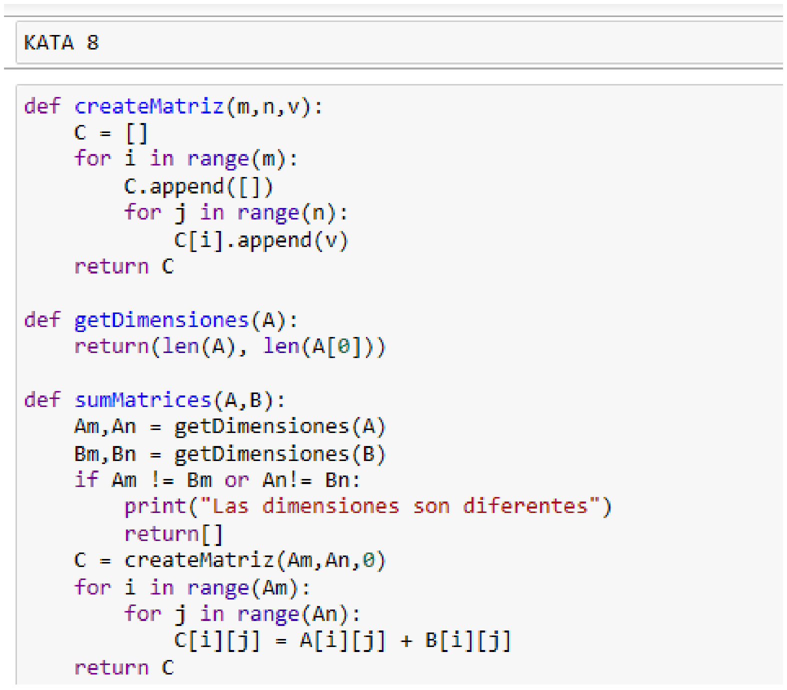
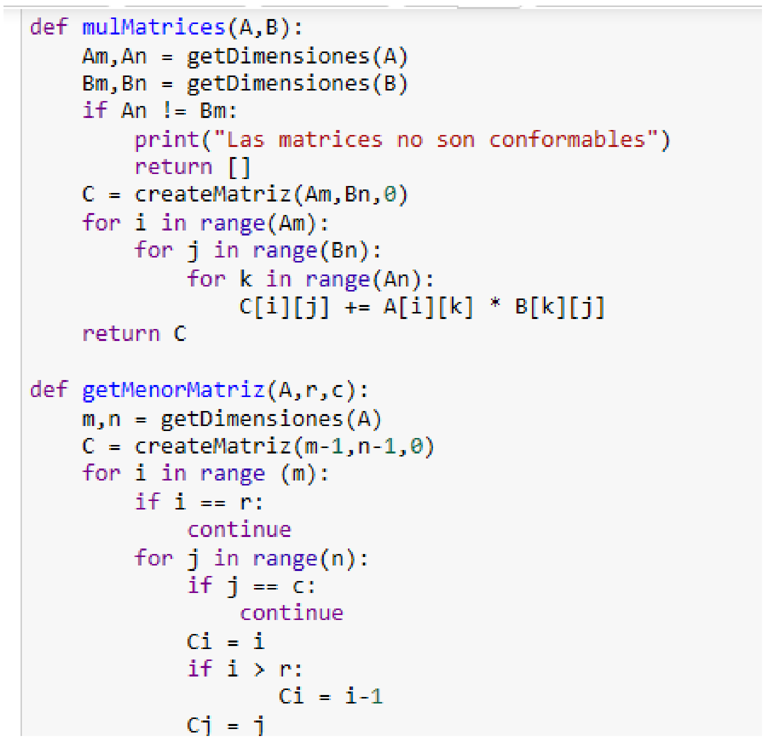
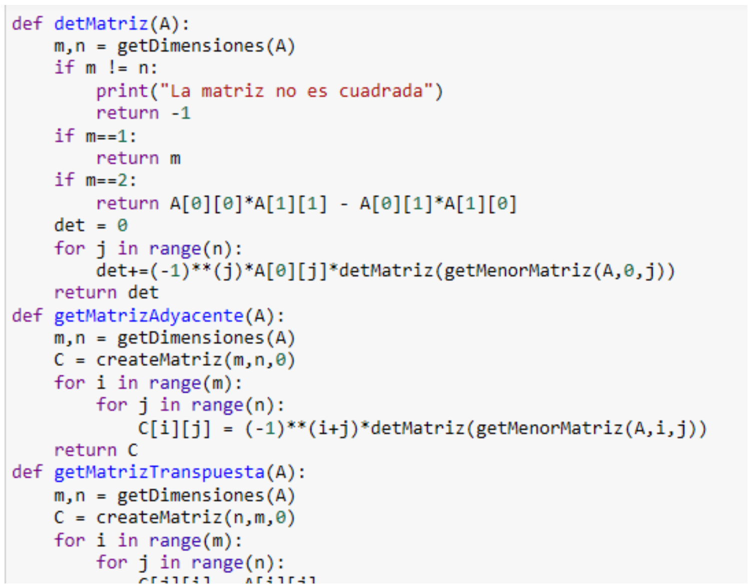


KATA 8

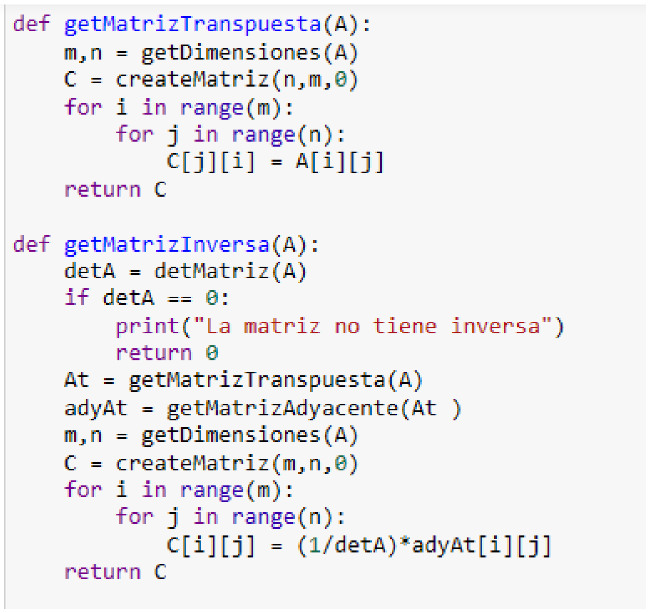
```python
def createMatriz(m,n,v):
    C = []
    for i in range(m):
        C.append([])
        for j in range(n):
            C[i].append(v)
    return C

def getDimensiones(A):
    return(len(A), len(A[0]))

def sumMatrices(A,B):
    Am,An = getDimensiones(A)
    Bm,Bn = getDimensiones(B)
    if Am != Bm or An!= Bn:
        print("Las dimensiones son diferentes")
        return[]
    C = createMatriz(Am,An,0)
    for i in range(Am):
        for j in range(An):
            C[i][j] = A[i][j] + B[i][j]
    return C
```



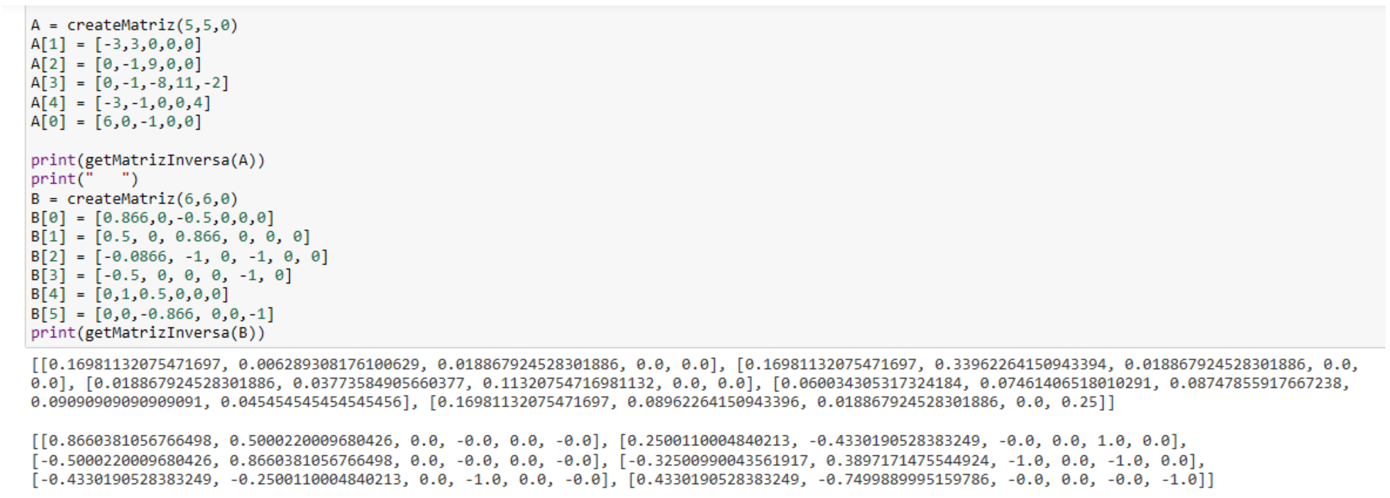
```python
def mulMatrices(A,B):
    Am,An = getDimensiones(A)
    Bm,Bn = getDimensiones(B)
    if An != Bm:
        print("Las matrices no son conformables")
        return []
    C = createMatriz(Am,Bn,0)
    for i in range(Am):
        for j in range(Bn):
            for k in range(An):
                C[i][j] += A[i][k] * B[k][j]
    return C

def getMenorMatriz(A,r,c):
    m,n = getDimensiones(A)
    C = createMatriz(m-1,n-1,0)
    for i in range (m):
        if i == r:
            continue
        for j in range(n):
            if j == c:
