FIR Code- Python 3.7

```
#FIR Filter
#Import libraries
import matplotlib.pyplot as plt
from numba import njit
import numpy as np
import scipy.io.wavfile as wavfile
import scipy.signal as signal
import numba as nb
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter import messagebox
```

```
#Filtering function
def fir_filter2(padded, impulse_response_rev):
    result=np.zeros(len(padded)-len(impulse_response_rev)+1) #define array of zeros to store the result
    #perform convolution (use 'dot' function instead of 'sum' and multiply to speed up computation)
    for i in range(len(padded)-len(impulse_response_rev)+1):
        result[i]=np.dot(impulse_response_rev, padded[i:i+len(impulse_response_rev)]) #calculate each output value
    return result
```

```
window = Tk()
window.title("FIR")
window.geometry('350x200')
```

```
#Define flags for the interface (these improve user experience)
flag =0
flag1=0
flag2=0
```

```
#Open the '.wav' file
def openwav():
    global flag1
    global Sound #array for audio
    global Header #array for header
    filename = askopenfilename()
    with open(filename,'rb') as f:
        buffer = f.read(44) #read first 44 bytes from file (header)
        Header = np.frombuffer(buffer,dtype=np.int16) #store header
        buffer = f.read() #read the rest of the file (audio)
        Sound = np.frombuffer(buffer,dtype=np.int16) #store audio
        flag1 = 1
```

```
#Import the coefficients
def import_coef():
    global flag2
    filename = askopenfilename()
    with open(filename, 'rb') as f:
        n=f.readlines() #read coefficients from file
        global impulse_response
        impulse_response=np.array(n, dtype=np.float64) #store the coefficients in 'double' array
        flag2 = 1
```

```
#Filter the '.wav' file
def filterwav():
    global flag
    global flag1
    global flag2
    global Sound
    global impulse_response
    #filtering can only be done if both the coefficients and the '.wav' file are imported
    if (flag1 and flag2):
        L=len(impulse_response)-1 #dimension of zero vector to pad the audio for convolution
        vector=np.zeros(L).astype(np.int16) #array of zeros for time reversal
        padded=np.concatenate((Sound,vector)) #forming the zero-padded Sound vector
        global filtered_sound
        messagebox.showinfo("FIR", "This will take a while. Press 'ok' to start!")
        filtered_sound=fir_filter2(padded, impulse_response) #call on fir function
        messagebox.showinfo("FIR", "Filtering finished. Now don't forget to save!")
        filtered_sound=np.asarray(filtered_sound, dtype=np.int16) #convert the filtered audio to 'int'
        flag = 1
        flag1=0
        flag2=0
    else:
        messagebox.showinfo("Error!", "Import necessary files for a new filtering OR save the already filtered file")
```

```
#Save the filtered audio
def savewav():
    global flag
    #saves the filtered audio only if filtering was performed
    if flag == 1:
        global filtered_sound
        global Header
        with open("header.bin","wb") as f:
            f.write(Header) #save header into a separate file
        with open("data.bin","wb") as f:
            f.write(filtered_sound) #save filtered audio into separate file
        with open("header.bin","rb") as h:
            song = h.read()
            with open("data.bin","rb") as d:
                song += d.read() #form the '.wav' file
        with open("new_fir.wav","wb") as f:
            song = np.array(song)
            f.write(song.tobytes()) #save the '.wav' file
            messagebox.showinfo("Congrats!", "File was saved in the same location as the original file under the name 'new_fir'. Rename the file and enjoy!")
            flag=0
    else:
        messagebox.showinfo("Error!", "There is no file to save")
```

```
#Define buttons
btn_openwav = Button(window, text="OPEN WAV FILE", command=openwav, height = 5, width = 24)
btn_openwav.grid(column=1, row=0)
btn_filterwav = Button(window, text="FILTER WAV FILE", command=filterwav, height = 5, width = 24)
btn_filterwav.grid(column=1, row=1)
btn_import_coef = Button(window, text="IMPORT FIR COEFFICIENTS", command=import_coef, height = 5, width = 24)
btn_import_coef.grid(column=2, row=0)
btn_savewav = Button(window, text="SAVE FILTERED WAV FILE", command=savewav, height = 5, width = 24)
btn_savewav.grid(column=2, row=1)
```

```
window.mainloop()
```

IIR Code- Python 3.7

```
#IIR Filter
#Import libraries
import matplotlib.pyplot as plt
from numba import njit
import numpy as np
import scipy.io.wavfile as wavfile
import scipy.signal as signal
import numba as nb
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter import messagebox
```

```
#function for reading numbers from file
def is_float(n):
    try:
        float(n)
        return True
    except:
        return False
```

```
#implement IIR filter
def iir_filter(Sound,zeros_real,zeros_imaginary,poles_real, poles_imaginary):
    N1=len(Sound)
    y=np.zeros(N1) #define array of zeros to store the output
    x=Sound.copy() #the input x is 'Sound'
    #there are two stages for a 4th order filter
    for m in range(2):
        #zeros/poles imaginary/real parts
        a0=zeros_real[m]
        b0=zeros_imaginary[m]
        a1=poles_real[m]
        b1=poles_imaginary[m]
        #calculate coefficients for difference equation
        e0=(a0*a0+b0*b0)
        e1=(a1*a1+b1*b1)
        coef1=1
        coef2=((-2)*a0)
        coef3=e0
        coef4=(-2)*a1
        coef5=e1
        maxi=np.max(abs(x))
        #peak normalisation for each stage to avoid overflow
        x=x/maxi
        for n in range(2,N1):
            #difference equation
            y[n]=(coef1*x[n]+coef2*x[n-1]+coef3*x[n-2]-coef4*y[n-1]-coef5*y[n-2])
        for n in range(N1):
            #make the output of the first stage, input for the second stage
            x[n]=y[n]
    return y
```

```
#Window for GUI
window = Tk()
window.title("IIR")
window.geometry('350x200')
```

```
#Define flags for the interface (these improve user experience)
flag1=0
flag2=0
flag=0
```

```
#Open the '.wav' file
def openwav():
    global flag1
    global Sound #array for audio
    global Header #array for header
    filename = askopenfilename()
    with open(filename,'rb') as f:
        buffer = f.read(44) #read first 44 bytes from file (header)
        Header = np.frombuffer(buffer,dtype=np.int16) #store header
        buffer = f.read() #read the rest of the file (audio)
        Sound = np.frombuffer(buffer,dtype=np.int16) #store audio
        flag1=1
```

```
#Read poles and zeros from file and store into array as 'float64'
def import_coef():
    global flag2
    filename= askopenfilename()
    with open(filename, 'rb') as f:
        lines = f.readlines() #read lines from txt file
        numbers = [float(n) for n in lines if is_float(n)] #keep only the numbers from the text file
        global zeros_real
        global zeros_imaginary
        global poles_real
        global poles_imaginary
        zeros_real = np.array(numbers[0:4]) #store the zero real part
        zeros_imaginary = np.array(numbers[4:8]) #store the zero imaginary part
        poles_real = np.array(numbers[8:12]) #store the poles real part
        poles_imaginary = np.array(numbers[12:16]) #store the poles imaginary part
        flag2=1
```

```
#Filter the '.wav'
def filterwav():
    global flag
    global flag1
    global flag2
    global Sound
    global zeros_real
    global zeros_imaginary
    global poles_real
    global poles_imaginary
    #filtering can only be done is both the pole/zeros and the '.wav' file are imported
    if (flag1 and flag2):
        maxi=np.max(abs(Sound)) #get the absolute maximum of the audio signal
        Sound=Sound/maxi #normalize the audio before filtering to avoid overflow
        global filtered_sound
        messagebox.showinfo("IIR", "This will take a while. Press 'ok' to start!")
        #Filter IIR
        filtered_sound=iir_filter(Sound, zeros_real, zeros_imaginary, poles_real, poles_imaginary) #call iir function
        maxi2=np.max(abs(filtered_sound))  #get maximum value of filtered sound
        maxi3=np.array(maxi/maxi2) #value for final normalisation
        filtered_sound=np.multiply(filtered_sound,maxi3) #normalize sound so that filtered_sound will have same peak as original audio
        filtered_sound=np.array(filtered_sound, dtype=np.int16) #convert filtered sound to int16 (16 bit word)
        messagebox.showinfo("IIR", "Filtering finished. Now don't forget to save!")
        flag=1
        flag1=0
        flag2=0
    else:
        messagebox.showinfo("Error!", "Import necessary files for a new filtering OR save the already filtered file")
```

```
#Save the filtered audio
def savewav():
    global flag
    #saves the filtered audio only if filtering was performed
    if flag == 1:
        global filtered_sound
        global Header
        with open("header.bin","wb") as f:
            f.write(Header) #save header into a separate file

        with open("data.bin","wb") as f:
            f.write(filtered_sound) #save filtered audio into separate file

        with open("header.bin","rb") as h:
            song = h.read()
            with open("data.bin","rb") as d:
                song += d.read() #form the '.wav' file

        with open("new_IIR.wav","wb") as f:
            song = np.array(song)
            f.write(song.tobytes()) #save the '.wav' file
            messagebox.showinfo("Congrats!", "File was saved in the same location as the original file under the name 'new_IIR'. Rename the file and enjoy!")
            flag=0
    else:
        messagebox.showinfo("Error!", "There is no file to save")
```

```
#Define buttons
btn_openwav = Button(window, text="OPEN WAV FILE", command=openwav, height = 5, width = 24)
btn_openwav.grid(column=1, row=0)
btn_filterwav = Button(window, text="FILTER WAV FILE", command=filterwav, height = 5, width = 24)
btn_filterwav.grid(column=1, row=1)
btn_import_coef = Button(window, text="IMPORT IIR COEFFICIENTS", command=import_coef, height = 5, width = 24)
btn_import_coef.grid(column=2, row=0)
btn_savewav = Button(window, text="SAVE FILTERED WAV FILE", command=savewav, height = 5, width = 24)
btn_savewav.grid(column=2, row=1)
```

```
window.mainloop()
```