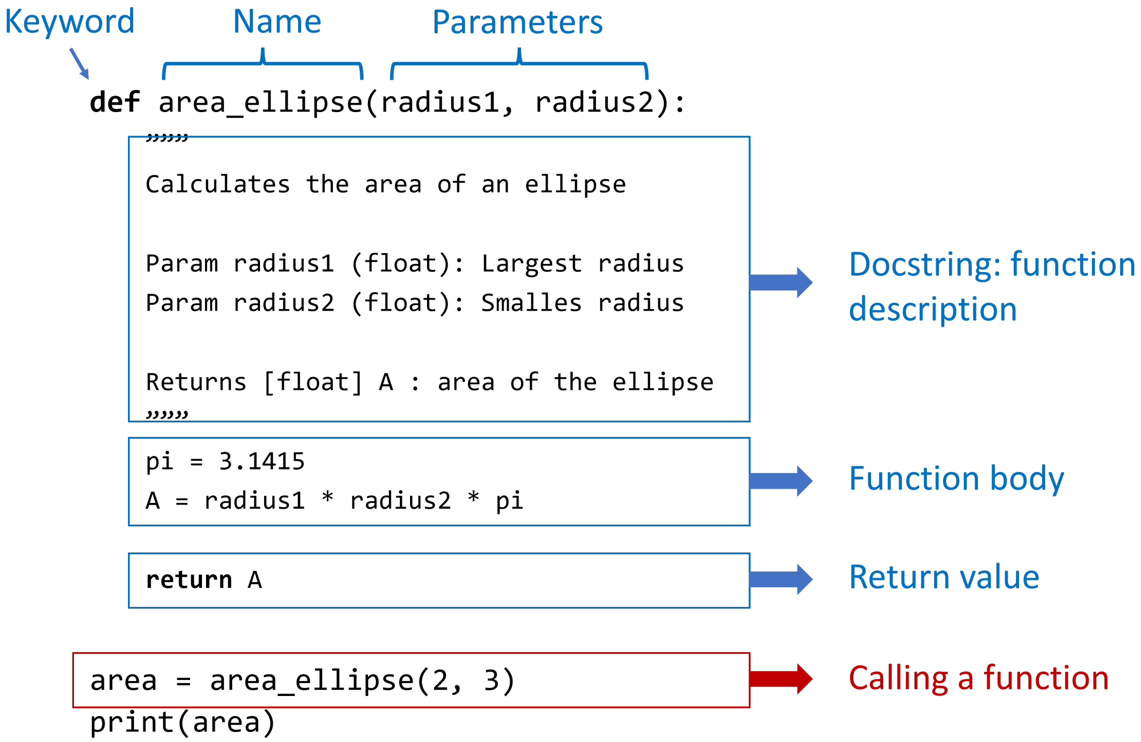
PHOT 110: Introduction to programming

Lecture 10: supporting materials

Michaël Barbier, Spring semester (2023-2024)

Exercises on functions

We will exercise creating functions and use them within problems similar to the ones we encountered before during recitation. Remember the different elements of the function definition:



```
def area_ellipse(radius1, radius2):
    ”””
    Calculates the area of an ellipse

    Param radius1 (float): Largest radius
    Param radius2 (float): Smalles radius

    Returns [float] A : area of the ellipse
    ”””
    pi = 3.1415
    A = radius1 * radius2 * pi

    return A

area = area_ellipse(2, 3)
print(area)
```

Figure 1: Function definition elements

At the beginning of the script you should first import the required libraries (numpy and matplotlib):

```
# Import numpy and matplotlib
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use("WebAgg")
```

Exercise 1: Calculate the factorial of a number

Calculate the value of the factorial of a number by defining a new function: `factorial(n)`. Use the following script as basis:

```
# input parameter
n = 5

# calculate the factorial
f = 1
for i in range(2, n+1):
  f = i * f

# print the factorial
print(f"The value of {n}! = {f}")
```

```
The value of 5! = 120
```

Exercise 2: Calculate the number of combinations

The number of combinations of k items out of a set of n objects is defined in statistics as

$$\mathcal{C}_k^n = \binom{n}{k} = \frac{n!}{(n-k)!\,k!}$$

where $k \leq n$.

- Use the function `factorial(n)` which you created in previous exercise 1 to calculate the number of combinations $\mathcal{C}_3^5$.
- Afterwards create a function `combinations(n, k)` to compute the combinations.

Exercise 3: Intersection of two lines

Find the intersection point $p = (x_p, y_p)$ between two lines with equations

$$\begin{cases} y = m_1\, x + c_1 \\ y = m_2\, x + c_2 \end{cases}$$

To find the intersection we extract the x-value by making use of the fact that at the intersection the y-values should be equal.

$$\begin{aligned} m_2\, x_p + c_2 &= m_1\, x_p + c_1 \\ \Rightarrow (m_2 - m_1)\, x_p &= c_1 - c_2 \\ \Rightarrow x_p &= -\frac{c_2 - c_1}{m_2 - m_1} \end{aligned}$$

then we substitute the found x_p coordinate into one of the equations of the system to obtain the y_p coordinate:

$$y_p = m_1\, x_p + c_1$$

As example parameters of the lines: pick $m_1 = 1/5$ and $m_2 = 7$ as the direction coefficients, and $c_1 = 2$ and $c_2 = -3$ the off-sets at $x = 0$.

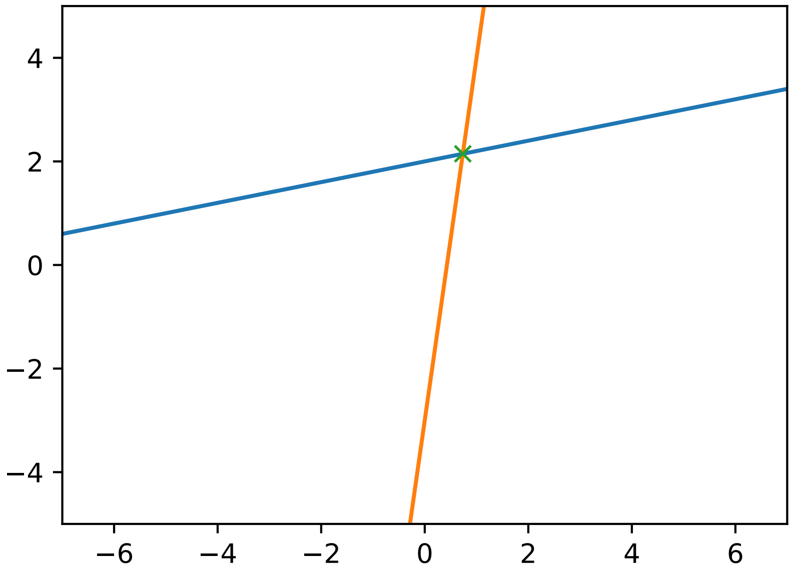
Convert the code to calculate the intersection point in following script into a function: `calc_intersection(m1, c1, m2, c2)` which returns `xp, yp`. Afterwards use your new function within this script.

```
# Parameters of the lines
m1 = 0.2; c1 = 2
m2 = 7; c2 = -3

# Calculate the intersection point
# (convert the next couple of lines into a function)
xp = -(c2 - c1) / (m2 - m1)
yp = m1 * xp + c1

# Calculate the coordinates for the lines to plot
x = np.linspace(-7, 7, 100)
y1 = m1 * x + c1
y2 = m2 * x + c2

# Plot the lines and the intersection point
fig, ax = plt.subplots()
ax.plot(x, y1)
ax.plot(x, y2)
ax.plot(xp, yp, marker="x")
ax.set_xlim([-7, 7])
ax.set_ylim([-5, 5])
ax.set_aspect("equal")
plt.show()
```



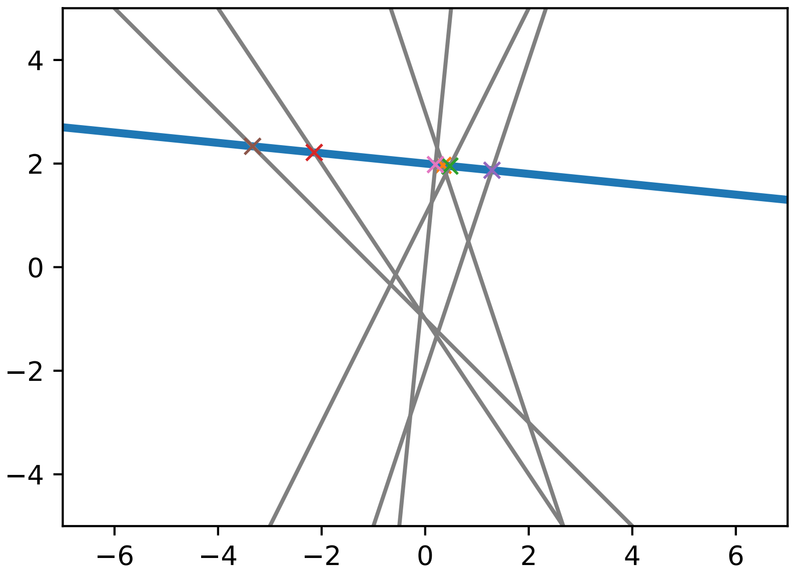
Exercise 4: Intersection of many lines

Use the function that you created in exercise 3 to calculate the intersection of a line with a list of other lines defined by:

```
# Parameters of the single line:
m1 = -0.1
c1 = 2

# Parameters of the other lines:
m_list = [-3, 2, -1.5, 3,  -1, 10];
c_list = [3,   1, -1,  -2, -1,   0]
```

The output plot should look as follows:


4
2
0
−2
−4
−6
−4
−2
0
2
4
6