Павшук Ярослав Денисович, ФІТ 3-14. Варіант 21  
  
**1.1  
  
2)**  
  
import numpy as np

# Введення матриць A і B

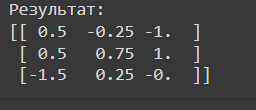
A = np.array([[2, 3, 1], [-1, 1, 0], [1, 2, -1]])

B = np.array([[1, 2, 1], [0, 1, 2], [3, 1, 1]])

# Розв'язання системи рівнянь AX = B

X = np.linalg.solve(A, B)

print("Результат:")

print(X)  
  
 **1.2  
  
2)**  
  
import numpy as np

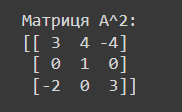
# Введення матриці

A = np.array([[-1, 0, 2], [0, 1, 0], [1, 2, -1]])

# Піднесення матриці до ступеня 2 (квадрат)

A\_squared = np.dot(A, A)

print("Матриця A^2:")

print(A\_squared)  
  
  
  
  
**1.3  
  
2)** import numpy as np

# Введення матриць

A = np.array([[5, 8, -4], [6, 9, -5], [4, 7, -3]])

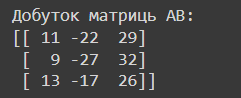
B = np.array([[3, 2, 5], [4, -1, 3], [9, 6, 5]])

# Знаходження добутку матриць

result = np.dot(A, B)

# result = A @ B

print("Добуток матриць AB:")

print(result)  
  
  
 **1.4**  
**2)**  
import numpy as np

# Введення матриць

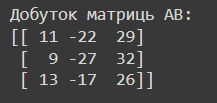
A = np.array([[5, 8, -4], [6, 9, -5], [4, 7, -3]])

B = np.array([[3, 2, 5], [4, -1, 3], [9, 6, 5]])

# Знаходження добутку матриць

result = np.dot(A, B)

print("Добуток матриць AB:")

print(result)  
  
  
**1.5  
2)**import numpy as np

# Введення матриці

matrix = np.array([[0, 2, 0], [3, 4, 5], [6, 7, 8]])

# Обчислення визначника

determinant = np.linalg.det(matrix)

print("Визначник матриці:")

print(determinant)  
   
  
**1.6  
2)**import numpy as np

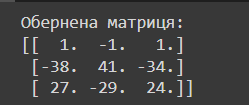
# Введення матриці

matrix = np.array([[2, 5, 7], [6, 3, 4], [5, -2, -3]])

# Знаходження оберненої матриці

inverse\_matrix = np.linalg.inv(matrix)

print("Обернена матриця:")

print(inverse\_matrix)  
  
  
**1.7  
2)**import numpy as np

# Введення матриці

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

# Знаходження рангу матриці

rank = np.linalg.matrix\_rank(matrix)

print("Ранг матриці:")

print(rank)  
  
  
**1.8  
2)**import numpy as np

# Введення коефіцієнтів системи

A = np.array([[4, 1, 4], [1, 1, 2], [2, 1, 2]])

B = np.array([-2, -1, 0])

# Метод Крамера

det\_A = np.linalg.det(A)

det\_X = np.linalg.det(np.column\_stack((B, A[:, 1:])))

det\_Y = np.linalg.det(np.column\_stack((A[:, 0], B, A[:, 2])))

det\_Z = np.linalg.det(np.column\_stack((A[:, :2], B)))

x\_kramer = det\_X / det\_A

y\_kramer = det\_Y / det\_A

z\_kramer = det\_Z / det\_A

print("Розв'язок за методом Крамера:")

print("x =", x\_kramer)

print("y =", y\_kramer)

print("z =", z\_kramer)

# Матричний метод (обчислення оберненої матриці)

A\_inv = np.linalg.inv(A)

solution\_matrix = np.dot(A\_inv, B)

print("\nРозв'язок за матричним методом:")

print("x =", solution\_matrix[0])

print("y =", solution\_matrix[1])

print("z =", solution\_matrix[2])

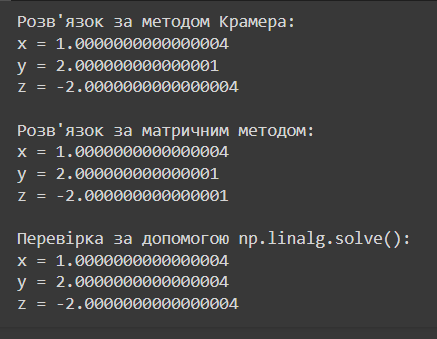
# Перевірка за допомогою np.linalg.solve()

solution\_check = np.linalg.solve(A, B)

print("\nПеревірка за допомогою np.linalg.solve():")

print("x =", solution\_check[0])

print("y =", solution\_check[1])

print("z =", solution\_check[2])  
  
  
**2.2**  
  
import numpy as np

# Задати розмірність матриці

N = 5 # кількість рядків

M = 4 # кількість стовпців

# Створити матрицю з випадкових елементів

matrix\_A = np.random.rand(N, M)

# Знайти середні значення для кожного рядка

row\_means = np.mean(matrix\_A, axis=1)

# Знайти найнижче значення серед середніх значень рядків

min\_row\_mean = np.min(row\_means)

print("Прямокутна матриця A:")

print(matrix\_A)

print("\nСередні значення для кожного рядка:")

print(row\_means)

print("\nНайнижче значення серед середніх значень рядків:")

print(min\_row\_mean)  
