الجمهورية العربية السورية
وزارة التعليم العالي والبحث العلمي
جامعة تشرين
كلية الهندسة الميكانيكية والكهربائية
قسم هندسة الاتصالات والالكترونيات

السنة الخامسة - برمجة الشبكات
الوظيفة الثانية

أسماء الطلاب :
نور لؤي موسى_2893
يوسف أسامة الشاعر _2129
حيدرة علي حامد _2246

Question 1: Bank ATM Application with TCP Server/Client and Multi-threading

Project Description:

Build a TCP server and client Bank ATM application using Python. The server should handle
multiple client connections simultaneously using multi-threading. The application should
allow clients to connect, perform banking operations (such as check balance, deposit, and
withdraw), and receive their updated account status upon completion.

Requirements:

A. The server should be able to handle multiple client connections concurrently.
B. The server should maintain a set of pre-defined bank accounts with balances.
C. Each client should connect to the server and authenticate with their account details.
D. Clients should be able to perform banking operations: check balance, deposit money, and withdraw money.
E. The server should keep track of the account balances for each client.
F. At the end of the session, the server should send the final account balance to each client.

Guidelines:

- Use Python's socket module without third-party packages.
- Implement multi-threading to handle multiple client connections concurrently.
- Store the account details and balances on the server side.

Notes:

- Write a brief report describing the design choices you made and any challenges faced during
implementation.
- You can choose to create a TCP Server/Client Bank ATM application or any other appropriate
application that fulfills all requirements.

الحل:

سنقوم بإنشاء كود السيرفر

```python
import socket
import sys
import threading

# إنشاء قاموس لتخزين أرصدة الحسابات (استبدله ببيانات حسابات فعلية)
account_balances = {"haedara":1000, "yousef":2000, "noor":1000}

def handle_client(client_socket):
    try:
        #  رسالة ترحيب للعميل
        client_socket.send("مرحبًا بك في خادم البنك!".encode())

        # المصادقة: العملاء يرسلون اسم المستخدم كأول رسالة
        username = client_socket.recv(1024).decode()




        # حلقة رئيسية لعمليات البنك
        while True:
            if username not in account_balances:
                client_socket.send("يرجى المصادقة باستخدام اسم مستخدم صالح.".encode())
                client_socket.close()

            client_socket.send("Available operations:\n1. Check balance\n2. Deposit funds\n3. withdraw funds\n4. exit\nenter your choise:".encode())
            choice = client_socket.recv(1024).decode()

            if choice == '1':
                # التحقق من الرصيد
                balance = account_balances[username]
                client_socket.send(f"Your balance: ${balance}".encode())
            elif choice == '2':
```

```python
            if choice == '1':
                # التحقق من الرصيد
                balance = account_balances[username]
                client_socket.send(f"Your balance: ${balance}".encode())
            elif choice == '2':
                # إيداع الأموال
                amount = float(client_socket.recv(1024).decode())
                account_balances[username] += amount
                client_socket.send(f"تم إيداع ${amount}. الرصيد الجديد: ${account_balances[username]}".encode())
            elif choice == '3':
                # سحب الأموال
                amount = float(client_socket.recv(1024).decode())
                if account_balances[username] >= amount:
                    account_balances[username] -= amount
                    client_socket.send(f"تم سحب ${amount}. الرصيد الجديد: ${account_balances[username]}".encode())
                else:
                    client_socket.send("رصيد غير كافٍ.".encode())
            elif choice == '4':
                # الخروج
                client_socket.send("وداعًا!".encode())
                break
            else:
                client_socket.send("اختيار غير صالح. يرجى إدخال خيار صالح.".encode())


            final_balance = account_balances[username]
            client_socket.send(f"رصيد حسابك النهائي: ${final_balance}".encode())

    except (ConnectionResetError, BrokenPipeError):
        print(f"تم فصل العميل {username}.")
    finally:
        client_socket.close()
```

```
    finally:
        client_socket.close()

def main():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    host = '127.0.0.1'
    port = 9999

    try:
        server_socket.bind((host, port))
    except OSError as err_msg:
        print(err_msg)
        sys.exit(1)

    server_socket.listen(5)
    print("[+] اتصالات جديدة TCP ينتظر خادم ...")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"[*] تم الاتصال بعميل جديد: {client_address}")
        client_thread = threading.Thread(target=handle_client, args=(client_socket,))
        client_thread.start()

if __name__ == "__main__":
    main()
```

