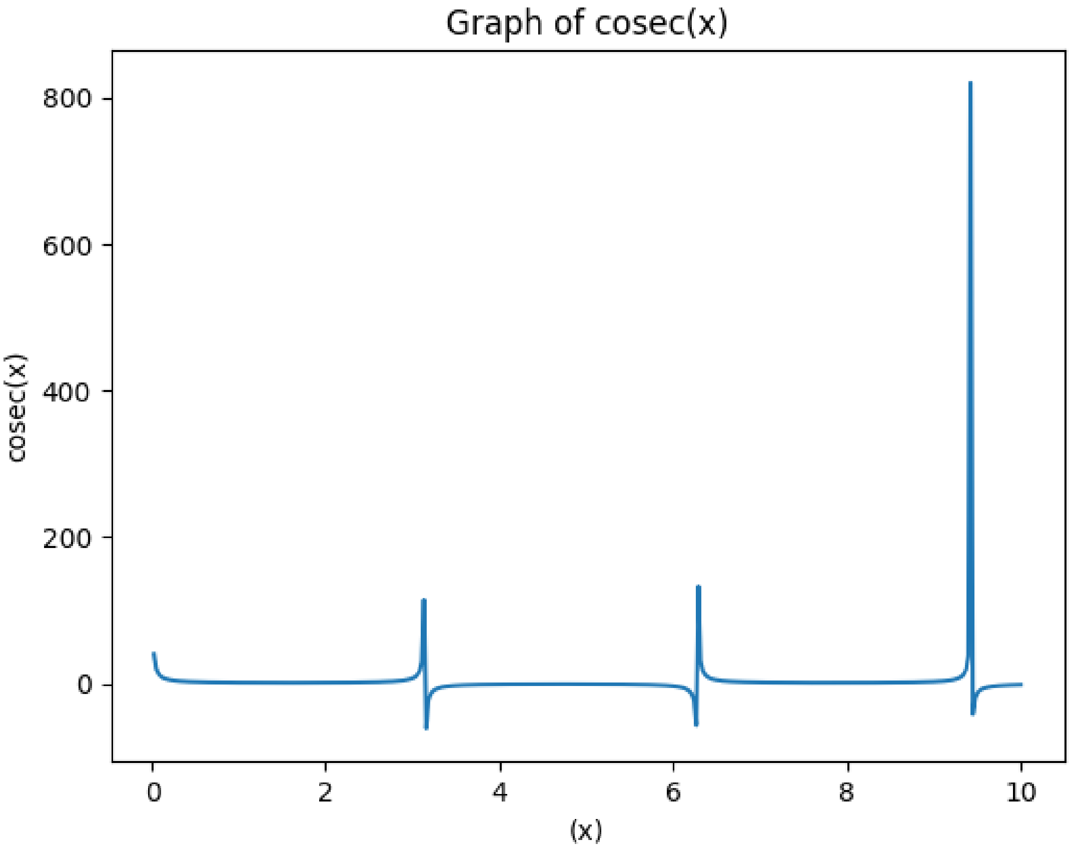
```
C:\Users\janam\AppData\Local\Temp\ipykernel_33172\2940608951.py:4:
RuntimeWarning: divide by zero encountered in divide
  y=1/np.tan(x)
```



[48]:

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,10,400)
y=1/np.sin(x)
plt.plot(x,y)
plt.title("Graph of cosec(x)")
plt.xlabel("(x)")
plt.ylabel("cosec(x)")
plt.show()
```

```
C:\Users\janam\AppData\Local\Temp\ipykernel_33172\3354266311.py:4:
RuntimeWarning: divide by zero encountered in divide
  y=1/np.sin(x)
```



[49]:
```
x=np.array([2,3,4,5])
```

[50]:
```
print(x)
```

```
[2 3 4 5]
```

[51]:
```
m1=np.array([[1,2,3],[4,5,6]])#2-dimensional array
```

[52]:
```
print(m1)
```

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

[53]:
```
m1=np.array([[1,2,3],[4,5,6]])#2-dimensional array
m2=np.array([[2,4,5],[4,9,8]])
```

[54]:
```
print(m1)
print(m2)
print(m1+m2)
```

```
[[1 2 3]
 [4 5 6]]
[[2 4 5]
 [4 9 8]]
[[ 3  6  8]
 [ 8 14 14]]
```

[55]:

```
m1[0]#Indexing in the matrix
```

[55]:

```
array([1, 2, 3])
```

[56]:

