Варіант 7

In [40]:
```
import numpy as np
import time
```

Завдання 1

3, 4, 5, 6, 7,8, 9, 13, 19

Вправа 3

3. Вводяться 4 числа n, m, r, c. Вивести масив розміру n×m, в якому в кожному рядку з номером r і в кожному стовпчику з номером c стоять 0, а інші елементи дорівнюють 1.

In [43]:
```
def task3_numpy(n, m, r, c):
    result_array = np.ones((n, m), dtype=int)
    result_array[r, :] = 0
    result_array[:, c] = 0
    return result_array


def task3_iterative(n, m, r, c):
    result_array = []
    for i in range(n):
        row = [0 if i == r or j == c else 1 for j in range(m)]
        result_array.append(row)
    return result_array


n, m, r, c = map(int, input("Введіть 4 числа через пробіл (n m r c): ").split())


start_time_numpy = time.time()
result_numpy = task3_numpy(n, m, r, c)
end_time_numpy = time.time()


print("Результат (NumPy):")
print(result_numpy)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")


start_time_iterative = time.time()
result_iterative = task3_iterative(n, m, r, c)
end_time_iterative = time.time()
```

```
print("\nРезультат (Ітеративно):")
for row in result_iterative:
    print(row)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть 4 числа через пробіл (n m r c): 6 7 2 3
Результат (NumPy):
[[1 1 1 0 1 1 1]
 [1 1 1 0 1 1 1]
 [0 0 0 0 0 0 0]
 [1 1 1 0 1 1 1]
 [1 1 1 0 1 1 1]
 [1 1 1 0 1 1 1]]
Час виконання (NumPy): 0.0 секунд

Результат (Ітеративно):
[1, 1, 1, 0, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1]
[0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1]
Час виконання (Ітеративно): 0.0 секунд
```

Вправа 4

4. Вводяться числа n і m. Вивести масив розміру n×m, в якому у першому рядку (рядок з нулевим індексом) йдуть числа від 0 до m–1, а всі інші елементи матриці дорівнюють 0.

In [49]:
```
def task4_numpy(n, m):
    result_array = np.zeros((n, m), dtype=int)
    result_array[0, :] = np.arange(m)
    return result_array


def task4_iterative(n, m):
    result_array = []
    for i in range(n):
        row = [j if i == 0 else 0 for j in range(m)]
        result_array.append(row)
    return result_array


n, m = map(int, input("Введіть 2 числа через пробіл (n m): ").split())


start_time_numpy = time.time()
result_numpy = task4_numpy(n, m)
end_time_numpy = time.time()


print("Результат (NumPy):")
print(result_numpy)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")
```

```
start_time_iterative = time.time()
result_iterative = task4_iterative(n, m)
end_time_iterative = time.time()


print("\nРезультат (Ітеративно):")
for row in result_iterative:
    print(row)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть 2 числа через пробіл (n m): 3 4
Результат (NumPy):
[[0 1 2 3]
 [0 0 0 0]
 [0 0 0 0]]
Час виконання (NumPy): 0.0009953975677490234 секунд

Результат (Ітеративно):
[0, 1, 2, 3]
[0, 0, 0, 0]
[0, 0, 0, 0]
Час виконання (Ітеративно): 0.0 секунд
```

Вправа 5

Вводиться число n. Вивести масив розміру n×n, в якому в рядках з парними індексами стоять 1, а в інших – 0.

In [50]:
```
def task5_numpy(n):
    result_array = np.zeros((n, n), dtype=int)
    result_array[1::2] = 1
    return result_array


def task5_iterative(n):
    result_array = []
    for i in range(n):
        row = [1 if i % 2 == 0 else 0 for j in range(n)]
        result_array.append(row)
    return result_array


n = int(input("Введіть число n: "))


start_time_numpy = time.time()
result_numpy = task5_numpy(n)
end_time_numpy = time.time()


print("Результат (NumPy):")
print(result_numpy)
```

```
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")


start_time_iterative = time.time()
result_iterative = task5_iterative(n)
end_time_iterative = time.time()


print("\nРезультат (Ітеративно):")
for row in result_iterative:
    print(row)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть число n: 5
Результат (NumPy):
[[0 0 0 0 0]
 [1 1 1 1 1]
 [0 0 0 0 0]
 [1 1 1 1 1]
 [0 0 0 0 0]]
Час виконання (NumPy): 0.0 секунд

Результат (Ітеративно):
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
[1, 1, 1, 1, 1]
Час виконання (Ітеративно): 0.0009987354278564453 секунд
```

Вправа 6

6. З клавіатури вводиться масив. Замінити всі ненульові елементи на –1.

