**Pandas**

1. List to dataframe

a=[[1,2,3,4,5],[6,7,8,9,0]]

a=pd.dataframe(a,columns=['a','b'])

1. Read csv

df=pd.read\_csv("167.Movie IMDb Data Analyzer.csv")

1. Drop duplicate

customers = customers.drop\_duplicates(subset="email", keep="first")

1. Drop none value

students.dropna(subset=['name'])

1. Rename columns

students.rename(columns={

'id': 'student\_id','first':'first\_name','last':'last\_name','age':'age\_in\_years'},

inplace=True)

1. Change type of columns

students["grade"] = students["grade"].astype(int)

1. Replace none value

products["quantity"] = products["quantity"].fillna(0)

1. Concation of 2 data frame

return pd.concat([df1, df2], ignore\_index=True)

1. Lenth

list(players.shape)

1. First three rows

employees[:3]

1. select a person with student id 101

return students.loc[students['student\_id']==101,['name','age']]

1. creating column

employees["bonus"]=employees['salary']\*2

1. updating column

employees['salary']=employees['salary']\*2

# Pivot the table to make cities the columns and months the rows

result = weather.pivot(index="month", columns="city", values="temperature")

# Reset the index to make 'month' a column and sort by month name if needed

result = result.reset\_index()

return result

# Melt the dataframe to transform quarters into rows

result = report.melt(id\_vars=["product"], var\_name="quarter", value\_name="sales")

return result

return animals[animals["weight"] > 100].sort\_values(by="weight", ascending=False)[["name"]]

**PyPDF2**

1. Open file

with open("198.Extract Text from PDF.pdf","rb") as file:

reader=PyPDF2.PdfReader(file)

1. Read pages

reader.pages

1. Extract text

page.extract\_text()

**File handling**

1. Read file

open("196.Favorite Quote Collector.txt",'r')

1. Read lines

k=f.readlines()

**Random**

1. random.choice(list)

**Pyqt**

import sys

from PyQt6.QtWidgets import QMainWindow,QLabel,QApplication,QTextEdit,QPushButton, QComboBox

1. create a window

app=QApplication(sys.argv)

window=QMainWindow()

window.setWindowTitle("voice typing notepad")

window.setGeometry(500,500,600,700)

window.show()

sys.exit(app.exec())

1. lable

lable=QLabel("voise typing notepad",window)

lable.setGeometry(10,10,400,30)

to change the text "voise typing notepad"

label.setText("no text to save!")

1. toload

QApplication.processEvents()

1. to create text box

textbox=QTextEdit(window)

to append some text

textbox.append(text)

1. button

listenbutton=QPushButton("start listening 🎤",window)

listenbutton.clicked.connect(lambda: hear(lable,textbox))

1. to exit

app.quit

1. combo box(select)

combobox=QComboBox(window)

combobox.addItems(["select your emotion","Happy","Sad","Energetic"])

1. get text from combobox

text=combobox.currentText()

**Speech recognize**

Each Recognizer instance has seven methods for recognizing speech from an audio source using various APIs. These are:

recognize\_bing(): Microsoft Bing Speech

recognize\_google(): Google Web Speech API

recognize\_google\_cloud(): Google Cloud Speech - requires installation of the google-cloud-speech package

recognize\_houndify(): Houndify by SoundHound

recognize\_ibm(): IBM Speech to Text

recognize\_sphinx(): CMU Sphinx - requires installing PocketSphinx

recognize\_wit(): Wit.ai

Of the seven, only recognize\_sphinx() works offline with the CMU Sphinx engine.

The other six all require an internet connection.

import speech\_recognition as sr

1. r=sr.Recognizer()
2. mic = sr.Microphone()
3. to print micphone avaliable

print(sr.Microphone.list\_microphone\_names())

1. Adjusting for ambient noise

r.adjust\_for\_ambient\_noise(source)

1. Listen

audio = r.listen(source, timeout=10)

text = r.recognize\_google(audio)

**Time**

import time

1. Time.sleep(seconds)

**Datetime**

import datetime

1. Current day

today = datetime.date.today()

current\_year = today.year

1. string to formate

fest\_date = datetime.datetime.strptime(f"{current\_year}-{date}", "%Y-%m-%d").date()

time=datetime.datetime.now().strftime("%Y-%m-%D %H:%M:%S")

1. to get day

current=(datetime.datetime.now().strftime("%A")).lower()

**Seaborn**

import seaborn as sns

1. sns.histplot(ratings)
2. sb.lineplot(x=x,y=y)

**Matplotlib**

import matplotlib.pyplot as plt

1. plt.title("distribution of movie ratings")
2. plt.xlabel("rating")
3. plt.ylabel("frequency")
4. plt.show()

**api**

1. currency exchange

url = f"https://api.exchangerate-api.com/v4/latest/{currency}"

1. weather

url =f”[https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API\_KEY}&units=metric](https://api.openweathermap.org/data/2.5/weather?q=%7bcity%7d&appid=%7bAPI_KEY%7d&units=metric)”

1. http://127.0.0.1:8000/forecast/London

**Requests**

import requests

1. response = requests.get(url)
2. response.json()

**re**

import re

1. re.search(r"[A-Z]", password))
2. re.search(r"[a-z]", password))
3. re.search(r"\d", password))
4. re.search(rf"[{re.escape(string.punctuation)}]", password)

**string**

import string

1. string.ascii\_uppercase
2. string.ascii\_lowercase
3. string.punctuation
4. string.digits