ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

ИАиИТ институты

Программная инженерия кафедрасы



ЛАБОРАТОРИЯЛЫҚ ЖҰМЫС

Тақырыбы: **«Функционалды бағдарламалау арқылы нақты жағдайларды талдау»**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Жұмысты орындау сапасы | Баға диапазоны | Орындалған  % |
| 1 | Орындалған жоқ | 0% |  |
| 2 | Орындалды | 0-50% |  |
| 3 | Материялдық өзіндік жүйелендіру | 0-10% |  |
| 4 | Талап етілген көлемде және көрсетілген мерзімде орындау | 0-5% |  |
| 5 | Қосымша ғылыми әдебиеттерді пайдалану | 0-5% |  |
| 6 | Орындаған тапсырманың ерекшелігі | 0-10% |  |
| 7 | СӨЖ-ді қорғау | 0-20% |  |
|  | Қорытынды: | 0-100% |  |

Оқытушы: Қасенхан А.М.

Студент: Нескенова А.К.

Мамандығы: Computer Science

Тобы: вт 12.10-14.05

Алматы 2024 ж

**8-зертханалық жұмыс:** Функционалды бағдарламалау арқылы нақты жағдайларды талдау

**Мақсат:** Студенттерге функционалдық тәсілдер қолданылатын жағдайлар мен нақты өмірлік мәселелерді талдау және шешу арқылы функционалдық бағдарламалау туралы терең түсінік беру. Жұмыстың мақсаты – бағдарламалық қамтамасыз етуді әзірлеудің әртүрлі салаларында функционалдық бағдарламалаудың практикалық құндылығы мен тиімділігін көрсету.

10. Функционалды аудио қолданбасы - Дыбысты түрлендіру және талдаудың функционалдық әдістерін қолдана отырып, аудио файлдарды өңдеуге арналған қосымшаны жасаңыз.

10. Функциональное Аудио-Приложение - Создать приложение для обработки аудиофайлов, используя функциональные техники для преобразования и анализа звука.

import tkinter as tk

from tkinter import filedialog

import librosa

import librosa.display

import matplotlib.pyplot as plt

import numpy as np

import pygame

import soundfile as sf

class AudioApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Audio Processing App")

# Аудионы ойнату үшін pygame.mixer модулі

pygame.mixer.init()

# Интерфейс кнопкасы

buttons = [

("Load Audio", self.load\_audio),

("Transform Audio", self.transform\_audio),

("Play Original Audio", self.play\_original\_audio),

("Play Transformed Audio", self.play\_transformed\_audio),

("Plot Spectrogram", self.plot\_spectrogram)

]

for text, command in buttons:

button = tk.Button(root, text=text, command=command) #объект кнопки tk.Button. Кнопка создается с указанием родительского виджета (root)

button.pack(pady=10) #добавляется в граф интерфейс с пом метода pack().pady=10,вертикальный отступ по 10 пикселей сверху и снизу кнопки

# Аудио деректерін, үлгі жылдамдығын, түрлендірілген деректерді және бастапқы аудио файлға жолды сақтау үшін айнымалы мәндері

self.audio\_data = None

self.sample\_rate = None

self.transformed\_data = None

self.original\_file\_path = None

def load\_audio(self):

file\_path = filedialog.askopenfilename(title="Select Audio File", filetypes=[("Audio Files", "\*.mp3;\*.wav")])

# Аудио файлды таңдау үшін терезе, тақырыбы, файл түрі, тек .mp3 және .wav, file\_path сақталады

if file\_path:

self.original\_file\_path = file\_path

self.audio\_data, self.sample\_rate = librosa.load(file\_path, sr=None)

print(f"Loaded {file\_path}")

self.transformed\_data = self.audio\_data.copy()

# Егер файл таңдалса, бастапқы аудио файлға жол сақталады, librosa.load арқылы аудио деректер мен үлгі жылдамдығын жүктеу, Жүктеп алынған файл туралы хабарды көрсету, Әрі қарай түрлендіру үшін бастапқы деректердің көшірмесін сақтау

def transform\_audio(self):

if self.transformed\_data is not None:

#строка проверяет, существует ли self.transformed\_data. Если self.transformed data не равно None, код внутри блока if будет выполнен

self.transformed\_data = librosa.effects.pitch\_shift(self.transformed\_data, sr=self.sample\_rate, n\_steps=1)

print("Audio transformed")

# Pitch\_shift көмегімен дыбыс биіктігін бір жарты тонға жоғары жылжыту

def plot\_spectrogram(self):

# Спектрограмманы визуализациялау

if self.transformed\_data is not None:

stft = librosa.stft(self.transformed\_data)

spectrogram = librosa.amplitude\_to\_db(np.abs(stft), ref=np.max)

# librosa.stft көмегімен спектрограмманы есептеу, Спектрограмманы дБ-ге децибель түрлендіру, максималды мәнге келтіру, np.abs — деректер массивінің абсолютті мәнін есептейтін NumPy кітапханасының функциясы.

