Piano Report

Made By

Name : Mazen Mohamed Sayed

ID : 52 - 2735

Tutorial : 26

Approach

My approach to solve the project is trying to make it user friendly by allowing the user to just produce any song he want by a combination of pair of notes from the Third and Fourth Octaves .

Just by typing Left note and Right note he want to choose and it is starting time and the period he/she took pressing note

SingleToneGeneration

```
def SingleToneGeneration(n,I,F,Ti,Tp):

    '''

    I  : Interval
    Ti : Initial Time
    Tp : Period Time
    F  : Tone Frequency

    '''

    t =  np.linspace(0, I, n)

    # UNIT STEP FUNCTION

    start_step = np.where(((t-Ti)>=0),1,0)

    end_step = np.where(((t-Ti-Tp)>=0),1,0)

    # Pulse

    pulse = start_step - end_step

    # Sin Function

    sin = np.sin(2 * np.pi * F * t)

    # Tone

    tone = sin * pulse

    # Display Tone

    '''

    plt.figure()

    plt.plot(t, tone , label = 'Tone')

    plt.grid(True)

    '''

    return tone
```

Description

The SingleToneGeneration method take as an input Number of Elements in Tone , Time Interval of Tone is being generated , and its Frequency , Initial Time of the Tone , and the Range (Period) for the Tone was clicked on.

then by applying the rule for tone generating in Milestone

$$x_i = sin(2\pi f_i t)[u(t - t_i) - u(t - t_i - T_i)]$$

and using this inputs to make two unit steps and subtract them to get a pulse then multiplying it with sin function to produce the Tone and at end return the tone.

SinglePairNote

```
def SinglePairNote( left_note , right_note, n, I, Ti, Tp):

    '''

     get() method of dictionary data type returns
     value of passed argument if it is present
     in dictionary otherwise second argument will
     be assigned as default value of passed argument

    '''
    # GET LEFT NOTE FREQUENCY

    switcher_left = {

        'C3' : 130.81,
        'D3' : 146.83,
        'E3' : 164.81,
        'F3' : 174.61,
        'G3' : 196,
        'A3' : 220,
        'B3' : 246.93,

    }

    left_freq = switcher_left.get(left_note,0)

   # GET RIGHT NOTE FREQUENCY

    switcher_right = {

        'C4' : 261.63,
        'D4' : 293.66,
        'E4' : 329.63,
        'F4' : 349.23,
        'G4' : 392,
```

```
        'A4' : 440,
        'B4' : 493.88,

    }

    right_freq = switcher_right.get(right_note,0)

    # Generate Left Tone

    left_tone = SingleToneGeneration(n, I, left_freq, Ti, Tp)

    # Generate RIGHT Tone

    right_tone = SingleToneGeneration(n, I, right_freq, Ti, Tp)

    # Generate Mixed Tone

    mixed_tone = left_tone + right_tone

    # Display Mixed Signal
    '''

    t =  np.linspace(0, I, n)

    plt.figure()

    plt.plot(t, mixed_tone,label = 'Mixed Note')

    plt.grid('True')

    '''

    return  mixed_tone
```

Description

By following a Technique similar to Java , where switcher act as a switch method (but it is not a method) I stored the values of Left Note Frequencies in “ switcher_left “ and Right Note Frequencies at “ switcher_right “ and having an input of

left_note : String represent name of left note

right_note : String represent name of right note

n : Number of elements

I : Interval of Time

Ti : Initial Time

Tp : Period of Time

and Using SingleToneGeneration method from previous to construct Left Note and Right Note and summing them up , to yield the “ mixed_tone “ and at end returning it back

Play

```
def Play():

    list = []

    n = int(input("Please Enter a Step : ")) # you can out value you want here

    I = int(input("Please Enter a Range : ")) # you can out value you want here

    # First Call

    in_1 = input("Please Enter a Left Note : ")

    in_2 = input("Please Enter a Right Note : ")

    if (in_1 == 'done' and in_2 =='done'):
        return 0

    Ti = float(input("Please Enter a Initial Time : "))

    Tp = float(input("Please Enter a Period : "))



    mixed = SinglePairNote(in_1, in_2, n, I, Ti, Tp)


    while(in_1 != 'done' or in_2 !='done'):

        mixed = SinglePairNote(in_1, in_2, n, I, Ti, Tp)

        list.append(mixed)

        in_1 = input("Please Enter a Left Note : ")

        in_2 = input("Please Enter a Right Note : ")

        if (in_1 == 'done' and in_2 =='done'):
            break

        Ti = float(input("Please Enter a Initial Time : "))
```

```
        Tp = float(input("Please Enter a Period : "))



    final_tone = 0


    for i in list :

        final_tone += i


    # Display Final Note

    t = np.linspace(0,I,n)

    plt.figure()

    plt.plot(t, final_tone)

    plt.grid('True')


    return final_tone
```

Description

Here is the Engine part , Where by using dynamic list predefined in python to store any number of pair of Notes till user enter “ done “as an input for both left and right note .

First , user is asked to enter the number of elements in every Tone , Time Interval of Song , then by using “ SinglePairNote “ method to make a pair of Note and store it in the list and using a “ while “ loop to keep asking the user to enter pair of Note at a time till he/she enters “ done “ .

Second , iterating over the list and summing up all the pair Notes to make the “ final_tone ”

and displaying and returning it back .

Code

```
"""
Created on Wed Apr 27 16:26:37 2022

@author: Mazen
"""




import numpy as np

import matplotlib.pyplot as plt

import sounddevice as sd



# 1st Function

def SingleToneGeneration(n,I,F,Ti,Tp):

    '''

    I  : Interval
    Ti : Initial Time
    Tp : Period Time
    F  : Tone Frequency

    '''

    t =  np.linspace(0, I, n)

    # UNIT STEP FUNCTION

    start_step = np.where(((t-Ti)>=0),1,0)

    end_step = np.where(((t-Ti-Tp)>=0),1,0)

    # Pulse

    pulse = start_step - end_step

    # Sin Function

    sin = np.sin(2 * np.pi * F * t)

    # Tone

    tone = sin * pulse

    # Display Tone
```

```
    '''

    plt.figure()

    plt.plot(t, tone , label = 'Tone')

    plt.grid(True)

    '''

    return tone


# 2nd Function


def SinglePairNote( left_note , right_note, n, I, Ti, Tp):

    '''

     get() method of dictionary data type returns
     value of passed argument if it is present
     in dictionary otherwise second argument will
     be assigned as default value of passed argument

    '''
    # GET LEFT NOTE FREQUENCY

    switcher_left = {

        'C3' : 130.81,
        'D3' : 146.83,
        'E3' : 164.81,
        'F3' : 174.61,
        'G3' : 196,
        'A3' : 220,
        'B3' : 246.93,

    }

    left_freq = switcher_left.get(left_note,0)

   # GET rIGHT NOTE FREQUENCY

    switcher_right = {

        'C4' : 261.63,
        'D4' : 293.66,
        'E4' : 329.63,
        'F4' : 349.23,
        'G4' : 392,
        'A4' : 440,
        'B4' : 493.88,
```

```
    }

    right_freq = switcher_right.get(right_note,0)

    # Generate Left Tone

    left_tone = SingleToneGeneration(n, I, left_freq, Ti, Tp)


    right_tone = SingleToneGeneration(n, I, right_freq, Ti, Tp)

    # Generate Mixed Tone

    mixed_tone = left_tone + right_tone

    # Display Mixed Signal
    '''

    t =  np.linspace(0, I, n)

    plt.figure()

    plt.plot(t, mixed_tone,label = 'Mixed Note')

    plt.grid('True')

    '''

    return  mixed_tone


#3rd Function

def Play():

    list = []

    n = int(input("Please Enter Number of Elements : ")) # you can out value you want here

    I = int(input("Please Enter a Range : ")) # you can out value you want here

    # First Call

    in_1 = input("Please Enter a Left Note : ")

    in_2 = input("Please Enter a Right Note : ")

    if (in_1 == 'done' and in_2 =='done'):
        return 0

    Ti = float(input("Please Enter a Initial Time : "))

    Tp = float(input("Please Enter a Period : "))
```

```
    mixed = SinglePairNote(in_1, in_2, n, I, Ti, Tp)


    while(in_1 != 'done' or in_2 !='done'):

        mixed = SinglePairNote(in_1, in_2, n, I, Ti, Tp)

        list.append(mixed)

        in_1 = input("Please Enter a Left Note : ")

        in_2 = input("Please Enter a Right Note : ")

        if (in_1 == 'done' and in_2 =='done'):
            break

        Ti = float(input("Please Enter a Initial Time : "))

        Tp = float(input("Please Enter a Period : "))



    final_tone = 0


    for i in list :

        final_tone += i


    # Display Final Note

    t = np.linspace(0,I,n)

    plt.figure()

    plt.plot(t, final_tone)

    plt.grid('True')


    return final_tone




# Driver Code
```

```
Piano = Play()

sd.play(Piano,3*1024)
```

Input

n = 12288

l = 3

Left	Right	Ti	Tp
C3	C4	0	0.5
C3	D4	0.75	0.15
C3	C4	1	0.15
A3	done		1.25
E3	A4	1.5	0.25
E3	A4	2	0.15
C3	F4	2.30	0.5
E3	A4	2.9	0.1

Terminal

Please Enter Number of Elements : 12288

Please Enter a Range : 3

Please Enter a Left Note : C3

Please Enter a Right Note : C4

Please Enter a Initial Time : 0

Please Enter a Period : 0.5

Please Enter a Left Note : C3

Please Enter a Right Note : D4

Please Enter a Initial Time : 0.75

Please Enter a Period : 0.15

Please Enter a Left Note : C3

Please Enter a Right Note : C4

Please Enter a Initial Time : 1

Please Enter a Period : 0.15

Please Enter a Left Note : C3

Please Enter a Right Note : C4

Please Enter a Initial Time : 1

Please Enter a Period : 0.15

Please Enter a Left Note : A3

Please Enter a Right Note : done

Please Enter a Initial Time : 1.25

Please Enter a Period : 0.15

Please Enter a Left Note : E3

Please Enter a Right Note : A4

Please Enter a Initial Time : 1.5

Please Enter a Period : 0.25

Please Enter a Left Note : E3

Please Enter a Right Note : A4

Please Enter a Initial Time : 2

Please Enter a Period : 0.15

Please Enter a Left Note : C3

Please Enter a Right Note : F4

Please Enter a Initial Time : 2.3

Please Enter a Period : 0.5

Please Enter a Left Note : E3

Please Enter a Right Note : A4

Please Enter a Initial Time : 2.9

Please Enter a Period : 0.1

Please Enter a Left Note : done

Please Enter a Right Note : done

Output