```
print(m2)#sub-matrix
m2[0:3,1:3]
```

```
[[2 4 5]
 [4 9 8]]
```

[56]:

```
array([[4, 5],
       [9, 8]])
```

[57]:

```
print(m1)
print(m2)
print(m1*m2)
```

```
[[1 2 3]
 [4 5 6]]
[[2 4 5]
 [4 9 8]]
[[ 2  8 15]
 [16 45 48]]
```

[58]:

```
m3=np.array([4,88,6])
m4=np.array([2,11,5])
j=np.mod(m3,m4)
print(j)
```

```
[0 0 1]
```

[59]:

```
mprod=m3*m4
print(mprod)
```

```
[  8 968  30]
```

[60]:

```
m_dot_prod=np.dot(m3,m4)
print(m_dot_prod)
```

```
1006
```

```
[61]: m_transpose=np.transpose(m2)
```

```
[62]: m_transpose
```

```
[62]: array([[2, 4],
             [4, 9],
             [5, 8]])
```

```
[63]: matrix=np.array([[2,3,7],[2,5,6],[12,52,54]])
      inverse=np.linalg.inv(matrix)
```

```
[64]: inverse
```

```
[64]: array([[-0.36206897,  1.74137931, -0.14655172],
             [-0.31034483,  0.20689655,  0.01724138],
             [ 0.37931034, -0.5862069 ,  0.03448276]])
```

```
[65]: det_matrix=np.linalg.det(matrix)
      print(det_matrix)
```

```
116.00000000000009
```

```
[66]: #solve 2x+y=8, 3x+4y=18 using matrix
      m1=np.array([[2,1],[3,4]])
      co_m2=np.array([8,18])
      solution=np.linalg.solve(m1,co_m2)
      print(solution)
```

```
[2.8 2.4]
```

```
[67]: #solve 3x+4y=10, x+y=5
      m1=np.array([[3,4],[1,1]])
      co_m2=np.array([10,5])
      solution=np.linalg.solve(m1,co_m2)
      print(solution)
```

```
[10. -5.]
```

```
[68]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      df=pd.read_csv("Amazon sales Data.csv",encoding='unicode_escape')
```

```
[69]: df
```

```
[69]:       User_ID    Cust_name Product_ID Gender Age Group  Age  Marital_Status  \
      0     1002903    Sanskriti  P00125942      F     26-35   28               0
      1     1000732       Kartik  P00110942      F     26-35   35               1
```

```
2      1001990        Bindu  P00118542      F     26-35   35               1
3      1001425       Sudevi  P00237842      M      0-17   16               0
4      1000588         Joni  P00057942      M     26-35   28               1
…        …          …        …    …       …  …          …
11246  1000695      Manning  P00296942      M     18-25   19               1
11247  1004089  Reichenbach  P00171342      M     26-35   33               0
11248  1001209        Oshin  P00201342      F     36-45   40               0
11249  1004023       Noonan  P00059442      M     36-45   37               0
11250  1002744      Brumley  P00281742      F     18-25   19               0

                State      Zone       Occupation Product_Category  Orders  \
0         Maharashtra   Western       Healthcare             Auto       1
1      Andhra Pradesh  Southern             Govt             Auto       3
2       Uttar Pradesh   Central       Automobile             Auto       3
3           Karnataka  Southern     Construction             Auto       2
4             Gujarat   Western  Food Processing             Auto       2
…                …       …              …              …      …
11246     Maharashtra   Western         Chemical           Office       4
11247         Haryana  Northern       Healthcare       Veterinary       3
11248  Madhya Pradesh   Central          Textile           Office       4
11249       Karnataka  Southern      Agriculture           Office       3
11250     Maharashtra   Western       Healthcare           Office       3

        Amount  Status  unnamed1
0      23952.0     NaN       NaN
1      23934.0     NaN       NaN
2      23924.0     NaN       NaN
3      23912.0     NaN       NaN
4      23877.0     NaN       NaN
…         …     …       …
11246    370.0     NaN       NaN
11247    367.0     NaN       NaN
11248    213.0     NaN       NaN
11249    206.0     NaN       NaN
11250    188.0     NaN       NaN

