UNIVERSIDAD PRIVADA DOMINGO SAVIO

Documento de Capturas #2

DOCENTE: Jimmy N. Requena Llorentty **TURNO:** Mañana

CARRERA: Ing. En Sistemas **MATERIA:** Programación II

ESTUDIANTE: Maria Fernanda Vidaurre Alvarado

Santa Cruz- Bolivia

Código #1 Ordenamiento burbuja

```python
#Codigo de ordenamiento burbuja
def ordenamiento_burbuja(lista):
    n = len(lista)  # Cantidad de elementos en la lista

    for i in range(n - 1):   # Bucle exterior para las pasadas
        hubo_intercambio = False  # Marca si hubo un intercambio en esta pasada

        # Bucle interior para las comparaciones e intercambios
        for j in range(n - 1 - i):  # Cada pasada evita revisar los últimos ya ordenados
            if lista[j] > lista[j + 1]:
                # ¡Intercambio!
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
                hubo_intercambio = True

        if not hubo_intercambio: # Si no hubo ningún intercambio, la lista ya está ordenada
            break

    return lista  # Opcional: también se puede omitir

if __name__ == "__main__":
    numeros = [6, 3, 8, 2, 5]
    print("Antes:", numeros)
    ordenamiento_burbuja(numeros)
```

```
python3 mai...   Ask Assistant
Antes: [6, 3, 8, 2, 5]
Después Ordenamiento Burbuja: [2, 3, 5, 6, 8]

--- Ejecutando pruebas con asserts ---
Caso 1 (Lista desordenada): PASSED
Caso 2 (Lista ya ordenada): PASSED
Caso 3 (Lista ordenada a la inversa): PASSED
Caso 4 (Lista con elementos duplicados): PASSED
Caso borde (Lista vacía): PASSED
Caso borde (Lista con un solo elemento): PASSED
/n Codigo realizado por: Maria F. Vidaurre Alvar
ado
```

Código #2 Ordenamiento por inserción

```python
#Codigo ordenamiento por insercion
def ordenamiento_insercion(lista):
    for i in range(1, len(lista)):
        valor_actual = lista[i]  # La "carta" que vamos a insertar
        posicion_actual = i

        # Desplazar elementos mayores hacia la derecha
        while posicion_actual > 0 and lista[posicion_actual - 1] > valor_actual:
            lista[posicion_actual] = lista[posicion_actual - 1]
            posicion_actual -= 1

        # Insertar la "carta" en su hueco correcto
        lista[posicion_actual] = valor_actual

    return lista

if __name__ == "__main__":
    numeros = [6, 3, 8, 2, 5]
    print("Antes:", numeros)
    ordenamiento_insercion(numeros)
    print("Después Ordenamiento Inserción:", numeros)

    print("\n--- Ejecutando pruebas con asserts ---")
```

```
python3 mai...   Ask Assistant
Antes: [6, 3, 8, 2, 5]
Después Ordenamiento Inserción: [2, 3, 5, 6, 8]

--- Ejecutando pruebas con asserts ---
Caso 1 (Lista desordenada): SUCCESS
Caso 2 (Lista ya ordenada): SUCCESS
Caso 3 (Lista ordenada a la inversa): SUCCESS
Caso 4 (Lista con duplicados): SUCCESS
Caso borde (Lista vacía): SUCCESS
Caso borde (Lista con un solo elemento): SUCCESS

Programa realizado por Maria F. Vidaurre Alvarad
o
```

Código #3 Merge Sort

```python
def merge_sort(lista):
  # Paso Vencer (Condición Base de la Recursividad):
  if len(lista) <= 1:
      return lista

  # Paso 1: DIVIDIR
  medio = len(lista) // 2
  mitad_izquierda = lista[:medio]
  mitad_derecha = lista[medio:]

  # Paso 2: VENCER (Recursión)
  izquierda_ordenada = merge_sort(mitad_izquierda)
  derecha_ordenada = merge_sort(mitad_derecha)

  # Paso 3: COMBINAR
  print(f"Mezclaría {izquierda_ordenada} y {derecha_ordenada}")
  return merge(izquierda_ordenada, derecha_ordenada)

def merge(izquierda, derecha):
  resultado = []
  i = j = 0

  # Comparar elementos de izquierda y derecha uno por uno
  while i < len(izquierda) and i < len(derecha):
```

```
python3 mai...                                    Ask Assistant
Mezclaría [5] y [2]
Mezclaría [1] y [2]
Mezclaría [3] y [1, 2]
Mezclaría [5] y [3]
Mezclaría [10] y [3, 5]
Mezclaría [6] y [2]
Mezclaría [8] y [2, 6]
Mezclaría [3, 5, 10] y [2, 6, 8]
Mezclaría [9] y [7]
Mezclaría [3] y [1]
Mezclaría [5] y [1, 3]
Mezclaría [7, 9] y [1, 3, 5]
Mezclaría [1] y [2]
Mezclaría [4] y [5]
Mezclaría [3] y [4, 5]
Mezclaría [1, 2] y [3, 4, 5]
Mezclaría [4] y [2]
Mezclaría [4] y [1]
Mezclaría [2] y [1, 4]
Mezclaría [2, 4] y [1, 2, 4]
Mezclaría [100] y [-50]
Mezclaría [50] y [-100]
Mezclaría [0] y [-100, 50]
Mezclaría [-50, 100] y [-100, 0, 50]
Mezclaría [1.2] y [3.8]
Mezclaría [2.5] y [1.2, 3.8]
¡Todas las pruebas con assert pasaron correctamente
!

Programa realizado por Maria F. Vidaurre Alvarado
```

Código # Matrices

```python
#Matrices
# 1. Crear la matriz de 3x3
teclado = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# 2. Imprimir la matriz completa
print("Matriz original:")
for fila in teclado:
    print(fila)

# 3. Acceder a elementos específicos
print("\nNúmero en el centro:", teclado[1][1])  # 5
print("Número en la esquina inferior derecha:", teclado[2][2])  # 9

# 4. Modificar el número en la esquina superior izquierda (1 por un 0)
teclado[0][0] = 0

# 5. Imprimir la matriz modificada
print("\nMatriz modificada:")
for fila in teclado:
```

```
python3 mai...   Ask Assistant

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

Número en el centro: 5
Número en la esquina inferior derecha: 9

Matriz modificada:
[0, 2, 3]
[4, 5, 6]
[7, 8, 9]
Matriz como cuadrícula:
0       8       9
4       5       6
1       2       3

Matriz 5x5 llena de ceros (con bucles):
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]

Programa realizado por Maria F. Vidaurre Alvarado
```

Código # Sumar todo elementos de una matriz

```python
# Codigo sumar todo elementos de una matriz
def sumar_total_matriz(matriz):

    # matriz = [[1, 2], [3, 4]]
    # resultado = 10

    total = 0
    for fila in matriz:
        for elemento in fila:
            total += elemento
    return total

# Función para probar que sumar_total_matriz funciona correctamente
def probar_suma_total():
    print("--- Probando sumar_total_matriz ---")

    # Caso 1: matriz normal
    m1 = [[1, 2, 3], [4, 5, 6]]
    try:
        assert sumar_total_matriz(m1) == 21, "Fallo en Caso 1: Matriz normal"
        print("Caso 1 (Matriz normal): PASSED")
    except AssertionError as e:
        print(f"Error: {e}")
```

```
python3 mai...   Ask Assistant

--- Probando sumar_total_matriz ---
Caso 1 (Matriz normal): PASSED
Caso 2 (Matriz con negativos y ceros): PASSED
Caso 3 (Matriz con una fila vacía): PASSED
Caso 4 (Matriz completamente vacía): PASSED
Caso 5 (Matriz de un solo elemento): PASSED
Todas las pruebas para sumar_total_matriz han final
izado

Programa realizado por Maria F. Vidaurre Alvarado
```

Código # Sumar por fila en matriz

```python
#Codigo para sumar por fila en matriz
# Definimos la función que suma los elementos por cada fila de la matriz
def sumar_por_filas(matriz):
    """
    Esta función recibe una matriz (lista de listas)
    y devuelve una lista con la suma de cada fila.

    Ejemplo:
    matriz = [[1, 2, 3], [4, 5, 6]]
    resultado = [6, 15]
    """
    resultado = []
    for fila in matriz:
        suma_fila = sum(fila)  # Suma todos los elementos de la fila
        resultado.append(suma_fila)
    return resultado

# Función de prueba para verificar que sumar_por_filas funciona correctamente
def probar_suma_por_filas():
    print("\n Probando sumar_por_filas")

    # Caso 1: matriz con 3 filas y 3 columnas
    m1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
    try:
```

```
python3 mai...   Ask Assistant

 Probando sumar_por_filas
Caso 1 (Matriz 3x3): success
Caso 2 (Matriz con pares repetidos): success
Caso 3 (Matriz con números negativos): success
Caso 4 (Matriz con filas de diferente longitud): su
ccess
Caso 5 (Matriz vacía): success
Caso 6 (Matriz con filas vacías): success
Todas las pruebas para sumar_por_filas han finaliza
do

Programa realizado por Maria F. Vidaurre Alvarado
```

Código # Sumar diagonal en matriz

```python
#Codigo para sumar diagonal en matriz
# Definimos la función que suma los elementos de la diagonal principal de una matriz cuadrada
def sumar_diagonal_principal(matriz):
    suma = 0
    for i in range(len(matriz)):
        suma += matriz[i][i]  # Accede al elemento en la posición (i, i)
    return suma

# Función de prueba para verificar que sumar_diagonal_principal funciona correctamente
def probar_suma_diagonal_principal():
    print("\nPrueba de sumar diagonal")

    # Caso 1: matriz 3x3 con números consecutivos
    m1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
    try:
        assert sumar_diagonal_principal(m1) == 15, "Fallo en Caso 1: Matriz 3x3 con números
consecutivos"
        print("Caso 1 (Matriz 3x3): PASSED")
    except AssertionError as e:
        print(f"Error: {e}")

    # Caso 2: matriz 2x2 con ceros y valores definidos
    m2 = [[10, 0], [0, 20]]
    try:
```

```
python3 mai...   Ask Assistant

Prueba de sumar diagonal
Caso 1 (Matriz 3x3): PASSED
Caso 2 (Matriz 2x2): PASSED
Caso 3 (Matriz 1x1): PASSED
Caso 4 (Matriz con números negativos): PASSED
Caso 5 (Matriz 4x4): PASSED
Todas las pruebas para sumar_diagonal_principal han
 finalizado

Programa realizado por Maria F. Vidaurre Alvarado
```