Program cs04 CS 5245: Data Programming
Fall 2022: Updated September 23, 2022 R. M. Parry

Read chapter 7 in Fundamentals of Python Programming and complete the following exercises.

Complete computer setup by following the directions here:
`http://cs.appstate.edu/~rmp/cs5245/setup.pdf`

Create a folder called `cs04` to store your files for this lab.

Complete the following activities using the **exact variable names** and save the files using the **exact file names** as specified.

To test your programs, run them from IDLE and check their output.

1 `get_times`

To plot an audio signal, we'll need appropriate units on the x-axis representing the time of each sample:

$x = [t_0, t_1, t_2, \ldots, t_{n-1}]$, where n is the number of audio samples

The index of each sample in the `list` of audio data indicates its relative position. We can convert the sample index i into a time in seconds t_i using the sample rate r:

$$t_i(\text{seconds}) = t_0 + i(\text{samples}) \times \frac{1 \text{ second}}{r \text{ (samples)}}$$

In the `cs04.py` file, add a function named `get_times` that takes the number of samples n, the starting time t_0, and the sample rate r, and returns a `list` of length n that starts at time t_0 and takes steps of size $1/r$:

```python
def get_times(num_samples, start_time, sample_rate):
    # write your code here to create the new list of times 'x'
    return x
```

For example, it should look like this (to within 6 decimal places):

```
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs04.py ==================
>>> n = 11
>>> t = 0
>>> r = 10
>>> get_times(n, t, r)
[0, 0.1, 0.2, 0.30000000000000004, 0.4, 0.5, 0.6, 0.7, 0.7999999999999999, 0.8999999999999999, 0.9999999999
999999]
>>> r = 5
>>> get_times(n, t, r)
[0, 0.2, 0.4, 0.6000000000000001, 0.8, 1.0, 1.2, 1.4, 1.5999999999999999, 1.7999999999999998, 1.99999999999
99998]
>>> t = 1
>>> get_times(n, t, r)
[1, 1.2, 1.4, 1.5999999999999999, 1.7999999999999998, 1.9999999999999998, 2.1999999999999997, 2.4, 2.6, 2.8
000000000000003, 3.0000000000000004]
>>> from audio_helpers import load
>>> rate, data = load('speech.wav')
>>> n = len(data)
>>> x = get_times(n, 0, rate)
>>> x[:3]
[0, 2.2675736961451248e-05, 4.5351473922902495e-05]
>>> x[-3:]
[3.2855555555573064, 3.285578231294268, 3.2856009070312293]
>>> len(x)
144896
>>> |
```

2 scale

If an audio file is too quiet so that you always have to increase the volume when you play it or too loud and you find yourself turning it down, you might want to scale the audio file. So, if the original audio is a `list` containing the amplitudes y, you can create a new `list` y' with the audio scaled by a factor of α:

$$y = [y_0, y_1, y_2, \ldots, y_{n-1}]$$
$$y' = [\alpha y_0, \alpha y_1, \alpha y_2, \ldots, \alpha y_{n-1}]$$

In the `cs04.py` file, add a function named `scale` that takes the audio data y and a scale factor α as input arguments (in that order) and returns a new `list` with the scaled amplitudes:

```python
def scale(y, alpha):
    # put your code here...
    return new_y
```

For example, it should look like this:

```
================== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs04.py ==================
>>> scale([-1, 0, 1], 2)
[-2, 0, 2]
>>> scale([-1, 0, 1], 10)
[-10, 0, 10]
>>> from audio_helpers import load
>>> rate, data = load('speech.wav')
>>> data2 = scale(data, 2)
>>> data[:3]
[10, 10, 4]
>>> data[-3:]
[34, 34, 33]
>>> data2[:3]
[20, 20, 8]
>>> data2[-3:]
[68, 68, 66]
>>>
```

3 clip

Amplitudes greater than 32767 or less than -32768 break the format and cause unnecessary distortion. So, we will want to clip the audio so that the maximum value is at most 32767 and minimum values is at least -32768. In the `cs04.py` file, add a function named `clip` that takes a `list` of amplitudes as the input argument and creates a new `list` with values greater than 32767 replaced with 32767 and values less than -32768 replaced with -32768. You are not allowed to use the `numpy.clip` function:

```python
def clip(y):
    # put your code here...
    return new_y
```

For example, it should look like this:

```
>>>
================== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs04.py ==================
>>> a = [-40000, -32769, -32768, -32767, 0, 32766, 32767, 32768, 40000]
>>> clip(a)
[-32768, -32768, -32768, -32767, 0, 32766, 32767, 32767, 32767]
>>>
```

4 zoom

Sometimes it is useful to zoom into a segment of audio by specifying the starting and stopping index. In the `cs04.py` file, add a function named `zoom` that takes a `list` of times x, `list` of amplitudes y, a start index, and the number of samples as the input arguments and creates two new lists. The first contains a `list` of times, and the second a `list` of amplitudes for the selected segment of audio:

```python
def zoom(x, y, start, stop):
    # put your code here...
    return x2, y2
```

For example, it should look like this:

```
=================== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs04.py ==================
>>> x = [0, 1, 2, 3, 4, 5, 6, 7]
>>> y = [1, 5, 6, 9, 3, 2, 1, 0]
>>> x2, y2 = zoom(x, y, 2, 5)
>>> x2
[2, 3, 4]
>>> y2
[6, 9, 3]
>>> from audio_helpers import load
>>> rate, data = load('speech.wav')
>>> x = get_times(len(data), 0, rate)
>>> x2, data2 = zoom(x, data, 50000, 65000)
>>> x2[:3]
[1.133786848073117, 1.1338095238100785, 1.13383219954704]
>>> data2[:3]
[398, 411, 604]
>>> 
```

5 double_pitch

We can increase the pitch by skipping samples and keeping the sample rate the same. In the `cs04.py` file, add a function named `double_pitch` that takes a `list` of amplitudes y and returns a new `list` containing every other sample in the original (i.e., the even indexes starting at zero). Remember: to check if an integer is even you can use the mod operator: `index % 2 == 0`.

```
def double_pitch(y):
    # put your code here...
    return y2
```

For example, it should look like this:

```
=================== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs04.py ==================
>>> double_pitch([1, 2, 3, 4, 5, 6])
[1, 3, 5]
>>> from audio_helpers import load
>>> rate, data = load('speech.wav')
>>> data2 = double_pitch(data)
>>> data[:6]
[10, 10, 4, 3, 4, 1]
>>> data2[:3]
[10, 4, 4]
>>> 
```

6 half_pitch

We can decrease the pitch by duplicating samples and keeping the sample rate the same. In the `cs04.py` file, add a function named `half_pitch` that takes a `list` of amplitudes y returns a new `list` containing every sample twice:

```
def half_pitch(y):
    # put your code here...
    return y2
```

For example, it should look like this:

```
=================== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs04.py ==================
>>> half_pitch([1, 2, 3])
[1, 1, 2, 2, 3, 3]
>>> from audio_helpers import load
>>> rate, data = load('speech.wav')
>>> data2 = half_pitch(data)
>>> len(data)
144896
>>> len(data2)
289792
>>> data[:3]
[10, 10, 4]
>>> data2[:6]
[10, 10, 10, 10, 4, 4]
>>> 
```

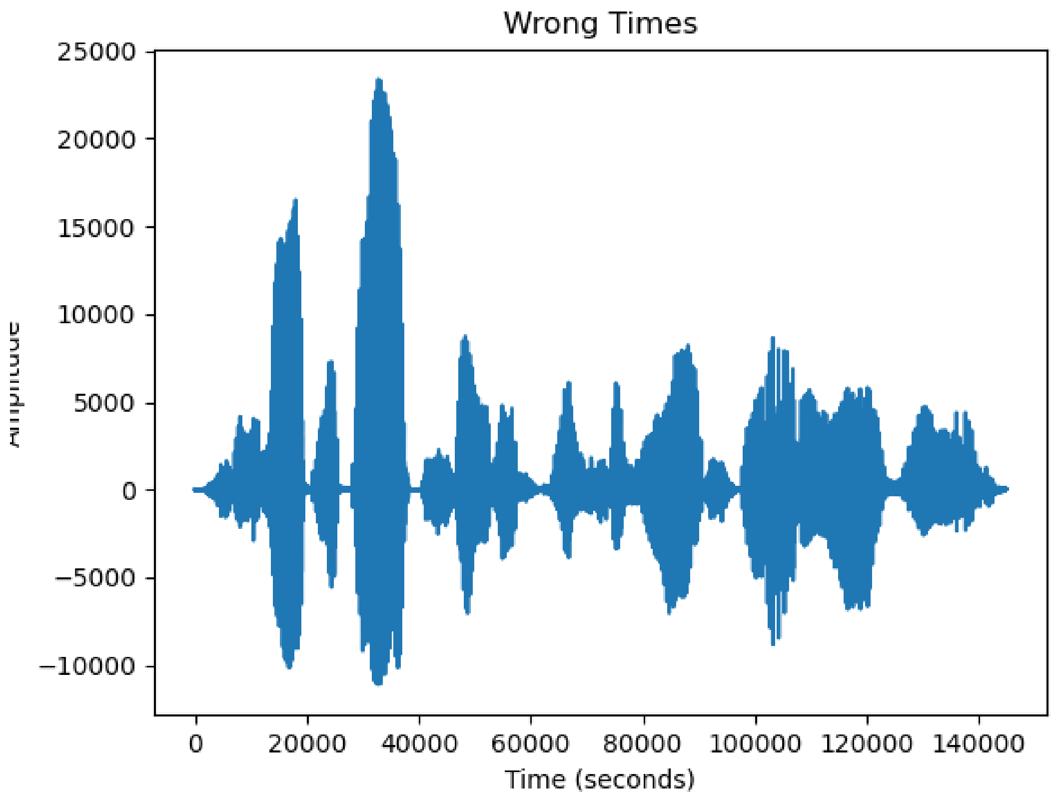
7 plot_audio

Plotting an audio signal. Now that we have a reasonable way to get the x coordinates in a `list`, y coordinates in a `list`, we can make a function that creates a plot and returns the `plt` object. In the `cs04.py` file, add a function named `plot_audio` that takes x, y, and a `title` as input arguments and plots them using `matplotlib.pyplot`. Add a label for the x-axis (`'Time (seconds)'`), label for the y-axis (`'Amplitude'`), and a title using the input argument. Save the file as `<title>.png` where `<title>` is the title passed as an input argument. For example, to save the figure as `file.png` you can use `plt.savefig('file.png')`. Finally, be sure to return `plt` at the end.

```
def plot_audio(x, y, title):
    import matplotlib.pyplot as plt
    # put your code here
    return plt
```

For example, it should look like this:

```
Python 3.8.3 (default, Jul  2 2020, 16:21:59)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs04.py ===
>>> from audio_helpers import load
>>> rate, data = load('speech.wav')
>>> plt = plot_audio(range(len(data)), data, 'Wrong Times')
>>> plt.show()
>>> 
```



8 wash

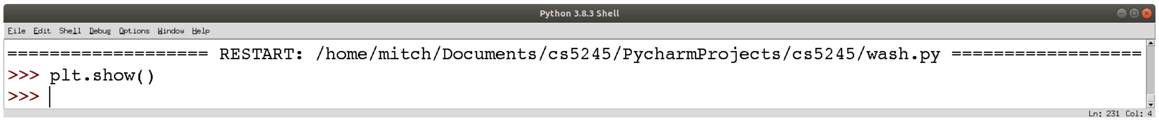
Putting it all together. Now we can generate some cool new figures using our new functions. In a file called `wash.py`, complete the following:

1. load the audio data from the `speech.wav`,
2. get the times,
3. zoom in to 150 samples starting at index 87300,
4. scale the audio by 5, and
5. generate an audio plot and store it in a variable `plt`.

```
# wash.py
import cs04

# your code here...
```

After running your script, you should be able to `plt.show()` the figure:

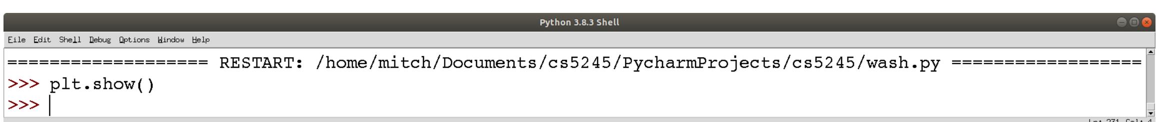


Showing the figure, should produce this:

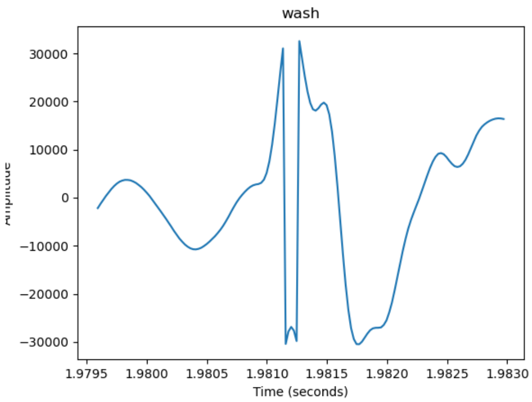
Now, edit your code so that the file is saved and loaded before creating the figure:

1. load the audio data from the `speech.wav`,
2. get the times,
3. zoom in to 150 samples starting at index 87300,
4. scale the audio by 5,
5. **save the new audio as the file `wash.wav`**,
6. **load the same data from the file `wash.wav`**, and
7. generate an audio plot using the same times as before and store it in a variable `plt`.

After running your script, you should have a new file `wash.wav` in the same folder as your script and you should be able to `plt.show()` the figure:



Showing the figure, should produce this:



What's happening here? What's with the "Batman" ears? Figure out what's wrong, fix it in your `wash.py` file so that the "Batman ears" go away.

It should look like this:

Submit to Web-CAT to compute your score!

1. Create a ZIP file for your `cs04` folder by right-clicking the folder and selecting:

   - **Send to → Compressed (zipped) folder** on Windows
   - **Compress Items** on MacOS
   - **Compress** on Linux

   You should find the new ZIP file in the same directory where `cs04` resides.

2. Login to http://webcatvm.cs.appstate.edu:8080/Web-CAT and submit your ZIP file for grading. You may submit as many times as you want before the deadline.