[11251 rows x 15 columns]
```

[70]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   User_ID           11251 non-null  int64
 1   Cust_name         11251 non-null  object
 2   Product_ID        11251 non-null  object
 3   Gender            11251 non-null  object
 4   Age Group         11251 non-null  object
 5   Age               11251 non-null  int64
 6   Marital_Status    11251 non-null  int64
 7   State             11251 non-null  object
 8   Zone              11251 non-null  object
 9   Occupation        11251 non-null  object
 10  Product_Category  11251 non-null  object
 11  Orders            11251 non-null  int64
 12  Amount            11239 non-null  float64
 13  Status            0 non-null      float64
 14  unnamed1          0 non-null      float64
dtypes: float64(3), int64(4), object(8)
memory usage: 1.3+ MB
```

[71]:

```
#Drop empty columns, status and unnamed1
df.drop(["Status","unnamed1"],axis=1,inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   User_ID           11251 non-null  int64
 1   Cust_name         11251 non-null  object
 2   Product_ID        11251 non-null  object
 3   Gender            11251 non-null  object
 4   Age Group         11251 non-null  object
 5   Age               11251 non-null  int64
 6   Marital_Status    11251 non-null  int64
 7   State             11251 non-null  object
 8   Zone              11251 non-null  object
 9   Occupation        11251 non-null  object
 10  Product_Category  11251 non-null  object
 11  Orders            11251 non-null  int64
 12  Amount            11239 non-null  float64
dtypes: float64(1), int64(4), object(8)
memory usage: 1.1+ MB
```

[72]:

```
df.isna().sum()
```

[72]:

```
User_ID             0
Cust_name           0
Product_ID          0
Gender              0
```

```
Age Group           0
Age                 0
Marital_Status      0
State               0
Zone                0
Occupation          0
Product_Category    0
Orders              0
Amount             12
dtype: int64
```

[73]:

```
df.dropna(inplace=True)
```

[74]:

```
df.info()
```

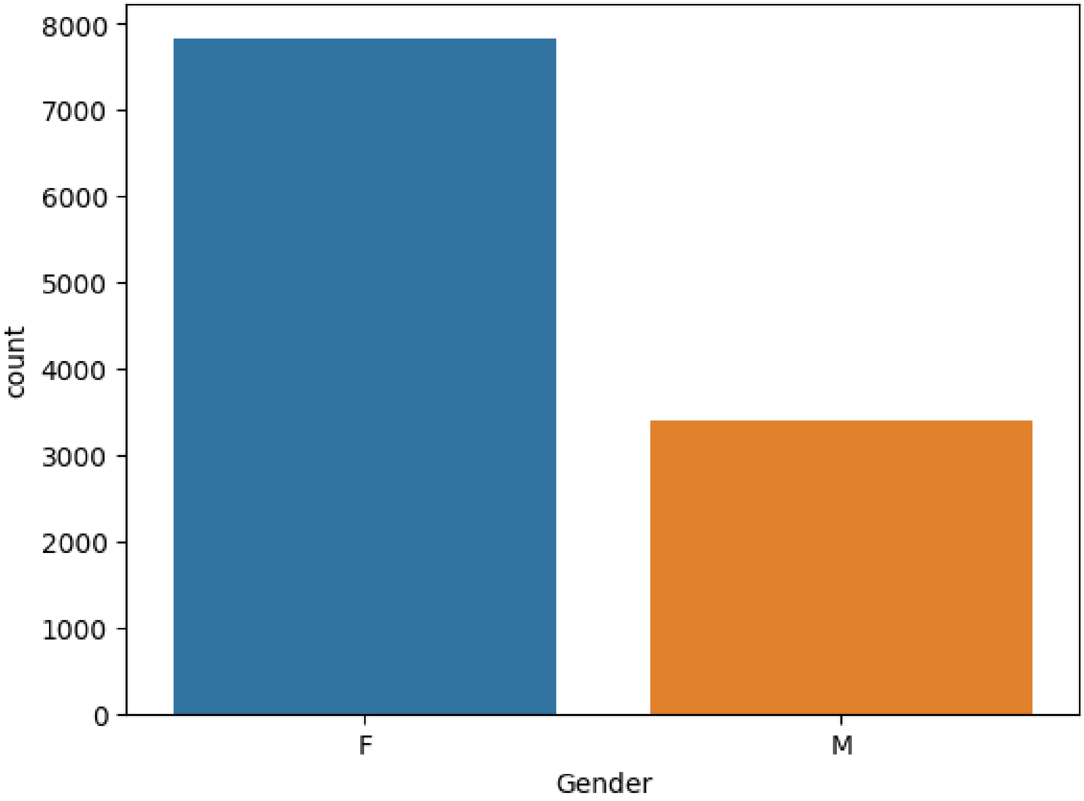
```
<class 'pandas.core.frame.DataFrame'>
Index: 11239 entries, 0 to 11250
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   User_ID           11239 non-null  int64
 1   Cust_name         11239 non-null  object
 2   Product_ID        11239 non-null  object
 3   Gender            11239 non-null  object
 4   Age Group         11239 non-null  object
 5   Age               11239 non-null  int64
 6   Marital_Status    11239 non-null  int64
 7   State             11239 non-null  object
 8   Zone              11239 non-null  object
 9   Occupation        11239 non-null  object
 10  Product_Category  11239 non-null  object
 11  Orders            11239 non-null  int64
 12  Amount            11239 non-null  float64
dtypes: float64(1), int64(4), object(8)
memory usage: 1.2+ MB
```

