Python: without numpy or sklearn

Q1: Given two matrices please print the product of those two matrices

```
Ex 1: A   = [[1 3 4]
             [2 5 7]
             [5 9 6]]
      B   = [[1 0 0]
             [0 1 0]
             [0 0 1]]
      A*B = [[1 3 4]
             [2 5 7]
             [5 9 6]]


Ex 2: A   = [[1 2]
             [3 4]]
      B   = [[1 2 3 4 5]
             [5 6 7 8 9]]
      A*B = [[11 14 17 20 23]
             [23 30 36 42 51]]

Ex 3: A   = [[1 2]
             [3 4]]
      B   = [[1 4]
             [5 6]
             [7 8]
             [9 6]]
      A*B =Not possible
```

In [13]:
```
# function takes A and B matrix as inputs and returns multiplication
def matrix_mul(A, B):
    cola = len(A[0])
    rowb = len(B)

    if cola != rowb:
      return "Not possible"

    rows = len(A)
    cols = len(B[0])
    result = [[0] * cols for _ in range(rows)]

    for i in range(rows):
      for j in range(cols):
        # loop in rows of B
        for k in range(len(B)):
          result[i][j] = result[i][j] + (A[i][k] * B[k][j])

    return(result)

A   = [[1, 2],
       [3, 4]]
B   = [[1, 2, 3, 4, 5],
       [5, 6, 7, 8, 9]]
C = matrix_mul(A, B)
print ("A*B = [", end =" ")
for r in range(len(C)):
  print (C[r])
print ("]")
```

```
A*B = [ [11, 14, 17, 20, 23]
[23, 30, 37, 44, 51]
]
```

Q2: Select a number randomly with probability proportional to its magnitude from the given array of n elements

consider an experiment, selecting an element from the list A randomly with probability proportional to its magnitude. assume we are doing the same experiment for 100 times with replacement, in each experiment you will print a number that is selected randomly from A.

```
Ex 1: A = [0 5 27 6 13 28 100 45 10 79]
let f(x) denote the number of times x getting selected in 100 experiments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f
(0)
```

In [1]:
```
import random

# function takes a list A and print random number based on its magnitu
de probability
def sampling_based_on_magnitued(A):
  A = sorted(A)
  sum = 0
  B = [0]*len(A)

  for i in range(len(A)):
    sum += A[i]
    B[i] = sum

  num = random.randint(0,sum)
  for i in range(len(A)):
    if B[i] >= num:
      return (A[i])


A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
for i in range(100):
  print (sampling_based_on_magnitued(A))
```

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

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

Q3: Replace the digits in the string with

consider a string that will have digits in that, we need to remove all the not digits and replace the digits with #

```
Ex 1: A = 234                Output: ###
Ex 2: A = a2b3c4             Output: ###
Ex 3: A = abc                Output:   (empty string)
Ex 5: A = #2a$#b%c%561#      Output: ####
```

In [15]:
```
import re

# function takes String as input and return string afetr removing non
digits and replace digits with #
def replace_digits(String):
    s = re.sub("[^0-9]", "", String)
    return(re.sub("\d","#",s))

String = "#2a$#b%c%561#"
print ("Output: ", replace_digits(String))
```

```
Output:  ####
```

Q4: Students marks dashboard

consider the marks list of class students given two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]
from the above two lists the Student[0] got Marks[0], Student[1] got Marks[1] and so on

your task is to print the name of students **a. Who got top 5 ranks, in the descending order of marks**
b. Who got least 5 ranks, in the increasing order of marks
d. Who got marks between >25th percentile <75th percentile, in the increasing order of marks

```
Ex 1:
Students=['student1','student2','student3','student4','student5','student6
','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
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
```

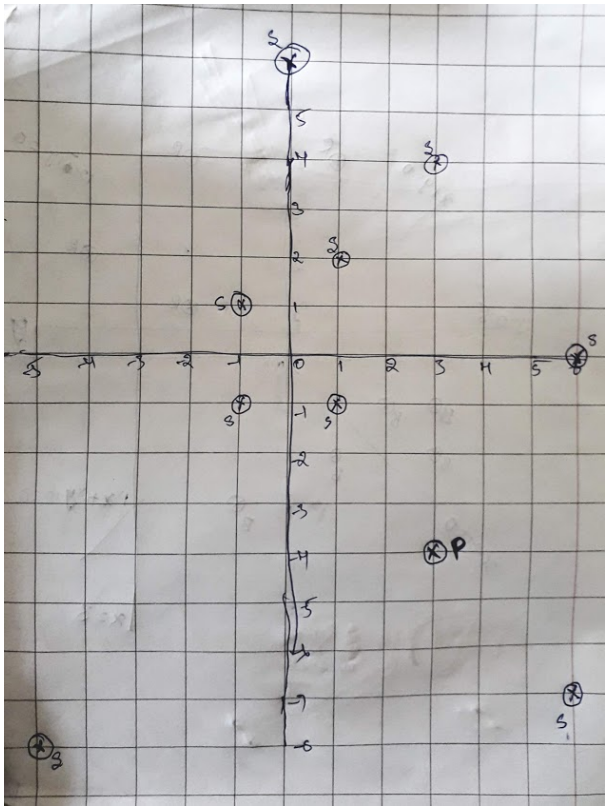
In [27]:
```
def display_dash_board(students, marks):
    res = list(zip(students, marks))
    res = sorted(res, key = lambda x: x[1], reverse=True)

    # write code for computing top top 5 students
    top_5_students = res[:5]
    max = top_5_students[0][1]

    # write code for computing top least 5 students
    total_element = len(students)
    least_5_students = res[:total_element-6:-1]
    min = least_5_students[0][1]

    # write code for computing students between >25th <75th percentile
    end = int(total_element - (.25 * total_element))
    start = int(total_element - (.75 * total_element))

    res = sorted(res, key = lambda x: x[1])
    students_within_25_and_75 = list()
    for i in range(start, end):
      students_within_25_and_75.append(res[i])

    return top_5_students, least_5_students, students_within_25_and_75

students=['student1','student2','student3','student4','student5','stud
ent6','student7','student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
top_5_students, least_5_students, students_within_25_and_75 = display_
dash_board(students, marks)

print ("Top 5 students")
for x in top_5_students:
  print (x[0],"  ", x[1])

print ("\nLeast 5 students")
for x in least_5_students:
  print (x[0],"  ", x[1])

print ("\nStudents between 25-75 percentile")
for x in students_within_25_and_75:
  print (x[0],"  ", x[1])
```

```
Top 5 students
student8    98
student10    80
student2    78
student5    48
student7    47

Least 5 students
student3    12
student4    14
student9    35
student6    43
student1    45

Students between 25-75 percentile
student9    35
student6    43
student1    45
student7    47
student5    48
```

Q5: Find the closest points

consider you have given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q)
your task is to find 5 closest points(based on cosine distance) in S from P
cosine distance between two points (x,y) and (p,q) is defind as $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

```
Ex:

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1)(6,0),(1,-1)]
P= (3,-4)
```




```
Output:
(6,-7)
(1,-1)
(6,0)
```

In [78]:
```
import math

# function takes S-list of points and tell the 5 nearest points from P
def cosine_similarity(S, P):
    num = (S[0] * P[0]) + (S[1] * P[1])
    den = (math.sqrt(S[0]**2 + S[1]**2)) * (math.sqrt(P[0]** 2 + P[1]
**2))
    cos = math.acos(num/den)
    return (cos)

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)
cos_dist = []

for i in range(len(S)):
  cos_dist.append((S[i], cosine_similarity(S[i],P)))

cos_dist1 = sorted(cos_dist, key = lambda x: float(x[1]))

for r in range(5):
  print (cos_dist1[r][0])
```

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

Q6: Find Which line separates oranges and apples

consider you have given two set of data points in the form of list of tuples like

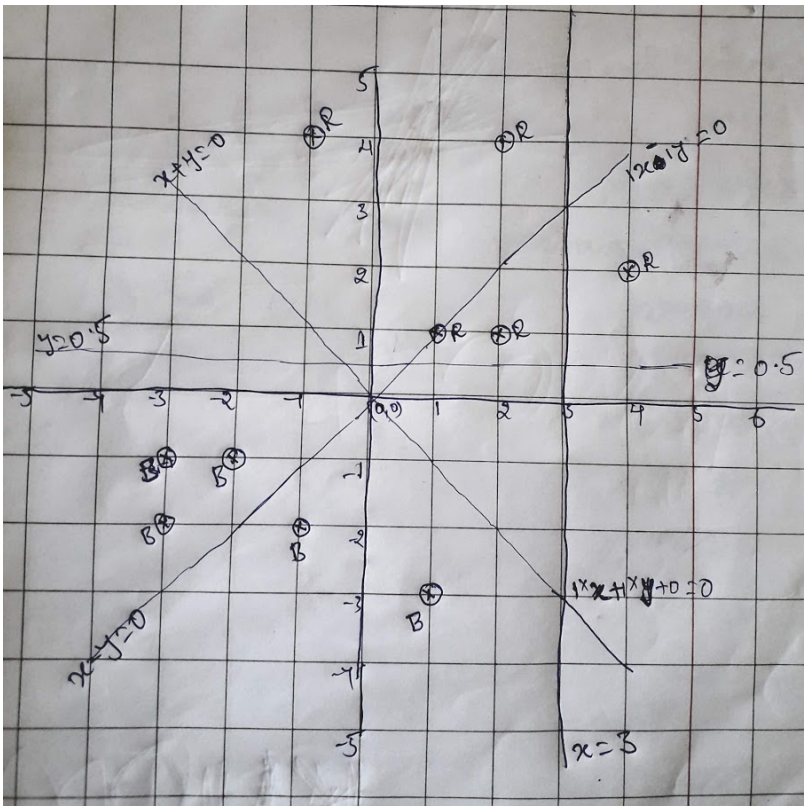
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

and set of line equations(in the string formate, i.e list of strings)

```
Lines = [a1x+b1y+c1,a2x+b2y+c2,a3x+b3y+c3,a4x+b4y+c4,..,K lines]
Note: you need to string parsing here and get the coefficients of x,y and
intercept
```

your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

```
Ex:
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"]
```



In [115]:
```
import math

# functions takes red and blue points and return Yes is line seperates
red and blue points
def i_am_the_one(red,blue,line):

  for i in range(len(red)):

    fxy_red = line[0] * float(red[i][0]) + line[1] * float(red[i][1])
+ line[2]
    fxy_blue = line[0] * float(blue[i][0]) + line[1] * float(blue
[i][1]) + line[2]
    if i == 0:
      if fxy_red == 0.0:
        sign_red = 1.0
      if fxy_blue == 0.0:
        sign_blue = 1.0
      sign_red = fxy_red
      sign_blue = fxy_blue
    else:
      if sign_red * fxy_red  < 0:
        return 'NO'
      if sign_blue * fxy_blue <0:
        return 'NO'
  return 'YES'

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"]

for line in Lines:
  a, b, c = [float(coef.strip()) for coef in re.split('x|y', line)]
  yes_or_no = i_am_the_one(Red, Blue, [a, b, c])
  print (yes_or_no)
```

```
YES
NO
NO
YES
```

Q7: Filling the missing values in the specified formate

You will be given a string with digits and '_'(missing value) symbols you have to replace the '_' symbols as explained

```
Ex 1: _, _, _, 24 ==> 24/4, 24/4, 24/4, 24/4 i.e we. have distributed the
24 equally to all 4 places

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/
5 ==> 20, 20, 20, 20, 20 i.e. the sum of (60+40) is distributed qually to
all 5 places

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16
i.e. the 80 is distributed qually to all 5 missing values that are right t
o it

Ex 4: _, _, 30, _, _, _, 50, _, _
==> we will fill the missing values from left to right
    a. first we will distribute the 30 to left two missing values (10, 10,
10, _, _, _, 50, _, _)
    b. now distribute the sum (10+50) missing values in between (10, 10, 1
2, 12, 12, 12, 12, _, _)
    c. now we will distribute 12 to right side missing values (10, 10, 12,
12, 12, 12, 4, 4, 4)
```

for a given string with comma seprate values, which will have both missing values numbers like ex: "_, _, x, _, _, _" you need fill the missing values Q: your program reads a string like ex: "_, _, x, _, _, _" and returns the filled sequence Ex:

```
Input1: "_,_,_,24"
Output1: 6,6,6,6

Input2: "40,_,_,_,60"
Output2: 20,20,20,20,20

Input3: "80,_,_,_,_"
Output3: 16,16,16,16,16

Input4: "_,_,30,_,_,_,50,_,_"
Output4: 10,10,12,12,12,12,4,4,4
```

In [122]:
```
# function takes a string with _ and retuen string with filled blankes
def curve_smoothing(string):
  list1 = list(string.split(','))
  n = len(list1)
  start = end = 0
  end += 1
  while start < n and end < n:

    while start < n  and list1[start] != '_' and  list1[start+1] != '_
':
      start += 1
    while end < n and list1[end] == '_' :
      end += 1
    if start == 0 and list1[start] == '_':
      list1[start] = 0
    if end == n:
      end = end -1
      list1[end] = 0
    if end == n-1 and list1[end] == '_':
      list1[end] = 0

    element = (int(list1[start]) + int(list1[end])) // ((end-start)+1)

    for i in range(start, end+1):
      list1[i] = element

    start = end
    end += 1

  return list1

S =  "_,_,30,_,_,_,50,_,_"
#S = "80,_,_,_,_"
#S = "40,_,_,_,60"
#S = "_,_,_,24"
smoothed_values= curve_smoothing(S)
print(*smoothed_values, sep=", ")
```

```
10, 10, 12, 12, 12, 12, 4, 4, 4
```

Q8: Filling the missing values in the specified formate

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns

1. the first column F will contain only 5 uniques values (F1, F2, F3, F4, F5)
2. the second column S will contain only 3 uniques values (S1, S2, S3)

```
your task is to find
a. Probability of P(F=F1|S==S1), P(F=F1|S==S2), P(F=F1|S==S3)
b. Probability of P(F=F2|S==S1), P(F=F2|S==S2), P(F=F2|S==S3)
c. Probability of P(F=F3|S==S1), P(F=F3|S==S2), P(F=F3|S==S3)
d. Probability of P(F=F4|S==S1), P(F=F4|S==S2), P(F=F4|S==S3)
e. Probability of P(F=F5|S==S1), P(F=F5|S==S2), P(F=F5|S==S3)
```

Ex:

```
[[F1,S1],[F2,S2],[F3,S3],[F1,S2],[F2,S3],[F3,S2],[F2,S1],[F4,S1],[F4,S
3],[F5,S1]]

a. P(F=F1|S==S1)=1/4, P(F=F1|S==S2)=1/3, P(F=F1|S==S3)=0/3
b. P(F=F2|S==S1)=1/4, P(F=F2|S==S2)=1/3, P(F=F2|S==S3)=1/3
c. P(F=F3|S==S1)=0/4, P(F=F3|S==S2)=1/3, P(F=F3|S==S3)=1/3
d. P(F=F4|S==S1)=1/4, P(F=F4|S==S2)=0/3, P(F=F4|S==S3)=1/3
e. P(F=F5|S==S1)=1/4, P(F=F5|S==S2)=0/3, P(F=F5|S==S3)=0/3
```

In [29]:
```
# Function takes A, f, s as input and print conditional probability of
elements of f set given elements in s set
def compute_conditional_probabilites(A, f, s):

  scount = 0
  fcount = 0

  for ele in A:
    if ele[1] == s:
      scount += 1
    if ele[0] == f and ele[1] == s:
      fcount += 1
  return ("P(F={0}|S={1})={2}".format(f, s, str(fcount)+'/'+str(scoun
t)))

def compute_sets(A):
  f_set = set() # to store F values
  s_set = set() # to store S values

  for element in A:
    f_set.add(element[0])
    s_set.add(element[1])

  f_set = sorted(f_set)
  s_set = sorted(s_set)
  for f in f_set:
    res = []
    for s in s_set:
      res.append(compute_conditional_probabilites(A, f, s))
    print(", ".join(res))

A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3
','S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]
compute_sets(A)
```

```
P(F=F1|S=S1)=1/4, P(F=F1|S=S2)=1/3, P(F=F1|S=S3)=0/3
P(F=F2|S=S1)=1/4, P(F=F2|S=S2)=1/3, P(F=F2|S=S3)=1/3
P(F=F3|S=S1)=0/4, P(F=F3|S=S2)=1/3, P(F=F3|S=S3)=1/3
P(F=F4|S=S1)=1/4, P(F=F4|S=S2)=0/3, P(F=F4|S=S3)=1/3
P(F=F5|S=S1)=1/4, P(F=F5|S=S2)=0/3, P(F=F5|S=S3)=0/3
```

Q9: Given two sentances S1, S2

You will be given two sentances S1, S2 your task is to find

```
a. Number of common words between S1, S2
b. Words in S1 but not in S2
c. Words in S2 but not in S1
```

Ex:

```
S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
Output:
a. 7
b. ['first','F','5']
c. ['second','S','3']
```

In [155]:
```
def string_features(S1, S2):
  s1 = set(S1.split(" "))
  s2 = set(S2.split(" "))

  a = len(s1 & s2)
  b = s1 - s2
  c = s2 - s1
  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.", a,"\nb.", b,"\nc.", c)
```

```
a. 7
b. {'F', '5', 'first'}
c. {'S', 'second', '3'}
```

Q10: Given two sentances S1, S2

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns

a. the first column Y will contain interger values
b. the second column Y_{score} will be having float values
Your task is to find the value of
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ here n is the number of rows in the matrix

```
Ex:
[[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1,
0.8]]
output:
0.4243099
```

$\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + \ldots + (1 \cdot log_{10}(0.8) + 0 \cdot log_{10}$

In [282]:
```
import math

# fnction takes list of lists A and return Loss function value
def compute_log_loss(A):
  l = 0
  for elements in A:
    l += elements[0] * (math.log10(elements[1]))
    l += (1-elements[0]) * (math.log10(1-elements[1]))
  return ((-1/len(A)) * l)

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]]
loss = compute_log_loss(A)
print(loss)
```

```
0.42430993457031635
```