1. Product of two matrices.

In [1]:

```
def matrix_mul(a,b):
    list2=[]
    for i in range(len(a)):
        list2.append(len(b[0])*[0])
    if len(a[0])==len(b):
        for i in range(len(a)):
            for j in range(len(b[0])):
                for k in range(len(b)):
                    list2[i][j]=list2[i][j]+(a[i][k]*b[k][j])
        return list2
    else:
        print("matrix multiplication is not possible")


A   = [[1 ,2],[3 ,4]]
B   = [[1 ,2 ,3 ,4 ,5],[5, 6, 7, 8 ,9]]

matrix_mul(A, B)
```

Out[1]:

```
[[11, 14, 17, 20, 23], [23, 30, 37, 44, 51]]
```

Q2: Proportional Sampling - Select a number randomly with probability proportional

In [15]:

```
from random import uniform
def pick_a_number_from_list(a):
    list1=[]
    list2=[]
    sum_1=sum(a)
    sum2=0
    for i in range(len(a)):
        list1.append(a[i]/sum_1)
    for j in range(len(list1)):
        sum2=sum2+list1[j]
        list2.append(sum2)
    r=uniform(0,1)
    for i in range(len(list2)):
        if r<=list2[i]:
            return a[i]
a=[0 ,5 ,27, 6 ,13 ,28 ,100, 45 ,10 ,79]
def sampling_based_on_magnitued():
    for i in range(0,100):
        number = pick_a_number_from_list(a)
        print(number)
sampling_based_on_magnitued()
6
100
100
79
79
```

100
45
100
100
28
79
27
100
28
79
79
100
10
5
79
28
100
13
27
45
45
45
27
28
100
6
45
79
10
100
100
79
100
100
13
28
79
45
45
79
28
45
100
28
79
79
28
27
79

```
45
45
100
100
100
100
79
79
100
79
45
100
45
79
28
45
100
45
45
100
100
79
28
100
45
79
10
79
79
100
5
13
100
79
45
79
45
45
27
27
27
28
27
27
100
28
```

3 .Replace the digits in the string with "# "

In [18]:

```
def replace_digits(a):
```

```
    list1=[]
    for i in range(len(a)):
        if a[i].isdigit():
            list1.append(a[i])
    if len(list1)==0:
        print("empty string")
    else:
        print(len(list1)*"#")

a="a2b3c4"
replace_digits(a)
###
```

4.Error Function

In [21]:

```
import math as m
def compute_log_loss(list1):
    sum1=0
    for i in range(len(list1)):
        sum1=sum1+(list1[i][0]*m.log10(list1[i][1]))+((1-
list1[i][0])*m.log10(1-list1[i][1]))
    print(sum1/(-len(list1)))

A = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9],
[1, 0.8]]
compute_log_loss(A)
0.42430993457031635
```

Q5: Operations on sentences You will be given two sentances S1, S2 your task is to find

In [22]:

```
def string_features(S1, S2):
    list1=S1.split()
    list2=S2.split()
    l3=[]
    l4=[]
    count=0
    for i in list1:
        for j in range(len(list2)):
            if  list2[j]==i:
                count+=1
    print(count)
    for j in range(len(list2)):
        if list1[j]!=list2[j]:
            l3.append(list1[j])
    print(l3)
    for j in range(len(list1)):
        if list1[j]!=list2[j]:
            l4.append(list2[j])
    print(l4)

S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
```

```
string_features(S1,S2)
7
['first', 'F', '5']
['second', 'S', '3']
```