[75]:

```
import seaborn as sns
ax=sns.countplot(x='Gender',data=df)
for bars in ax.containers:
    sns.bar.label(bars)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[75], line 4
      2 ax=sns.countplot(x='Gender',data=df)
      3 for bars in ax.containers:
----> 4     sns.bar.label(bars)
```

```
AttributeError: module 'seaborn' has no attribute 'bar'
```



[76]:
```
sales_gen=df.groupby(['Gender'],as_index=False)['Amount'].sum()
```

[77]:
```
print(sales_gen)
```

```
  Gender       Amount
0      F  74335856.43
1      M  31913276.00
```

[78]:
```
sales_state=df.groupby(['State'],as_index=False)['Amount'].sum()
```

[79]:
```
print(sales_state)
```

```
                State       Amount
0      Andhra Pradesh   8037146.99
1               Bihar   4022757.00
2               Delhi  11603819.45
3             Gujarat   3946082.00
4             Haryana   4220175.00
```

```
5   Himachal Pradesh   4963368.00
6          Jharkhand   3026456.00
7          Karnataka  13523540.00
8             Kerala   3894491.99
9     Madhya Pradesh   8101142.00
10       Maharashtra  14427543.00
11            Punjab   1525800.00
12         Rajasthan   1909409.00
13         Telangana   1151490.00
14     Uttar Pradesh  19374968.00
15       Uttarakhand   2520944.00
```

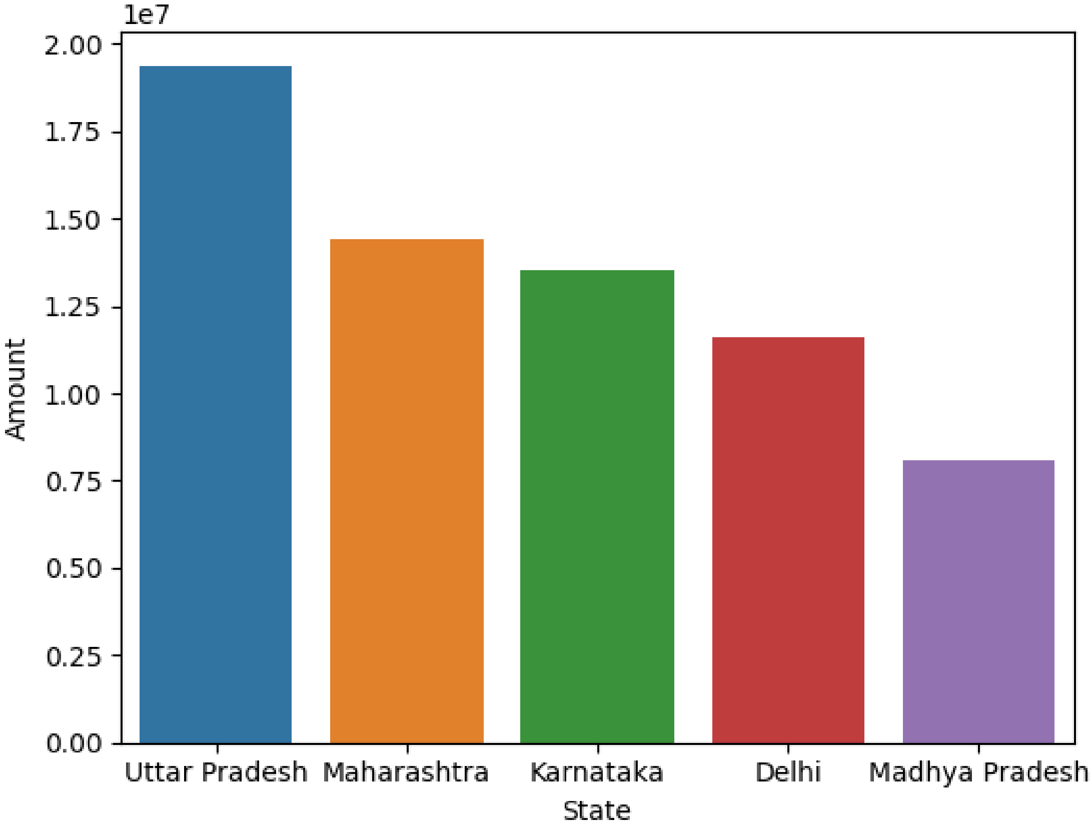
[80]:

```
sales_state=df.groupby(['State'],as_index=False)['Amount'].sum().
 ↪sort_values(ascending=False,by="Amount").head()
print(sales_state.head())
sns.barplot(x="State",y="Amount",data=sales_state)
```

```
             State       Amount
14   Uttar Pradesh  19374968.00
10     Maharashtra  14427543.00
7        Karnataka  13523540.00
2            Delhi  11603819.45
9   Madhya Pradesh   8101142.00
```

