In [437]:

```
a = 9

for i in range(0,7):
    for j in range(0,int(a/2)):
        if(i%2==0):
            print(" ",end="")

            print(" ",end="")

        else:
            for x in range(0,j):
                print(" ",end="")




    if(i%2!=0):
        if(i!=1):
            print(" ",end="")
        print(" ",end="")


    for k in range(0,i):
        print("*", end = "")
        if(i%2!=0):
             print("*", end = "")


        else:
            print("*", end = "")

    print("*", end = "")



    for l in range(0,int(a/2)):

        print(" ", end="")

    a= a-1
```

```
    print()
```

```
        *
       ***
      *****
     *******
    *********
   ***********
  *************
```

In [438]:
```
class test:
    pass
```

In [439]:
```
import pandas as pd
import numpy as np

df = pd.read_csv(r"D:\PAP_prep\Placement Test\Python\Churn_Modelling.csv")
```

In [440]:
```
df.head()
```

Out[440]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 |
| **2** | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 |

Q3. (1)Details of Customer there location is Spain,France and Excited (it is column in dataset) from Bank

In [441]:
```
df[(df["Geography"]=="Spain") & df["Exited"]==1]
```

Out[441]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMem |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **5** | 6 | 15574012 | Chu | 645 | Spain | Male | 44 | 8 | 113755.78 | 2 | 1 | |
| **22** | 23 | 15699309 | Gerasimov | 510 | Spain | Female | 38 | 4 | 0.00 | 1 | 1 | |
| **30** | 31 | 15589475 | Azikiwe | 591 | Spain | Female | 39 | 3 | 0.00 | 3 | 1 | |
| **58** | 59 | 15623944 | T'ien | 511 | Spain | Female | 66 | 4 | 0.00 | 1 | 1 | |
| **86** | 87 | 15762418 | Gant | 750 | Spain | Male | 22 | 3 | 121681.82 | 1 | 1 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9718** | 9719 | 15704053 | T'ang | 710 | Spain | Male | 62 | 3 | 131078.42 | 2 | 1 | |
| **9756** | 9757 | 15662698 | Ko | 648 | Spain | Female | 43 | 7 | 81153.82 | 1 | 1 | |
| **9800** | 9801 | 15640507 | Li | 762 | Spain | Female | 35 | 3 | 119349.69 | 3 | 1 | |
| **9852** | 9853 | 15718765 | Maclean | 501 | Spain | Male | 43 | 6 | 104533.24 | 1 | 0 | |
| **9962** | 9963 | 15594612 | Flynn | 702 | Spain | Male | 44 | 9 | 0.00 | 1 | 0 | |

413 rows × 14 columns

(2)Define another column within dataframe If salary >100000 - ‘you are eligible for loan’ Else: - ‘you are not eligible for loan’(column name - Loan Eligibility)

In [442]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [ ]:
```

```

In [445]:
```
df['Loan Eligibility'] = np.where(df['EstimatedSalary']>100000, "Eligible", "Not Eligible")
df.head()
```

Out[445]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 |
| **2** | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 |

(3)In this Dataframe Region and Gender are in Categorical(String) format - Convert it into the Numerical format so that we can apply ML Algo on it.

In [447]:
```
df.Geography.value_counts()
```

Out[447]:
```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

In [448]:
```
df.Gender.value_counts()
```

Out[448]:
```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

In [452]:
```
Geography = {"France": 1, "Germany": 2, "Spain": 3}
Gender = {"Male": 1, "Female": 2}

df = df.replace({"Geography": Geography})
df = df.replace({"Gender":Gender})
```

