Linear Algebra Project (Week #2)

Reduced Row-Echelon Form using Python

In this project, we want to solve a system of linear equations by reducing the system's augmented matrix to reduced row-echelon form (RREF).

As an example, let's solve the following system of equations by hand:

$$\begin{cases} 3x + 2y = 11 \\ 5x - 7y = 8 \end{cases}$$

1) Converting the system to matrix form, we have:

$$\begin{bmatrix} 3 & 2 \\ 5 & -7 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 11 \\ 8 \end{bmatrix}$$

2) Converting the matrix form to the augmented matrix, we have:

$$M = \left[\begin{array}{cc|c} 3 & 2 & 11 \\ 5 & -7 & 8 \end{array}\right]$$

3) Reducing the augmented matrix to RREF, we have:

$$M = \left[\begin{array}{cc|c} 1 & 0 & 3 \\ 0 & 1 & 1 \end{array}\right]$$

4) This tells us that the system's solution is $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$ or $\begin{cases} x = 3 \\ y = 1 \end{cases}$

Now, let's solve the same system of equations using Python.

1) First, we'll need to import the `sympy` library.

```
import sympy as sy
```

2) We'll still need to manually create our system's augmented matrix, but we then enter it into the code using `sympy`'s `matrix` method. (We'll also print out our newly-entered matrix to make sure it is correct.)

```
M = sy.Matrix([ [3,2,11],
                [5,-7,8]])
sy.pprint(M)
```

3) Conveniently, `sympy` will automatically reduce a matrix to RREF by using the `rref` method.

```
R = M.rref( pivots = False )
sy.pprint(R)
```

Note that we had to give an extra instruction to the `rref` method: `pivots = False`. By default, `rref` returns both the RREF matrix and the column numbers containing pivots. This week's project has no need of the pivots, so we tell `rref` to skip them.

We're also printing the RREF matrix, so we can compare it to our manually calculated version (on the first page).

4) Finally, we create a new variable (which we'll call `c`) to store our solutions, and extract those solutions from the RREF matrix using the `col` method. This returns the indicated column. In this case, despite the solutions being in the matrix's third column, is considered Column #2. (Python starts counting at 0, so a three-column matrix has Column #0, Column #1, and Column #3.)

```
c = R.col(2)
sy.pprint(c)
```

5) Now, we're ready to run the code!

Your output should look like this:

```
⎡3  2   11⎤
⎢         ⎥
⎣5  -7  8 ⎦
⎡1  0  3⎤
⎢       ⎥
⎣0  1  1⎦
⎡3⎤
⎢ ⎥
⎣1⎦
```

Solving Systems of Equations Using RREF and Python

Using Python, **solve the system of linear equations** shown below. Display your answer as a column vector.

$$\begin{cases} x + 3y - z + 2t + u = 19 \\ x + y + 2z + t + u = 18 \\ 3x - 2y - z + 2t - 3u = 5 \\ 2x + y - 5z - 7t - u = -16 \\ 5x + y + 3z - 2t + 2u = 36 \end{cases}$$

Systems Word Problems with RREF and Python

We're going to prepare a meal using only five different foods. However, the meal needs to provide exact amounts of calories and nutrients. The following table shows the nutritional values of the five foods (labeled "A" through "E") and the meal's required values.

	Food A	**Food B**	**Food C**	**Food D**	**Food E**	**Required**
Calories (per ounce)	100	20	50	60	200	1200
Protein (in grams/ounce)	5	6	7	4	3	200
Vitamin C (in mg/ounce)	10	2	5	5	6	500
Vitamin B (in mg/ounce)	12	4	10	1	13	600
Vitamin D (in mg/ounce)	7	9	3	5	10	300

Use Python to determine how many ounces of each food will be needed to exactly meet the meal's required values.

Hints:

1. Use the table to create a system of linear equations.
2. Convert the system to an augmented matrix.
3. Use Python to reduce the augmented matrix to RREF.
4. Print the answer as a vector.

END