Q6 Find the probabilities

In [25]:

```
a = [['F1', 'S1'], ['F2', 'S2'], ['F3', 'S3'], ['F1', 'S2'], ['F2', 'S3'],
['F3', 'S2'], ['F2', 'S1'], ['F4', 'S1'], ['F4', 'S3'], ['F5', 'S1']]
def compute_conditional_probabilites(a):
    count1=0
    count2=0
    count3=0
    count4=0
    count5=0
    count6=0
    count7=0
    count8=0
    count9=0
    count11=0
    count12=0
    count13=0
    count14=0
    count15=0
    count16=0
    count17=0
    count18=0
    count19=0
    F_1=[]
    S=[]
    list1=[]
    for i in range(len(a)):
        F_1.append(a[i][0])
        S.append(a[i][1])
        list1.append(a[i][0]+a[i][1])
    for s in range(len(list1)):  #code for taking count of s1,s2,s3
        if S[s]=="S1":
            count1+=1
        elif  S[s]=="S2":
            count2+=1
        else:
            count3+=1
    for i in range(len(list1)):
        if list1[i]=="F1S1":
            count4+=1
        if list1[i]=="F2S2":
            count5+=1
        if list1[i]=="F3S3":
            count6+=1
        if list1[i]=="F1S2":
```

```
            count7+=1
        if list1[i]=="F2S3":
            count8+=1
        if list1[i]=="F3S2":
            count9+=1
        if list1[i]=="F2S1":
            count11+=1
        if list1[i]=="F4S1":
            count12+=1
        if list1[i]=="F4S3":
            count13+=1
        if list1[i]=="F5S1":
            count14+=1
        if list1[i]=="F1S3":
            count15+=1
        if list1[i]=="F3S1":
            count16+=1
        if list1[i]=="F4S2":
            count17+=1
        if list1[i]=="F5S2":
            count18+=1
        if list1[i]=="F5S3":
            count19+=1
    print('Probability of P(F=F1|S==S2) =',(count7/count2))
    print('Probability of P(F=F1|S==S3) =',(count15/count3))
    print('Probability of P(F=F2|S==S1) =',(count11/count1))
    print('Probability of P(F=F2|S==S2) =',(count5/count2))
    print('Probability of P(F=F2|S==S3) =',(count8/count3))
    print('Probability of P(F=F3|S==S1) =',(count16/count1))
    print('Probability of P(F=F3|S==S2) =',(count9/count2))
    print('Probability of P(F=F3|S==S3) =',(count6/count3))
    print('Probability of P(F=F4|S==S1) =',(count12/count1))
    print('Probability of P(F=F4|S==S2) =',(count17/count2))
    print('Probability of P(F=F4|S==S3) =',(count13/count3))
    print('Probability of P(F=F5|S==S1) =',(count14/count1))
    print('Probability of P(F=F5|S==S2) =',(count18/count2))
    print('Probability of P(F=F5|S==S3) =',(count19/count3))
compute_conditional_probabilites(a)
Probability of P(F=F1|S==S2) = 0.3333333333333333
Probability of P(F=F1|S==S3) = 0.0
Probability of P(F=F2|S==S1) = 0.25
Probability of P(F=F2|S==S2) = 0.3333333333333333
Probability of P(F=F2|S==S3) = 0.3333333333333333
Probability of P(F=F3|S==S1) = 0.0
Probability of P(F=F3|S==S2) = 0.3333333333333333
Probability of P(F=F3|S==S3) = 0.3333333333333333
Probability of P(F=F4|S==S1) = 0.25
Probability of P(F=F4|S==S2) = 0.0
Probability of P(F=F4|S==S3) = 0.3333333333333333
```

```
Probability of P(F=F5|S==S1) = 0.25
Probability of P(F=F5|S==S2) = 0.0
Probability of P(F=F5|S==S3) = 0.0
```

Q7: Find Which line separates oranges and apples

In [29]:

```
import re
Blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]
Red= [(1,1),(2,1),(4,2),(2,4),(-1,4)]
red=[]
blue=[]
for line in Lines:
    red_1=[]
    blue_1=[]
    a, b, c = [float(x.strip()) for x in re.split('x|y', line)] #from
stackoverflow
    for i in range(len(Red)):
        d=(a*Red[i][0]) + ((b*Red[i][1])+c)
        red_1.append(d)
    red.append(red_1)

    for i in range(len(Blue)):
        e=((a*Blue[i][0]) + (b*Blue[i][1])+c)
        blue_1.append(e)
    blue.append(blue_1)
list2=[]
list3=[]
for i in range(len(red)):
    list2.append(len(red[0])*[0])
for i in range(len(blue)):
    list3.append(len(blue[0])*[0])
for i in range(len(red)):
    for j in range(len(red[i])):
        if red[i][j]>0:
            list2[i][j]=1
        else:
            list2[i][j]=0
for i in range(len(blue)):
    for j in range(len(blue[i])):
        if blue[i][j]<0:
            list3[i][j]=1
        else:
            list3[i][j]=0
for i in range(len(list2)):
    if list2[i]==list3[i]:
        print("Yes")
    else:
        print("No")
Yes
```

```
No
No
Yes
```