                continue
            Ci = i
            if i > r:
                    Ci = i-1
            Cj = j
```



```python
def detMatriz(A):
    m,n = getDimensiones(A)
    if m != n:
        print("La matriz no es cuadrada")
        return -1
    if m==1:
        return m
    if m==2:
        return A[0][0]*A[1][1] - A[0][1]*A[1][0]
    det = 0
    for j in range(n):
        det+=(-1)**(j)*A[0][j]*detMatriz(getMenorMatriz(A,0,j))
    return det
def getMatrizAdyacente(A):
    m,n = getDimensiones(A)
    C = createMatriz(m,n,0)
    for i in range(m):
        for j in range(n):
            C[i][j] = (-1)**(i+j)*detMatriz(getMenorMatriz(A,i,j))
    return C
def getMatrizTranspuesta(A):
    m,n = getDimensiones(A)
    C = createMatriz(n,m,0)
    for i in range(m):
        for j in range(n):
```



```python
def getMatrizTranspuesta(A):
    m,n = getDimensiones(A)
    C = createMatriz(n,m,0)
    for i in range(m):
        for j in range(n):
            C[j][i] = A[i][j]
    return C

def getMatrizInversa(A):
    detA = detMatriz(A)
    if detA == 0:
        print("La matriz no tiene inversa")
        return 0
    At = getMatrizTranspuesta(A)
    adyAt = getMatrizAdyacente(At )
    m,n = getDimensiones(A)
    C = createMatriz(m,n,0)
    for i in range(m):
        for j in range(n):
