TP 4
Graph plot with Matplotlib

Goal : Learn to plot different types of graph with the library matplotlib

1 First plot

To plot figures with Python we use **matplotlib** and **matplotlib.pyplot**

```
import matplotlib.pyplot as plt # Import the module and give him a short name
import numpy as np # Idem for numpy

N=50 # Number of points
start = -2
stop = 2
step = (stop - start) / N
X = np.arange(-2, 2, step)
Y = np.power(X, 2)
plt.plot(X, Y)
```

1. Using the previous example, plot on the same figure the fonction *cosinus* and *sinus*;
2. Fix the size of the plot windows to $[-\pi, \pi]$ for X axis and $[-1, 1]$ for Y axis;
3. Change the color of the two curves and add different marker for each of the curve;
4. Add a legend for each of the curve and add the grid;
5. Add a title on the figure. The title should have the number of plotting points as the interval of the X axis
6. Create a function **plot_sinus_cosinus** which takes as argument, the number of points N, the starting point *start* and the end point *stop* and plot the curve on a new figure;
7. Using the function **plot_sinus_cosinus** and a for loop, make different plots with various number of points, and then start and stop point. We can choose for instance $N = [5, 20, 50, 100]$.

Help : plt.plot, plt.title, plt.legend, plt.xlim, plt.ylim, color, label, marker

2 Parametric curve

- Define a function **plot circle(x, y, r, N)** which will plot the circle of center (x, y) , radius r and N points. To see a circle and not an ellipse, you should use the command `plt.axis('equal')`. We remind that the parametric expression of a circle is :

$$\begin{cases} X = x + r * \cos(\theta) \\ Y = y + r * \sin(\theta) \\ \theta = [-\pi, \pi] \end{cases}$$

- Use the previous function and a for loop to create the three following plots on figure 1.

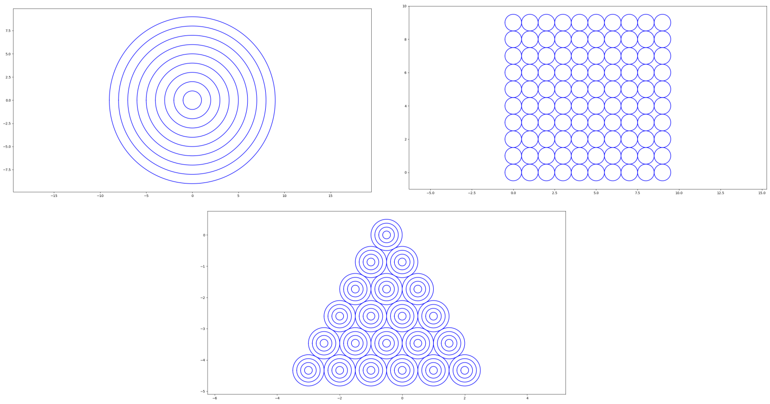

Figure 1: Parametric curve to plot

3 Polar curve

- A polar curve is a curve defined by :

$$\begin{cases} X = x + r(\theta) * \cos(\theta) \\ Y = y + r(\theta) * \sin(\theta) \\ \theta = [-\pi, \pi] \end{cases}$$

  The circle is a special case of polar curve where $r(\theta) = 1$.

  Create a function **polar_curve(r, N=1000)** which take as argument a function r and a number of point N, and plot the curve polar.

- For each of the following functions, plot the polar curve. Change the different parameters and see what they control :
  - $f(\theta) = \cos(\cos(a * \theta))$
  - $g(\theta) = (\cos(a * \theta))^2$
  - $h(\theta) = k$
  - $i(\theta) = \sin(a * x)$
  - $j(\theta) = \cos(a * x) + k$
  - $k(\theta) = \frac{p}{1+e*\cos(\theta)}$

- Using some of the previous functions and good values to parameter try to reproduce the figure 2.

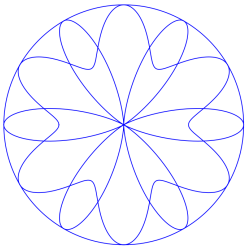

Figure 2: Polar curve

4 2D Images

To plot a 2D images, we will use the function **plt.imshow()** which takes as input a 2D numpy matrix. The goal of the exercice is to complete the following code to obtain the graph represented on figure 3. You will have to use the following functions :

- `ax.set_xticks()`
- `ax.set_yticks()`
- `ax.set_xtickslabels()`
- `ax.set_ytickslabels()`
- `ax.text()`
- `fig.colorbar()`
- `ax.set_title()`

```
school_subjects = ['Maths', 'Physique', 'Sport',
                   'Informatique', 'Musique', 'Histoire']
students = ['John', 'Sarah', 'Tim', 'Lea', 'Hector', 'Natacha']


grades = np.random.randint(0, 21, size=(len(students), len(school_subjects)))

fig, ax = plt.subplots()
im = ax.imshow(grades)

pass
```

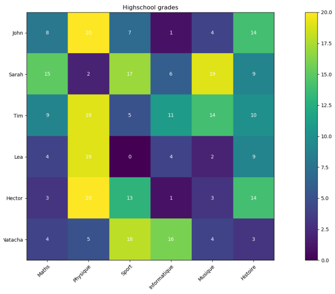

Figure 3: 2D Figure

5 3D curves

Execute the following code to plot a 3D surface. Change the function Z and try new functions.

```
N = 1000
x = np.linspace(-1, 1, N)
y = np.linspace(-2, 2, N)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.pi*np.sqrt(X**2 + Y**2))

fig = plt.figure()
ax = fig.gca(projection='3d')

ax.plot_surface(X=X, Y=Y, Z=Z)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)

plt.show()
```

6 And more ....

You can do a lot more with matplotlib : animations (movies), histogram,