Coverage for audio_calculations.py: 100%

Show/hide keyboard shortcuts

Shortcuts on this page

`r m x` toggle line displays

`j k` next/prev highlighted chunk

`0` (zero) top of page

`1` (one) first highlighted chunk

46 statements

46 run | 0 missing | 0 excluded

```
1import soundfile as sf
2import pyloudnorm as pyln
3import numpy as np
4import scipy
5import math
6import resampy
7import statistics
8import moviepy.editor as mp
9
10class audio_calculations():
11 """Opening file and calculations for LUF and True Peak.
12
13 Will open a wav or mp4 file, test its peak value against
14 a standard's peak value, and test its LUF value against a standard's LUF value.
15
16 Attributes:
17 file_path: The current audio file path to be tested. Can change frequently. String object.
18 """
19
```

```
20 def __init__(self, file_path):
21 """Initializes the file to be tested.
22
23 Will store the file to be tested.
24
25 Args:
26 self: The main object
27 file_path: String containing the name of the file
28
29 Raises:
30 Any errors raised should be put here
31
32 """
33 # initialize path of the file being passed in
34 self.file_path = file_path
35
36 def get_file_path(self):
37 """ Gives the current file name being stored by the object
38
39 Returns the filename attribute being stored.
40
41 Args:
42 self: Instance of main object
43
44 Returns:
45 file_path: the file path of the audio file selected by the user
46
47 Raises:
48 Any errors raised should be put here
```

```
49
50 """
51 return self.file_path
52
53 def open_wav_file(self):
54 """Opens the wav file and fetches its needed information.
55
56 Opens the selected wav file and fetches its sample rate, data itself, length of data, and number of channels.
57
58 Args:
59 self: A main Object.
60
61 Returns:
62 A tuple containing the selected wav file's sample rate, data, length of data, and number of channels.
63
64 Raises:
65 Add possible errors here.
66
67 """
68 fileType = self.get_file_path().split('.') #split file path on '.'
69 fileType = fileType[-1] #take the last entry in the list from split as the file extension
70
71 #if the file is an MP4 file then open using moviepy and extract the audio
72 if fileType.upper() == 'MP4':
73 clip = mp.VideoFileClip(self.get_file_path())
74 audioFile = clip.audio
75 data = audioFile.to_soundarray(None,44100)
76 rate = 44100
77 else: #else open as an audio (wav/flac) file
```

78 data, rate = sf.read(self.get_file_path())

79

80 length_file = len(data)

81

82 if len(data.shape) > 1:

83 n_channels = data.shape[1]

84 else:

85 n_channels = 1

86

87 wav_info = (data, rate, length_file, n_channels)

88

89 return wav_info

90

91 def get_luf(self, wav_info):

92 """Returns the integrated loudness in LUFS of an audio file.

93

94 Uses pyloudnorm to find the LUFS of an audio file found at a file path passed.

95

96 Args:

97 self: Instance of main object

98 wav_info: a tuple of the selected wav file's sample rate, data, length of data, number of channels

99

100 Returns:

101 The integrated loudness of the audio file in LUFS

102

103 Raises:

104 Any errors raised should be put here

105

106 """

```
107
108 meter = pyln.Meter(wav_info[1]) # create meter; wav_info[1] is the rate
109 lufs = meter.integrated_loudness(wav_info[0]) # get lufs value; wav_info[0] is the data
110 return lufs
111
112 def get_peak(self, wav_info):
113 """Returns the true peak in dB of an audio file.
114
115 Uses some method to find the peak of an audio file found at a file path passed.
116
117 Args:
118 self: Instance of main object
119 wav_info: a tuple of the selected wav file's sample rate, data, length of data, number of channels
120
121 Returns:
122 The true peak of the audio file in dB
123
124 Raises:
125 Any errors raised should be put here
126
127 """
128 resampling_factor = 4 # use a resampling factor of 4
129
130 # calculate number of samples in resampled file
131 samples = wav_info[2] * resampling_factor # wav_info[2] is the length of the data
132
133 # resample using FFT
134 new_audio = scipy.signal.resample(wav_info[0], samples)
135 current_peak1 = np.max(np.abs(new_audio)) # find peak value
```

```
136 current_peak1 = math.log(current_peak1, 10) * 20 # convert to decibels
137
138 # resample using resampy
139 new_audio = resampy.resample(wav_info[0], wav_info[2], samples, axis=-1)
140 current_peak2 = np.max(np.abs(new_audio)) # find peak value
141 current_peak2 = math.log(current_peak2, 10) * 20 # convert to decibels
142
143 # resample using polynomial
144 new_audio = scipy.signal.resample_poly(wav_info[0], resampling_factor, 1)
145 current_peak3 = np.max(np.abs(new_audio)) # find peak value
146 current_peak3 = math.log(current_peak3, 10) * 20 # convert to decibels
147
148 # get and return median of the three techniques
149 peak = statistics.median([current_peak1, current_peak2, current_peak3])
150 return peak
```

« index coverage.py v6.3.2, created at 2022-04-08 11:54 -0400