In [52]:
```
def task6_numpy(arr):
    return np.where(arr != 0, -1, arr)


def task6_iterative(arr):
    result_array = [x if x == 0 else -1 for x in arr]
    return result_array


arr = np.array(list(map(int, input("Введіть масив чисел через пробіл: ").split())))


start_time_numpy = time.time()
result_numpy = task6_numpy(arr)
end_time_numpy = time.time()


print("Результат (NumPy):", result_numpy)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")
```

```
start_time_iterative = time.time()
result_iterative = task6_iterative(arr)
end_time_iterative = time.time()


print("Результат (Ітеративно):", result_iterative)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть масив чисел через пробіл: 3 4 0 9 7 0 6 0 4 0 3
Результат (NumPy): [-1 -1  0 -1 -1  0 -1  0 -1  0 -1]
Час виконання (NumPy): 0.0020003318786621094 секунд
Результат (Ітеративно): [-1, -1, 0, -1, -1, 0, -1, 0, -1, 0, -1]
Час виконання (Ітеративно): 0.0 секунд
```

Вправа 7

7. З клавіатури вводиться масив. Замінити всі нульові елементи на –1.

In [53]:
```
def task7_numpy(arr):
    return np.where(arr == 0, -1, arr)


def task7_iterative(arr):
    result_array = [-1 if x == 0 else x for x in arr]
    return result_array


arr = np.array(list(map(int, input("Введіть масив чисел через пробіл: ").split())))


start_time_numpy = time.time()
result_numpy = task7_numpy(arr)
end_time_numpy = time.time()


print("Результат (NumPy):", result_numpy)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")


start_time_iterative = time.time()
result_iterative = task7_iterative(arr)
end_time_iterative = time.time()


print("Результат (Ітеративно):", result_iterative)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть масив чисел через пробіл: 3 4 0 6 5 0 3 0 4
Результат (NumPy): [ 3  4 -1  6  5 -1  3 -1  4]
Час виконання (NumPy): 0.0 секунд
Результат (Ітеративно): [3, 4, -1, 6, 5, -1, 3, -1, 4]
Час виконання (Ітеративно): 0.0 секунд
```

Вправа 8

8. З клавіатури вводиться масив. Підрахувати в ньому кількість нульових та ненульових елементів.

In [54]:
```
def task8_numpy(arr):
    num_zeros = np.count_nonzero(arr == 0)
    num_nonzeros = np.count_nonzero(arr != 0)
    return num_zeros, num_nonzeros

def task8_iterative(arr):
    num_zeros = sum(1 for x in arr if x == 0)
    num_nonzeros = len(arr) - num_zeros
    return num_zeros, num_nonzeros


arr = np.array(list(map(int, input("Введіть масив чисел через пробіл: ").split())))


start_time_numpy = time.time()
num_zeros_numpy, num_nonzeros_numpy = task8_numpy(arr)
end_time_numpy = time.time()


print("Нулів:", num_zeros_numpy)
print("Не нулів:", num_nonzeros_numpy)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")


start_time_iterative = time.time()
num_zeros_iterative, num_nonzeros_iterative = task8_iterative(arr)
end_time_iterative = time.time()


print("Нулів:", num_zeros_iterative)
print("Не нулів:", num_nonzeros_iterative)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть масив чисел через пробіл: 3 4 0 9 8 2 4 0 8 4 0
Нулів: 3
Не нулів: 8
Час виконання (NumPy): 0.0019974708557128906 секунд
Нулів: 3
Не нулів: 8
Час виконання (Ітеративно): 0.0 секунд
```

Вправа 9

9. Вводиться число n. Створити масив значень від n до 0.

