Programming Assignments General Guidelines

Course on Audio Signal Processing for Music Applications

This document outlines some general guidelines for the programming assignments in the course. Before you start, please make sure you have a working installation of python 2.7.*. We recommend you to install ipython so that you can interactively test/run your code. Go through this document for guidelines on python coding style, setting up sms-tools repository, the working directory to use for assignments, testing your code, and instructions for submitting your answers.

Python coding style

To maintain uniformity in the python code we write in assignments, we recommend the use of **four spaces** for each indentation level of the code.

Setting up sms-tools

Throughout the course we will be using many functions of the sms-tools repository. It is therefore important to setup sms-tools correctly and make sure it works. You can download the repository directly as a zip archive from github:

`https://github.com/MTG/sms-tools/archive/master.zip`

For students familiar with git, we recommend to clone the repository using git. Since the repository evolves over time and undergoes changes, it is easier to keep the code synchronized with the latest version on github through git. For the course, you just need to use the clone and pull commands of git.

Installation

To start working with sms-tools you need to install some dependencies. Read the instructions in the README.md file inside the sms-tools repository:

`https://github.com/MTG/sms-tools/blob/master/README.md`

You can use the following forum thread for discussions related to the installation of sms-tools.

`https://class.coursera.org/audio-002/forum/thread?thread_id=4`

Working directory for programming assignments

In sms-tools there is a folder called 'workspace'. We will use this folder as our working directory for all the programming assignments. For every assignment you need to download a zip folder from the course website and unzip it in 'workspace' directory. Taking the example of the first programming assignment, the directory structure should look like this:

`/sms-tools/workspace/A1`

The programming assignment folder of each week will contain templates for answer scripts (*.py files) corresponding to every part of the assignment, a pickle file (*.pkl) containing the test cases, a PDF file explaining the assignment, and a script to submit your answers (submitA*.py). You have to complete the code (functions) in the template script without altering the name of the functions or variables already specified in the template. You should not rename or delete any file inside this folder.

Sound files

Most assignments require the use of sound files. You can use your own, making sure that they are in the right format, or you can use the ones in the folder 'sounds' of sms-tools, which are all in the required format.

To facilitate the work we have restricted the formats of sound files to use. You should always use wav audio files that are mono (one channel), sampled at 44100 Hz and stored as 16 bit integers.

Within the code, we use floating point arrays to store the sound samples, thus the samples in the audio files are converted to floating point values with a range from -1 to 1. The wavread function from the utilFunctions module in the sms-tools reads a wav file and returns a floating point array with the sample values normalized to the range -1 to 1, which is what we want.

Testing your code

After you complete a part of an assignment, make sure you run and test the code before submitting it for automatic evaluation on the coursera servers. All the programming assignments questions will require that you write a python function. The question will specify an input, or a type of input, and what output the function has to return. These are the steps to test a function (the example here considers the first programming assignment):

1. Open the terminal and go to the directory where your code is.

   ```
   $ cd */sms-tools/workspace/A1
   ```

2. Open the ipython shell by typing:

   ```
   $ ipython
   ```

3. Import the module (name of the '.py' file) to test

```
In [1]: import A1Part1
```

4. Run the code by calling the function with appropriate input arguments.

```
In [2]: A1Part1.readAudio('../../sounds/piano.wav')
```

Using test cases

In the explanation of each assignment question we describe some example test cases that you can use to run and test your code and make sure that it gives meaningful results. At times, your answers (output of your code) may differ from the reference output provided in the question description. **Ignore these differences if they are beyond the third decimal place as they are insignificant and will not be penalized**.

From the third programming assignment (A3) onwards we provide two test cases with inputs and the expected outputs. To use these test cases follow these instructions (the example here considers the third programming assignment):

1. In ipython shell import the script `loadTestCases.py` to fetch the test cases.

   ```
   In [3]: import loadTestCases
   ```

2. Obtain the test cases for a part of the assignment specifying the `partId` and the `caseId`.

   ```
   In [4]: testcase = loadTestCases.load(partId, caseId)
   ```

   For example, to obtain the second test case of A3Part4 specify `partId` = 4 and `caseId` = 2.

3. This function returns a python dictionary, where `testcase['input']` contains the input argument/s and `testcase['output']` is the expected output from your code. You can import an assignment part (A3Part4 in previous example) and use the `testcase['input']` as input argument to test your code as:

   ```
   In [5]: output = A3Part4.suppressFreqDFTmodel(**testcase['input'])
   ```

4. Finally, you can compare your `output` with `testcase['output']` and verify your implementation.

Submission Instructions

You will need a working internet connection to submit your answers. Once you complete a specific part of an assignment, and after testing it on few example test cases, you can submit your answer from the terminal by typing (the example considers the first programming assignment):

```
$ python submitA1.py
```

Follow the instructions that appear on the terminal. You need to enter the email address you used to register for the course and the Submission Password that is generated on the Programming Assignments page of the course (`https://class.coursera.org/audio-002/assignment`). Answers to each part are submitted separately. e.g. For Part-1, you will complete the code in A1Part1.py and submit it by choosing '1' when prompted for the part you wish to submit. If the submission is successful, you will see the following message on the terminal:

```
$ == Your submission has been accepted and will be graded shortly.
```

The answers are then submitted to a queue and we use a grader script that evaluates your submission and provides you an appropriate score and feedback, which can be seen on the assignment page. It might take a few minutes for the evaluation to complete and the score to get updated. You can submit your answers as many times as you wish but there is a penalty after a certain number of attempts, as explained in the grading policy. Please wait to see the feedback and score before making the next attempt at submission. Good luck coding!