# График үшін фигураны құру

plt.figure(figsize=(8, 6))

librosa.display.specshow(spectrogram, sr=self.sample\_rate, x\_axis='time', y\_axis='log')

#sr - дыбыс жылдамдығы (sempple\_rate), x\_axis='time' және y\_axis='log' графикалық ось: Xуақыт, Yжиілік.

plt.colorbar(label='dB')

plt.title("Spectrogram")

plt.show()

def play\_audio(self, file\_path):

pygame.mixer.music.load(file\_path)

pygame.mixer.music.play()

#Pygame көмегімен аудио файлды жүктеу және ойнату

def play\_original\_audio(self):

if self.original\_file\_path is not None:

self.play\_audio(self.original\_file\_path)

#Бастапқы аудио файлға жол көрсетілсе, түпнұсқа дыбыс ойнатылады

def play\_transformed\_audio(self):

if self.transformed\_data is not None:

#Түрлендірілген деректердің бар-жоғын тексереді,аудио файл үшін (временный файл)жасайды

transformed\_file\_path = "transformed\_audio.wav"

sf.write(transformed\_file\_path, self.transformed\_data, self.sample\_rate)

# Түрлендірілген аудио файлды уақытша файлға жазy

self.play\_audio(transformed\_file\_path)

if \_\_name\_\_ == "\_\_main\_\_":

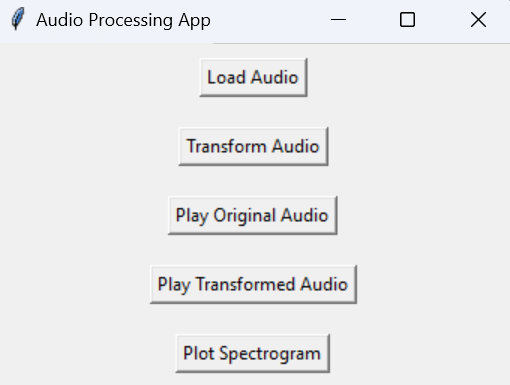
root = tk.Tk()

app = AudioApp(root)

root.mainloop()  
# Бағдарламаның негізгі блогы: граф интерфейс үшін root виджетті жасау.

AudioApp қолданбасын root инициализациялау.

граф интерфейс көрсету және оны жаңарту үшін root.mainloop() негізгі циклін іске қосу.

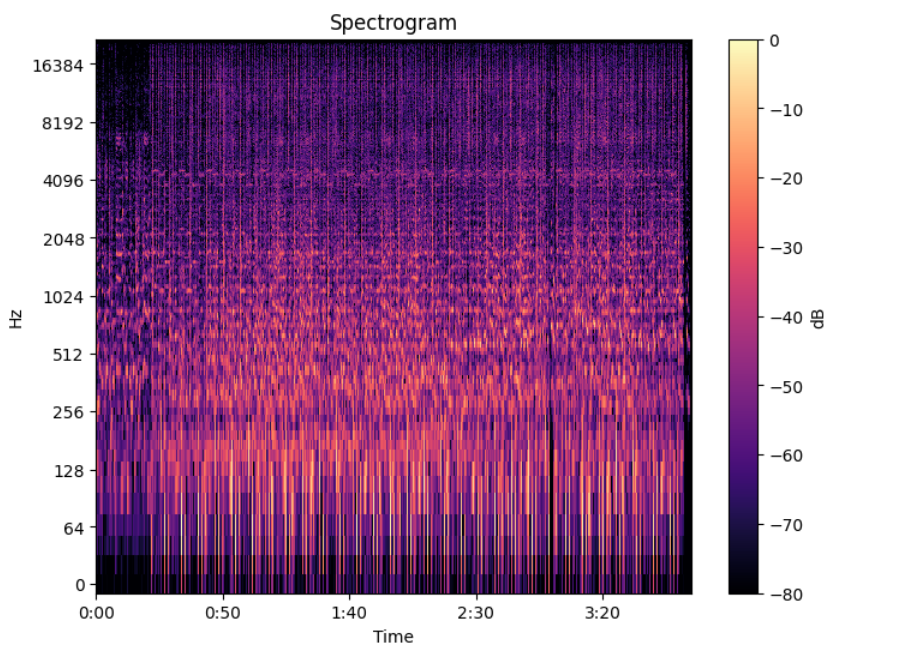
 Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание









Дыбыс және дыбыстық сигналдар контекстінде **децибел** сигналдың салыстырмалы дыбыс деңгейін немесе күшін көрсетеді. Децибел мәні неғұрлым жоғары болса, дыбыс соғұрлым күшті болады.

Дыбыстық сигналдар контекстінде **Герц** дыбыс жиілігін өлшеу үшін қолданылады, ол оның биіктігіне немесе үніне сәйкес келеді.

Спектрограмманы визуализациялау дыбысты талдау тәсілдерінің бірі болып табылады. Спектрограмма дыбыстық сигналдың жиілік мазмұны уақыт бойынша қалай өзгеретінін көрсетеді.

Спектрограмма дыбыстық сигналды уақыт пен жиілік доменіне түрлендіру арқылы құрастырылады. Әдетте бұл мақсат үшін Фурье түрлендіруі пайдаланылады. Бұл уақыттың әртүрлі нүктелерінде әртүрлі жиіліктердегі дыбыс энергиясын елестетуге мүмкіндік береді.

Спектрограмманың негізгі аспектілеріне мыналар жатады:

**Жиілік облысы:** Спектрограмманың тік осі жиіліктер ауқымын көрсетеді (әдетте герц, Гц).

**Уақыт аймағы:** спектрограмманың көлденең осі уақытты көрсетеді.

**Қарқындылық:** Спектрограммадағы түс немесе жарықтық белгілі бір уақытта белгілі бір жиіліктердегі дыбыстың қарқындылығын немесе қуатын білдіреді.

Осылайша, спектрограмма визуализациясы дыбысты талдаудың пайдалы құралы болып табылады, себебі ол дыбыстың уақыт пен жиіліктер бойынша қалай өзгеретінін көруге мүмкіндік береді.

**Дайындық сұрақтары:**

**1тапс. Функционалдық бағдарламалау принциптерін жобаңызға қалай кіріктірдіңіз?**

**Функционалды бағдарламалау** - таза функцияларды, деректердің өзгермейтіндігін және кодтың түсініктілігі мен сапасын жақсартатын басқа концепцияларды қолдануға ерекше мән беретін бағдарламалау парадигмасы.

Менің AudioApp жобамда кодты жақсартуға және оның оқылуын арттыруға көмектесетін кейбір функционалды бағдарламалау принциптері біріктірілген:

**Таза функциялар:** load\_audio, transform\_audio, plot\_spectrogram, play\_original\_audio және play\_transformed\_audio сияқты класстағы әдістердің көпшілігі AudioApp нысанының күйін өзгертпейді және параметрлерде немесе айнымалылармен жұмыс істейді. Олар тек бір тапсырманы орындайды және болжамды нәтижелерді қайтарады, бұл оларды таза функцияларға жақындатады.

**Міндеттерді бөлу:** Әрбір әдіс бір нақты тапсырмаға жауап береді (мысалы, дыбысты жүктеу, дыбысты түрлендіру, спектрограмманы визуализациялау және т.б.), кодты құрылымды етеді.

**Айнымалыларды инициализациялау:** класс конструкторында (\_\_init\_\_) audio\_data, sample\_rate, transformed\_data және original\_file\_path сияқты барлық айнымалылар None мәніне инициализацияланады, бұл инициализацияланбаған айнымалылармен жұмыс кезінде қателерді болдырмайды.

**play\_audio функциясы:** бастапқы және түрлендірілген дыбысты ойнату үшін кодты қайта пайдалануға мүмкіндік беретін, көрсетілген жолы бар аудио файлды ойнату үшін пайдаланылады.

**2тапс. Жобаны жасау кезінде қандай функционалдық үлгілерді немесе конструкцияларды пайдаландыңыз?**

Жобаны әзірлеу кезінде мен кодты құрылымдауға, оның оқылуын арттыруға және техникалық қызмет көрсетуді жеңілдетуге көмектесетін бірнеше функционалдық үлгілер мен конструкцияларды қолдандым:

**Жоғары ретті функциялар:** жоба түйме кнопкалары түрінде жоғары ретті функцияларды пайдаланады. Мысалы, кнопкалар тізімінде (мәтін, функция) жұптарының тізімін анықтаймын, мұнда функция әрбір кнопка үшін оқиға өңдеушісі(обработчик событий) болып табылады.

*buttons = [*

*("Load Audio", self.load\_audio),*

*("Transform Audio", self.transform\_audio),*

*("Play Original Audio", self.play\_original\_audio),*

*("Play Transformed Audio", self.play\_transformed\_audio),*

*("Plot Spectrogram", self.plot\_spectrogram)*

*]*

Бұл тізім әр функцияны сәйкес кнопкаға тағайындау арқылы пайдаланушы интерфейсін оңай және ықшам анықтауға мүмкіндік береді. Кейінірек интерфейсті жасаған кезде тізім бойынша қайталауға және команда аргументі ретінде сәйкес функцияны бере отырып, әрбір кнопканы жасауға болады:

*for text, command in buttons:*

*button = tk.Button(self.root, text=text, command=command)*

*button.pack(pady=10)*

Осылайша, (мәтін, функция) жұптарының тізімін пайдалану пайдаланушы интерфейсін ықшам және ыңғайлы басқаруға мүмкіндік береді, әсіресе бізде бірнеше кнопкалар және олармен байланысты функциялар болған кезде.

**Бір реттік функциялар:** AudioApp класындағы әрбір әдіс дыбысты жүктеу, дыбысты түрлендіру, спектрограмманы көрсету және дыбысты ойнату сияқты бір нақты тапсырманы орындайды. Бұл кодты құрылымды және түсінікті етеді.

**Күйді функциялар арқылы өзгерту:** AudioApp сыныбының функциялары дыбысты жүктеу немесе түрлендіру сияқты белгілі бір әрекеттерді орындау арқылы нысанның күйін өзгертеді (audio\_data және transformed\_data).

Бұл элементтер кодты икемді, оқылатын және қолдауға болатын етуге көмектеседі.