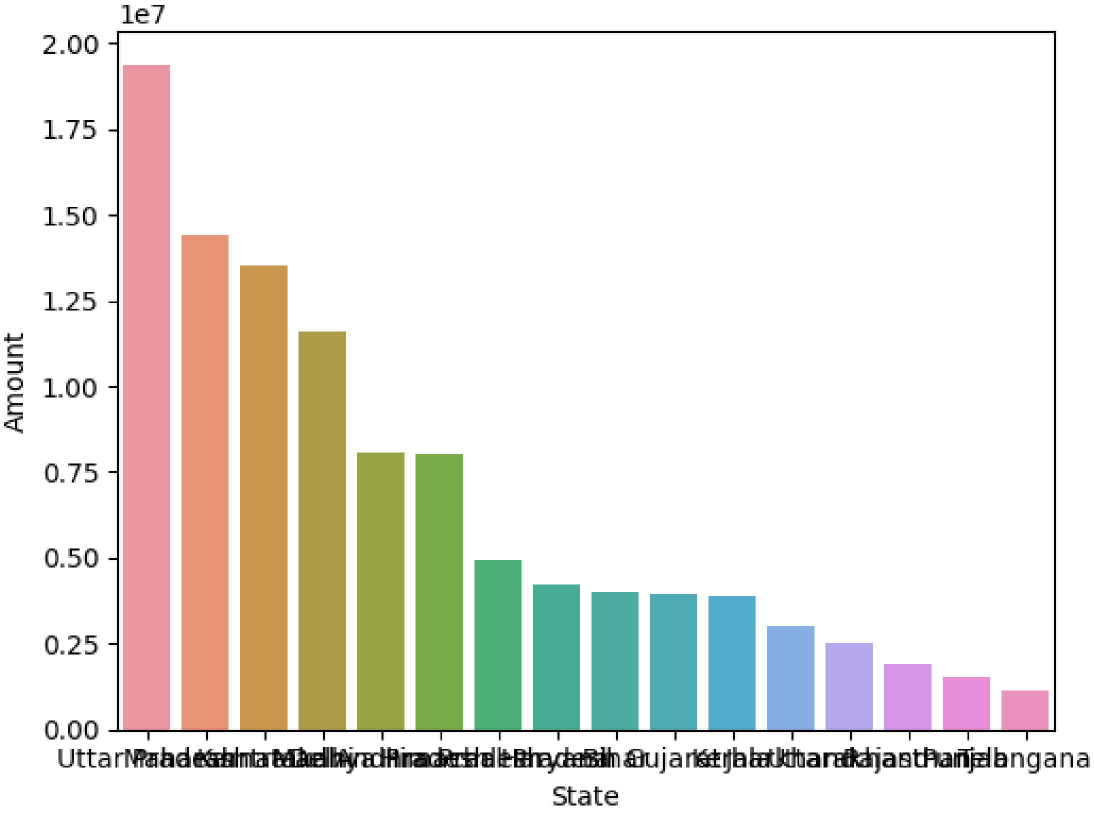
[80]:

```
<Axes: xlabel='State', ylabel='Amount'>
```



```
[82]: sales_state=df.groupby(['State'],as_index=False)['Amount'].sum().
        ↪sort_values(ascending=False,by="Amount")
      sales_state
      sns.barplot(x="State",y="Amount",data=sales_state)
```

```
[82]: <Axes: xlabel='State', ylabel='Amount'>
```


1e7
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Amount
Telangana
State

```
[ ]:
```