In [455]:
```
df.head(25)
```

Out[455]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMembe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | 1 | 2 | 42 | 2 | 0.00 | 1 | 1 | |
| **1** | 2 | 15647311 | Hill | 608 | 3 | 2 | 41 | 1 | 83807.86 | 1 | 0 | |
| **2** | 3 | 15619304 | Onio | 502 | 1 | 2 | 42 | 8 | 159660.80 | 3 | 1 | |
| **3** | 4 | 15701354 | Boni | 699 | 1 | 2 | 39 | 1 | 0.00 | 2 | 0 | |
| **4** | 5 | 15737888 | Mitchell | 850 | 3 | 2 | 43 | 2 | 125510.82 | 1 | 1 | |
| **5** | 6 | 15574012 | Chu | 645 | 3 | 1 | 44 | 8 | 113755.78 | 2 | 1 | |
| **6** | 7 | 15592531 | Bartlett | 822 | 1 | 1 | 50 | 7 | 0.00 | 2 | 1 | |
| **7** | 8 | 15656148 | Obinna | 376 | 2 | 2 | 29 | 4 | 115046.74 | 4 | 1 | |
| **8** | 9 | 15792365 | He | 501 | 1 | 1 | 44 | 4 | 142051.07 | 2 | 0 | |
| **9** | 10 | 15592389 | H? | 684 | 1 | 1 | 27 | 2 | 134603.88 | 1 | 1 | |
| **10** | 11 | 15767821 | Bearce | 528 | 1 | 1 | 31 | 6 | 102016.72 | 2 | 0 | |
| **11** | 12 | 15737173 | Andrews | 497 | 3 | 1 | 24 | 3 | 0.00 | 2 | 1 | |
| **12** | 13 | 15632264 | Kay | 476 | 1 | 2 | 34 | 10 | 0.00 | 2 | 1 | |
| **13** | 14 | 15691483 | Chin | 549 | 1 | 2 | 25 | 5 | 0.00 | 2 | 0 | |
| **14** | 15 | 15600882 | Scott | 635 | 3 | 2 | 35 | 7 | 0.00 | 2 | 1 | |
| **15** | 16 | 15643966 | Goforth | 616 | 2 | 1 | 45 | 3 | 143129.41 | 2 | 0 | |
| **16** | 17 | 15737452 | Romeo | 653 | 2 | 1 | 58 | 1 | 132602.88 | 1 | 1 | |

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMembe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **17** | 18 | 15788218 | Henderson | 549 | 3 | 2 | 24 | 9 | 0.00 | 2 | 1 | |
| **18** | 19 | 15661507 | Muldrow | 587 | 3 | 1 | 45 | 6 | 0.00 | 1 | 0 | |
| **19** | 20 | 15568982 | Hao | 726 | 1 | 2 | 24 | 6 | 0.00 | 2 | 1 | |
| **20** | 21 | 15577657 | McDonald | 732 | 1 | 1 | 41 | 8 | 0.00 | 2 | 1 | |
| **21** | 22 | 15597945 | Dellucci | 636 | 3 | 2 | 32 | 8 | 0.00 | 2 | 1 | |
| **22** | 23 | 15699309 | Gerasimov | 510 | 3 | 2 | 38 | 4 | 0.00 | 1 | 1 | |
| **23** | 24 | 15725737 | Mosman | 669 | 1 | 1 | 46 | 3 | 0.00 | 2 | 0 | |
| **24** | 25 | 15625047 | Yen | 846 | 1 | 2 | 38 | 5 | 0.00 | 1 | 1 | |

(II) (1) Find relation between x and y (Hint - Matplotlib)

In [464]:
```
from scipy import stats
X = np.array([1,4,2,6,7])
Y = np.array([7,5,89,1,90])
correlation, p_value = stats.pearsonr(X, Y)
```