8 Find the closest points

In [44]:

```
import math as m
S = s= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
p= (3,-4)
def closest_points_to_p(S, P):
    point=[]
    for i in range(len(S)):

c=m.acos(((p[0]*S[i][0])+(p[1]*S[i][1]))/(m.sqrt((m.pow(S[i][0],2))+(m.pow(
S[i][1],2)))*m.sqrt((m.pow(p[0],2)+(m.pow(p[1],2))))))
        point.append(c)
        cob=[(S[i],point[i]) for i in range(0,len(point))]
    def sort(x):
        return x[1]
    cob.sort(key=sort)
    for i in range(0,5):
        print(cob[i][0])

closest_points_to_p(S,p)
```

```
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
```

9 Students marks dashboard

In [4]:

```
def display_dash_board(Students, Marks):
    list1=[]
    count=0
    for i in range(len(Marks)):
        list1.append((Students[i],Marks[i]))
    def myfunc(n):
        return n[1]
    list1.sort(key=myfunc,reverse=True)
    for c ,d  in list1:
        print(c , " " , d)
        count+=1
        if count==5:
            break
    top_5_students = count
    print("-----------------------------------------")
    list1.sort(key=myfunc,reverse=False)
    count1=0
    for c ,d  in list1:
        print(c , " " , d)
        count1+=1
```

```
        if count1==5:
            break
    least_5_students =count1
    print("-----------------------------------------")
    max1 = max(Marks)
    min1 = min(Marks)
    diff = max1 - min1
    pre_25 = diff*0.25
    pre_75 = diff*0.75
    for a,b in list1:
        if (b>pre_25) and (b<pre_75):
            print(a,b)

    return top_5_students, least_5_students
Students=['student1','student2','student3','student4','student5','student6'
,'student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]
a= display_dash_board(Students, Marks)
student8   98
student10   80
student2   78
student5   48
student7   47
-----------------------------------------
student3   12
student4   14
student9   22
student6   43
student1   45
-----------------------------------------
student9 22
student6 43
student1 45
student7 47
student5 48
```

10 Filling the missing values in the specified format

In [41]:

```
def curve_smoothing(a):
    b=a.split(",")
    list1=[]
    list2=[]
    count=0
    for i in range(len(b)):
        if b[i]=='_' and b[len(b)-1]!="_":
            list1.append(0)
        elif b[i]=='_' and b[len(b)-1]=="_":
            list1.append(0)
        else:
            list1.append(int(b[i]))
```

```
    if list1[0]==0 and list1[len(list1)-1]==0:
        first=list1[0]
        start=1
        for i in range(start,len(list1)):
            if list1[i]!=0:
                k=i
                for j in range(start-1,k+1) :
                    list1[j]=(int(first)+int(list1[i]))/int((k-start+2))
                start=k+1
                first=list1[k]
        if list1[-1]==0:
            k=int(len(list1))//2
            for i in range(k+2,len(list1)):
                list1[i]=((list1[k]+list1[-1])/k+1)
        print(list1)

    else:
        for j in range(len(list1)):
            if list1[0]==0 and list1[len(list1)-1]!=0:
                list2.append((list1[0]+list1[len(list1)-1])/len(list1))
            elif list1[0]!=0 and list1[len(list1)-1]!=0:
                list2.append((list1[0]+list1[len(list1)-1])/len(list1))
            elif list1[0]!=0 and list1[len(list1)-1]==0:
                list2.append((list1[0]+list1[len(list1)-1])/len(list1))
        print(list2)


a="_,_,30,_,_,_,50,_,_"
b="40,_,_,_,60"
c="80,_,_,_,_"
curve_smoothing(a)
curve_smoothing(b)
curve_smoothing(c)
[10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.0, 4.0, 4.0]
[20.0, 20.0, 20.0, 20.0, 20.0]
[16.0, 16.0, 16.0, 16.0, 16.0]
```

In [ ]: