THE UNIVERSITY OF SYDNEY

INFO1110 Assignment 1

Due: Week 6, Monday 10pm (16th April, 10pm AEDT)

This assignment is worth 5%% of your final assessment

Task Description

In this assignment you will develop a noise wave simulator. You will be given a number of noise sources such as instruments, periods over which the noise is emitted and you will have to use these to display the resulting sound wave.

As with any assignment, make sure that the work is your own and do not share your code or solutions with other students.

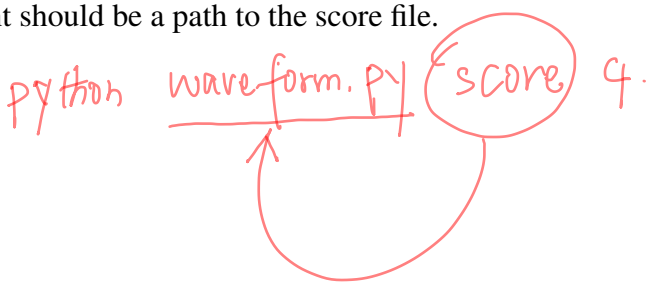
Implementation Details

The only Python libraries you are permitted to use for this assignment are 'os' and 'sys'. You are to write a Python Program that takes a command line argument indicating the path to a 'score' file containing the periods over which the noise sources are playing. Using this and a number of sources in a provided 'sources' folder you are to then construct what the resulting wave is.

It is important to note that your program will be marked automatically, so make sure that you follow the assignment specification carefully, your program's output must exactly match that shown in the examples.

An arbitrarily placed command line argument should be a path to the score file.

```
python waveform.py
No score file specified.

python waveform.py qwop
Invalid path to score file.

python waveform.py --character=b
No score file specified.
```



With a score file ‘score’:

```
piano
|**********************|
```

And instrument file ‘piano’:

```
3	   ___
2	  /   \
1	 /     \
0	___     \           ___
-1	         \         /
-2	          \       /
-3	           \_____/
```

Note that the character directly after the number is a tab character (‘\t’) and the rest of the characters in the line are either spaces, slashes or dashes.

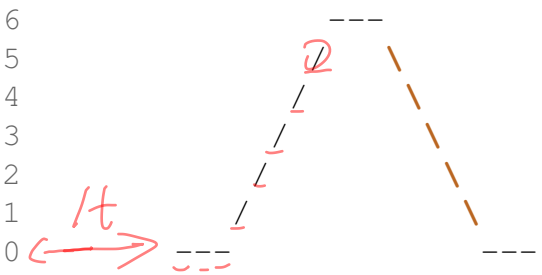
We can then print the final waveform

```
python waveform.py score
piano:
 3:              ***
 2:             *   *
 1:            *     *
 0:         ***       *       ***
-1:                   *     *
-2:                  *   *
-3:                   ***
```

Modifying the score we can see that the source gets interrupted and **restarted**.

```
piano
|*********-***********|

piano:
 3:              ***       ***
 2:             *   *     *   *
 1:            *     *   *     *
 0:         ***       ****       *
```

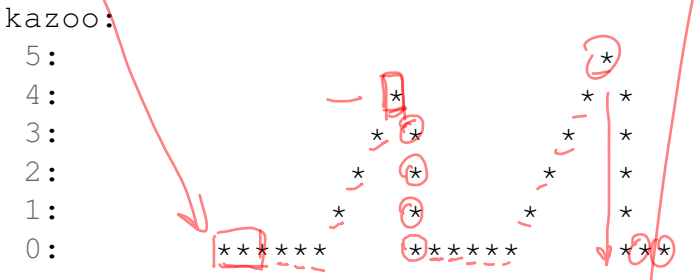
Taking a second example, we have the following kazoo file in the instruments folder. Once again the character directly after the amplitude is a tab.



Here's the score for our kazoo:

```
kazoo
|---*******---*********|
```

And we can see how the kazoo wave gets restarted when the asterisks stop in the score.



If an instrument is specified in the score file that doesn't have a corresponding file in the instruments folder it should print 'Unknown Source'.

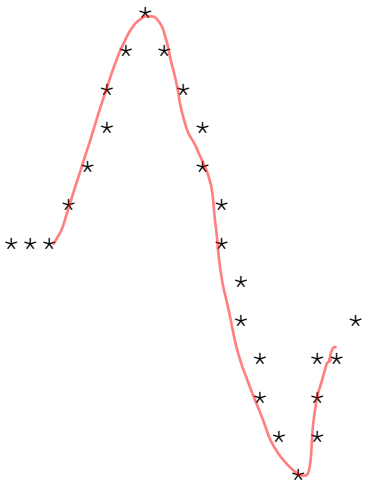
```
pian0000
|*********----------|

python waveform.py score
Unknown source.
```

Our score file should be able to specify multiple 'channels', these waves should be additive.

```
piano
|*********----------|
|----------********|
```

```
piano
|*****************---|
|---*****************|
```

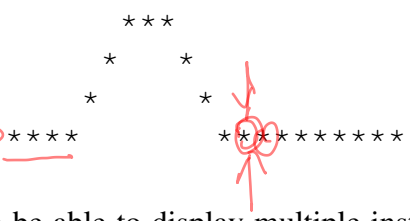
```
piano:
 6:
 5:
 4:
 3:
 2:
 1:
 0:
-1:
-2:
-3:
-4:
-5:
-6:
```



With a judiciously selected score we can also see the waves cancel out completely.

```
piano
|-*******************-|
|---------***********-|
```

```
piano:
 3:
 2:
 1:
 0:
```



You should also be able to display multiple instruments separately (in the same order that they are specified in the score file).

```
piano
|*****************---|
violin
|--******************|
```

```
piano:
 3:               ***
 2:              *   *
 1:             *     *
 0:          ***       *     ***
-1:                    *     *
-2:                   *     *
-3:                    ***
violin:
 3:                  ***
 2:                 *   *
 1:                *     *
 0:          *****       ***
-1:                        *
-2:                       *
-3:                        **
```

The '–character' flag should be able to change the character that is used to print the wave.

```
piano
|***************---|

python waveform.py score --character=q
piano:
 3:              qqq
 2:             q   q
 1:            q     q
 0:          qqq       q  qqq
-1:                    q q
-2:                     q
```

If more than one character is specified, only the first is printed

```
python waveform.py score --character=wq
piano:
 3:                 www
 2:                w   w
 1:               w     w
 0:             www       w  www
-1:                       w w
-2:                        w
```

The '–total' flag instead adds all the instruments into a single wave

```
piano
|**************---|
violin
|**************|

python waveform.py ./score --total
Total:
 6:              ***
 5:              *  *
 4:             *   *
 3:              *    *
 2:            *      *
 1:            *       *
 0:          ***        *    **
-1:                      * *
-2:                       **
```

Of course

Both of these flags may be combined in any order

```
python waveform.py --total score --character=2
Total:
 6:              222
 5:              2  2
 4:             2   2
 3:             2    2
 2:           2      2
 1:           2       2
 0:         222       2   22
-1:                    2 2
-2:                     2
```

Evaluation and submission

You will submit your assignment using Edstem. You are encouraged to write your own test cases and files to check the validity of your program while also submitting it to Edstem.

Your submission will be evaluated on two components:

- **4 Marks** are based on automatic marking of the assessment, automatic test cases includes test cases that are both visible and hidden on Edstem assignment submission page.
- **1 Mark** is based on style and explanation of code within your tutorial. Your code should closely conform to PEP-8 standard and you should be able to explain a function or segment of your code to your tutor in your tutorial

Warning: Any attempts to deceive or disrupt the marking system will result in an immediate zero for the entire assignment. Negative marks can be assigned if you do not properly follow the assignment specification, or your code is unnecessarily or deliberately obfuscated.

Academic declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgment from other sources, including published works, the Internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Information Technologies, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program, and that a copy of the assignment may be maintained by the service or the School of IT for the purpose of future plagiarism checking.