#Ex1

```
import numpy as np
import sympy as sp

#An example - Solve the following linear system:
# x + y = 0
# x - y = 2

x, y = sp.symbols('x,y')
# defining equations
eq1 = sp.Eq( x + y, 0)
eq2 = sp.Eq( x - y, 2)
#eq2 = sp.Eq( x + y, 2)

sol = sp.solve( (eq1, eq2), (x, y))
print("solution = ",  sol)

if x in sol:
  print("x=", sol[x])

if y in sol:
  print("y=", sol[y])

#The "sp.solve" function return a Dictionary, check more about Python Dictionary in:
https://www.w3schools.com/python/python_dictionaries.asp

#How about Ex1a?
x, y, z = sp.symbols('x,y,z')
# defining equations
eq1 = sp.Eq( x + 2*y + z, 0)
#eq2 = ...
#eq3 = ...
#sol = sp.solve...
```

#Ex2

```
#REMOVE the line "a3x + b3y = c3" in the exercise content

import matplotlib.pyplot as plt
import numpy as np

def plot2DEquation(x_arr, a, b, c):
  """
  This function plots the linear equation ax + by = c in a 2D coordinate system.

  x_arr: values on the x-axis
  """

  if b != 0:
    fx = lambda x: (c - a*x)/b  #note that y=f(x)
    y_arr = list( map(fx, x_arr) )
    plt.plot(x_arr, y_arr, label = str(a) + "*x + " + str(b) + "*y = " + str(c))
```

```
 else:
   #You handle this case


#Try the function "plot2DEquation" in the following linear systems:
#Based on the resulted graph, we print out the number of solutions for the system
#System 1 ------------------------------
# x + y = 0
# x - y = 2

#Firstly, we manually create x-values on the x-axis as follows:
x_arr = np.arange(-10, 10.1, 0.1)

plot2DEquation(x_arr, 1, 1, 0)
plot2DEquation(x_arr, 1, -1, 2)

plt.title("System 1 - ONE solution")
plt.legend()
plt.show()

#System 2 ------------------------------
# x + y = 5
# 2x + 2y = 12

#System 3 ------------------------------
# x + y = 5
# 3x + 3y = 15

#System 4 ------------------------------
# x = 0
# x - y = 2
```

#Ex3

```
import matplotlib.pyplot as plt
import numpy as np

def plotEquation3D(ax, x_arr, a, b, c, d, color):
 """
 This function plots the linear equation ax + by + cz = d in a 3D coordinate system.

 ax: axes
 x_arr: values on the x-axis
 color: color of the graph
 """
 if c != 0:
   z_func = lambda x, y: (d - a*x - b*y) / c

   y_arr = x_arr.copy()
   X,Y = np.meshgrid(x_arr, y_arr) #a grid of points created from all combinations of x-values and y-values
   Z = z_func(X, Y) # evaluation of the z-function on the grid

   ax.plot_surface(X, Y, Z, color = color)
 else:
   #Google search "python plot a 2D line in 3d"
   #You handle this case


#Try the function "plotEquation3D" in the following linear systems:
```

#Based on the resulted graph, **it is difficult to find out the number of solutions** for the system

```
#System 1 ------------------------------
# 25x + 5y + z = 106.8
# 64x + 8y + z = 177.2
# 144x+ 12y+ z = 279.2

#Firstly, we manually create x-values on the x-axis as follows:
x_arr = np.arange(-10, 10.1, 0.1)
fig = plt.figure()
ax= fig.add_subplot(projection= '3d')

plotEquation3D(ax, x_arr, 25, 5, 1, 106.8, "blue")
plotEquation3D(ax, x_arr, 64, 8, 1, 177.2, "green")
plotEquation3D(ax, x_arr, 144, 12, 1, 279.2, "red")

plt.title("System 1")
ax.set_xlim([-10, 10])
ax.set_ylim([-20, 20])
#ax.set_zlim([-50, 50])
plt.show()

#System 2 ------------------------------
# x + y + z = 10
# x + y + z = 20
# 3x + 3y + 3z = 40

#System 3 ------------------------------
# x + 2y + z = 0
# 2x - y + z = 0
# 2x + y = 0
```

#2D in 3D example

```
import numpy as np
import matplotlib.pyplot as plt

x= np.random.random(100)
y= np.random.random(100)
z= np.sin(3*x**2+y**2)

fig= plt.figure()
ax= fig.add_subplot(111, projection= '3d')
ax.scatter(x,y,z)

ax.plot(x, z, 'r+', zdir='y', zs=1.5)
ax.plot(y, z, 'g+', zdir='x', zs=-0.5)
ax.plot(x, y, 'k+', zdir='z', zs=-1.5)

ax.set_xlim([-0.5, 1.5])
ax.set_ylim([-0.5, 1.5])
ax.set_zlim([-1.5, 1.5])

plt.show()
```

#IGNORE Ex4 and Ex5

#Ex6

```
# t     p(t)
# 1     6     ----> a0 + a1 + a2 = 6
# 2     15   ----> ??
# 3     38   ----> ??
```

#Ex7

#Try to make a linear system with two variables from the exercise content.

#IGNORE Ex8, Ex9

#Ex10

#p = Ap + d --> p - Ap = d --> (I - A)*p = d

--> $p = (I - A)^{-1} * d$

#**np.eye**(...) --> identity matrix I

```
import numpy as np
A = np.array([
    [0.25, 0.15, 0.1],
    ...,
    ...
])
d = np.array([
    [100],
    ...,
    ...
])
print("p = ", ...)
```

#Ex11

# for carbon   --> 3*x_1= x_3

# for hydrogen -->

# for oxygen   -->

Additional Exercises for Week5

Input

Create a 10×10 matrix **A** with random integers $\in [1, 99]$.

