Практическая работа №3

Выполнил студент группы БВТ2003 Глазков Даниил

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import csv
from PIL import Image
```

Задача 1

Дано множество из p матриц (n,n) и множество из p векторов (n,1). Написать функцию для рассчета суммы p произведений матриц (результат имеет размерность (n,1))

In [2]:
```
def mult(mat, vec, n ,p):
    ans = np.zeros([n, 1], dtype=int)
    for i in range(p):
        ans = ans + mat[i].dot(vec[i])
    return ans

try:
    data = np.genfromtxt(r"C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\input1
except IndexError:
    print("Ошибка входных данных")

m = len(data)
n = int(len(data[0]))

if m % (n + 1) != 0:
    print("Ошибка входных данных")
else:
    p = int(m / (n + 1))
    matrix = np.reshape(data[:m - p], [p, n, n])
    vectors = np.reshape(data[m - p:], [p, n, 1])
    np.savetxt("C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\output1.csv", mul
```

Задача 2

Написать функцию преобразовывающую вектор чисел в матрицу бинарных представлений.

In [13]:
```
import numpy as np
import csv
input_file_path = np.genfromtxt(r'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\

B = ((input_file_path.reshape(-1,1) & (2**np.arange(8))) != 0).astype(int)

output_file_path = 'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\result2.csv'
```

```
np.savetxt(output_file_path, B[:,::-1], delimiter=',', fmt='%f')
print(B[:,::-1])
```

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 1]
 [0 0 0 0 1 1 1 1]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

Задача 3

Написать функцию, которая возвращает все уникальные строки матрицы

In [7]:
```
input_file_path = np.genfromtxt(r'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\
unique_rows = np.vstack({tuple(row) for row in input_file_path})

output_file_path = 'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\result3.csv'
np.savetxt(output_file_path, unique_rows, delimiter=',', fmt='%f')
print(unique_rows)
```

```
[[1 1 1 0 0 0]
 [1 1 1 1 1 0]
 [0 1 1 1 0 0]]
C:\Users\loprz\AppData\Local\Temp\ipykernel_27564\1816678958.py:2: FutureWarning:
arrays to stack must be passed as a "sequence" type such as list or tuple. Suppor
t for non-sequence iterables such as generators is deprecated as of NumPy 1.16 an
d will raise an error in the future.
  unique_rows = np.vstack({tuple(row) for row in input_file_path})
```

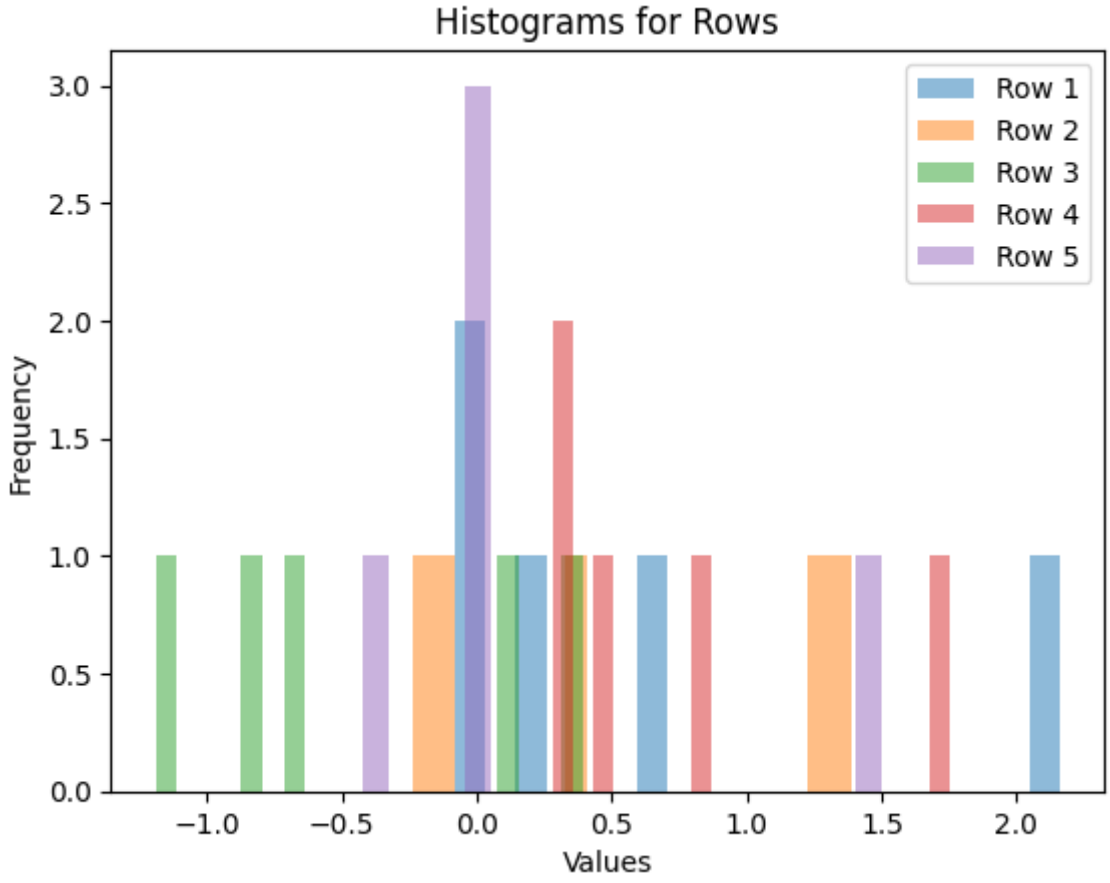
Задача 4

Написать функцию, которая заполняет матрицу с размерами (M,N) случайными числами распределенными по нормальному закону. Затем считает мат. ожидание и дисперсию для каждого из столбцов, а также строит для каждой строки стоит гистограмму значений (использовать функцию hist из модуля matplotlib.plot)

In [14]:
```
def generate_and_analyze_matrix(M, N):

    matrix = np.random.randn(M, N)
    print(matrix)

    means = np.mean(matrix, axis=0)

    variances = np.var(matrix, axis=0)

    for i in range(M):
        plt.hist(matrix[i, :], bins=20, alpha=0.5, label=f'Row {i+1}')

    plt.xlabel('Values')
    plt.ylabel('Frequency')
```

```
    plt.legend(loc='upper right')
    plt.title('Histograms for Rows')
    plt.show()
    print(means, variances)

if __name__ == '__main__':
    M = int(input("Введите количество строк (M): "))
    N = int(input("Введите количество столбцов (N): "))

    generate_and_analyze_matrix(M, N)
```

```
[[ 2.15869532  0.63787063  0.25180606 -0.07893922  0.01043733]
 [ 0.33508874  1.26629266 -0.08585652  1.38628111 -0.24034474]
 [ 0.15038924 -1.18844991 -0.66141932 -0.86347402  0.39173579]
 [ 0.85056705  0.43719632  0.29227079  1.75232628  0.28239316]
 [-0.42556205  0.05347966  0.03813589  1.49971466 -0.02735205]]
```




```
[ 0.61383566  0.24127787 -0.03301262  0.73918176  0.0833739 ] [0.76309263 0.66514
297 0.11793611 1.05228744 0.05141389]
```

Задача 5

Написать функцию, которая заполняет матрицу (M,N) в шахматном порядке заданными числами a и b.

In [15]:
```
def fill_chessboard_matrix(M, N, a, b):
    matrix = [[0 for _ in range(N)] for _ in range(M)]

    for i in range(M):
        for j in range(N):
            if (i + j) % 2 == 0:
                matrix[i][j] = a
```

```
            else:
                matrix[i][j] = b

    return matrix

def save_matrix_to_csv(matrix, filename):
    with open(filename, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        for row in matrix:
            writer.writerow(row)

M = int(input("Введите количество строк (M): "))
N = int(input("Введите количество столбцов (N): "))
a = int(input("Введите число (a): "))
b = int(input("Введите число (b): "))

matrix = fill_chessboard_matrix(M, N, a, b)

csv_filename = "C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\result4.csv"
np.savetxt(csv_filename, matrix, delimiter=',', fmt='%f')
print(matrix)
```

```
[[0, 1, 0, 1, 0], [1, 0, 1, 0, 1], [0, 1, 0, 1, 0], [1, 0, 1, 0, 1], [0, 1, 0, 1,
0]]
```

Задача 6

Написать функцию, которая возвращает тензор представляющий изображение круга с заданным цветом и радиусом в схеме rgd на черном фоне.

In [16]:
```
def create_circle_image(radius, color):

    width = height = radius * 2
    image = np.zeros((height, width, 3), dtype=np.uint8)

    center_x, center_y = width // 2, height // 2

    y, x = np.ogrid[:height, :width]
    mask = (x - center_x) ** 2 + (y - center_y) ** 2 <= radius ** 2

    image[mask] = color

    pil_image = Image.fromarray(image)

    return pil_image

radius = int(input("Введите радиус: "))
r = int(input("Введите крвсный цвет: "))
g = int(input("Введите зеленый цвет: "))
b = int(input("Введите синий цвет: "))
color = (r, g, b)  #10, 186, 181

circle_image = create_circle_image(radius, color)

circle_image.save("circle.png")
circle_image.show()
```

Задача 7

Написать функцию, которая стандартизирует все значения тензор (отнять мат. ожидание и поделить на СКО)

In [17]:
```
def standardize_tensor(tensor):
    mean = np.mean(tensor)
    std = np.std(tensor)
    standardized_tensor = (tensor - mean) / std

    return standardized_tensor

if __name__ == '__main__':

    tensor = np.genfromtxt(r"C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\vect


    standardized_result = standardize_tensor(tensor)
    csv_filename = "C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\result7.csv"
    np.savetxt(csv_filename, standardized_result, delimiter=',', fmt='%f')

    print("Исходный тензор:")
    print(tensor)
    print("\nСтандартизированный тензор:")
    print(standardized_result)
```

```
Исходный тензор:
[  0   1   2   3  15  16  32  64 128]

Стандартизированный тензор:
[-0.72424597 -0.69927197 -0.67429797 -0.64932397 -0.34963599 -0.32466199
  0.074922    0.87408996  2.4724259 ]
```

Задача 8

Написать функцию, выделяющую часть матрицы фиксированного размера с центром в данном элементе (дополненное значением fill если необходимо)

In [18]:
```
def extract_submatrix(matrix, center_row, center_col, size, fill):
    rows, cols = matrix.shape

    start_row = max(0, center_row - size // 2)
    end_row = min(rows, center_row + (size + 1) // 2)
    start_col = max(0, center_col - size // 2)
    end_col = min(cols, center_col + (size + 1) // 2)

    submatrix = np.full((size, size), fill)

    row_offset = center_row - size // 2 - start_row
    col_offset = center_col - size // 2 - start_col

    submatrix[row_offset:row_offset + end_row - start_row, col_offset:col_offset

    return submatrix
matrix = np.genfromtxt(r'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\input8.c
```

```
output_filename = 'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\result8.csv'
center_row = 2
center_col = 2
size = 3
fill_value = 0
result = extract_submatrix(matrix, center_row, center_col, size, fill_value)
print(result)

np.savetxt(output_filename, result, delimiter=',', fmt='%f')
```

```
[[ 7  8  9]
 [12 13 14]
 [17 18 19]]
```

Задача 9

Написать функцию, которая находит самое часто встречающееся число в каждой строке матрицы и возвращает массив этих значений

In [19]:
```
def most_common_number(arr):
    unique, counts = np.unique(arr, return_counts=True)
    index = np.argmax(counts)
    return unique[index]

def process_csv_file(input_file, output_file):
    data = []

    with open(input_file, 'r') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            data.append([int(x) for x in row])

    data = np.array(data)

    result = np.apply_along_axis(most_common_number, axis=1, arr=data)

    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        for value in result:
            writer.writerow([value])

if __name__ == "__main__":
    input_file = 'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\input9.csv'
    output_file = 'C:\\Users\\loprz\\OneDrive\\Рабочий стол\\pr3\\result9.csv'
    process_csv_file(input_file, output_file)

print(result)
```

```
[[ 7  8  9]
 [12 13 14]
 [17 18 19]]
```

Задача 10

Дан трёхмерный массив, содержащий изображение, размера (height, width, numChannels), а также вектор длины numChannels. Написать функцию, которая

складывает каналы изображения с указанными весами, и возвращает результат в виде матрицы размера (height, width)

In [20]:
```
def combine_channels(image, channel_weights):
    if len(image.shape) != 3 or image.shape[2] != len(channel_weights):
        raise ValueError("Размерность массива изображения и длина вектора весов

    weighted_channels = image * channel_weights

    combined_image = np.sum(weighted_channels, axis=2)

    return combined_image

image = np.array([[[1, 2, 3], [4, 5, 6]],
                  [[7, 8, 9], [10, 11, 12]]])

channel_weights = np.array([0.5, 0.3, 0.2])

result = combine_channels(image, channel_weights)
print(result)
```

```
[[ 1.7  4.7]
 [ 7.7 10.7]]
```