: **Python and Linear Algebra** :

: Assignment 4 :

Matrix Decomposition :

A matrix decomposition is a way of reducing a matrix into its constituent parts. It is an approach that can simplify more complex matrix operations that can be performed on the decomposed matrix rather than on the original matrix itself.

A common analogy for matrix decomposition is the factoring of numbers, such as the factoring of 25 into 5 × 5. For this reason, matrix decomposition is also called matrix factorization. Like factoring real values, there are many ways to decompose a matrix, hence there are a range of different matrix decomposition techniques.

LU Matrix Decomposition -

The LU decomposition is for square matrices and decomposes a matrix into L and U components.

$$A = L \cdot U$$

A - square matrix that we wish to decompose,
L is the lower triangle matrix, and
U is the upper triangle matrix.

A variation of this decomposition that is numerically more stable to solve in practice is called the LUP decomposition, or the LU decomposition with partial pivoting.

$$A = L \cdot U \cdot P$$

The rows of the parent matrix are re-ordered to simplify the decomposition process and the additional P matrix specifies a way to permute the result or return the result to the original order.

There are also other variations of the LU. The LU decomposition is often used to simplify the solving of systems of linear equations, such as finding the coefficients in a linear regression. The LU decomposition can be implemented in Python with the lu() function.

```
# LU decomposition
from numpy import array
from scipy.linalg import lu
# define a square matrix
A = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(?)
# LU decomposition
P, L, U = lu(A)
print(P)
print(?)
print(?)
# reconstruct
B = P.dot(L).dot(U)
print(B)
```

Task 1 : For this lesson, you must implement small examples of other simple methods for matrix factorization,
such as the **QR decomposition**, the **Cholesky decomposition** and the **eigendecomposition**.

Singular-Value Decomposition :

The Singular-Value Decomposition, or **SVD** for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler.

$$\mathbf{A = U \cdot \Sigma \cdot V^{T}}$$

A is the real m × n matrix that we wish to decompose,
U is an m × m matrix,
Σ (sigma) is an m × n diagonal matrix, and
V T is the transpose of an n × n matrix where T is a superscript.

The SVD can be calculated by calling the svd() function.
The function takes a matrix and returns the U , Σ and V T elements.

The Σ diagonal matrix is returned as a vector of singular values. The V matrix is returned in a **transposed form**, e.g. V.T.

```
# Singular-value decomposition
from numpy import array
from scipy.linalg import svd
# define a matrix
A = array([[1, 2], [3, 4], [5, 6]])
print(A)
# SVD
U, s, V = svd(A)
print(U)
print(s)
print(V)
```

Task 2 : List 5 applications of the **SVD**. Demonstrate each with a small example in Python.