إنشاء كود العميل :

```
import socket

def main():
    host = '127.0.0.1'
    port = 9999

    try:
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_socket.connect((host, port))
        print(client_socket.recv(1024).decode())  # رسالة ترحيب من السيرفر

        # ادخال اسم المستخدم الخاص به
        username = input("Enter your username: ")
        client_socket.send(username.encode())

        while True:
            print(client_socket.recv(1024).decode())  # العمليات المتاحة
            choice = input("(1/2/3/4): ")
            client_socket.send(choice.encode())

            if choice == '1':
                # التحقق من الرصيد
                print(client_socket.recv(1024).decode())
            elif choice == '2':
                # ايداع اموال
                amount = float(input("Enter the amount to deposit: "))
                client_socket.send(str(amount).encode())
                print(client_socket.recv(1024).decode())
            elif choice == '3':
                # سحب اموال
                amount = float(input("Enter the amount to withdraw: "))
                client_socket.send(str(amount).encode())
                print(client_socket.recv(1024).decode())
```

```
            elif choice == '3':
                # سحب اموال
                amount = float(input("Enter the amount to withdraw: "))
                client_socket.send(str(amount).encode())
                print(client_socket.recv(1024).decode())
            elif choice == '4':
                # الخروج
                print(client_socket.recv(1024).decode())
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    except ConnectionRefusedError:
        print("Error: Could not connect to the server.")
    finally:
        client_socket.close()

if __name__ == "__main__":
    main()
```

تم إنشاء كودي السيرفر والكلاينت

السيرفر:

يتم استدعاء كلا socket لإنشاء سوكيت السيرفر واستدعاء كلاس النظام للتعامل مع الأخطاء وكلاس threading لتنظيم عمل السيرفر مع العملاء

- أولا يتم إنشاء قاموس لتخزين أرصدة الحسابات .

- في التابع الرئيسي يتم اشتقاق سوكيت من كلاس سوكيت

-يتم إجراء Bind مع عنوان ورقم بورت

لتجنب حالة خطأ تؤدي لفشل البرنامج يتم استخدام try , except حيث وجود رسالة خطأ يؤدي لطباعتها ثم الخروج من النظام .

-يحدد عدد الكلاينت الذين يمكنهم الانتظار في السيرفر ب 5

-عند اتصال كلاينت ما بالسيرفر يخزن السيرفر عنوانه ورقم البورت

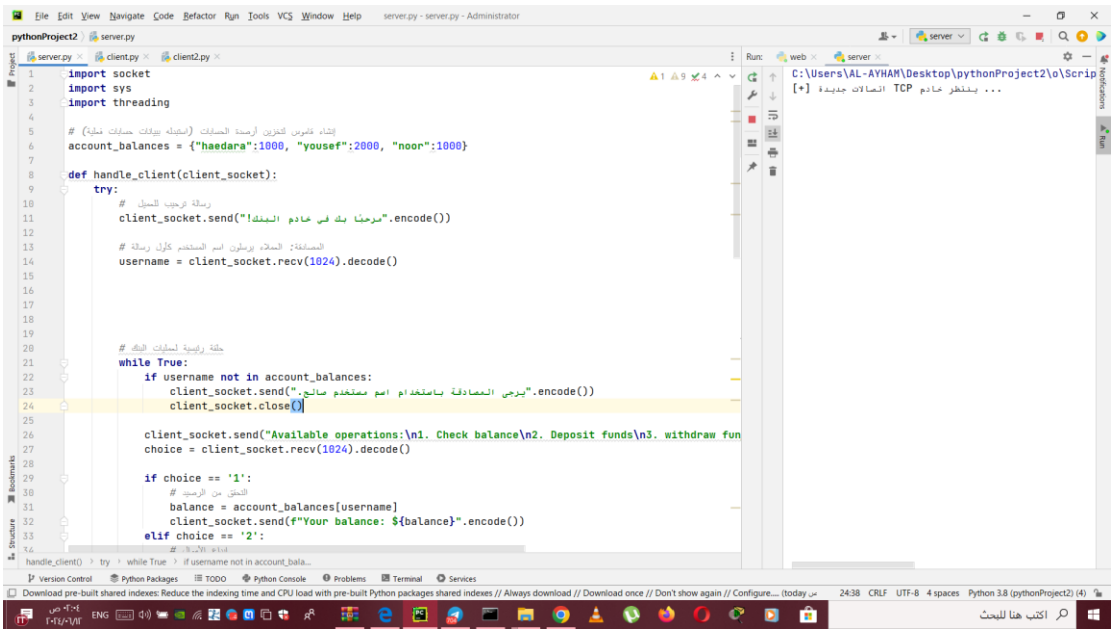
يتم انشاء thread واستدعاء تابع للتعامل مع العملاء