In [55]:
```
def task9_numpy(n):
    return np.arange(n, -1, -1)

def task9_iterative(n):
    return [i for i in range(n, -1, -1)]
```

```
n = int(input("Введіть число n: "))


start_time_numpy = time.time()
result_numpy = task9_numpy(n)
end_time_numpy = time.time()


print("Результат (NumPy):", result_numpy)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")


start_time_iterative = time.time()
result_iterative = task9_iterative(n)
end_time_iterative = time.time()


print("Результат (Ітеративно):", result_iterative)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть число n: 10
Результат (NumPy): [10  9  8  7  6  5  4  3  2  1  0]
Час виконання (NumPy): 0.0 секунд
Результат (Ітеративно): [10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
Час виконання (Ітеративно): 0.0010006427764892578 секунд
```

Вправа 13

13. Розмістити на полі 8×8 нулі та одиниці в шахматному порядку, використовуючи функцію повторення (*).

In [56]:
```
def task13_numpy():
    chessboard = np.zeros((8, 8), dtype=int)
    chessboard[1::2, ::2] = 1
    chessboard[::2, 1::2] = 1
    return chessboard

def task13_iterative():
    chessboard = []
    for i in range(8):
        row = [1 if (i+j) % 2 == 1 else 0 for j in range(8)]
        chessboard.append(row)
    return chessboard


start_time_numpy = time.time()
result_numpy = task13_numpy()
end_time_numpy = time.time()


print("Результат (NumPy):")
print(result_numpy)
```

```
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")


start_time_iterative = time.time()
result_iterative = task13_iterative()
end_time_iterative = time.time()


print("\nРезультат (Ітеративно):")
for row in result_iterative:
    print(row)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Результат (NumPy):
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
Час виконання (NumPy): 0.0009999275207519531 секунд

Результат (Ітеративно):
[0, 1, 0, 1, 0, 1, 0, 1]
[1, 0, 1, 0, 1, 0, 1, 0]
[0, 1, 0, 1, 0, 1, 0, 1]
[1, 0, 1, 0, 1, 0, 1, 0]
[0, 1, 0, 1, 0, 1, 0, 1]
[1, 0, 1, 0, 1, 0, 1, 0]
[0, 1, 0, 1, 0, 1, 0, 1]
[1, 0, 1, 0, 1, 0, 1, 0]
Час виконання (Ітеративно): 0.0 секунд
```

Вправа 19

19. Згенерувати вектор із n елементів, що рівномірно розміщені на інтервалі (0,1) – тобто обидва кінці інтервалу не включені. Значення вивести до 3 знаків після коми.

In [60]:
```
def task19_numpy(n):
    return np.round(np.linspace(0, 1, n, endpoint=False), 3)

def task19_iterative(n):
    interval_length = 1 / n
    return [round(interval_length * i, 3) for i in range(n)]


n = int(input("Введіть кількість елементів вектора: "))


start_time_numpy = time.time()
result_numpy = task19_numpy(n)
end_time_numpy = time.time()
```

```
start_time_iterative = time.time()
result_iterative = task19_iterative(n)
end_time_iterative = time.time()


print("Результат (NumPy):", result_numpy)
print("Результат (Ітеративно):", result_iterative)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се
```

```
Введіть кількість елементів вектора: 10
Результат (NumPy): [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
Результат (Ітеративно): [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
Час виконання (NumPy): 0.0010011196136474 61 секунд
Час виконання (Ітеративно): 0.0 секунд
```

