Beautiful pictures

Goals:

The goal of this lab assignment is to learn how to use the scientific computing tools in Python. If you have any question about the syntax or how to write a part of your code, please use the “memo”, the Python documentation or any other resource. For this course and its lab assignments, you will use the PyCharm Edu Integrated Development Environment (IDE). Thus, your first step here is to create a new project named `SIP_LAB7` in PyCharm Edu.

1 Introduction to Visualization

This first part will guide you through the basics of `matplotlib`, a Python library that is used to display graphs and images.

```python
from matplotlib import pyplot as plt
```

The name `plt` refers to an object that allows the visualization of graphs. In order to plot a graph, the function `plt.show()` is used, as follows.

```python
plt.show() # Won't do anything since there is nothing to plot
```

Of course, the previous code won’t display anything because we didn’t specify anything to plot, which can be done with the following code:

```python
plt.plot([1,2,3,4],[1,4,9,16])
plt.show() # Don't forget it to display your graph !
```

Question 1 Arguments of plt.plot()

What happens if you provide only one array ? How would you draw more lines ?
Looking at the graph in Figure 1, could you try to understand what it actually represents?

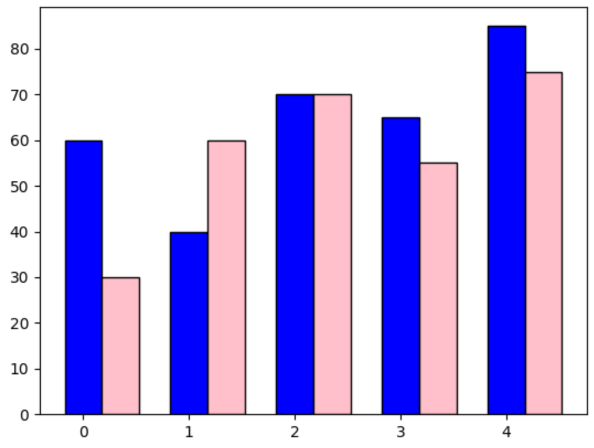


Figure 1: What does this graph represent?

It is virtually impossible to guess the meaning of the graph without additional information, such as a title, a legend explaining the meaning of the values on the x- and y-axis and the colours.

This information can be easily added to a graph in `matplotlib`, as the following example shows.

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

city=['Delhi','Beijing','Washington','Tokyo','Moscow']
Gender=['Male','Female']
pos = np.arange(len(city))
bar_width = 0.35
Happiness_Index_Male=[60,40,70,65,85]
Happiness_Index_Female=[30,60,70,55,75]

plt.bar(pos,Happiness_Index_Male,bar_width,color='blue',edgecolor='black')
plt.bar(pos+bar_width,Happiness_Index_Female,bar_width,color='pink',
        edgecolor='black')
plt.xticks(pos, city)
plt.xlabel('City', fontsize=16)
plt.ylabel('Happiness_Index', fontsize=16)
plt.title('Group Barchart - Happiness index across cities By Gender',fontsize=18)
plt.legend(Gender,loc=2)
plt.show()
```

In order to enrich a graph with useful information, here are few useful functions:

- `plt.title()`. Adds a title to the graph. You can look at the documentation for more information.
- `plt.xlabel()`. Adds a label to the x-axis. You can look at the documentation for more information.
- `plt.ylabel()`. Adds a label to the y-axis. You can look at the documentation for more information.
- `plt.legend()`. Adds a legend to the graph. `loc=2` means that the legend will be placed in the upper left corner of the graph. You can look at the documentation for more information.
- `plt.xticks()`. Gets or sets the tick locations and labels on the x-axis. You can look at the documentation for more information.

An introduction to `matplotlib` can be found here.
A description of the types of plots, with the relative code samples, can be found here.

1.1 Visualizing Functions

In order to plot mathematical functions with `matplotlib`, we can use the function `linspace()` provided by the package `numpy` to generate an array X of values for the x-axis. The advantage is that we can apply to X any mathematical function (e.g., `sin`, `log10`) provided by the package `numpy`.

Here is an example to plot a parabola.

```python
def center_axes(fig):
    ax = plt.figure().add_subplot(1, 1, 1)

    # Move left y-axis and bottim x-axis to centre, passing through (0,0)
    ax.spines['left'].set_position('center')
    ax.spines['bottom'].set_position('center')

    # Eliminate upper and right axes
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')

    # Show ticks in the left and lower axes only
    ax.xaxis.set_ticks_position('bottom')
    ax.yaxis.set_ticks_position('left')

def plot_parabola():
    x = np.linspace(-2,2,200) # 200 linearly spaced numbers
    y = x**2
    center_axes(plt)
    plt.plot(x,y)
    plt.show()

plot_parabola()
```

The function `center_axis()` is used to center the y-axis on the origin.

Question 2 Visualizing Sines

Based on the notions learned so far, write a function `plot_sines()` that plots the functions $f(x) = \sin(\pi \cdot x)$ and $g(x) = \sin(2\pi \cdot x)$ on an interval $[-2; +2]$ with 200 linearly spaced numbers.

Question 3 Visualizing multiple figures

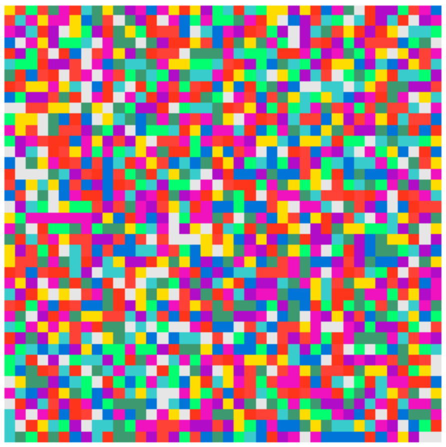
Looking at the example in the documentation, plot the two previous functions in two separate figure.
Hint. Look at the documentation to understand the meaning of the parameters of the function `subplot()`.

Question 4 Changing style

Looking at the documentation of the function `plot()`, change the style of the two curves in such a way that one uses a red "triangle down" (v) marker and the other a blue 'plus" (+) marker.

2 Displaying digits of Pi

We try to see the randomness of Pi decimals by displaying them with colors.



To do this, we will use `matplotlib.pyplot.imshow(M)` where `M` is a color coded RGB matrix.

Question 5 A plain red square

Display a deep red (128,29,58) square of size 40 by 40.

Question 6 Fractions: repeating decimal

Create a color scheme for each of the digits (a list of RBG colors). Then, display the colors of a fraction (so it has a repeating pattern) of your choice.

Question 7 Displaying π

Find the first N digits of Pi and display them. What happens if you use `math.pi`? Why?

To go further: display $e, \phi = \frac{1+\sqrt{5}}{2}$. Using the Bailey–Borwein–Plouffe formula, display the digits of π in base 16 by computing them on the fly.

3 The Koch Snowflake

A fractal is a geometric figure that reproduces a specific pattern indefinitely. There are many known fractals, namely Sierpinski's Triangle, Koch's snowflake and Mandelbrot's set. We will be working on this last one in this part.

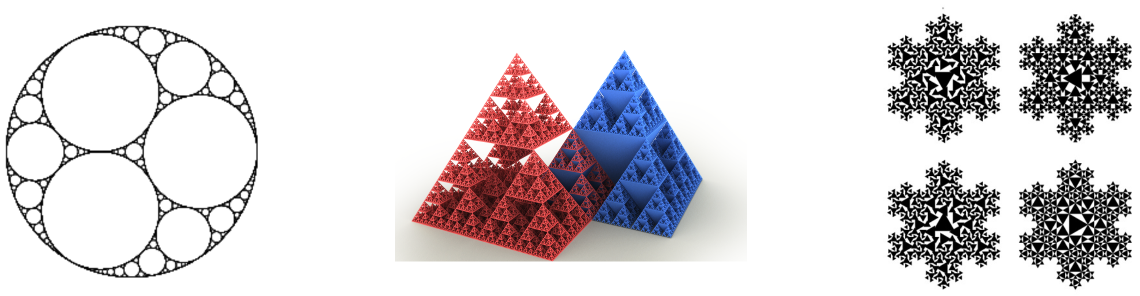
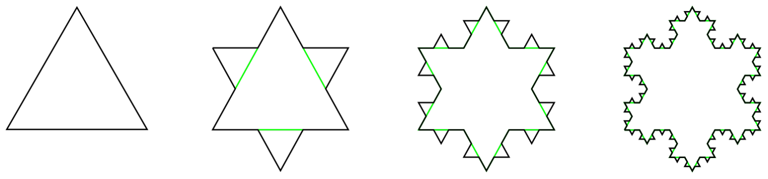


Figure 2: Examples of fractals (Apollonian Gasket D_3, Sierpinski Pyramid and Koch snowflake)

The Koch snowflake is a fractal that can be easily constructed recursively. Start with an equilateral triangle, then modify each segment as follows:

1. divide the segment into three segments of equal length.
2. draw an equilateral triangle that has the middle segment from step 1 as its base and points outward.
3. remove the line segment that is the base of the triangle from step 2.

The figure below illustrates the first iterations of the fractal.



To draw it, we will be using `matplotlib.pyplot.plot(xdata, ydata)` which will plot the lines connecting the points `(xdata[i], ydata[i])`

Question 8 Initialization

Draw the initial equilateral triangle.

Question 9 Modifying a segment

Create a function that, given a segment, modify it as described in steps 1-3. If the segment is $\vec{AB}$, then the new line is $ACDEB$ with:

$$\begin{cases} \vec{AC} = \vec{AB}/3 \text{ ; } \vec{AE} = 2\vec{AB}/3 \\ \vec{AD} = \vec{AB}/2 + \sqrt{3}\vec{AB}^{\perp}/6, \text{ with } \vec{AB}^{\perp} \text{ a vector of norm } \|\vec{AB}\| \text{ perpendicular to } \vec{AB} \end{cases}$$

Question 10 The whole iteration

Create a function that modifies each segment and do the whole iteration. You can modify the previous function so that it works on a `(xdata, ydata)` at a given position.

Question 11 Wrapping up

Finally, draw the 6th iteration of the Koch snowflake.