1. Product of two matrices.

In [24]:
```
def matrix_mul(a,b):
    row1=len(a)
    col1=len(a[0])
    row2=len(b)
    col2=len(b[0])
    if col1==row2:
        result = [[0 for x in range(col2)] for y in range(row1)]
        for i in range (row1):
             for j in range (col2):
                     for k in range(row2):
                             result[i][j] += a[i][k] * b[k][j]
        return result
    else:
         print("matrix multiplication is notr possible")

a  = [[1 ,2],
      [3 ,4]]
b   = [[1 ,2, 3, 4, 5],
      [5 ,6 ,7, 8, 9]]

matrix_mul(a,b)
```

Out[24]:
```
[[11, 14, 17, 20, 23], [23, 30, 37, 44, 51]]
```

2. Proportional sampling

In [25]:
```
from random import uniform
a=[2.0,6.0,1.2,5.8,20.0]
def pick_a_number_from_list(a):
    p=uniform(0,1)
```

```
    s=sum(a)
    b=[]
    for i in range(len(a)):
        b.append(a[i]/s)
    c=[]
    for i in range(0,len(a)):
        c.append(sum(b[0:(i+1)]))
    for j in range(0,len(a)):
        if p<=c[j]:
            number=a[j]
            return number

def sampling_based_on_magnitued():
    for i in range(0,100):
        number = pick_a_number_from_list(a)
        print(number)

sampling_based_on_magnitued()
```

```
20.0
6.0
20.0
20.0
6.0
20.0
6.0
6.0
20.0
20.0
6.0
20.0
20.0
2.0
20.0
5.8
6.0
5.8
20.0
20.0
20.0
```

```
20.0
20.0
20.0
20.0
20.0
20.0
5.8
20.0
20.0
5.8
5.8
6.0
2.0
6.0
6.0
20.0
20.0
5.8
20.0
20.0
6.0
20.0
20.0
6.0
6.0
6.0
20.0
5.8
20.0
20.0
20.0
5.8
20.0
1.2
20.0
20.0
20.0
6.0
20.0
```

```
5.8
20.0
20.0
20.0
20.0
20.0
20.0
20.0
20.0
20.0
6.0
20.0
6.0
5.8
20.0
6.0
6.0
20.0
5.8
20.0
6.0
5.8
1.2
5.8
6.0
5.8
5.8
6.0
1.2
5.8
20.0
20.0
20.0
20.0
20.0
20.0
20.0
1.2
```

```
20.0
20.0
```

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

In [30]:
```
def replace_digits(input):
    count=0
    for i in range(len(input)):
        if input[i].isdigit():
            count=count+1

    if count==0:
        return("empty string")
    else:
        return(count*"#")

replace_digits("#2a$#b%c%561#")
```

Out[30]:
```
'####'
```

4 . Students marks dashboard

In [3]:
```
def display_dash_board(students, marks):
    marklist=[(students[i],marks[i]) for i in range(0,len(marks))]
    print("-------a-------")
    print("\n")
    marklist.sort(key=lambda x:x[1],reverse=True)
    count=0
    for a,b in marklist:
        print(a,b)
        count=count+1
        if count==5:
            break
    print("\n")
    print("-------b-------")
```

```
    print("\n")
    count=0
    marklist.sort(key=lambda x:x[1],reverse=False)
    for a,b in marklist:
        print(a,b)
        count=count+1
        if count==5:
            break
    print("\n")
    print("-------c-------")
    print("\n")
    for a,b in marklist:
        if (b>25) and (b<75):
            print(a,b)
```

In [4]:
```
Students=['student1','student2','student3','student4','student5','stude
nt6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
```

In [5]:
```
display_dash_board(Students,Marks)
```

```
-------a-------


student8 98
student10 80
student2 78
student5 48
student7 47


-------b-------


student3 12
student4 14
student9 35
student6 43
```

```
student1 45


-------c-------


student9 35
student6 43
student1 45
student7 47
student5 48
```

5. Find the closest points

In [33]:
```
import math as m
def closest_points_to_p(s,p):
    point=[]
    for i in s:
        c=m.acos(((p[0]*i[0])+(p[1]*i[1]))/(m.sqrt((m.pow(i[0],2))+(m.p
ow(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))]
    cob.sort(key=lambda x:x[1])
    result=[]
    for i in range(0,5):
        result.append(cob[i][0])
    return result

s= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
p= (3,-4)
closest_points_to_p(s,p)
```

Out[33]:
```
[(6, -7), (1, -1), (6, 0), (-5, -8), (-1, -1)]
```

6.Find line which seperate given set of points

In [28]:
```
import re
def i_am_the_one(red,blue,lines):

    for i in lines:
        b=[]
        r=[]
        z = re.findall(r'[\d\.\-\+]+', i)
        for a in blue:
            c=(float(z[0])*a[0])+(float(z[1])*a[1])+float(z[2])
            if c>0:
                b.append(1)
            else:
                b.append(0)
        lr=len(set(b))
        for a in red:
            c=(float(z[0])*a[0])+(float(z[1])*a[1])+float(z[2])
            if c>0:
                r.append(1)
            else:
                r.append(0)
        lb=len(set(r))

        if(lr==1 and lb==1):
            print("yes")
        else:
            print("no")

red = [(1,1),(2,1),(4,2),(2,4), (-1,4)]
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"]
i_am_the_one(red,blue,lines)
```

```
yes
no
no
yes
```