#Ex12

Save odd columns of the matrix A into a new matrix.

Hint: odd columns of the matrix are the 1st column, 3rd column, 5th column, and so on.

- For ex., input matrix = $\begin{bmatrix} 2 & 5 & 9 \\ 3 & 5 & 8 \\ 4 & 4 & 7 \end{bmatrix}$ → output = $\begin{bmatrix} 2 & 9 \\ 3 & 8 \\ 4 & 7 \end{bmatrix}$

#Ex13

Save even rows of the matrix A into a new matrix.

Hint: even rows of the matrix are the 2nd row, 4th row, 6th row, and so on.

- For ex., input matrix = $\begin{bmatrix} 2 & 5 & 9 & 1 \\ 3 & 5 & 8 & 9 \\ 4 & 4 & 7 & 1 \\ 7 & 6 & 5 & 4 \end{bmatrix}$ → output = $\begin{bmatrix} 3 & 5 & 8 & 9 \\ 7 & 6 & 5 & 4 \end{bmatrix}$

#Ex14

Save even integer numbers in the matrix A into a vector.

- For ex., input matrix = $\begin{bmatrix} 2 & 5 & 9 & 1 \\ 3 & 5 & 8 & 9 \\ 4 & 4 & 7 & 1 \\ 7 & 6 & 5 & 12 \end{bmatrix}$ → output = $\begin{bmatrix} 2 & 8 & 4 & 4 & 6 & 12 \end{bmatrix}$

#Ex15

Reverse elements in the even rows of the matrix A, and save the results into a new matrix.

- For ex., input matrix = $\begin{bmatrix} 2 & 5 & 9 & 1 \\ \mathbf{3} & \mathbf{5} & \mathbf{7} & \mathbf{9} \\ 4 & 4 & 7 & 1 \\ \mathbf{6} & \mathbf{2} & \mathbf{10} & \mathbf{14} \end{bmatrix}$ → output = $\begin{bmatrix} 2 & 5 & 9 & 1 \\ \mathbf{9} & \mathbf{7} & \mathbf{5} & \mathbf{3} \\ 4 & 4 & 7 & 1 \\ \mathbf{14} & \mathbf{10} & \mathbf{2} & \mathbf{6} \end{bmatrix}$

#Ex16

Replace odd integer numbers in the matrix A by the maximum odd integer number of A.

- For ex., input matrix = $\begin{bmatrix} 2 & 5 & \mathbf{9} \\ 3 & 5 & 8 \\ 4 & 4 & 7 \end{bmatrix}$ → output = $\begin{bmatrix} 2 & 9 & 9 \\ 9 & 9 & 8 \\ 4 & 4 & 9 \end{bmatrix}$

#Ex17

Swap the first column and the last column of the matrix A, and save the results into a new matrix.

Note: it is different with matrix flipping.

- For ex., input matrix = $\begin{bmatrix} 2 & 5 & 9 \\ 3 & 5 & 8 \\ 4 & 4 & 7 \end{bmatrix}$ → output = $\begin{bmatrix} 9 & 5 & 2 \\ 8 & 5 & 3 \\ 7 & 4 & 4 \end{bmatrix}$

#Ex18

Calculate sum of all Boundary elements of the matrix A, and save the result into a variable.

Hint:

- Boundary elements are those elements that are not surrounded by elements in all four directions, i.e. elements in the first row, first column, last row, and last column.
- To avoid duplicated calculation, each element must be used only one time in the sum calculation.

- For ex., input matrix = $\begin{bmatrix} 2 & 5 & 9 \\ 3 & 5 & 8 \\ 4 & 4 & 7 \end{bmatrix}$ → output = $2+5+9+8+7+4+4+3$

#Ex19

Calculate sum of all major and minor Diagonal elements of the matrix A, and save the result into a variable.

Hint:

- Major diagonal elements are the ones that occur from TOP LEFT of matrix down to BOTTOM RIGHT corner. Minor diagonal elements are the ones that occur from TOP RIGHT of matrix down to BOTTOM LEFT corner.
- To avoid duplicated calculation, each element must be used only one time in the sum calculation.
- For ex., input matrix = $\begin{bmatrix} 2 & 5 & 9 \\ 3 & 5 & 8 \\ 4 & 4 & 7 \end{bmatrix}$ → output = $2+5+7+9+4$

#Ex20

Given a table containing the values of the coordinate points of the graph $(x) = c_1 + c_2x + c_3x^2 + c_4x^3$.

x	1	2	3	4
$f(x)$	1000	2000	3000	4000

Find the coefficients c_1, c_2, c_3, c_4.

#Ex21

Try interesting examples of plotting vector fields in the following link:
https://www.geeksforgeeks.org/how-to-plot-a-simple-vector-field-in-matplotlib/

Vector fields represent **fluid flow** (among many other things). You can think of a vector field as representing **a multivariable function** whose input and output spaces each have the same dimension.
Read more about vector fields:

- https://www.khanacademy.org/math/multivariable-calculus/thinking-about-multivariable-function/ways-to-represent-multivariable-functions/a/multivariable-functions
- https://www.khanacademy.org/math/multivariable-calculus/thinking-about-multivariable-function/ways-to-represent-multivariable-functions/a/vector-fields

END

Feel tired, relax here:

3D vector field - https://www.youtube.com/watch?v=xi0P5JZgUHo

Fluid flow and vector fields - https://www.youtube.com/watch?v=VJ2ZDLQk3IQ

3d vector fields, introduction - https://www.youtube.com/watch?v=Go3UvzbsSGo

THE LINEAR EQUATIONS SONG - https://www.youtube.com/watch?v=2OpuD67v8QY

a simple equation - https://www.youtube.com/watch?v=tTJuOgmVmOY