**3тапс. Жобамен жұмыс істеу барысында функционалдық тәсілдің қандай артықшылықтарын байқадыңыз?**

Жобада бағдарламалауға функционалдық тәсілді пайдалану кезінде байқауға болатын бірнеше артықшылықтар:

**Кодтың оқылымдылығы мен қарапайымдылығының жоғарылауы:** Функционалдық тәсіл жобаны әрқайсысы белгілі бір тапсырманы орындайтын бірнеше қарапайым және түсінікті функцияларға бөлуге мүмкіндік береді. Бұл кодты басқа әзірлеушілер үшін оқуға және түсінікті етеді.

**Кодты қайта пайдалану:** функцияларды жобаның әртүрлі бөліктерінде қайта пайдалануға болады. Мысалы, кнопка өңдегіштері(обработчики для кнопок) әртүрлі кнопкалар немесе әрекеттер үшін пайдаланылуы мүмкін.

**Ықшам және қысқа код:** Жоғары ретті функциялар және басқа функционалдық конструкциялар қайталанатын кодтың көлемін азайта отырып, кодты қысқа жазуға мүмкіндік береді.

**Ыңғайлы күйді басқару:** функционалдылықты бөлек функцияларға бөлу жобаның әртүрлі бөліктерін нақты бөлуге және бағдарлама күйін тиімдірек басқаруға көмектеседі.

**Тестілеудің қарапайымдылығы:** Жеке функцияларды оқшауланған түрде тексеру оңайырақ, нәтижесінде код сапасы жақсарады.

**4тапс. Функционалдық бағдарламалауды қолдану кезінде қандай қиындықтарға тап болдыңыз және оларды қалай жеңдіңіз?**

**Функция құрамы:** Функция құрамын басқарудың күрделілігі кодты мұқият ұйымдастыру арқылы шешіледі. Мен тапсырмаларды біріктіруді және түсінуді жеңілдетіп, кішірек функцияларға бөлдім. Бұған қоса, жоғары ретті функциялар бізге әрбір кнопка үшін оқиға өңдегіштерін ықшамырақ анықтауға мүмкіндік берді.

**Файл түріне шектеу:** Бағдарлама аудио файлдарды тек белгілі пішімдерде (.mp3 және .wav) жүктеп алуға мүмкіндік береді. Пайдаланушы басқа пішімдегі файлды жүктеп алуға әрекеттенсе, бұл ақаулық тудыруы мүмкін.

Аудио қолданбаға қатысты тапсырма болғандықтан, мен mp3 форматын қойдым.

**Кодты оқу мүмкіндігі:** Функционалдық стиль оқуды және түсінуді жеңілдететін сызықтық код құрылымына әкелді. Нақты анықталған жауапкершіліктері бар функциялар және нақты кірістер мен шығыстар деректер ағынын бақылауды жеңілдетеді.

**Шешімі, кодты модульдеу:** Кодты бөлек модульдерге немесе файлдарға бөлу, олардың әрқайсысы белгілі бір функционалдық бөлікке жауап береді (мысалы, дыбысты жүктеу, дыбысты түрлендіру, спектрограмманы визуализациялау және т.б.). Бұл кодты құрылымды және оқуға жеңілдетеді.

**Код тазалығы:** Функционалды тәсілді пайдалану кодтың оқылу және техникалық қызмет көрсету мүмкіндігін жақсартты. Әрбір функцияның нақты анықталған тапсырмалары бар, бұл бағдарламаны түсінуді жеңілдетеді және кодтың сапасын жақсартады.

**Түсініктемелер мен құжаттама(комметарий):** Кодтың күрделі бөліктеріне олардың не істейтінін сипаттау үшін түсініктеме беру. Жақсы құжатталған функциялар мен әдістер кодтың не істейтінін түсінуге көмектеседі.

**Кері байланыс жоқ:** пайдаланушы аудио түрлендіру немесе файлды жүктеп алу сияқты тапсырмалардың орындалу барысы туралы білмеуі мүмкін. Орындалу көрсеткіштерін немесе күй хабарларын қосу пайдаланушы үшін жақсы болатын еді.

**5тапс. Сіздің жобаңыздың қандай аспектілерінде функционалдық бағдарламалау әсіресе пайдалы болды?**

Функционалды бағдарламалау бірнеше себептерге байланысты AudioApp жобасында әсіресе пайдалы болуы мүмкін:

**Кодтың қарапайымдылығы мен анықтығы:** дыбысты жүктеу, дыбысты түрлендіру немесе визуализация сияқты әрбір тапсырмаға арналған функциялар кодты түсінуді және оқуды жеңілдетеді.

**Кодты қайта пайдалану:** Тапсырмаларды бөлек функцияларға бөлу оларды қайта пайдалануға мүмкіндік береді. Мысалы, дыбысты ойнатуға немесе дыбысты түрлендіруге арналған функцияларды қолданбадағы әртүрлі контексттерде пайдалануға болады.

**Тапсырманы оқшаулау:** Әрбір функция бір нақты тапсырмаға жауап береді (мысалы, load\_audio, transform\_audio или plot\_spectrogram), бұл күрделілікті басқаруды және қателерді табуды жеңілдетеді.

**Түзетудің қарапайымдылығы:** әртүрлі тапсырмаларды бөлек функцияларға бөлу отладтау процесінде көмектеседі. Ақаулық орын алса, мәселенің көзін жылдам табу үшін белгілі бір мүмкіндікке назар аударуға болады.

**Икемділік:** функционалдық тәсіл жобаны оңай өзгертуге немесе кеңейтуге мүмкіндік береді. Мысалы, жаңа функционалдылықты қосу немесе бар функционалдылықты өзгерту кодтың басқа бөліктерін өзгертпестен жасалуы мүмкін.

AudioApp жобасында функционалдық тәсілді пайдалану кодты жеңілдетеді және оны сенімді және түсінікті етеді, бұл жобаның сәтті және тиімді дамуына ықпал етеді.

**6тапс. Функционалдық бағдарламалаудың тестілеуге және кодты жөндеуге қалай әсер ететінін сипаттай аласыз ба?**

Функционалды бағдарламалау бірнеше себептер бойынша кодты тестілеуге (тексеруге) және жөндеуге оң әсер етеді:

**Таза функциялар:** Функционалды бағдарламалау жанама әсерлері жоқ және тек кіріс параметрлеріне тәуелді таза функцияларды пайдалануды қолдайды. Мұндай функцияларды сынау оңайырақ, себебі олар болжамды және олардың қандай нәтиже қайтару керектігін оңай анықтауға болады.

**Логикалық оқшаулау:** Функционалдық бағдарламалауда тапсырмалар шағын, тәуелсіз функцияларға бөлінеді. Бұл әрбір функция үшін сынақтарды бөлек жазуды жеңілдетеді. Әрбір функцияны оқшаулау қателерді дәлірек және тиімдірек анықтауға мүмкіндік береді.

**Қайта пайдалану және модульдік:** Функционалды бағдарламалау қайта пайдалануға болатын модульдерді жасауға ықпал етеді. Бұл функцияларды кодтың әртүрлі бөліктерінде қолдануға болатынын білдіреді, бұл тестілеу көлемін азайтады, өйткені бір функцияны бір рет тексеріп, содан кейін басқа контексттерде пайдалануға болады.

**Жанама әсерлер жоқ:** таза функциялардың жанама әсерлері болмағандықтан, тестілеу және жөндеу оңайырақ болады. Мұндай функциялардағы қателерді анықтау және түзету оңайырақ.

Тұтастай алғанда, функционалдық бағдарламалау сынауға және жөндеуге оңай кодты жасауға ықпал етеді, бұл әзірлеушілерге қателерді тезірек табуға және түзетуге және сенімдірек, жоғары сапалы бағдарламалық құрал жасауға көмектеседі.

**7тапс. Жобаңызда модульдік пен композицияны қалай қамтамасыз еттіңіз?**

Менің кодымда ұсынылған дизайнда модульдік пен композиция бірнеше негізгі аспектілермен қамтамасыз етілген:

**Класстағы кодты ұйымдастыру:** Менің жобам қолданбаның жұмыс істеуі үшін қажетті барлық әдістер мен айнымалы мәндерді қамтитын AudioApp класын пайдаланады. Бұл бағдарламаның функционалдығын нақты бөлуге және кодты құрылымдауға мүмкіндік береді.

**Тапсырмаларды функция бойынша бөлу:** Әрбір функция дыбысты жүктеу, түрлендіру, ойнату және спектрограмма визуализациясы сияқты белгілі бір тапсырманы орындайды. Бұл кодты айқынырақ және жөндеуді жеңілдетеді.

**Кодты қайта пайдалану:** play\_audio және load\_audio әдістерін play\_original\_audio және play\_transformed\_audio сияқты басқа функцияларда қайта пайдалануға болады, бұл кодты тиімдірек етеді.

**Композиция:** қолданба аудио жүктеу, дыбысты түрлендіру, спектрограмманы көрсету және дыбысты ойнату сияқты әртүрлі функцияларды жүзеге асыру үшін librosa, pygame, soundfile, tkinter және басқалар сияқты үшінші тарап кітапханаларын пайдаланады. Бұл бәрін нөлден әзірлеу қажеттілігін болдырмай, әртүрлі тапсырмалар үшін дайын компоненттерді пайдалануға мүмкіндік береді.

**Бірыңғай жауапкершілік қағидатын ұстану:** AudioApp класындағы әдістерде нақты анықталған тапсырмалар бар. Мысалы, load\_audio әдісі тек аудио файлды жүктеуге жауап береді, ал transform\_audio әдісі тек дыбысты түрлендіруге жауап береді. Бұл кодты айқынырақ және техникалық қызмет көрсетуді жеңілдетеді.

**8тапс. Функционалдық бағдарламалау үшін қандай Python құралдары мен кітапханаларын ең пайдалы деп таптыңыз?**

**tkinter:** Python-да графикалық пайдаланушы интерфейстерін жасауға арналған стандартты кітапхана.

Ол интерактивті қолданбаларды жасауға мүмкіндік беретін терезелер, түймелер, белгілер, енгізу өрістері және т.б. сияқты әртүрлі виджеттерді қамтамасыз етеді.

**functools:** reduce, partial және lru\_cache сияқты пайдалы функцияларды қамтитын кітапхана.

reduce деректер ретін өңдеуге және бір мәнді алуға көмектеседі.

partial жаңа функцияны жасау кезінде функцияның кейбір аргументтерін фиксировать(есте сақтау) мүмкіндік береді.

**itertools:** итераторлармен жұмыс істеу үшін көптеген функцияларды қамтамасыз етеді, мысалы, chain, combinations, product және т.б.

Ол күрделі деректер ретін жасауға және біріктіруге көмектеседі.

**map, filter и reduce:** Функцияны реттілік (map) элементтеріне қолдануға, шарт бойынша элементтерді сүзуге (filter) және ретті бір мәнге келтіруге (reduce) мүмкіндік беретін кіріктірілген Python функциялары.

**Лямбда функциялары:** Қарапайым функцияларды жылдам және қысқаша жасауға мүмкіндік беретін бір қатарлы функциялар.

**librosa:** Дыбысты жүктеу, дыбысты түрлендіру, мүмкіндіктерді шығару, визуализация және басқа тапсырмалар үшін функцияларды қамтамасыз ететін аудио өңдеу кітапханасы.

librosa функциялары дыбысты ыңғайлы өңдеуге және оған әртүрлі түрлендірулерді қолдануға мүмкіндік береді.

**numpy:** Мәліметтердің массивтерімен және матрицаларымен жұмыс істеуге ыңғайлы функцияларды қамтамасыз ететін ғылыми есептеулерге арналған кітапхана.

np.array, np.sum, np.mean және басқа да көптеген функциялар деректермен жұмыс істеуді жеңілдетеді және операцияларды тиімді орындауға мүмкіндік береді.

**pandas:** Деректерді талдау және өңдеуге арналған кітапхана.

Ыңғайлы деректер құрылымдарын (DataFrame, Series) және деректерді сүзуге, түрлендіруге, біріктіруге арналған функцияларды қамтамасыз етеді.

**9тапс. Функционалдық бағдарламалаудың қандай аспектілерін түсіну немесе қолдану сізге қиынға соғады?**

**Ерекше синтаксис:** Функция коды мен үйренгеннен басқаша болуы мүмкін. Мысалы, басқа функцияларды аргумент ретінде қабылдайтын функцияларды пайдалану.

**Функциялар композициясы:** Функция композициясы- бұл бірнеше функциялардың бір функцияға бірігуі. Бұл кодтың оқылуын және модульділігін жақсартқанымен, ұзын композиция тізбегі жөндеуді және түсінуді қиындатады.

**Циклдердің орнына рекурсия:** Рекурсия жиі циклдардың орнына пайдаланылады, бұл үйреншіксіз және жөндеуге қиын болуы мүмкін.

**10тапс. Жобамен жұмыс істеу барысында функционалдық бағдарламалаудың қандай шектеулерін таптыңыз?**

**Өнімділік:** Кейде бағдарламалаудың функционалдық стилі деректерді орнында өзгертудің орнына үнемі көшіру қажеттілігіне байланысты баяу болуы мүмкін.

**Түзету:** кодтың абстрактылығына және жанама әсерлердің болмауына (деректерге немесе айнымалы мәндерге өзгертулер) байланысты қателерді табу және түзету қиынырақ болуы мүмкін.

**Жанама әсерлер:** Функциялар таза болуы керек, яғни жанама әсерлері жоқ, бұл кейбір үшінші тарап кітапханаларын пайдалануды шектеуі мүмкін.

**Оқыту:** Егер сіз басқа бағдарламалау стильдеріне үйренген болсаңыз, функционалдық тәсілге бейімделу үшін біраз уақыт алуы мүмкін.

**Императивті код:** Функционалдық стиль декларативті болу үшін жасалған, бұл дәстүрлі, императивті кодпен жұмыс істеуді қиындатады. Функциялардың жанама әсерлері болмауы керек болатын функционалдық тәсілге үйрену және императивтік стильден бас тарту дәстүрлі тәсілдерге үйренгендер үшін қиындық тудыруы мүмкін

**Рекурсияны түсіну қиындығы:** Функционалдық код жиі циклдардың орнына рекурсияны пайдаланады. Егер сіз рекурсивті тәсілдерге үйренбеген болсаңыз, басқару ағыны мен деректерді қадағалау қиынырақ болуы мүмкін.

**Өзгергіштік бойынша шектеулер:** Функционалды бағдарламалау өзгермейтін деректер құрылымдарын пайдалануды ынталандырады. Бұл әр операцияда деректердің жаңа көшірмелерін жасау керек дегенді білдіреді, бұл көбірек жадты қажет етуі мүмкін.

**11тапс. Функционалды бағдарламалау жобаңыздың өнімділігіне қалай әсер етті?**

**Аудио жүктеп алу:**

Функционалды тәсіл аудио файлдарды (load\_audio) оқуға ыңғайлы және модульдік түрде жүктеуге арналған кодты жазуға мүмкіндік береді.

**Аудио түрлендіру:**

Transform\_audio әдісі дыбысқа әсерді қолданады. Бұл ресурсты қажет ететін процесс болуы мүмкін, әсіресе үлкен файлдарды түрлендіретін болсаңыз. Дегенмен, функционалдық тәсіл бұл әдісті тәуелсіз функция ретінде жазуға мүмкіндік береді, бұл кодты түсінікті етеді.

**Спектрограмма визуализациясы:**

plot\_spectrogram әдісі дыбыстың спектрограммасын бейнелейді. Бұл процесс, әсіресе үлкен аудио файлдар үшін, есептеу қарқынды болуы мүмкін. Дегенмен, функционалдық тәсіл бұл кодты жазуды және жөндеуді жеңілдетеді.

**Аудио ойнату:**

play\_original\_audio және play\_transformed\_audio әдістері дыбысты ойнатады. Үлкен аудио файлдарды ойнату уақыт пен ресурстарды алуы мүмкін. Мұнда функционалдық бағдарламалау өнімділікке тікелей әсер етпейді, бірақ кодты неғұрлым анық құрылымдауға мүмкіндік береді.

**12тапс. Функционалдық тұжырымдамаларды пайдалана отырып, жобаңызға жақсартуларды ұсына аласыз ба?**

**1.Жоғары ретті функцияларды пайдалану:**

Карта, сүзу және азайту сияқты функциялар деректерді функционалды түрде өңдеуге мүмкіндік береді.

Мысал: Бөлек-бөлек аудио деректерге түрлендіруді қолдану үшін map пайдалануға болады.

*def apply\_transformation(audio\_data, sample\_rate):*

*transformed\_data = map(lambda x: librosa.effects.pitch\_shift(x, sr=sample\_rate, n\_steps=1), audio\_data)*

*return list(transformed\_data)*

**2.Lambda функцияларын пайдалану:**

Lambda функциялары кодты жеңілдетуге көмектесетін анонимді функциялар болып табылады.

Мысал: plot\_spectrogram функциясында спектрограмманы есептеу үшін ламбда функциясын пайдалануға болады.

*def plot\_spectrogram(self, data, sample\_rate):*

*stft = librosa.stft(data)*

*spectrogram = librosa.amplitude\_to\_db(np.abs(stft), ref=np.max)*

*plot\_func = lambda: librosa.display.specshow(spectrogram, sr=sample\_rate, x\_axis='time', y\_axis='log')*

*return plot\_func*

**3. Өнімділікті оңтайландыру:**

Өнімділікті оңтайландыру үшін функционалдық құралдарды пайдалану.

Мысал: plot\_spectrogram функциясында есептеулерді жылдамдату үшін numpy пайдалануға болады.

*def plot\_spectrogram(self):*

*if self.transformed\_data is not None:*

*stft = librosa.stft(self.transformed\_data)*

*spectrogram = librosa.amplitude\_to\_db(np.abs(stft), ref=np.max)*

*plt.figure(figsize=(8, 6))*

*librosa.display.specshow(spectrogram, sr=self.sample\_rate, x\_axis='time', y\_axis='log')*

*plt.colorbar(label='dB')*

*plt.title("Spectrogram")*

*plt.show()*

**13тапс. Жобаңызды масштабтау немесе жаңа мүмкіндіктерді қосу үшін қалай бейімдеуге болады?**

**Кодты модульдерге бөлу:**

Кодты дыбысты жүктеу, түрлендіру немесе көрсету сияқты нақты тапсырмаларды орындайтын шағын, тәуелсіз бөліктерге бөлу.

Бұл кодты түсінуді және жаңа мүмкіндіктерді қосуды жеңілдетеді.

**Әрбір тапсырма үшін класс жасау:**

Әртүрлі тапсырмалар үшін класс жасау, мысалы, дыбысты өңдеуге арналған класс (AudioHandler), визуализацияға арналған класс (AudioVisualizer).

Бұл кодты ұйымдастыруға және оны құрылымды етуге көмектеседі.

**Асинхронды орындауды пайдалану:**

Асинхронды функциялар бір уақытта бірнеше тапсырмаларды орындауға мүмкіндік береді, бұл аудио файлдармен жұмысты жылдамдатуға көмектеседі.

Мысалы, бір уақытта бірнеше аудио файлдарды жүктеп алуға және өңдеуге болады.

**Интерфейстерді енгізу:**

Интерфейстер негізгі кодты өзгертпестен жаңа мүмкіндіктерді оңай қосуға көмектеседі.

Мысалы, жаңа дыбыс эффекті қосу үшін интерфейс жасау.

Осы әдістерді пайдалана отырып, сіз жобаңызды жақсарта аласыз, оны масштабтауға және жаңа мүмкіндіктерді қосуды жеңілдетуге болады.

**Жаңа мүмкіндіктерді қосу:**

1. Жаңа эффект немесе фильтр қосу.
2. Мәліметтердің жаңа түрлерін визуализациялау.
3. Жаңа аудио форматтарын қосу.
4. Кодты ұйымдастыру:

Түсінуді және басқаруды жеңілдету үшін кодты кішірек функциялар мен класстарға бөлу.