يقوم التابع بإرسال رسالة رسالة ترحيب للكلاينت الذي قام بالاتصال .

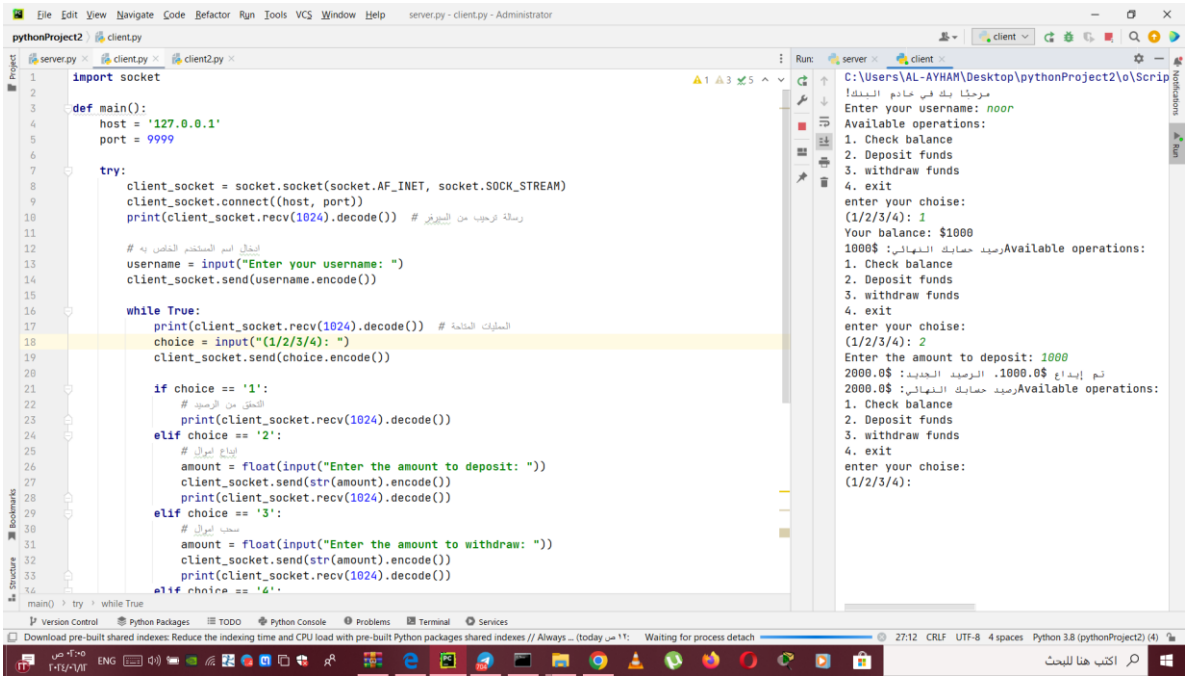
يقوم باستقبال اسم المستخدم من الكلاينت والتحقق من وجوده في الحسابات
في حال كان الاسم موجود يقوم بإرسال رسالة للمستخدم تطلب منه اختيار خيار من أربع عمليات ممكنة
ثم ينتظر استقبال رسالة الاختيار
باستخدام التعليمة الشرطية if يتم إجراء العملية التي تم اختيارها من قبل العم
1 التحقق من الرصيد وطباعته
2 ايداع أموال وإظهار الرصيد الجديد
3 سحب أموال
4 الخروج من العمليات

\\ يتم التعامل مع ورود أخطاء في الإدخال في كل بلوك ووضع شروط للتأكد من صحة العمليات الحسابية \\

