UNIVERSIDAD NACIONAL SAN CRISTOBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Laboratorio 02

Resolución de ejercicios

ASIGNATURA : Estructura de Datos Fundamentales y Algorítmicos

SIGLA : IS - 182

DOCENTE : Ing. EDEM JERSSON TERRAZA HUAMAN

INTEGRANTES : HUARI HERRERA, Quin Werner
MUÑOZ CORICHAHUA, Yeltsin Wilber

AYACUCHO – 2024

Ejercicio 9:

ejercicio_09.py > ...

```python
"""Accede al elemento central de una matriz"""

print("para ser poder acceder al elemento central, la matris tiene que ser de longitus impar, MxN M=N , impar")

import random        # importamos el modulo random
A = []       #creaos una lista basia


def pedir_f():                                                  # funcion para pedir al usuario el numero de filas
    filas = int(input("Digite el tamaño de la matris (impar):"))
    return filas


def llenar_mostrar(matriz):        # función que va crear la matris
    filas = pedir_f()           #se llama a la funcion pedir_f
    for i in range(filas):
        matriz.append([])       #esto hace que agregue una nueva fila
        for j in range(filas):
            matriz[i].append(random.randint(1, 10))        # va iterar las N veces que pusimos agregando un numero random
    print("matriz A")

    for row in matriz: # va imprimir fila por fila
        print(row)

def encontrar_elemento_central(matriz):  #funcion que ubicará al elemento central
    filas = len(matriz)  #lees cuanot elementos hay
    fila_central = filas // 2        #hace divicion entera y el resultado es el indice en la fila (como empiea en INDICE 0)

    columna_central = filas // 2         #hace divicion entera y el resultado es el indice en la columna (como empiea en INDICE 0)

    elemento_central = matriz[fila_central][columna_central]      #busca al elemeto central
    print(f"El elemento central de la matriz es: {elemento_central}")


llenar_mostrar(A)                #llamamos a la función
encontrar_elemento_central(A)    #llamamos a la función

```

```
para ser poder acceder al elemento central, la matris tiene que ser de longitus impar, MxN M=N , impar
Digite el tamaño de la matris (impar):5
matriz A
[1, 1, 10, 3, 8]
[4, 4, 4, 4, 9]
[8, 10, 2, 10, 5]
[9, 4, 9, 1, 3]
[3, 8, 1, 1, 6]
El elemento central de la matriz es: 2
PS E:\laboTerraza\LABO_terraza_#02>
```

Ejercicio 10:

ejercicio_10.py > ...

```python
"""multiplicacion de  dos matrices de diferentes tamaños"""

import numpy as np

# Definir dos matrices con dimensiones compatibles para multiplicación
matriz1 = np.array([[8, 2, 18],
                    [6, 5, 1],
                    [64, 8, 9],
                    [10, 11, 12]])

matriz2 = np.array([[83, 4, 45, 16],
                    [1, 18, 9, 20],
                    [51, 22, 3, 24]])


resultado_multiplicacion = np.dot(matriz1, matriz2) #multipliando raises


print("Matriz 1:") #mostramos la matriz 1
print(matriz1)

print("\nMatriz 2:") #mostramos la matriz 2
print(matriz2)

print("\nResultado de la multiplicación:")  #
print(resultado_multiplicacion)    #mostramos el resultado de la multiplicación

```

```
PS E:\laboTerraza\LABO_terraza_#02> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python310/python.exe e
Matriz 1:
[[ 8  2 18]
 [ 6  5  1]
 [64  8  9]
 [10 11 12]]

Matriz 2:
[[83  4 45 16]
 [ 1 18  9 20]
 [51 22  3 24]]

Resultado de la multiplicación:
[[1584  464  432  600]
 [ 554  136  318  220]
 [5779  598 2979 1400]
 [1453  502  585  668]]
PS E:\laboTerraza\LABO_terraza_#02>
```

Ejercicio 11:

ejercicio_11.py > ...

```python
"""Multiplica una matriz por un número"""

import numpy as np   #importamos el módulo numpy

                                    # ya definimos  una matriz
matriz = np.array([[12, 9, 34],
                   [4, 51, 6],
                   [31, 2, 9]])


n=int(input("Dijiste el multiplicador : "))
# Multiplicar la matriz por el escalar
matriz_resultante = n * matriz
                                    # Mostrar el resultado
print("Matriz original:")
print(matriz)

print("\nResultado de la multiplicación por el escalar:") # nueva linea
print(matriz_resultante)                                  #imprimimos la nueva matriz
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS E:\laboTerraza\LABO_terraza_#02> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python310/python
Dijiste el multiplicador : 5
Matriz original:
[[12  9 34]
 [ 4 51  6]
 [31  2  9]]

Resultado de la multiplicación por el escalar:
[[ 60  45 170]
 [ 20 255  30]
 [155  10  45]]
PS E:\laboTerraza\LABO_terraza_#02>
```

Ejercicio 12 :

ejercicio_12.py > ...

```python
""" Calcula la media de los elementos de una matriz""" #la suma de sus elementos entre la cantidad

import numpy as np #importamos el módulo

# Definir una matriz
matriz = np.array([[5, 10, 6],
                   [4, 5, 4],
                   [2, 8, 6]])

media_matriz = np.mean(matriz)      # Calcular la media de los elementos de la matriz

# Mostrar el resultado
print("Matriz:")
print(matriz)

print("\nMedia de los elementos de la matriz:", media_matriz) #imprime el resultado de la media
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS E:\laboTerraza\LABO_terraza_#02> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python310/python.exe e:/
Matriz:
[[ 5 10  6]
 [ 4  5  4]
 [ 2  8  6]]

Media de los elementos de la matriz: 5.555555555555555
PS E:\laboTerraza\LABO_terraza_#02>
```

Ejercicio 13:

```python
"""Crea una matriz de números aleatorios de tamaño 100x100"""

import random                # importamos el modulo random
import numpy as np            # importamos modulo matematico
A=[]
def pedir_filas():                                      # funcion para pedir al usuario el numero de filas
    filas = int(input("Digite el número de filas:"))
    return filas


def pedir_columnas():                                   # funcion para pedir al usuario el numero de columnas
    columnas = int(input("Digite el número de columnas: "))
    return columnas


def llenar(matriz):                         # función que va crear la matriz
    filas = pedir_filas()
    columnas = pedir_columnas()
    for i in range(filas):                # va iterar las N veces que pusimos agregando un numero random
        matriz.append([])
        for j in range(columnas):
            matriz[i].append(random.randint(1, 100))       #agrega un numero random

llenar(A)           #pasa la lista basia a la función llenar

print("Matriz A con numpy")   #imprimimos
matriz_array = np.array(A)    #creamos una varibales...llamamos a la matriz mediante el modulo np.array (creando una matriz )
print(matriz_array)           # imprimimos la matriz..
```

```
Digite el número de filas:100
Digite el número de columnas: 100
Matriz A con numpy
[[38 73 74 ... 22 63 70]
 [57 93 48 ...  4 59 62]
 [12 60 37 ... 22 14 44]
 ...
 [92 24 48 ... 17 53 51]
 [30 66 35 ... 37  5 94]
 [36 21 20 ... 75 84 60]]
PS E:\laboTerraza\LABO_terraza_#02>
```

```
Digite el número de filas:15
Digite el número de columnas: 15
Matriz A con numpy
[[ 99  29  98  33  58  96  16  33  16  48  50  33  95  58  70]
 [ 61  34  23  29  59  93  89  76 100  91  61  53 100  46  64]
 [  1  29  84  37  76  57 100  60  59  76  28  95   6  23  25]
 [ 27  24  40  32  42  58  69  85  95  25  90   2  53  39  46]
 [ 25  38  58  27  25   8  92  37  14  37  33  58  75  43  52]
 [ 48  85  65  12  60  95  95  94  90  92  29  17  16  93  29]
 [  3  95  27  22  12   1   1  12  73  40  28  25  82  89  70]
 [ 40  25  89  43  40  68  65  35  61  31  12  43  12  75  52]
 [ 16  51  66  50  76  88   9  37  62  98  47  48  11  42   7]
 [ 78  43  31  19   1  31  86  68   5  75  29  42  76  57   5]
 [ 35  61  31  88 100  28  91  16  33  54  77  81  65  76  12]
 [ 37  19  38  87  45  66  92  24  15  27  68  17  88  49  19]
 [  9  72  38  69  21  77  80  33  24  60  82   9  79  32  41]
 [ 97  99  72  34  68   2   3  73  25  43  54  68  67  54  56]
 [ 17  88  67  60  67  73  95  21  92  84  25  73  49  43  87]]
PS E:\laboTerraza\LABO_terraza_#02>
```

Ejercicio 14:

ejercicio_14.py > ...

```python
"""Calcula la media, la mediana y la desviación estándar de los elementos de una matriz."""

import statistics     #importamos este modulo ya que es una pregunta de estadistica

# Definir una matriz
matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

todos_los_elementos = [elemento for fila in matriz for elemento in fila]  # intera la matriz para obtener los elementos

                                                                          # Calcular la media, mediana y desviacion estandar
media= sum(todos_los_elementos) / len(todos_los_elementos)
mediana= statistics.median(todos_los_elementos)
desviacion_estandar = statistics.stdev(todos_los_elementos)

print("Matriz:")
for fila in matriz:
    print(fila)     #imprime la fila y así toda la matriz

print("\nMedia de los elementos de la matriz:", media)           #imprimen los resusltados
print("Mediana de los elementos de la matriz:", mediana)
print("Desviación estándar de los elementos de la matriz:", desviacion_estandar)
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS E:\laboTerraza\LABO_terraza_#02> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python310/python.exe e:/laboTerraza/LABO_terraza
Matriz:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

Media de los elementos de la matriz: 5.0
Mediana de los elementos de la matriz: 5
Desviación estándar de los elementos de la matriz: 2.7386127875258306
PS E:\laboTerraza\LABO_terraza_#02>
```

Ejercicio 15:

```python
"""Escribe una función que encuentre el elemento máximo de una matriz."""

def encontrar_maximo(matriz):
                                        # verificamos si la matriz tiene dimensiones incorrectas
    if not matriz or not matriz[0]:
        raise ValueError("La matriz está vacía o tiene dimensiones incorrectas")

    maximo = matriz[0][0]  # inicializamos con el primer elemento

    for fila in matriz:            # Iteramos  las filas y columnas de la matriz
        for elemento in fila:
            if elemento > maximo:               # Comparar cada elemento con el máximo actual
                maximo = elemento

    return maximo

matriz_ejemplo = [             #matriz ejemplo
    [1, 5, 3],
    [9, 2, 8],
    [4, 7, 6]
]

maximo = encontrar_maximo(matriz_ejemplo)

print("Matriz:")
for fila in matriz_ejemplo:    #imprimimos la matriz
    print(fila)

print("\nElemento máximo de la matriz:", maximo)   #imprimimos el resultado
```

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

PS E:\laboTerraza\LABO_terraza_#02> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python310/python.exe e:/laboTerraza/LA
Matriz:
[1, 5, 3]
[9, 2, 8]
[4, 7, 6]

Elemento máximo de la matriz: 9
PS E:\laboTerraza\LABO_terraza_#02>
```

Ejercicio 16:

```python
"""Escribe una función que encuentre la submatriz de mayor suma de una matriz.
"""
def submatriz_maxima(matriz):  #creamos una función para saber si esta correcto o no
    if not matriz or not matriz[0]:
        raise ValueError("La matriz está vacía o tiene dimensiones incorrectas")

    filas, columnas = len(matriz), len(matriz[0])  #cuenta los eleementos

    maxima_suma = float('-inf')      # Inicializar con un valor negativo infinito
    resultado = None

    for i in range(filas):
                                    # Inicializar un array para almacenar las sumas acumulativas de la columna actual
        suma_acumulativa = [0] * columnas

        for j in range(i, filas):
                                                                # Actualizar la suma acumulativa con la nueva fila
            for k in range(columnas):
                suma_acumulativa[k] += matriz[j][k]