Завдання 2

In [61]:
```
# Задана матриця коефіцієнтів A
A = np.array([[2, 1, -5, 1],
              [1, -3, 0, -6],
              [0, 2, -1, 2],
              [1, 4, -7, 6]])

# Заданий вектор вільних членів B
B = np.array([8, 9, -5, 0])

# Вирішення системи рівнянь за допомогою формул Крамера
def solve_with_cramer(A, B):
    det_A = np.linalg.det(A)
    X = []
    for i in range(len(B)):
        Ai = A.copy()
        Ai[:, i] = B
        X.append(np.linalg.det(Ai) / det_A)
    return np.array(X)

# Вирішення системи рівнянь за допомогою матричного множення
def solve_with_matrix_multiplication(A, B):
    return np.linalg.inv(A) @ B

# Вирішення системи рівнянь за допомогою оберненої матриці
def solve_with_inverse_matrix(A, B):
    return np.linalg.inv(A) @ B

# Вирішення системи рівнянь за допомогою функції numpy.linalg.solve()
def solve_with_numpy_linalg_solve(A, B):
    return np.linalg.solve(A, B)

# Отримання розв'язків
solution_cramer = solve_with_cramer(A, B)
solution_matrix_multiplication = solve_with_matrix_multiplication(A, B)
```

```
solution_inverse_matrix = solve_with_inverse_matrix(A, B)
solution_numpy_linalg_solve = solve_with_numpy_linalg_solve(A, B)

# Виведення результатів
print("Розв'язок за допомогою формул Крамера:", solution_cramer)
print("Розв'язок за допомогою матричного множення:", solution_matrix_multiplication
print("Розв'язок за допомогою оберненої матриці:", solution_inverse_matrix)
print("Розв'язок за допомогою numpy.linalg.solve():", solution_numpy_linalg_solve)

# Порівняння розв'язків
print("\nПорівняння розв'язків:")
print("Формули Крамера == Матричне множення:", np.allclose(solution_cramer, solutio
print("Формули Крамера == Обернена матриця:", np.allclose(solution_cramer, solution
print("Формули Крамера == numpy.linalg.solve():", np.allclose(solution_cramer, solu
```

```
Розв'язок за допомогою формул Крамера: [ 3. -4. -1.  1.]
Розв'язок за допомогою матричного множення: [ 3. -4. -1.  1.]
Розв'язок за допомогою оберненої матриці: [ 3. -4. -1.  1.]
Розв'язок за допомогою numpy.linalg.solve(): [ 3. -4. -1.  1.]

Порівняння розв'язків:
Формули Крамера == Матричне множення: True
Формули Крамера == Обернена матриця: True
Формули Крамера == numpy.linalg.solve(): True
```

Завдання 3

In [62]:

```
# Визначення матриць A і B
A = np.array([[5, 3, -1],
              [2, -2, 0],
              [3, -1, 2]])

B = np.array([[1, 4, 16],
              [-3, -2, 0],
              [5, 7, 2]])

# Обчислення значення матричного виразу за допомогою універсальних функцій бібліоте
start_time_numpy = time.time()
result_numpy = 2 * (A - 0.5 * B) + A * B
end_time_numpy = time.time()

# Обчислення значення матричного виразу за допомогою ітеративних конструкцій
start_time_iterative = time.time()
rows, cols = A.shape
result_iterative = np.zeros((rows, cols))
for i in range(rows):
    for j in range(cols):
        result_iterative[i, j] = 2 * (A[i, j] - 0.5 * B[i, j]) + A[i, j] * B[i, j]
end_time_iterative = time.time()

# Виведення результатів та часу виконання
print("Результат за допомогою універсальних функцій бібліотеки NumPy:")
print(result_numpy)
print("Час виконання (NumPy):", end_time_numpy - start_time_numpy, "секунд")
```

```
print("\nРезультат за допомогою ітеративних конструкцій:")
print(result_iterative)
print("Час виконання (Ітеративно):", end_time_iterative - start_time_iterative, "се

# Порівняння результатів за допомогою функції numpy.allclose()
print("\nПорівняння результатів за допомогою numpy.allclose():", np.allclose(result
```

```
Результат за допомогою універсальних функцій бібліотеки NumPy:
[[ 14.  14. -34.]
 [  1.   2.   0.]
 [ 16. -16.   6.]]
Час виконання (NumPy): 0.0 секунд

Результат за допомогою ітеративних конструкцій:
[[ 14.  14. -34.]
 [  1.   2.   0.]
 [ 16. -16.   6.]]
Час виконання (Ітеративно): 0.002999544143676758 секунд

Порівняння результатів за допомогою numpy.allclose(): True
```

In [ ]: