CS101Audio Functions
CS101: Fall 2020

1. `variable1 = Audio()`
   (a) Make *variable1* an empty *Audio* object.
2. `variable1 = Audio(time)`
   (a) Make *variable1* an *Audio* object of silence for *time* **milliseconds**.
3. `variable1.open_audio_file(filename)`
   (a) Make *variable1* an *Audio* object with *audio segment* of the audio within *filename*. The file named *filename* must be in the same folder as the file you are working on and must contain the file extension (ie: `.wav`).
4. `variable1.play()`
   (a) Play the audio saved to *variable1*.
5. `variable1 = variable2 * n`
   (a) Make *variable1* the audio of *variable2* repeated n times.
6. `variable1 *= n`
   (a) Make *variable1* the its original audio repeated n times.
7. `variable1 = variable2 + variable3`
   (a) Make *variable1* the audio of *variable2* concatenated with the audio of *variable3*.
8. `variable1 += variable2`
   (a) Make *variable1* its original audio concatenated with the audio saved to *variable2*.
9. `len(variable1)`
   (a) Print the length of *variable1* (an *Audio* object) in **milliseconds**.
10. `variable1.change_speed(n)`
    (a) Multiply the current speed of *variable1* by n. Note that if n is less than 1, the speed will **decrease**.
11. `variable1.fade_in(n)`
    (a) Change the *Audio* object variable1 to fade in for n milliseconds.
12. `variable1.fade_out(n)`
    (a) Change the *Audio* object variable1 to fade out for n milliseconds.

13. `variable1.fade(n1,n2)`

    (a) Change the *Audio* object *variable1* to fade in for *n1* milliseconds and fade out for *n2* milliseconds. If either *n1* or *n2* are not provided, they default to 0 milliseconds.

14. `variable1[n1:n2]`

    (a) Slice the *Audio* object *variable1* from millisecond *n1* until *n2*. Note that, similar to string slicing, the slice will **include** millisecond *n1*, but will **not include** millisecond *n2*.

15. `variable1.apply_gain(n)`

    (a) Change the amplitude (loudness) of *variable1* by *n* decibels. Note that the amplitude can be decreased by specifying a negative *n* value.

16. `variable1.overlay(variable2, n, loop)`

    (a) Lay the *Audio* object *variable2* over the *Audio* object variable1 at millisecond *n*. If *n* is not provided, it defaults to 0. If *loop* is **True**, *variable2* will loop until the end of *variable1*. If it is **False**, it *variable1* will be overlayed with *variable2* for the duration of *variable2*.

17. `variable1.from_generator(n, time, shape)`

    (a) Takes *variable1*, an empty *Audio* object, and assigns it the sound of frequency *n* for *time* milliseconds with waves of type *shape*. The *shape* parameter can be one of: Sine, Square, Sawtooth, or Triangle. These wave types are case insensitive.

18. `variable1 = generate_music_note(note, time, shape, n)`

    (a) Similar to the `from_generator()` function, the `generate_music_note()` function generates sound. You pass it a *note* as a string, the *time* in milliseconds that you want it to last, and wave type of *shape*, which can be any of the types listed above. This will make *variable1* an *Audio* object with the above features. The *gain* parameter, if not specified, defaults to 0. If given, the function will also apply a gain of size *n*, just as the `apply_gain()` function does above. Note that unlike the `from_generator()` function, this function does not require *variable1* to be previously defined. Refer to the table below for a list of valid values for the *note* parameter, and the frequency that they correspond to.

19. `variable1.get_sample_list()`

    (a) Returns a list of the samples of *variable1*. The way sound is stored in computers is by taking many different measurements of the amplitude of a sound wave, called samples, and storing them together. These samples are taken at a certain sampling rate, or frame rate. A common frame rate is 44,100 samples/second. The `get_sample_list` method returns the list of samples, which could then be modified and turned back into audio with the `from_sample_list` method.

20. `variable1.from_sample_list(sampleList)`

    (a) Given a list of samples, it sets the sample list for the predefined *Audio* object *variable1* to *sampleList*.

Note String	**Frequency**	**Note String**	**Frequency**	**Note String**	**Frequency**
C0	16	C#0	17	Db0	17
D0	18	D#0	19	Eb0	19
F0	22	F#0	23	Gb0	23
G0	25	G#0	26	Ab0	26
A0	28	A#0	29	Bb0	29
B0	31	C1	33	C#1	35
Db1	35	D1	37	D#1	39
E1	41	F1	44	F#1	46
Gb1	46	G1	49	G#1	52
Db2	69	D2	73	D#2	78
Eb2	78	E2	82	F2	87
F#2	92	Gb2	92	G2	98
G#2	104	Ab2	104	A2	110
A#2	116	Bb2	116	B2	123
C3	131	C#3	139	Db3	139
D3	147	D#3	156	Eb3	156
E3	165	F3	175	F#3	185
Gb3	185	G3	196	G#3	208
Ab3	208	A3	220	A#3	233
Bb3	233	B3	247	C4	262
C#4	277	Db4	277	D4	294
D#4	311	Eb4	311	E4	330
F4	349	F#4	370	Gb4	370
G4	392	G#4	415	Ab4	415
A4	440	A#4	466	Bb4	466
B4	494	C5	523	C#5	554
Db5	554	D5	587	D#5	622
Eb5	622	E5	659	F5	699
F#5	740	Gb5	740	G5	784
G#5	831	Ab5	831	A5	880
A#5	932	Bb5	932	B5	988
C6	1047	C#6	1109	Db6	1109
D6	1175	D#6	1245	Eb6	1245
E6	1319	F6	1397	F#6	1480
Gb6	1480	G6	1568	G#6	1661
Ab6	1664	A6	1760	A#6	1865
Bb6	1865	B6	1976	C7	2093
C#7	2217	Db7	2217	D7	2349
D#7	2489	Eb7	2489	E7	2637
F7	2794	F#7	2960	Gb7	2960
G7	3136	G#7	3322	Ab7	3322
A7	3520	A#7	3729	Bb7	3729
B7	3951	C8	4186	C#8	4435
Db8	4435	D8	4699	D#8	4978
Eb8	4978	E8	5274	F8	5588
F#8	5920	Gb8	5920	G8	6272
G#8	6645	Ab8	6645	A8	7040
A#8	7459	B8	7902		