Computer Vision 1: Exercise Sheet 1

All of the computer exercise will be done using the Linux operating system. As a first step, log into Linux using your student credentials. You will have an option to choose Linux from a menu that appears when switching on the computer.
You can use your own laptop as well, but it is your responsibility to install all required software.

1 Hello World in Python

If you already know basics of Python, you can skip ahead to Section 3.

Now you should be logged in the graphical user interface (GUI) of Linux. To get started, open a text editor such as `gedit`, for example by clicking on the top left dash button and typing the name of the editor. Write the following Python code into a new text file:

```
print('Hello world!')
```

Save the file as `hello.py` into some directory in your home folder.

1.1 Running your code: using the terminal

We will now learn how to run your code. First, open a terminal window. You can do this either by clicking on the dash button and typing in "terminal" or using the keyboard shortcut Ctrl +Alt +t (press all keys down together). On German keyboards Ctrl is indicated by Strg .

Now that the terminal window is open, let's look at some basic ways to operate on the command line. You should see something like this on the command line:

```
username@computername:[~]:
```

This indicates your user name, the computer you are logged on to, and the current directory – the tilde symbol (~) is short for your home directory.

You can run a command by typing it into the command line and pressing Enter . Try

```
ls
```

to print out a listing of the contents of your current directory.

Now, navigate to the directory where you saved your Python code. You can do this by using the `cd` shell command. If you saved the file to `~/Documents/CV1/hello.py`, you can give the two commands

```
cd Documents
cd CV1
```

or the single command `cd ~/Documents/CV1` to move the command line to the respective directory. Modify the commands above to navigate to the directory where you saved your code.

Now we are ready to run your code. Give the command

```
python hello.py
```

and observe the output. Congratulations, you have just implemented the classic "Hello world!" program in Python and successfully run it through the Linux command line.

1.2 Running your code: Python interpreter

There is another way to run code. You can type in the command

```
python
```

to launch the interactive Python interpreter. Then, you can start to write Python code one line at a time, while observing the outputs after every line. You can type in the hello world code into the interpreter and then see the output.

We will not be using the Python interpreter much in the exercises, since it is convenient to save your code to a text file to save it and to be able to run it again later. It is useful to be aware of the interpreter however if you ever want to run some quick tests in Python and you do not want to save the code for later.

You can exit the Python interpreter by giving the `quit()` command, or by the keyboard shortcut Ctrl + d .

2 Python: Tutorial

You should already know some programming language by now, and be familiar with basic concepts such as variables, data types, functions and perhaps classes. We will now learn how these basic concepts appear in Python.

Go back to the text editor, and open a new file to start working with. It is a good idea to create a new file for each separate task that you do, name the files meaningfully and organize them so that they are easy to found.

You can start to work with a Python tutorial that will gradually introduce you to the language. We refer you to the online tutorial by Justin Johnson found at: `http://cs231n.github.io/python-numpy-tutorial/` (click the link to open the URL).

Start from the section "Basic data types" and work down until the section "Numpy".

- Take the time to understand what you are doing. It's easy to just copy and paste the code, but you will not learn much this way.
- If you get stuck, you can ask a teacher or one of your fellow students for help.

If you want to get a deeper understanding, you can study the more comprehensive tutorial found at `https://docs.python.org/3.5/tutorial/index.html`.

After completing the tutorial, you should be able to solve the following small problems.

1. You are given a list

   ```
   a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
   ```

   Write one line of Python that creates a new list that has only the even elements of `a` in it. Hint: list comprehension.

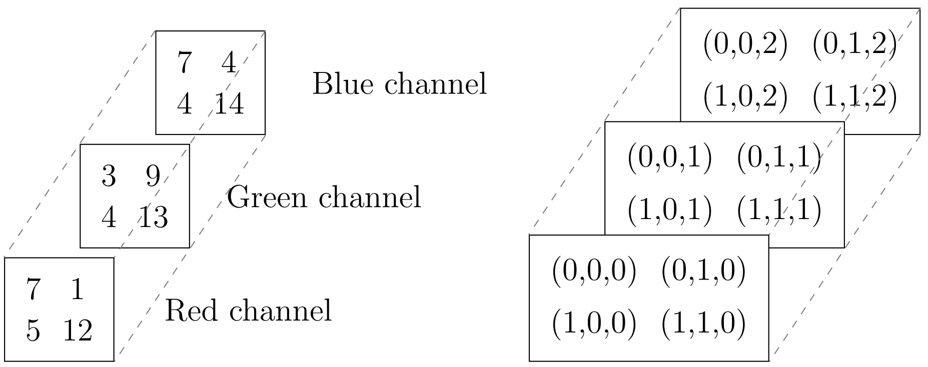
2. Write a function that takes as input a string and counts how many times each letter appears in it and outputs the data in a dictionary. For example, for `'mississippi'`, the desired output dictionary is

   ```
   {'m': 1, 'i': 4, 's': 4, 'p': 2}
   ```

   Hint: use a loop over letters in the string.

3 Python: Modules and images as arrays

A Python module is a file with definitions (e.g., of functions or classes) that can be *imported* into other modules, or into the Python main module. If you want to create a module yourself, you can write a file, let's say `mymodule.py`, and put some definitions inside it. You may then import it in another file by the command `import mymodule`.



(a) Values in a 3-dimensional array. (b) Indices in a 3-dimensional array.

Figure 1: A color image I can be represented as a 3-dimensional array such as shown in Figure 1a. Each layer of the array corresponds to a color channel such as red, green, or blue. The value $I(i, j, c)$ is indexed by (i, j, c), where i indicates the row, j the column, and c the channel. The indices are illustrated in Figure 1b. For example, $I(1, 1, 2) = 14$ means that the 2nd row, 2nd column, 3rd (blue) channel intensity is 14. Note that Python/NumPy use 0-based indexing, but when speaking we usually use 1-based indexing.

Here is an example. File `hello.py`:

```python
def say_hi():
  print('Hello World!')
```

File `mycode.py`:

```python
import hello # import the contents of the module hello
hello.say_hi() # call the function say_hi from module hello
```

You can also give a shorter alias for the module by using `import as`:

```python
import hello as hi # import module hello under alias hi
hi.say_hi() # call function say_hi from module hello, using alias hi
```

3.1 Representing images as arrays and the NumPy module

For computer vision, 2-dimensional and 3-dimensional arrays are useful for representing images[1]. A 2-dimensional array is a matrix, e.g.,

$$I = \begin{bmatrix} 10 & 0 \\ 50 & 1 \end{bmatrix}.$$

This kind of an array can represent a grayscale image I: the rows i and columns j correspond to the rows and columns of an image, and the value $I(i, j)$ of an entry is the grayscale intensity of the pixel. Usually, intensities are represented as *8-bit unsigned integers* (`uint8`) with values ranging from 0 (darkest) to 255 (brightest). Color images can be represented as 3-dimensional arrays such as shown in Figure 1.

NumPy is a fundamental package for scientific computing with Python. The basic object that we use in NumPy is a multidimensional, or in other words N-dimensional, array. We now learn the basics of using NumPy. The very first thing you must do to start using NumPy in your python code is to import the module:

```python
import numpy as np
```

[1]A 0-dimensional array is a scalar, such as 42. A 1-dimensional array is a vector, e.g., $\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$.

Continue the tutorial `http://cs231n.github.io/python-numpy-tutorial/#numpy`. After you are done with the NumPy section, make sure you are able to solve the following small problems. If you want to know more, or need to figure out what a NumPy function does or how to accomplish a particular task in NumPy, study the NumPy reference page: `https://docs.scipy.org/doc/numpy/reference/`

1. Create a vector of size 10 with all elements equal to zero.
2. Create a 3-by-3 matrix with the numbers from 0 to 8. Multiply all elements in the matrix by 3.
3. Calculate the matrix product $A^T \cdot B$ where $A = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$[2].
4. Add the row vector $\begin{bmatrix} -1 & 1 \end{bmatrix}$ to each row of $A^T \cdot B$.
5. Create a 3-by-3 matrix X with random values in the range $[0, 1)$. Create a binary matrix C of size 3-by-3 where the value of an element is `False` if the value of the corresponding element in X is less than 0.5, and `True` otherwise. Set all values in X at indices where C is `True` to -1.

3.2 The imageio module & indexing image arrays

The `imageio` module can be used to load and write images from disk or from the Internet. Access the module documentation at `https://imageio.github.io/` and solve the next two tasks.

1. Read a color image from disk or from a http address, and print out its shape and data type.
2. Read a grayscale image from disk or from a http address, and print out its shape and data type.
3. For both cases, can you see how the array shape relates to Figure 1?

Consider the color image array `X`.

1. What is the meaning of `X[:,:,0]`? How about `X[0,:,0]`? And `X[:10,:10,2]`? Can you tell what the shape of the results will be?
2. How can you index `X` to extract the 10-by-10 area in the lower right corner of the green color channel? Hint: use negative indices in NumPy.

[2]The correct answer is $A^T \cdot B = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}$.