                                                            # Aplicar el algoritmo de Kadane para encontrar la sublista de suma máxima
            maxima_suma_local = float('-inf')
            inicio_local = 0
            for k in range(columnas):
                if suma_acumulativa[k] > maxima_suma_local + suma_acumulativa[k]:
                    maxima_suma_local = suma_acumulativa[k]
                    inicio_local = k
                else:
                    maxima_suma_local += suma_acumulativa[k]

                                                                # Actualizar el resultado si encontramos una suma mayor
            if maxima_suma_local > maxima_suma:
                maxima_suma = maxima_suma_local
                resultado = [(i, inicio_local), (j, k)]

    return resultado
```

```python
# Ejemplo de uso
matriz_ejemplo = [
    [1, 2, -1, -4, -20],
    [-8, -3, 4, 2, 1],
    [3, 8, 10, 1, 3],
    [-4, -1, 1, 7, -6]
]

submatriz_maxima_resultado = submatriz_maxima(matriz_ejemplo)

                                                            # Mostrar la matriz y la submatriz de ma
print("Matriz:")        # Mostrar la matriz y la submatriz de mayor suma

for fila in matriz_ejemplo:
    print(fila)

print("\nSubmatriz de mayor suma:")  # Verificar si hay una submatriz de mayor suma

if submatriz_maxima_resultado:
    inicio, fin = submatriz_maxima_resultado
    for i in range(inicio[0], fin[0] + 1):      # Mostrar la submatriz

        print(matriz_ejemplo[i][inicio[1]:fin[1] + 1])
else:
    print("No hay submatriz de mayor suma.")
```

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL   PUERTOS

PS E:\laboTerraza\LABO_terraza_#02> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python310/python
Matriz:
[1, 2, -1, -4, -20]
[-8, -3, 4, 2, 1]
[3, 8, 10, 1, 3]
[-4, -1, 1, 7, -6]

Submatriz de mayor suma:
[-3, 4, 2]
[8, 10, 1]
[-1, 1, 7]
```

Ejercicio 17:

ejercicio_17.py > ...

```python
"""Escribe una función que encuentre la matriz de covarianza de dos matrices."""

def media(lista):                   #saca la media
    return sum(lista) / len(lista)

def covarianza(matriz1, matriz2):
    if len(matriz1) != len(matriz2):
        raise ValueError("Las matrices deben tener el mismo número de filas")

    n = len(matriz1)  # número de filas
    m = len(matriz1[0])  # número de columnas

    # Calcular medias de cada columna en ambas matrices
    media_matriz1 = [media(matriz1[i]) for i in range(n)]
    media_matriz2 = [media(matriz2[i]) for i in range(n)]

    # Inicializar la matriz de covarianza
    cov_matrix = [[0] * m for _ in range(m)]

    # Calcular la covarianza para cada par de columnas
    for i in range(m):
        for j in range(m):

            # Fórmula de covarianza

            cov_ij = sum((matriz1[k][i] - media_matriz1[i]) * (matriz2[k][j] - media_matriz2[j]) for k in range(n)) / (n - 1)
            cov_matrix[i][j] = cov_ij

    return cov_matrix

# Ejemplo de uso
matriz1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matriz2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]

matriz_cov = covarianza(matriz1, matriz2)

print("Matriz 1:")
for fila in matriz1:
    print(fila)

print("\nMatriz 2:")
for fila in matriz2:
    print(fila)

print("\nMatriz de covarianza:")      #imprime los resultados
for fila in matriz_cov:
    print(fila)
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
Matriz 2:
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]

Matriz de covarianza:
[-15.0, -9.0, -3.0]
[-9.0, -9.0, -9.0]
[-3.0, -9.0, -15.0]
PS E:\laboTerraza\LABO_terraza_#02>
```