            C[i][j] = (1/detA)*adyAt[i][j]
    return C
```



```python
A = createMatriz(5,5,0)
A[1] = [-3,3,0,0,0]
A[2] = [0,-1,9,0,0]
A[3] = [0,-1,-8,11,-2]
A[4] = [-3,-1,0,0,4]
A[0] = [6,0,-1,0,0]

print(getMatrizInversa(A))
print("   ")
B = createMatriz(6,6,0)
B[0] = [0.866,0,-0.5,0,0,0]
B[1] = [0.5, 0, 0.866, 0, 0, 0]
B[2] = [-0.0866, -1, 0, -1, 0, 0]
B[3] = [-0.5, 0, 0, 0, -1, 0]
B[4] = [0,1,0.5,0,0,0]
B[5] = [0,0,-0.866, 0,0,-1]
print(getMatrizInversa(B))
```

```
[[0.16981132075471697, 0.006289308176100629, 0.018867924528301886, 0.0, 0.0], [0.16981132075471697, 0.33962264150943394, 0.018867924528301886, 0.0,
0.0], [0.018867924528301886, 0.03773584905660377, 0.11320754716981132, 0.0, 0.0], [0.060034305317324184, 0.07461406518010291, 0.08747855917667238,
0.09090909090909091, 0.045454545454545456], [0.16981132075471697, 0.08962264150943396, 0.018867924528301886, 0.0, 0.25]]

[[0.8660381056766498, 0.5000220009680426, 0.0, -0.0, 0.0, -0.0], [0.2500110004840213, -0.4330190528383249, -0.0, 0.0, 1.0, 0.0],
[-0.5000220009680426, 0.8660381056766498, 0.0, -0.0, 0.0, -0.0], [-0.32500990043561917, 0.3897171475544924, -1.0, 0.0, -1.0, 0.0],
[-0.4330190528383249, -0.2500110004840213, 0.0, -1.0, 0.0, -0.0], [0.4330190528383249, -0.7499889995159786, -0.0, 0.0, -0.0, -1.0]]
```