Lecture 2 exercises

Advanced Python - Spring 2022

Introduction

In this exercise set, we will use the concepts covered in the second lecture. In addition, you will need to make use of the modules *scipy* and *matplotlib*. To install these, run *pip install scipy matplotlib*, in the terminal, or !*pip install scipy matplotlib* in Jupyter.

Download the exercise materials (*lecture_2_materials.zip*) from the course page on Ilias.

The exercise will be marked as OK if you get **13 / 19** points or more. Points are only awarded for exercises where your code produce the expected result, and where you provide comments describing what the code does.

If you have any questions, send an e-mail to *sigurd.alnes@inf.unibe.ch*.

Questions that do not ask for code should be answered as comments.

Exercises must be handed in via ILIAS (*Exercise 2 delivery*). Only if submission via ILIAS is not possible, you may submit via e-mail to *sigurd.alnes@inf.unibe.ch*. Deliver your submission as a compressed file (zip) containing one .py file for each main exercise (exercise_X.py, exercise_2.py, etc.). Use comments to indicate which sub-task your are answering within the main exercise (# Exercise 1a, etc.).

Name the zip archive according to the following format: **lecture-X_group-ID.zip**
Where X is the lecture number indicated in the title of this PDF, and ID is the ID of your group.

The exercises must be handed in by two students working together. If you do not have someone to collaborate with, please refer to *this* document to find another student without a group. If all groups have two members, add your information to an empty row, and preferably also create a forum post on Ilias announcing that you are looking for someone to collaborate with.

Collaboration outside the group (i.e. submitting the code of other groups as your own) will result in 0 points for the plagiarized exercises.

Deadline: 16:00, March 10.

Exercises

1. **Working with continuous signals** ..... 4 points

   (a) **Load and investigate sound** ..... 0.5 point
   Load the file *sounds.wav* from the lecture 2 materials:

   ```
   # Load the sound file
   from scipy.io.wavfile import read
   fs, sounds = read('./sounds.wav')
   ```

   The signal contains two sounds of equal duration. To better understand the signal:

   - Plot the signal

     ```
     # (a) Plot the sound signal
     import matplotlib.pyplot as plt
     plt.plot(signal)
     plt.show()
     ```

   - Play the signal

     ```
     # (b) Play the sound
     import sounddevice as sd
     import time
     sd.play(signal, fs)
     time.sleep(1)
     ```

   (b) **Split the *sounds* array in two equal parts** ..... 1 point
   Split the sounds array in two equal parts to divide the sounds, and store the two halves in separate variables. The first half is the sound of a froq ribbit. The second, the sound of a door being slammed shut.

   (c) **Create a silent period** ..... 1 point
   Together with the sounds, we also want a silent period, where amplitude = 0. Create a 1d-array called silence filled with zeros, that is as long as each of the auditory signals you made above. The data type should be a float.

   (d) **Concatenate the signal** ..... 1.5 points
   Create a variable *new_sound* that contains the following sounds, in the described order, using *np.concatenate*():

   i. Frog
   ii. Silence
   iii. Frog
   iv. Silence
   v. Door

   Both plot and play your 1d array *new_sound*, as described in Exercise 1a, to confirm that your solution is correct.

2. **Manipulating arrays** ..... 10 points

(a) **Load bitmap image and change colors** ..... 1 point
Load in the bitmap image *python.bmp* from the lecture 2 materials (code below).

The image is a 3-dimensional array, where the 1st and 2nd dimensions represent positions on the X and Y axis and the 3rd saturation values between 0 and 255 for that specific position, as [red, green, blue].

Complete the example below so that it sets the pixels on the 50th to 60th row and and 10th to 20th column to red, and displays the resultant image. The data-type of the array should be an 8-bit unsigned integer.

```
import matplotlib.pyplot as plt
import numpy as np
im = plt.imread('./python.bmp')
im = np.array(im)
im[ ..., ... ] = np.array([ ..., ..., ... ], dtype= ... )
plt.imshow(im)
plt.show()
```

(b) **Flipping** ..... 2 points
Make a copy of the top half of the image, flip it along the 2nd axis using the appropriate numpy method, and plot the result.

(c) **Splicing** ..... 2 points
Make a copy of the bottom half of your image, splice it together with the first half you flipped in Exercise 2b, and display the result. Also plot your original image and check whether it has changed.

(d) **Shallow copy** ..... 3 points
Make a shallow copy (view) of the top half of your image. Change every black pixel ([0, 0, 0]) to a red pixel ([255, 0, 0]. Plot the base of your shallow copy and also the original image.

(e) **Copies and pointers** ..... 2 points
In Exercise 2c, your original image should not have changed, while in Exercise 2d, the original image should show the same change as when plotting the base of your shallow copy. Give a brief explanation of why the original image changed in one instance, and not in the other.

3. **Working with arrays** ..... 5 points

(a) **Amplify signal** ..... 1 point
Import the sound used in exercise 1, and double its amplitude using *np.multiply*.

(b) **Sub-exercise title** ..... 2 points
In the following, you will make use of the temperature dataset found in the file *temperatures.npy*. This synthetic data contains information on day-time average temperatures for the month of July in Bern, Zurich, Amsterdam (NL), Reykjavik (IS), and Rome (IT).

```
import numpy as np
temperatures = np.load('./temperatures.npy', allow_pickle=True)
temperatures[0] #==> 31 data-points for Bern
temperatures[1] #==> 31 data-points for Zurich
temperatures[2] #==> 31 data-points for Amsterdam
temperatures[3] #==> 31 data-points for Reykjavik
temperatures[4] #==> 31 data-points for Rome
```

Use built-in numpy methods to obtain an array with the maximum temperature for each city, and the average temperature *across* the five cities for each day.

(c) **Maximum temperature** ..... 2 points
Obtain the maximum temperature of the entire dataset found in *temperatures.npy*. Next, use a numpy method to find out at which row and column in the dataset you would find this maximum temperature at.