Әрбір жаңа мүмкіндікті бөлек модульдер ретінде оңай қосуға болатындай етіп жобаны құрылымдау.

**14тапс. Жобаңызда жанама әсерлерді басқару үшін қандай әдістерді қолдандыңыз?**

**Логикалық инкапсуляция:**

Әрбір әдіс өз функциясын инкапсуляциялайды және оның жанама әсерлерін басқарады. Мысалы, load\_audio, play\_original\_audio және play\_transformed\_audio әдістері файлдармен жұмыс істеуге және дыбысты ойнатуға жауап береді, ал plot\_spectrogram деректерді визуализациялауға жауап береді.

Бұл функцияларды бөлу жанама әсерлерді оқшаулауға көмектеседі және кодты таза және түсінікті етеді.

**Таза функциялар:**

Transform\_audio, plot\_spectrogram және load\_audio сияқты кейбір әдістерді таза функциялар деп санауға болады, себебі олар енгізуді қабылдайды және күйді өзгертпестен нәтижені қайтарады. Мысалы, transform\_audio аудиоға түрлендірулерді қолданады және оны бастапқы деректерді өзгертпей қайтарады.

Таза функцияларды тексеру және түсіну оңай, өйткені олардың әрекеті болжамды және сыртқы күйге тәуелсіз.

**Түсінікті код құрылымы:**

Функционалды тұжырымдамаларды пайдалану және әдістердегі тапсырмаларды нақты бөлу кодты түсінікті етеді. Әрбір әдістің нақты міндеті және оқшауланған жауапкершілігі бар.

**Файлдармен жұмыс:**

librosa (self.load\_audio) арқылы аудио файлды жүктегенде, сіз аудио деректерін аласыз. Дыбыстық деректер одан әрі өңдеу үшін көшіріледі, бұл бастапқы (оригинал) деректердің өзгеруін болдырмауға көмектеседі.

self.play\_transformed\_audio ішінде түрлендірілген дыбыс деректерін ойнатудан бұрын уақытша файлға сақтайсыз. Бұл бастапқы (оригинал) аудио деректерін сақтауға және ойнату процесін кодтың басқа бөліктерінен оқшаулауға көмектеседі.

**15тапс. Болашақ жобалар үшін функционалдық бағдарламалау әлеуетін қалай бағалайсыз?**

Функционалдық бағдарламалау өзінің артықшылықтарына байланысты болашақ жобалар үшін үлкен әлеуетке ие:

**Жақсартылған кодты оқу мүмкіндігі:**

Функционалды бағдарламалау таза функцияларға және оған жету қадамдарынан гөрі қажетті нәтижені көрсетуге бағытталған. Бұл түсінікті және оқылатын кодты береді.

**Тестілеудің қарапайымдылығы:**

Функционалдық бағдарламалауға тән таза функциялар кодты болжауға және тексеруді жеңілдетеді. Функциялар оқшауланған және тек кіріс деректеріне тәуелді болғандықтан, олардың әрекетін тексеру оңай.

**Параллельді қолдау:**

Функционалды бағдарламалаудағы таза функциялар, әрине, параллелизмді қолдайды, өйткені олар бір-бірінен тәуелсіз орындалуы мүмкін.

**Жанама әсерлердің төмендеуі:**

Функционалды бағдарламалау кодтағы жанама әсерлердің санын азайтып, оны сенімдірек және қателерге төзімді етеді.

**Күйді басқаруды жеңілдету:**

Функционалды бағдарламалау өзгермейтін деректерді қолдайды, бұл бағдарлама күйін басқаруды жеңілдетеді және күйді өзгертуге байланысты қателер қаупін азайтады.

**Жақсы шешімдер:**

Функционалды бағдарламалау рекурсияны пайдалану және жалқау бағалау сияқты мәселелерді шешудің жақсы жолдарын ұсынады.

**Құралдар мен кітапханалар:**

Функционалды бағдарламалауды әзірлеу қауымдастығы күрделі бағдарламаларды әзірлеуді жеңілдететін құралдар мен кітапханаларды белсенді түрде жасауда, әсіресе деректерді талдау, машиналық оқыту және веб-әзірлеу салаларында.

**Мәселені шешу тәсілдеріндегі инновациялар:**

Функционалды бағдарламалау жалқау бағалау, замыкания және лямбда-функциялар сияқты мәселелерді шешудің жаңа тәсілдерін ұсынады. Бұл жаңа, тиімдірек және инновациялық шешімдерді жасауға әкелуі мүмкін.

Тұтастай алғанда, функционалдық бағдарламалау код сапасын жақсарту және әзірлеу тиімділігін арттыру үшін үлкен әлеуетке ие. Оның тұжырымдамалары көптеген салаларда сәтті қолданылуы мүмкін және болашақ жобаларда маңызды рөл атқарады, әсіресе параллель өңдеуді, сенімділікті және тестілеудің қарапайымдылығын талап ететін жобаларда.