الكلاينت :
_نستدعي كلاس socket
نقوم بإنشاء سوكيت خاصة بالكلاينت ووضع بورت وعنوان السيرفر
_تصل الكلاينت رسالة الترحيب من السيرفر
_يقوم الكلاينت بإدخال اسم مستخدم
وترسل الرسالة الى السيرفر
_يصل للكلاينت رسالة تحديد العملية
_يرسل الكلاينت إحدى الخيارات من 1 إلى 4
_في حال اختار عملية رقم 2 (ايداع الاموال) تحدد كمية الاموال التي سيتم ايداعها وترسل للسيرفر
وفي حال اختار العملية الثالثة كذلك تحدد كمية السحب وترسل للسيرفر
_يطبع الكلاينت الرسالة التي تصله من السيرفر حسب نتيجة العملية التي ارسلها
أو يقوم بطباعة خطأ الادخال في حال كان الاختيار غير موجود.

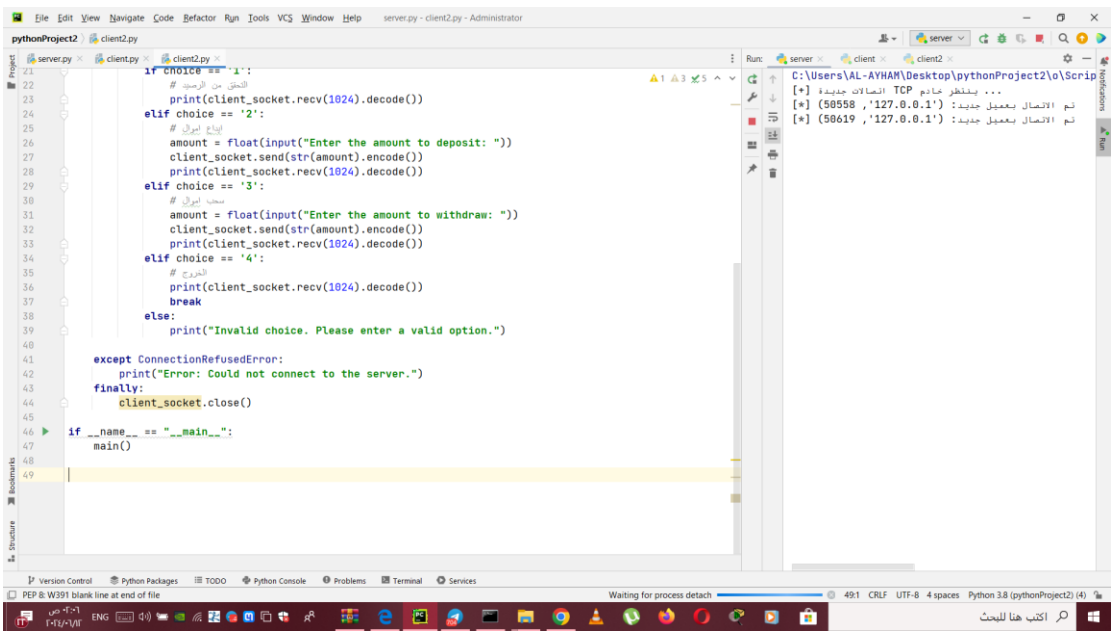


```python
import socket
import sys
import threading

# إنشاء قاموس لتخزين أرصدة الحسابات (استبدله ببيانات حسابات فعلية)
account_balances = {"haedara":1000, "yousef":2000, "noor":1000}

def handle_client(client_socket):
    try:
        # رسالة ترحيب للعميل
        client_socket.send("!مرحبًا بك في خادم البنك".encode())

        # المصادقة: العملاء يرسلون اسم المستخدم كأول رسالة
        username = client_socket.recv(1024).decode()




        # حلقة رئيسية لعمليات البنك
        while True:
            if username not in account_balances:
                client_socket.send(".يرجى المصادقة باستخدام اسم مستخدم صالح".encode())
                client_socket.close()

            client_socket.send("Available operations:\n1. Check balance\n2. Deposit funds\n3. withdraw fun
            choice = client_socket.recv(1024).decode()

            if choice == '1':
                # التحقق من الرصيد
                balance = account_balances[username]
                client_socket.send(f"Your balance: ${balance}".encode())
            elif choice == '2':
```



```python
import socket

def main():
    host = '127.0.0.1'
    port = 9999

    try:
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_socket.connect((host, port))
        print(client_socket.recv(1024).decode())  # رسالة ترحيب من السيرفر

        # إدخال اسم المستخدم الخاص به
        username = input("Enter your username: ")
        client_socket.send(username.encode())

        while True:
            print(client_socket.recv(1024).decode())  # العمليات المتاحة
            choice = input("(1/2/3/4): ")
            client_socket.send(choice.encode())

            if choice == '1':
                # التحقق من الرصيد
                print(client_socket.recv(1024).decode())
