Клас Matrix використовується для реалізації матриць m×n та векторів як матриць вигляду 1×n, або n×1.

Кожна матриця представляється у вигляді списку list() у мові програмування Python. Елементами цього списку також є списки, тобто підсписки. Відповідно кількість підсписків відповідає кількості рядків матриці, а кількість елементів вже у самих підсписках це кількість стовпців матриці. Всі підсписки однакових розмірів.
Щоб отримати відповідну матрицю, для створення екземпляру класу необхідно зазначити кількість рядків, кількість стовпців, і список і значеннями матриці, які виписати по рядках і подані у вигляді списку. Далі відповідно до кількості рядків і стовпців цей список значень буде розділено на підсписки й додано до основного списку, який і є вже готовою матрицею.

Перед виконанням додавання та віднімання матриць виконується перевірка чи однакових розмірів ці матриці, якщо так, то додавання (віднімання) виконується покомпонентно. У даній реалізації дії відбуваються між відповідними елементами підсписків.

Для множення спочатку йде стандартна перевірка чи можливо виконати множення, чи кількість стовпців першої матриці дорівнює кількості рядків другої. Якщо так, то переходимо до множення. Спочатку створюється відповідна матриця з нулів, де будуть зберігатися значення після виконання множення. Далі виконується стандартне множення матриць.

LUP - розклад реалізується окремою функцією в класі, і є реалізацією алгоритму наведеному на лекції з урахуванням особливостей представлення матриць в класі. На вхід подається матриця A, виконується її LUP - розклад і після цього ще окремо створюються порожні матриці для матриць L та U, які потім заповнюються відповідно як верхня та нижня трикутні матриці зі значень вже розкладеної матриці A. Функція повертає матриця L, U та перестановку P, яка представлена списком. На третьому кроці розкладу використовується вбудована в Python функція abs(), яка повертає абсолютне значення, виконує роль модуля.

Для розв'язання системи лінійних рівнянь використовується алгоритм для розв'язання трикутних систем рівнянь. На вхід подається матриця A з коефіцієнтів біля змінних і вектор констант b, який подається як матриця виду n×1. Далі виконується LUP - розклад, і ми отримуємо матриці L, U і перестановку P з допомогою якої робимо вектор b' (P_b). А далі за формулами спочатку розв'язуємо Ly = b', а потім Ux = y. І отримуємо вектор розв'язків x.
Для перевірки було знайдено систему рівнянь 6×6:

$$
\begin{aligned}
a + b - 2c + d + 3e - f &= 4 \\
2a - b + c + 2d + e - 3f &= 20 \\
a + 3b - 3c - d + 2e + f &= -15 \\
5a + 2b - c - d + 2e + f &= -3 \\
-3a - b + 2c + 3d + e + 3f &= 16 \\
4a + 3b + c - 6d - 3e - 2f &= -27
\end{aligned}
$$

Отримані розв’язки з допомогою реалізованого алгоритму:

```
[Running] python -u "c:\Users\ivann\Documents\GitHub\task_5\matrix.py"
[0.9999999999999994, -2.000000000000013, 3.000000000000001, 3.999999999999998, 2.000000000000002, -0.9999999999999998]

[Done] exited with code=0 in 0.126 seconds
```

Якщо змусити Python заокруглити отримані значення, то отримаємо той же результат, що видає WolframAlpha, а саме: