```
def skalercarpim(a,b):
    t=0
    for i in range(len(a)):
        t+=a[i]*b[i]
    return t

k=[4,6,2]
l=[9,5,3]
print(skalercarpim(k,l))
a=[8,6,2]
b=[1,2,3]
print(skalercarpim(a,b))
```

==========

```
from copy import *
def topmat(a,b):
    #a ve b nin satir ve sutun sayilarinin ayni oldugunu varsayalim
    c=deepcopy(a)
    for i in range(len(a)):
        for j in range(len(a[i])):
            c[i][j]+=b[i][j]
    return c

def topmat(a,b):
    #a ve b nin satir ve sutun sayilarinin ayni oldugunu varsayalim
    #Sifirlardan olusan bir c matrisi kuralim

    c=[[0 for i in range(len(a[j]))] for j in range(len(a))]

    for i in range(len(a)):
        for j in range(len(a[i])):
            c[i][j]+=a[i][j]+b[i][j]
    return c

k=[[3,2,1],[4,7,9]] # iki satir uc sutun
m=[[-1,2,5],[9,2,1]] # iki satir uc sutun

print(topmat(k,m))
```

==========

deepcopy kullanmadan bir a matrisinin kopyasini oluşturalım.

```
a=[[2,3],[5,1,6],[1,4]]
# a matrisinin bir kopyasini olusturalim
b=[[a[i][j] for j in range(len(a[i]))] for i in range(len(a))]
```

Bir matrisin kopyasının oluşturulması:

```
#Shallow copy works for the following
mat=[3,4,5,7,3]
print("orijinal mat:",mat)
k=mat[:]
k[3]=0
print("k degisikligi sonrasi mat:",mat)


#Shallow copy does not work for the following
mat=[[2,3,4],[3,4]]
print("orijinal mat:",mat)
k=mat[:]
k[1][0]=4
print("k degisikligi sonrasi mat:",mat)

#Deep copy works always
from copy import *
mat=[[2,3,4],[3,4]]
print("orijinal mat:",mat)
k=deepcopy(mat)
k[1][0]=4
print("k degisikligi sonrasi mat:",mat)
```

==========

```
def determinant(mat):
    if len(mat)>2:
        print("matris boyutu buyuk")
    else:
        a=mat[0][0]
        b=mat[0][1]
        c=mat[1][0]
        d=mat[1][1]
        return a*d-b*c
    return False
mat=[[1,2],[3,4]]
print(determinant(mat))
```

==========

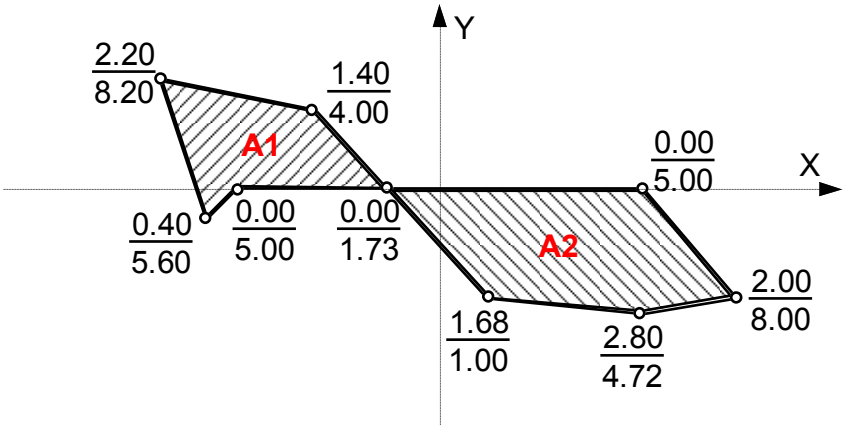
Koordinatlara dayalı olan bu hesap yönteminin dayandığı formül verilmiştir. Tablo ile çözümün nasıl yapıldığı gösterilmiştir. Buna göre, x ve y dizilerini değişken olarak kullanan "cross" adında bir fonksiyon yazalım. Fonksiyona istenilen sayıda ve değerde x ve y koordinatları verilebilmeli ve buna göre alanı hesaplayıp döndürebilmelidir. Örnek olarak,

```
print("A1 alanı:",cross([-4,-8.2,-5.6,-5,-1.73],[1.4,2.2,-0.4,0,0]))
```

şeklinde kullanılabilmelidir.

$$A = \frac{1}{2}\sum_{i=1}^{n} X_i\left(Y_{i+1} - Y_{i-1}\right)$$

Burada; **A:** Enkesit alanı , **X_i,Y_i :** Koordinatlar , **n:**Enkestteki nokta sayısı



A_1 Alanı için koordinatlar:

p_0=(-4.00,1.40), p_1=(-8.20,2.20), p_2=(-5.60,-0.40), p_3=(-5.00,0.00), p_4=(-1.73,0.00)

A_1 alanı hesabı:

i	x_i	y_i	$y_{i+1}-y_{i-1}$	$x_i(y_{i+1}-y_{i-1})$
4		0.00		
0	-4.00	1.40	2.20	-8.800
1	-8.20	2.20	-1.80	14.760
2	-5.60	-0.40	-2.20	12.320
3	-5.00	0.00	0.40	-2.000
4	-1.73	0.00	1.40	-2.422
0		1.40		
			Toplam	13,858

A_1=13.858/2=6.93m^2

A_2 Alanı hesabı için nokta koordinatları:

p_0=(1.00,-1.68) , p_1=(4.72,-2.80), p_2=(8.00,-2.00) ,p_3=(5.00,0.00), p_4=(-1.73,0.00)

A_2 alanı hesabı:

i	x_i	y_i	$y_{i+1}-y_{i-1}$	$x_i(y_{i+1}-y_{i-1})$
4		0.00		
0	1.00	-1.68	-2.80	-2.8000
1	4.72	-2.80	-0.32	-1.5104
2	8.00	-2.00	2.80	22.4000
3	5.00	0.00	2.00	10.0000
4	-1.73	0.00	-1.68	2.9064
0		-1.68		
			Toplam	30.9960

A_2=30.996/2=15.498m^2

```
def cross(x,y):
    t=0
    n=len(x) # n: Nokta sayısı
    for i in range(n):
        if i==n-1:
            ys=y[0]
        else:
            ys=y[i+1]
        t+=x[i]*(ys-y[i-1])
    return t/2
print("A1 alanı:",cross([-4,-8.2,-5.6,-5,-1.73],[1.4,2.2,-0.4,0,0]))
print("A2 alanı:",cross([1,4.72,8,5,-1.73],[-1.68,-2.8,-2,0,0]))
```

Liste oluşturma uygulamaları:

```
>>> [i for i in range(0,30,2)]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
>>> [0 for i in range(5)]
[0, 0, 0, 0, 0]
>>> [[i for i in range(3)] for j in range(5)]
[[0, 1, 2], [0, 1, 2], [0, 1, 2], [0, 1, 2], [0, 1, 2]]
>>> [[i+j for i in range(3)] for j in range(5)]
[[0, 1, 2], [1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6]]
>>> [[j for i in range(3)] for j in range(5)]
[[0, 0, 0], [1, 1, 1], [2, 2, 2], [3, 3, 3], [4, 4, 4]]
```

İki matrisin çarpımı:

```
   def çarp(x,y):
      # x matrisinin satır sayısı
      xsat=len(x)
      # x matrisinin sütun sayısı
      xsüt=len(x[0])
      ysat=len(y)
      ysüt=len(y[0])
      if xsüt!=ysat:
         print('Çarpım yapılamaz !')
         return None
      # z: çarpımın sonucu olan matris
      zsat=xsat
      zsüt=ysüt
      # z matrisini 0 lardan oluşturalım.
      z=[[0 for i in range(zsüt)] for j in range(zsat)]
      # Çarpma işlemini yapalım.
      for i in range(zsat):
         for k in range(zsüt):
             for j in range(ysat):
                 z[i][k]+=x[i][j]*y[j][k]
      return z
```