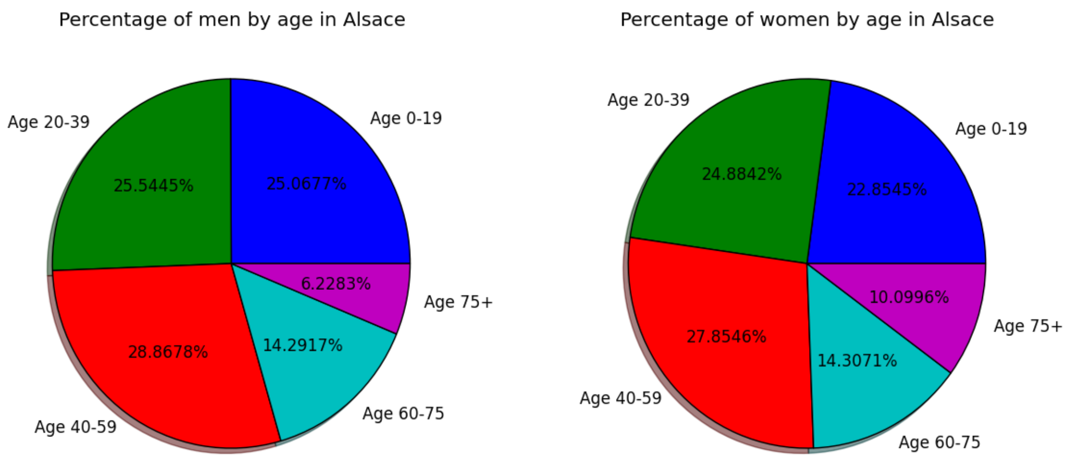
Introduction to programming, Lesson 6: matplotlib

matplotlib is probably the single most used Python package for 2D-graphics. It provides both a very quick way to visualize data from Python and publication-quality figures in many formats.

1. Let us start with an example of a pie chart drown with pyplot.

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

plt.figure(figsize = (6,6))
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
fracs = np.array([15, 30, 45, 10])
plt.pie(fracs,  labels = labels, autopct = '%1.1f%%', shadow = True)
plt.title('Raining Hogs and Dogs')
plt.savefig('raining.png')
plt.show()
```

2. **Pyramid of ages:** Draw a pie chart of female and male age groups in Alsace.

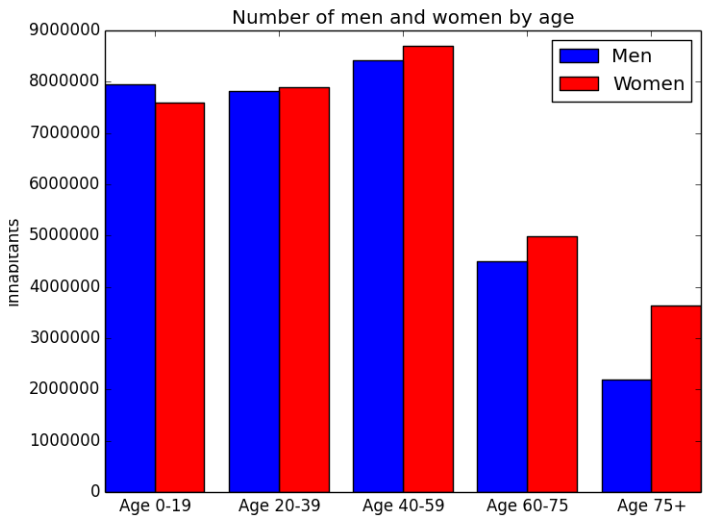


3. Here is an example of a bar chart for programming language usage.

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

languages = np.array(['Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp'])
number_of_users = np.array([10, 8, 6, 4, 2, 1])

plt.title('Programming language usage')
x_pos = np.linspace(1, len(languages), len(languages)) #Return a numpy array [1, 2, …, len(languages)]
bc = plt.bar(x_pos - 0.4, number_of_users, color = 'b')
plt.xticks(x_pos, languages) #label the ticks on the x-axis
plt.ylabel('Usage')

plt.savefig('programming_languages.png')
plt.show()
```

Introduction to programming, Lesson 6: matplotlib

4. **Pyramid of ages:** Your task is to draw a bar chart of male and female populations by ages. To add a legend to your graph, use plt.legend((bc_men[0], bc_women[0]), ('Men', 'Women')).



5. Here is an example of drawing the graph of a line.

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

'''Each graph consists of a collection of (x, y) points.
The array X contains the x-coordinates of the points,
and the array y contains the y-coordinates of the points
(in the same order).'''

X = np.linspace(0, 10, 10)
Y = np.linspace(20, 40, 10)
plt.plot(X, Y, color = "green", linewidth = 1.0, linestyle = "-", label = "line")

plt.xlim(0, 10) #set x limits
plt.xticks(np.linspace(0, 10, 6, endpoint = True)) #set x ticks
plt.xlabel('x') #label of the x-axis

plt.ylim(15, 45) #set y limits
plt.yticks(np.linspace(15, 45, 7, endpoint = True)) #set y ticks
plt.ylabel('y')

plt.legend(loc = 'upper left', frameon = False) #label of the graph

plt.savefig('graph.png')
plt.show()
```

6. **Carbon dioxide levels:** Mauna Loa Observatory (MLO) is a premier atmospheric research facility that has been continuously monitoring and collecting data related to atmospheric change since the 1950's. Our task will be to draw a graph of average carbon dioxide levels from 1951 to 2017.
   - Download co2-annmean-mlo.csv
   - Create a function that parses this data into two numpy arrays, years and co2_level

- Draw a graph following Exercise 5.

7. **Animation: rain drops**
NB: If you want an extra challenge, try to come up with a solution on your own. Otherwise, follow the steps below and try to figure out what each of them does. Use print to understand how the arrays are defined.

A very simple rain effect can be obtained by having small growing rings randomly positioned over a figure. Of course, they won't grow forever since the wave is supposed to damp with time. To simulate that, we can use a more and more transparent color as the ring is growing, up to the point where it is no more visible. At this point, we remove the ring and create a new one.

First, let us draw a static picture. We start by creating a blank figure. Next, we need to create several rings. For this, we can use the scatter plot object that is generally used to visualize points cloud, but we can also use it to draw rings by specifying we don't have a facecolor (i.e., the rings are not filled). We have also to take care of initial size and color for each ring such that we have all sizes between a minimum and a maximum size and also to make sure the largest ring is almost transparent.

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

fig = plt.figure(figsize = (6,6), facecolor = 'white') #new figure with white background
ax = fig.add_axes([0, 0, 1, 1], frameon = False, aspect = 1) #new axis over the whole figure, no frame and a
1:1 aspect ratio

n = 50 #number of rings
size_min = 50
size_max = 50 * 50

P = np.random.uniform(0,1,(n,2)) #ring position
C = np.ones((n,4)) * (0,0,0,1) #ring colors
C[:,3] = np.linspace(0,1,n) #alpha color channel goes from 0 (transparent) to 1 (opaque)

S = np.linspace(size_min, size_max, n) #ring sizes

#scatter plot
scat = ax.scatter(P[:, 0], P[:, 1], s = S, lw = 0.5, edgecolors = C, facecolors = 'None')

#ensure limits are [0,1] and remove ticks
ax.set_xlim(0,1), ax.set_xticks([])
ax.set_ylim(0,1), ax.set_yticks([])

plt.show()
```

Now, we need to write the update function for our animation. We know that at each time step each ring should grow to be more transparent while largest ring should be totally transparent and thus removed. Of course, we won't actually remove the largest ring but re-use it to set a new ring at a new random position, with nominal size and color. Hence, we keep the number of rings constant.

```
def update(frame):
    global P, C, S

    #every ring is made more transparent
    C[:,3] = np.maximum(0, C[:,3] - 1.0/n)

    #each ring is made larger
    S += (size_max - size_min) / n

    #reset ring specific ring (relative to frame number)
    i = frame % 50
    P[i] = np.random.uniform(0,1,2)
    S[i] = size_min
    C[i,3] = 1

    #update scatter object
    scat.set_edgecolors(C)
    scat.set_offsets(P)

    #return the modified object
    return scat
```

Last step is to tell matplotlib to use this function as an update function for the animation and display the result or save it as a movie:

```
animation = an.FuncAnimation(fig, update, interval=10, blit=False, frames=200)
animation.save('rain.gif', writer='imagemagick', fps=30, dpi=40)
plt.show()
```