Python advanced

*# import pandas as pd  
# import numpy as np  
# import matplotlib.pyplot as plt  
# import seaborn as sns  
# from matplotlib.patches import StepPatch  
# np.random.seed(0)  
# h,edges=np.histogram(np.random.normal(5,3,5000),bins=np.linspace(0,10,20))  
# fig,axs=plt.subplots(3,1,figsize=(7,15))  
# axs[0].stairs(h,edges,label='simple histogram')  
# axs[0].stairs(h,edges+5,baseline=50,label='Modified histogram')  
# axs[0].stairs(h,edges+10,baseline=None,label='No edges')  
# axs[0].set\_title("Step Histograms")  
# for ax in axs:  
# ax.legend()  
# plt.show()  
# print("hello"[0])  
# num\_char=len(input("what is your name ?"))  
# new\_num\_char=str(num\_char)  
# print("your name has " + new\_num\_char + " characters.")  
# # print(type(num\_char))  
# a=str(123)  
# print(type(a))  
# a=float(123)  
# print(type(a))  
# print(70+ float("100.5"))# 170.5  
# print(str(70)+ str(100.5))#70100.5  
# enter a two digit number and add their digits with each other  
# two\_digit\_number=input("enter two digit numbers")  
# first\_digit=int(two\_digit\_number[0])  
# second\_digit=int(two\_digit\_number[1])  
# result=first\_digit+second\_digit  
# print(result)  
# print(type(9/3))  
# print(3\*3+3/3-3)  
# print(3\*1+9/3-3)  
# print(3\*(3+3)/3-3)  
# How to calculate Body index rom a person weight and Height  
#BMI=weight(kg)/Height^2(m)  
# height=input("enter your height in m:")  
# weight=input("enter your weight in kg:")  
# new\_height=float(height)  
# new\_weight=float(weight)  
# new\_height2=pow(new\_height,2)  
# BMI= new\_weight / new\_height2  
# print(BMI)  
#OR  
# height=input("enter your height in m:")  
# weight=input("enter your weight in kg:")  
# BMI= int(weight) / float(height)\*\*2  
# bmi\_int=int(BMI)  
# print(bmi\_int)  
# how to use round functions  
# print(round(8/3))  
# print(round(8/3,2))  
# print(round(8.34582673533,2))  
# # use of floor  
# print(8//3)  
# result=8//3  
# result //=2  
# print(result)  
# score=0  
# score +=1  
# score -=1  
# print("This is your score "+str(score))  
# score=0  
# height=1.0  
# iswinning=True  
# f-string  
# print(f"your score is {score},your height is {height},you are winning{iswinning}")  
# how to calculate your life from 90 years life expectation  
# age=input("what is your current age in normal year ? : ")  
# age\_as\_int=int(age)  
# years\_remainaing=90 - age\_as\_int  
# days\_remainaing=years\_remainaing\*365  
# weeks\_remaining=years\_remainaing\*52  
# months\_remaining=years\_remainaing\*12  
# hours\_remaining=years\_remainaing\*8760  
# minutes\_remaining=years\_remainaing\*525600  
# secounds\_remaining=years\_remainaing\*31536000  
# message=f"You have {days\_remainaing} days,{weeks\_remaining} weeks, " \  
# f"{weeks\_remaining} months,{hours\_remaining} hours ," \  
# f"{minutes\_remaining} minutes,and {secounds\_remaining} seconds left."  
# print(message)  
# now build a tip calculator  
# print("Welcome to the tip calculator !")  
# bill=float(input("What was the total bill? $"))  
# tip=int(input("How much tip would like to give? 10,12,or 15? "))  
# people=int(input("How many people to splits the bill?"))  
# tip\_as\_percent=tip /100  
# total\_tip\_amount=bill\*tip\_as\_percent  
# total\_bill=bill+total\_tip\_amount  
# # print(total\_bill)  
# bill\_per\_person=total\_bill / people  
# final\_amount=round(bill\_per\_person,2)  
# final\_amount="{:.2f}".format(bill\_per\_person)  
# print(f"Each person should pay: $ {final\_amount}")  
# Treasure islands  
#this is given below  
# print('''  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# | | | |  
# \_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.=""\_;=.\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_  
# | | ,-"\_,="" `"=.| |  
# |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_"=.\_o`"-.\_ `"=.\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
# | `"=.\_o`"=.\_ \_`"=.\_ |  
# \_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:=.\_o "=.\_."\_.-="'"=.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_  
# | | \_\_.--" , ; `"=.\_o." ,-"""-.\_ ". |  
# |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_.\_" ,. .` ` `` , `"-.\_"-.\_ ". '\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
# | |o`"=.\_` , "` `; .". , "-.\_"-.\_; ; |  
# \_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_| ;`-.o`"=.\_; ." ` '`."\` . "-.\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_  
# | | |o; `"-.o`"=.\_`` '` " ,\_\_.--o; |  
# |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_| ; (#) `-.o `"=.`\_.--"\_o.-; ;\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
# \_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_|o;.\_ " `".o|o\_.--" ;o;\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_  
# /\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_"=.\_o--.\_ ; | ; ; ;/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_  
# \_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_"=.\_o--.\_ ;o|o; \_.\_;o;\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_  
# /\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_"=.\_o.\_; | ;\_.--"o.--"\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_  
# \_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_"=.o|o\_.--""\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_  
# /\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_ /  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# ''')  
# print("Welcome to Treasure Island.")  
# print("Your mission is to find the treasure.")  
#  
# #Write your code below this line   
#  
# choice1 = input('You\'re at a cross road. Where do you want to go? Type "left" or "right" \n').lower()  
# if choice1 == "left":  
# choice2 = input('You\'ve come to a lake. There is an island in the middle of the lake. Type "wait" to wait for a boat. Type "swim" to swim across. \n').lower()  
# if choice2 == "wait":  
# choice3 = input("You arrive at the island unharmed. There is a house with 3 doors. One red, one yellow and one blue. Which colour do you choose? \n").lower()  
# if choice3 == "red":  
# print("It's a room full of fire. Game Over.")  
# elif choice3 == "yellow":  
# print("You found the treasure! You Win!")  
# elif choice3 == "blue":  
# print("You enter a room of beasts. Game Over.")  
# else:  
# print("You chose a door that doesn't exist. Game Over.")  
# else:  
# print("You get attacked by an angry trout. