Exercise I. Setting up a Python framework (20 points)

(a) For Linux/Mac users: use your favorite distribution to install a Python-3 environment. Make sure to also install NumPy, SciPy, and Matplotlib. For Debian/Ubuntu

```
sudo apt-get install python3 python3-numpy python3-scipy python3-matplotlib
```

A great way to organize your code and present results are ipython (jupyter) notebooks. On Debian/Ubuntu run

```
sudo apt-get install ipython3
```

Alternatively, you can find installation instructions at [https://jupyter.org/](https://jupyter.org/).

(b) For Windows/Mac users: Probably the best way to go is to install Anaconda,

[https://www.continuum.io/downloads/](https://www.continuum.io/downloads/)

Exercise II. Introduction and Resources (10 points)  Have a look at some useful online resources:

- [https://python.org](https://python.org)
- [https://wiki.python.org/moin/BeginnersGuide/Programmers](https://wiki.python.org/moin/BeginnersGuide/Programmers)

(How can this be worth 10 points? Write a short philosophical statement.)

Exercise III. The Mandelbrot set (70 points)  The Mandelbrot set is defined as the set of complex numbers \( c \) for which the iteration

\[
z_0 = 0, \quad z_{n+1} = z_n^2 + c,
\]

converges. It can be shown that for a convergent sequence the condition \( |z_n| \leq 2 \) has to hold true for all \( n \).

(a) Write a function
Python code

```python
def mandelbrot_iteration(c, max_iter=200):
```

that takes a single $c$ and a maximal iteration count as arguments. The function shall return 0 if $|z_n| \leq 2$ holds true for $\text{max}_\text{iter}$ iterations. Otherwise it shall return the smallest iteration number $n$ for which $|z_n| > 2$.

Test your implementation for different values of $c$.

(b) Use `mandelbrot_iteration` to compute all points on a regular grid for a given lower left corner $a$, upper right corner $b$ and “resolution” res. Visualize the following regions, given as $(a, b, \text{res})$:

$(-2 - 1.5i, 1 + 1.5i, 500),$

$(-0.7462 + 0.1103i, -0.7460 + 0.1105i, 500),$

$(-0.08825 + 0.65425i, -0.0885 + 0.6545i, 500).$

(c) How does the output behave for different values of $\text{max}_\text{iter}$? Document your findings.

Hints.

– Complex number can be written in the form $z = x + yj$, or can be created via `complex(x, y)`. The real (imaginary) part is accessed via $z.\text{real}$ ($z.\text{imag}$).

– numpy has a function `linspace` to create an array of numbers between two boundaries with a given resolution:

```python
import numpy as np
np.linspace(left, right, resolution)
```

A list comprehension is a great way of creating a two dimensional array:

```python
[[ function(i, j) for i in ... ] for j in ... ]
```

– matplotlib makes the visualization of a two dimensional array with the help of a heat map very simple:

```python
import matplotlib.pyplot as plt
plt.axis("off")
plt.imshow(array, cmap="hot")
plt.colorbar()
```

Obsessive overcompletion I. (10 bonus points)    Computer and visualize another fractal set, e.g. the Julia set.

Obsessive overcompletion II. (20 bonus points)   Tweak the code you have written in Exercise III such that computing a 1600x1600 array takes under 5 seconds. (Alternatively, write the same code in julia and brag about how much faster it is compared to native Python.)