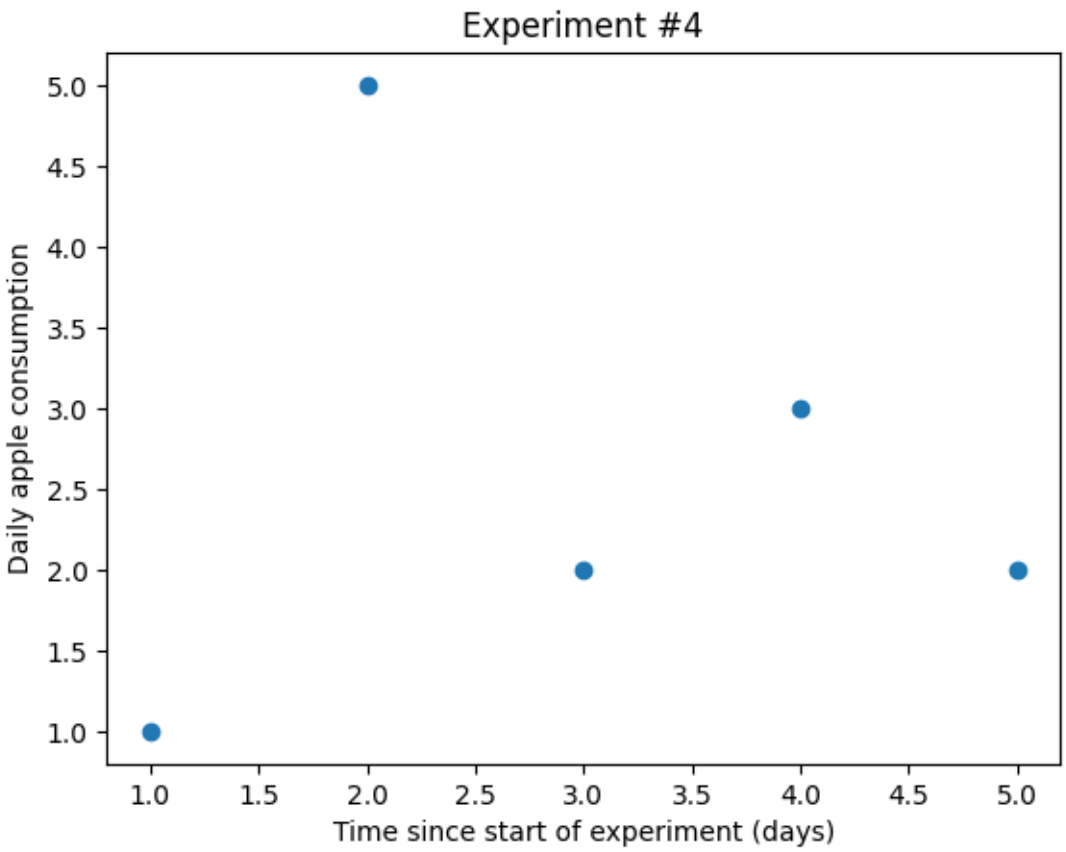
PythonNotebook4_solution_2023

December 6, 2023

0.0.1 Exercise 4.1.1

What if your data does not represent a curve, but are just points in x-y space? For example in a scatter-plot. Plot the data from above, using the function `plt.scatter()`.

[1]:
```
import matplotlib.pyplot as plt
import math

x = [1, 2, 3, 4, 5]
y = [1, 5, 2, 3, 2]

plt.scatter(x, y)
plt.xlabel('Time since start of experiment (days)')
plt.ylabel('Daily apple consumption')
plt.title('Experiment #4')
plt.show()
```



0.0.2 Exercise 4.2.1

- Create plots of the three temperature time series. Just do three plt.plot() calls after each other. Like this: `plt.plot(month, T_NL)`

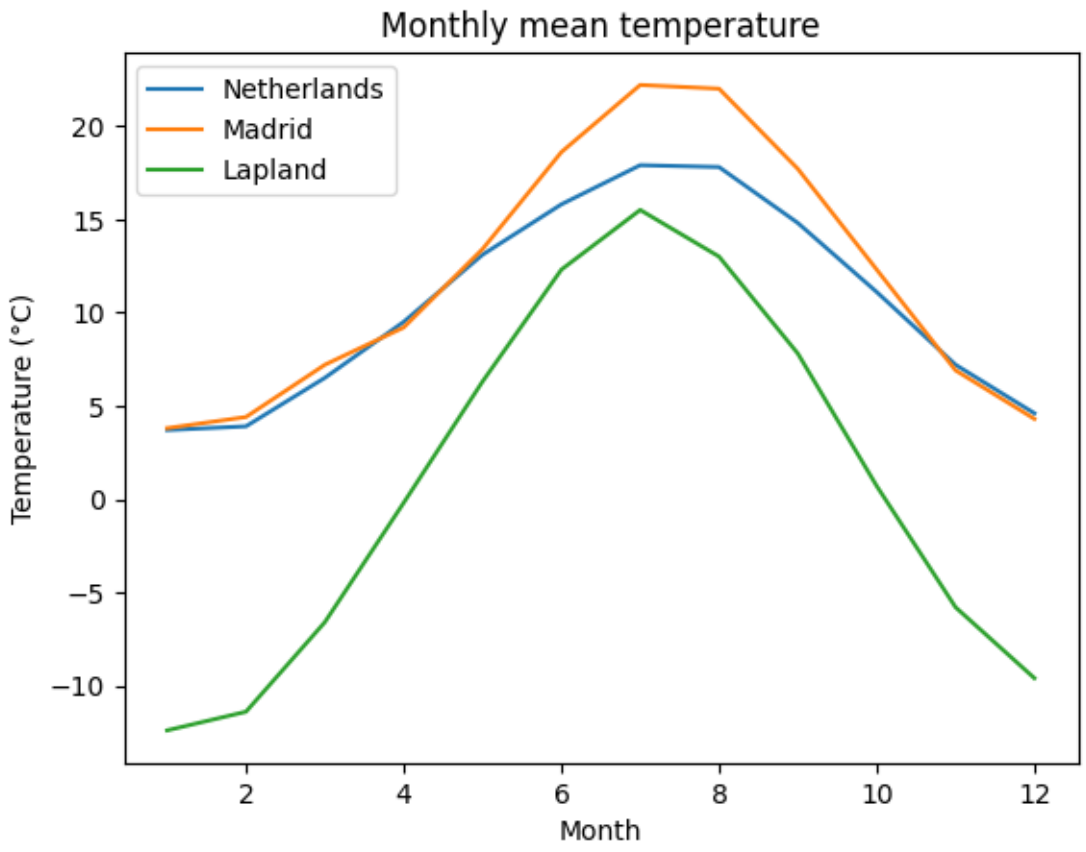
You see that Matplotlib selects different colors for the three curves. But which is which? You need a legend.

- add a label argument to each plot call: `plt.plot(month, T_NL, label='Madrid')`. Then add the command `plt.legend()` at the end of your script to create a legend which shows which colors and labels belong together.
- add axis labels and units.

[2]:
```
month=range(1,13)
T_NL      = [  3.7,   3.9,  6.5,  9.5, 13.1, 15.8, 17.9, 17.8, 14.8, 11.1,  7.
 ↪2,  4.6]
T_Madrid  = [  3.8,   4.4,  7.2,  9.2, 13.4, 18.6, 22.2, 22.0, 17.7, 12.3,  6.
 ↪9,  4.3]
T_Lapland = [-12.4, -11.4, -6.6, -0.2,  6.3, 12.3, 15.5, 13.0,  7.8,  0.7, -5.
 ↪8, -9.6] # Sodankylä, Finland
```

```
# monthly mean temperature, °C, 1990 - 2019
# Data from https://climatecharts.net/

plt.plot(month, T_NL, label='Netherlands')
plt.plot(month, T_Madrid, label='Madrid')
plt.plot(month, T_Lapland, label='Lapland')

plt.xlabel('Month')
plt.ylabel('Temperature (°C)')
plt.title('Monthly mean temperature')
plt.legend()
```

[2]: <matplotlib.legend.Legend at 0x1b5248d5580>

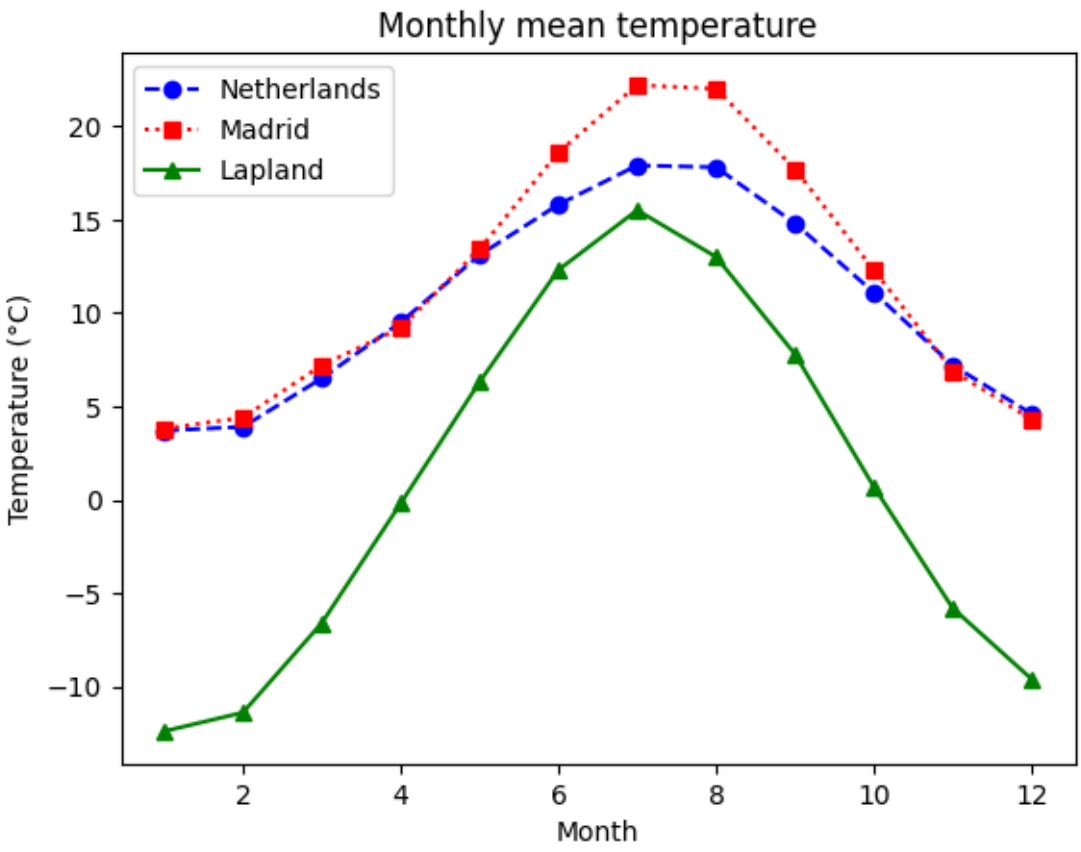


0.0.3 Exercise 4.2.2 Plotting in style

In the Matplotlib documentation (see above), find out how you can set the *line style* (solid line, dashed line, dotted line, …) and *line color* in a plot. Copy thetemperature plot above, and choose line styles and line colors for the curves.

[3]:
```
month=range(1,13)
T_NL      = [  3.7,   3.9,  6.5,  9.5, 13.1, 15.8, 17.9, 17.8, 14.8, 11.1,  7.
 ↪2,  4.6]
T_Madrid  = [  3.8,   4.4,  7.2,  9.2, 13.4, 18.6, 22.2, 22.0, 17.7, 12.3,  6.
 ↪9,  4.3]
T_Lapland = [-12.4, -11.4, -6.6, -0.2,  6.3, 12.3, 15.5, 13.0,  7.8,  0.7, -5.
 ↪8, -9.6] # Sodankylä, Finland
# monthly mean temperature, °C, 1990 - 2019
# Data from https://climatecharts.net/

plt.plot(month, T_NL, label='Netherlands', linestyle='--', marker='o',␣
 ↪color='blue')
plt.plot(month, T_Madrid, label='Madrid', linestyle=':', marker='s',␣
 ↪color='red')
plt.plot(month, T_Lapland, label='Lapland', linestyle='-', marker='^',␣
 ↪color='green')

plt.xlabel('Month')
plt.ylabel('Temperature (°C)')
plt.title('Monthly mean temperature')
plt.legend()
```

[3]:
```
<matplotlib.legend.Legend at 0x1b544d08a30>
```



0.1 Exercise 4.3.1

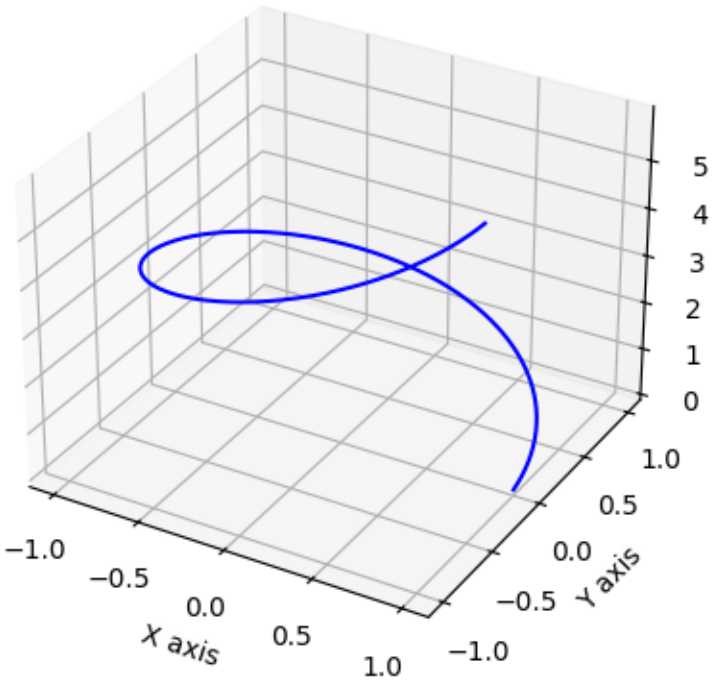
Copy the parametric curve example above, modify it to draw something else. Color the curve in your favourite color. Add labels to the x, y, and z axes.

Label hint: unfortunately there is no function `plot.zlabel()`. Instead you have to use `ax.set_zlabel('...')`. You can use `ax.set_xlabel()` and `ax.set_ylabel()` too. Your script looks nicer if you handle all axes in the same way.

Note: the z-label doesn't necessary show up anyway, perhaps it's hidden behind the plot. If you could rotate the plot it might show up. For now let's choose our battles and not worry about this.

[4]:
```
ax = plt.axes(projection='3d')

x, y, z = [], [], []
for i in range(600):
    t = i / 100
    x.append(math.cos(t))
    y.append(math.sin(t))
    z.append(t)
```

```
ax.plot3D(x, y, z, color='blue')

ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')

plt.show()
```



0.2 Exercise 4.4.1 Bar graph

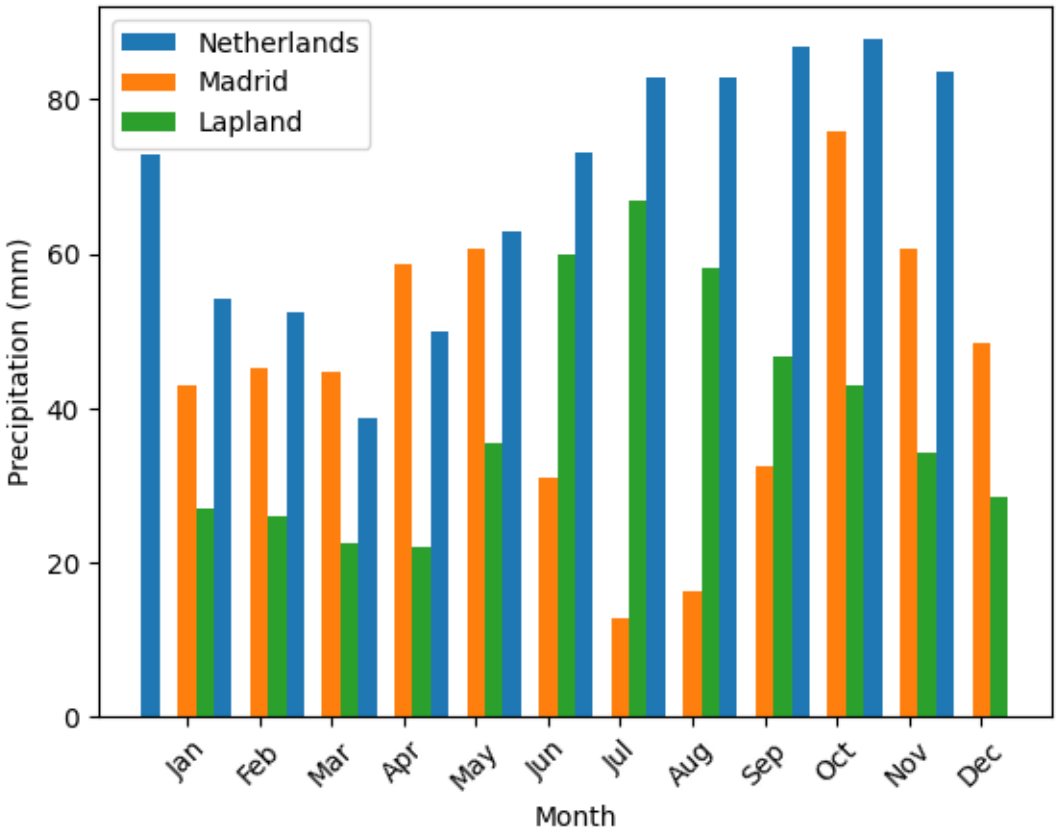
In the Matplotlib documentation, find out how to make bar graphs. Plot the monthly mean rainfall data below in a bar graph. Add labels on the axes.

Hint: you can plot the different locations one by one, one plot command each.

For now, let's accept that the bars sometimes overlap and hide each other. (One can in principle shift the bars in x so that they don't overlap).

[5]:
```
month = range(1, 13)
Pr_NL = [72.8, 54.1, 52.5, 38.7, 50.0, 63.0, 73.1, 82.9, 82.9, 86.8, 87.7, 83.
↪6]  # Netherlands
Pr_Madrid = [43.0, 45.1, 44.8, 58.6, 60.7, 31.0, 12.7, 16.3, 32.5, 75.8, 60.7,␣
↪48.5]  # Spain, around Madrid
```

```
Pr_Lapland = [27.0, 25.9, 22.5, 22.1, 35.6, 60.0, 66.9, 58.1, 46.6, 42.9, 34.2,␣
 ↪28.4]  # Finland, Lapland, around Sodankylä

# Width of a bar
width = 0.25

# Calculating the position of bars
month_NL = [x - width for x in month] # <= option 1 with list comprehension

for i in range(len(month)):
    month_NL[i] = month_NL[i] - width # <= option 2 with a for loop

month_Madrid = month
month_Lapland = [x + width for x in month]

plt.bar(month_NL, Pr_NL, width=width, label='Netherlands')
plt.bar(month_Madrid, Pr_Madrid, width=width, label='Madrid')
plt.bar(month_Lapland, Pr_Lapland, width=width, label='Lapland')

plt.xlabel('Month')
plt.ylabel('Precipitation (mm)')

# Month names and setting x-ticks
month_names = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',␣
 ↪'Oct', 'Nov', 'Dec']
plt.xticks(month, month_names, rotation=45)  # Rotate labels by 45 degrees

plt.legend()
plt.show()
```


Netherlands
Madrid
Lapland
Precipitation (mm)
0
20
40
60
80
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month