            elif choice == '2':
                # إيداع أموال
                amount = float(input("Enter the amount to deposit: "))
                client_socket.send(str(amount).encode())
                print(client_socket.recv(1024).decode())
            elif choice == '3':
                # سحب أموال
                amount = float(input("Enter the amount to withdraw: "))
                client_socket.send(str(amount).encode())
                print(client_socket.recv(1024).decode())
            elif choice == '4':
```

```
C:\Users\AL-AYHAM\Desktop\pythonProject2\o\Scrip
مرحبًا بك في خادم البنك!
Enter your username: noor
Available operations:
1. Check balance
2. Deposit funds
3. withdraw funds
4. exit
enter your choise:
(1/2/3/4): 1
Your balance: $1000
1000$ :رصيد حسابك النهائيAvailable operations:
1. Check balance
2. Deposit funds
3. withdraw funds
4. exit
enter your choise:
(1/2/3/4): 2
Enter the amount to deposit: 1000
2000.0$ :الرصيد الجديد .1000.0$ تم إيداع
2000.0$ :رصيد حسابك النهائيAvailable operations:
1. Check balance
2. Deposit funds
3. withdraw funds
4. exit
enter your choise:
(1/2/3/4):
```



```python
            if choice == '1':
                # التحقق من الرصيد
                print(client_socket.recv(1024).decode())
            elif choice == '2':
                # إيداع أموال
                amount = float(input("Enter the amount to deposit: "))
                client_socket.send(str(amount).encode())
                print(client_socket.recv(1024).decode())
            elif choice == '3':
                # سحب أموال
                amount = float(input("Enter the amount to withdraw: "))
                client_socket.send(str(amount).encode())
                print(client_socket.recv(1024).decode())
            elif choice == '4':
                # الخروج
                print(client_socket.recv(1024).decode())
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    except ConnectionRefusedError:
        print("Error: Could not connect to the server.")
    finally:
        client_socket.close()

if __name__ == "__main__":
    main()
```

```
C:\Users\AL-AYHAM\Desktop\pythonProject2\o\Scrip
[+] بانتظار خادم TCP اتصالات جديدة ...
[*] (50558 ,'127.0.0.1') :تم الاتصال بعميل جديد
[*] (50619 ,'127.0.0.1') :تم الاتصال بعميل جديد
```

Question 2:

Voice recognition by deep learning

- نقوم اولا باستيراد المكتبات:

```
[1]  import librosa
     import os
     import numpy as np
     import IPython.display as ipd
     from sklearn.preprocessing import LabelEncoder

     !apt-get -qq install -y utils && pip install np_utils
     !pip install np_utils
     !pip install keras


     from sklearn.model_selection import train_test_split
     from keras.models import load_model
     from keras.layers import Dense, Dropout, Flatten, Conv1D, Input, MaxPooling1D

     from keras.models import Model

     from keras.callbacks import EarlyStopping, ModelCheckpoint

     from keras import backend as K
     !sudo apt-get install libportaudio2
     !pip install sounddevice
     import soundfile as sf
     import sounddevice as sd
     import np_utils
     from keras.utils import to_categorical
     import shutil
```

يتم استيراد جميع المكتبات الضرورية إلى دفتر ملاحظاتنا و منها LibROSA و keras هما مكتبات Python المستخدمة لمعالجة الإشارات الصوتية.

في الخطوة التالية نقوم بحفظ مشروع التدريب في درايف خلال العمل في بيئة كولاب ولحفظ النموذج المدرب

ثم انشاء مجلد data من اجل حفظ الملفات الصوتية التي سنقوم بتدريب النموذج عليها

```
[3]  # in this way ...during we are using collab we are saving (training file for neural network)  on drive
     #to continue training from the last point we reachedin training
     from google.colab import drive

     drive.mount('/content/drive',force_remount=True)
```

```
Mounted at /content/drive
```

```
[4]  # making file and named it "data" to save dataset in it then extract dataset in data file

     # Create the "data" folder inside the parent directory
     data_dir = os.path.join("data")
     os.mkdir(data_dir)

     # Verify the current working directory
     current_directory = os.getcwd()
```

نقوم بعدها بتغيير المسار الى المجلد الذي قمنا بإنشائه و تنزيل ملف التدريب الصوتي من مكتبة tensorflow واستخراجه

```
[5]  # Change the current working directory to "/content/data"
     os.chdir("/content/data")
     !pwd
```

```
/content/data
```

```
[6]  #import training files from the "tensorflow"
     !wget http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz
```

```
--2024-06-11 22:07:44--  http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 172.253.117.207, 142.250.99.207, 142.250.107.207, ...
Connecting to download.tensorflow.org (download.tensorflow.org)|172.253.117.207|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1489096277 (1.4G) [application/gzip]
Saving to: ‘speech_commands_v0.01.tar.gz’

speech_commands_v0. 100%[===================>]   1.39G   100MB/s    in 14s

2024-06-11 22:07:58 (104 MB/s) - ‘speech_commands_v0.01.tar.gz’ saved [1489096277/1489096277]
```

```
[ ]  #extract data in  "/content/data"
     !tar -xvf '/content/data/speech_commands_v0.01.tar.gz'
```

```
./happy/402e2977_nohash_0.wav
./happy/d486fb84_nohash_1.wav
./happy/c38720cb_nohash_0.wav
./happy/b49caed3_nohash_0.wav
./happy/a40c62f1_nohash_0.wav
./happy/d312f481_nohash_4.wav
./happy/7b2e879e_nohash_0.wav
./happy/71aa5b54_nohash_0.wav
./happy/70a00e98_nohash_0.wav
./happy/71f9bba8_nohash_0.wav
./happy/05739450_nohash_0.wav
./happy/4bb1244f_nohash_1.wav
./happy/4249c833_nohash_0.wav
./happy/2d92f18b_nohash_0.wav
./happy/89ed36ab_nohash_1.wav
./happy/f297e878_nohash_0.wav
./happy/c0e0f834_nohash_1.wav
./happy/f875f965_nohash_0.wav
./happy/e72aa705_nohash_0.wav
./happy/0ac15fe9_nohash_0.wav
./happy/cab100c9_nohash_0.wav
./happy/4254621e_nohash_1.wav
./happy/fc28c8d8_nohash_0.wav
./happy/a108341b_nohash_0.wav
./happy/d750966e_nohash_0.wav
./happy/c2df23b2_nohash_0.wav
./happy/c6389ab0_nohash_0.wav
./happy/28e47b1a_nohash_0.wav
./happy/ea7ca285_nohash_0.wav
```

مرحلة تصنيف التسجيلات الصوتية والكلمات (الدخل والخرج)

Insert code cell below
Ctrl+M B

```
h = '/content/data'
iles in the path below
tdir(train_audio_path)

all_wave = []

all_label = []

for label in labels:
    print(label)
    # give what the file have in inside (all files such as (bird has sounds >> enter to it and show me this sound ))

    waves = [f for f in os.listdir(train_audio_path + '/'+ label) if f.endswith('.wav')]
    #after give me all files.wav
    #librosa reading soundfile & return samples & sample rate

    for wav in waves:
        samples, sample_rate = librosa.load(train_audio_path + '/' + label + '/' + wav, sr = 16000)
        samples = librosa.resample(samples, orig_sr=sample_rate, target_sr=8000)
        #samples = librosa.resample(samples, sample_rate, 8000)
        if len(samples)== 8000 :
            all_wave.append(samples)
            all_label.append(label)
# in training we give him input and output
```

```
six
five
three
go
right
```

```
[10] print(all_label)
```

```
['six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'six', 'si
```

```
[11] # from sk learn make data (samples) ready  for inter to neural  network
le = LabelEncoder()
y=le.fit_transform(all_label)#numberd samples
classes= list(le.classes_)
```

```
[12] print(classes)
```

```
['bed', 'bird', 'cat', 'dog', 'down', 'eight', 'five', 'four', 'go', 'happy', 'house', 'left', 'marvin', 'nine', 'no', 'off', 'on', 'one', 'righ
```

```
[13] import np_utils
from keras.utils import to_categorical
y=to_categorical(y, num_classes=len(labels))
```

```
[15] all_wave = np.array(all_wave).reshape(-1,8000,1)
#because list cant inter to neural  network we make all_wave an array  and reshape it
```

```
[17] x_tr, x_val, y_tr, y_val = train_test_split(np.array(all_wave),np.array(y),stratify=y,test_size = 0.2,random_state=777,shuffle=True)
     #sk learn after train go to validation to evalution
```

```
[18] model=load_model('/content/drive/MyDrive/best_model.hdf5')
```

```
K.clear_session()

inputs = Input(shape=(8000,1))

#First Conv1D layer
conv = Conv1D(8,13, padding='valid', activation='relu', strides=1)(inputs)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Second Conv1D layer
conv = Conv1D(16, 11, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Third Conv1D layer
conv = Conv1D(32, 9, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Fourth Conv1D layer
conv = Conv1D(64, 7, padding='valid', activation='relu', strides=1)(conv)
conv = MaxPooling1D(3)(conv)
conv = Dropout(0.3)(conv)

#Flatten layer
conv = Flatten()(conv)

#Dense Layer 1
conv = Dense(256, activation='relu')(conv)
conv = Dropout(0.3)(conv)

#Dense Layer 2
conv = Dense(128, activation='relu')(conv)
conv = Dropout(0.3)(conv)

outputs = Dense(len(labels), activation='softmax')(conv)

model = Model(inputs, outputs)
model.summary()
```

```
[20] model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
[21] from keras.callbacks import EarlyStopping, ModelCheckpoint
     es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10, min_delta=0.0001)
     mc = ModelCheckpoint('/content/best_model.hdf5', monitor='val_acc', verbose=1, save_best_only=True, mode='max')
```

```
[22] history=model.fit(x_tr, y_tr ,epochs=100, callbacks=[es,mc], batch_size=32, validation_data=(x_val,y_val))
```

```
Epoch 1/100
1457/1457 [==============================] - ETA: 0s - loss: 2.9223 - accuracy: 0.1316WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 22s 10ms/step - loss: 2.9223 - accuracy: 0.1316 - val_loss: 2.4325 - val_accuracy: 0.2753
Epoch 2/100
1451/1457 [============================>.] - ETA: 0s - loss: 1.9561 - accuracy: 0.3856WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 13s 9ms/step - loss: 1.9544 - accuracy: 0.3861 - val_loss: 1.5638 - val_accuracy: 0.5224
Epoch 3/100
1454/1457 [============================>.] - ETA: 0s - loss: 1.5113 - accuracy: 0.5282WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 17s 11ms/step - loss: 1.5109 - accuracy: 0.5283 - val_loss: 1.2760 - val_accuracy: 0.6182
Epoch 4/100
1455/1457 [============================>.] - ETA: 0s - loss: 1.3166 - accuracy: 0.5931WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 15s 10ms/step - loss: 1.3168 - accuracy: 0.5930 - val_loss: 1.0431 - val_accuracy: 0.6975
Epoch 5/100
1457/1457 [==============================] - ETA: 0s - loss: 1.1933 - accuracy: 0.6349WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 13s 9ms/step - loss: 1.1933 - accuracy: 0.6349 - val_loss: 0.9504 - val_accuracy: 0.7235
Epoch 6/100
1457/1457 [==============================] - ETA: 0s - loss: 1.0893 - accuracy: 0.6655WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 13s 9ms/step - loss: 1.0893 - accuracy: 0.6655 - val_loss: 0.8178 - val_accuracy: 0.7611
Epoch 7/100
1452/1457 [============================>.] - ETA: 0s - loss: 1.0303 - accuracy: 0.6859WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 13s 9ms/step - loss: 1.0298 - accuracy: 0.6860 - val_loss: 0.8372 - val_accuracy: 0.7547
Epoch 8/100
1456/1457 [============================>.] - ETA: 0s - loss: 0.9926 - accuracy: 0.6970WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 13s 9ms/step - loss: 0.9927 - accuracy: 0.6970 - val_loss: 0.7579 - val_accuracy: 0.7845
Epoch 9/100
1453/1457 [============================>.] - ETA: 0s - loss: 0.9486 - accuracy: 0.7129WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 15s 10ms/step - loss: 0.9486 - accuracy: 0.7128 - val_loss: 0.7764 - val_accuracy: 0.7768
Epoch 10/100
1455/1457 [============================>.] - ETA: 0s - loss: 0.9049 - accuracy: 0.7240WARNING:tensorflow:Can save best model only with val_acc
1457/1457 [==============================] - 15s 10ms/step - loss: 0.9050 - accuracy: 0.7240 - val_loss: 0.7387 - val_accuracy: 0.7914
Epoch 11/100
```

```
[23] model.save(filepath="/content/drive/MyDrive/best_model.hdf5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save
  saving_api.save_model(
```

```
[24] def predict(audio):
         prob=model.predict(audio.reshape(1,8000,1))
         index=np.argmax(prob[0])
         return classes[index]
```

```
[30] samples, sample_rate = librosa.load("/content/data/up/00b01445_nohash_0.wav", sr = 16000)
     samples = librosa.resample(samples, orig_sr=sample_rate, target_sr=8000)
     ipd.Audio(samples, rate=8000)
```


```
[31] print("Text:",predict(samples))
```

```
1/1 [==============================] - 0s 368ms/step
Text: up
```

الشرح :

نقوم باستدعاء جميع المكتبات اللازمة للتعامل مع الصوت ، ونقوم بإنشاء بيئة عمل وربطها مع الدرايف لحفظ النموذج في الدرايف أثناء عمليات التدريب .
ننشئ مجلد لحفظ الملفات الصوتية المراد استخراج الكلام منها .
نقوم بإنشاء قائمتين واحدة للأصوات والأخرى للكلام المقابل
نقوم بقراءة ملفات الصوت من المسار المحدد train_audio_pathو نقوم بتحميلها باستخدام مكتبة librosa. هذه المكتبة تقوم بقراءة ملفات الصوت وإرجاع العينات ومعدل العينات (sample rate). ثم يتم تغيير معدل العينات إلى 8000 باستخدام librosa.resample. إذا كانت العينات تحتوي على 8000 نقطة، يتم إضافتها إلى قائمة all_waveويتم إضافة العلامة المناسبة(الكلمة) إلى قائمة all_label.
يتم إنشاء غرض leباستخدام :LabelEncoder()
LabelEncoderهو مكوّن من مكتبة scikit-learnيستخدم لترميز العلامات (labels) بأرقام.
يتم تعيين العلامات الموجودة في قائمة all_labelإلى الغرض le.
يتم تحويل العلامات إلى أرقام باستخدام :le.fit_transform(all_label)
يتم تعيين الأرقام المقابلة لكل علامة في القائمة all_label.
يتم تخزين هذه الأرقام في المتغير y.
يتم إنشاء قائمة :classes
تحتوي هذه القائمة على العلامات الفعلية (بدون ترميز) الموجودة في all_label.
يتم تحويل الأرقام لتمثيل ثنائي
تحويل قائمة all_waveإلى مصفوفة (array) باستخدام مكتبة NumPy.
تحتوي المصفوفة على نفس العناصر الموجودة في قائمة all_wave.
يتم استدعاء:reshape(-1, 8000, 1).
ومن ثم إعادة تشكيل المصفوفة إلى شكل جديد
البارامتر الأول -1يعني أن الحجم سيتم حسابه تلقائيًا بناءً على الحجم الأصلي للمصفوفة.
البارامترين الآخرين هما 8000 و 1، مما يعني أن المصفوفة ستكون لها شكل (عدد العينات، 8000، 1).
يتم إضافة بعد إضافي (1) للمصفوفة للتأكيد على أنها تمثل البيانات الصوتية.

يتم تقسيم البيانات لمجموعتين (تدريب واختبار) بحيث يكون 20% من البيانات للاختبار .
نقوم بإنشاء ملف لحفظ نموذج التدريب
وانشاء نموذج تدريبي Conv1d عبارة عن شبكة عصبونية تقوم بالالتواء (التدريب و التنفيذ)
ثم نقوم بتدريب النموذج السابق وحفظ النسخة المدربة في درايف و اختباره بتمرير التسجيلات الصوتية وقراءتها وطباعة الكلمة الموافقة لها .