7. Filling the missing values in the specified format

In [34]:
```
def blanks(s):
    a=s.split(",")
    l=len(a)
    if a[0]=="_":
        a[0]="0"      # converting 3 cases into single case
    if a[l-1]=="_":
        a[l-1]="0"
    first=a[0]
    start=1
    for i in range(start,l):
        if a[i]!="_":
            k=i
            for j in range(start-1,k+1) :
                a[j]=(int(first)+int(a[i]))/int((k-start+2))
            start=k+1
            first=a[k]
    return(a)

blanks( "_,_,_,24")
```

Out[34]:
```
[6.0, 6.0, 6.0, 6.0]
```

In [146]:
```
blanks("40,_,_,_,60")
```

Out[146]:
```
[20.0, 20.0, 20.0, 20.0, 20.0]
```

In [147]:
```
blanks("80,_,_,_,_")
```

Out[147]:
```
[16.0, 16.0, 16.0, 16.0, 16.0]
```

In [148]:
```
blanks("_,_,30,_,_,_,50,_,_")
```

Out[148]:
```
[10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.0, 4.0, 4.0]
```

8. Conditional probability

In [35]:
```
def compute_conditional_probabilites(a):
    f=[]
    s=[]
    m=[]
    for i in a:
        f.append(i[0])
        s.append((i[1]))
    s=set(s)
    s=list(s)
    for k in f:
        for j in range (0,len(s)):
            m.append(k+s[j])
        mydict1 = {k:0 for k in m}
    for k in s:
        mydict2={k:0 for k in s}

    for i in range(len(a)):
        k = a[i][0] + a[i][1]    # I REFFERED THESE BLOCK OF STATEMENT
 FROM STACKOVERFLOW
        mydict1[k] += 1          #https://stackoverflow.com/questions/5
7160252/find-conditional-probabilities-using-python
        mydict2[a[i][1]] += 1


    for i in f:
        print(" _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _")
        print("\n")
        for j in s:
            print("probability of p(F=%s|S=%s) = "%(i,j),(mydict1[str(i
+j)]/mydict2[j]))

a = [['F1', 'S1'], ['F2', 'S2'], ['F3', 'S3'], ['F1', 'S2'], ['F2', 'S
3'], ['F3', 'S2'], ['F2', 'S1'], ['F4', 'S1'], ['F4', 'S3'], ['F5', 'S
1']]
compute_conditional_probabilites(a)
```

```
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

```
probability of p(F=F1|S=S2) =  0.3333333333333333
probability of p(F=F1|S=S1) =  0.25
probability of p(F=F1|S=S3) =  0.0
 – – – – – – – – – – – – – – – – –


probability of p(F=F2|S=S2) =  0.3333333333333333
probability of p(F=F2|S=S1) =  0.25
probability of p(F=F2|S=S3) =  0.3333333333333333
 – – – – – – – – – – – – – – – – –


probability of p(F=F3|S=S2) =  0.3333333333333333
probability of p(F=F3|S=S1) =  0.0
probability of p(F=F3|S=S3) =  0.3333333333333333
 – – – – – – – – – – – – – – – – –


probability of p(F=F1|S=S2) =  0.3333333333333333
probability of p(F=F1|S=S1) =  0.25
probability of p(F=F1|S=S3) =  0.0
 – – – – – – – – – – – – – – – – –


probability of p(F=F2|S=S2) =  0.3333333333333333
probability of p(F=F2|S=S1) =  0.25
probability of p(F=F2|S=S3) =  0.3333333333333333
 – – – – – – – – – – – – – – – – –


probability of p(F=F3|S=S2) =  0.3333333333333333
probability of p(F=F3|S=S1) =  0.0
probability of p(F=F3|S=S3) =  0.3333333333333333
 – – – – – – – – – – – – – – – – –


probability of p(F=F2|S=S2) =  0.3333333333333333
```

```
probability of p(F=F2|S=S1) =  0.25
probability of p(F=F2|S=S3) =  0.3333333333333333

 - - - - - - - - - - - - - - - - - -


probability of p(F=F4|S=S2) =  0.0
probability of p(F=F4|S=S1) =  0.25
probability of p(F=F4|S=S3) =  0.3333333333333333
 - - - - - - - - - - - - - - - - - -


probability of p(F=F4|S=S2) =  0.0
probability of p(F=F4|S=S1) =  0.25
probability of p(F=F4|S=S3) =  0.3333333333333333
 - - - - - - - - - - - - - - - - - -


probability of p(F=F5|S=S2) =  0.0
probability of p(F=F5|S=S1) =  0.25
probability of p(F=F5|S=S3) =  0.0
```

9 .Operations on sentences

In [36]:

```python
def string_features(s1, s2):
    set1=set(s1.split())
    set2=set(s2.split())
    a=len(set1&set2)
    b=list(set1-set2)
    c=list(set2-set1)
    return a,b,c

s1= "the first column F will contain only 5 uniques values"
s2= "the second column S will contain only 3 uniques values"
a,b,c=string_features(s1, s2)
print(a)
```

```
print(b)
print(c)
```

```
7
['F', 'first', '5']
['S', 'second', '3']
```

10 . Error function

In [38]:
```
import math as m
def compute_log_loss(a):
    sum=0
    for s in a:
        sum=sum+(s[0]*m.log10(s[1])+((1-s[0])*m.log10(1-s[1])))
    return (sum/-len(a))

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)
```

Out[38]:
```
0.42430993457031635
```