In [465]:
```
correlation
```

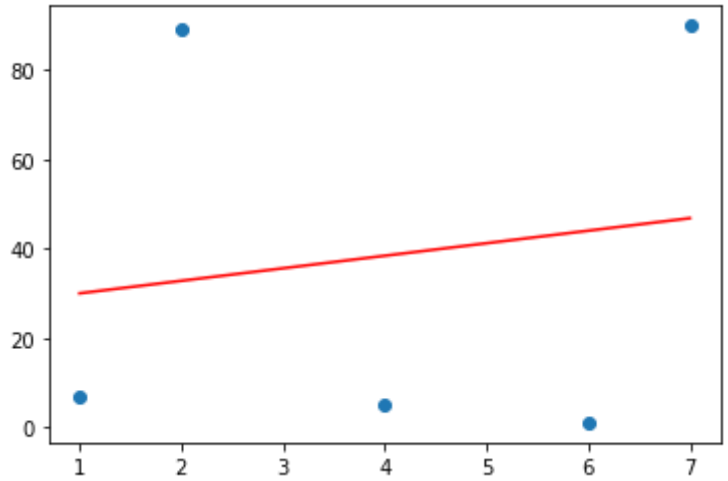
Out[465]: 0.1532845104631264

In [470]:
```
import matplotlib.pyplot as plt

plt.scatter(X,Y)

plt.plot(np.unique(X), np.poly1d(np.polyfit(X, Y, 1))
         (np.unique(X)), color='red')
```

Out[470]: [<matplotlib.lines.Line2D at 0x1b12a71aee0>]

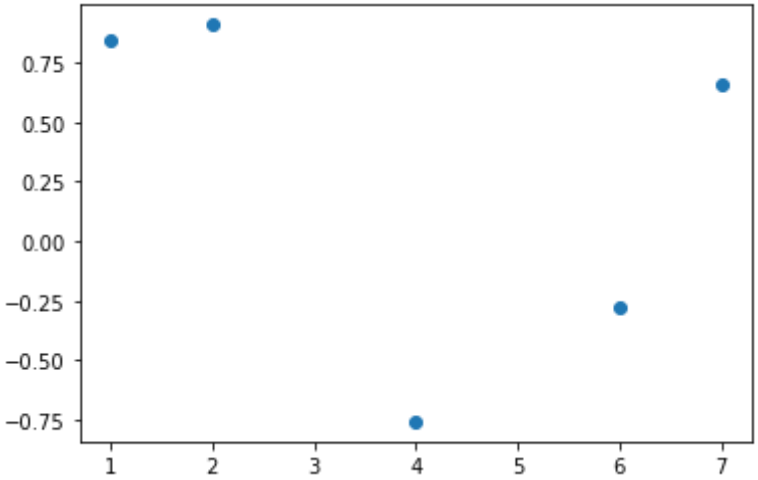



(2) plot graph between X and sinX

In [474]:
```
import matplotlib.pyplot as plt

plt.scatter(X,np.sin(X))
```

Out[474]: <matplotlib.collections.PathCollection at 0x1b12b821040>




Q4-)1)Factorial of 13Using Recursion and without iteration

In [ ]:

In [521]:
```
def fact(num):
    fact=1
    if num == 0:
        return 1

    for i in range(1,num+1):
        fact = fact*i

    return fact
```

In [523]:
```
def factr(n):

    if n == 0:
        return 1

    return n * factr(n-1)
```

In [522]:
```
fact(2)
```

Out[522]: 2

In [ ]:
```
(2)Check number of Factorial or not 720,120,80
Output should be like this -
720 - yes
120 - yes
80 - no
```

In [554]:
```
def fact(num):
    lst=[]
    fact=1
    if num == 0:
        return 1

    for i in range(1,int(num/4)):
        fact = fact*i
        lst.append(fact)
    if(num in lst):
        print("Yes")
    else:
        print("No")
    return
```

In [560]:
```
fact(720)
```

```
Yes
```

In [558]:
```
fact(80)
```

```
No
```

Q5 -What does this mean: *args, **kwargs? And why would we use it?

**args and kargs are both used to supply multiple parameters to a function. Here, args is non-key pair parameters, and kwargs are key-value based params.**

```
Q6 - Write a code in python
P R E P L E A F
P R E P L E A
P R E P L E
P R E P L
P R E P
P R E
```

```
P R
P
```

In [561]:
```
text = "PREPLEAF"
```

In [571]:
```
i = len(text)
text[7]
```

Out[571]: 'F'

In [578]:
```
for j in range(0,8):
    for i in range(0,j+1):
        print(text[i], end="")
    print()
```

```
P
PR
PRE
PREP
PREPL
PREPLE
PREPLEA
PREPLEAF
```

In [595]:
```
for i in range(len(text)):
    print(text[:len(text)-i])
```

```
PREPLEAF
PREPLEA
PREPLE
PREPL
PREP
PRE
PR
P
```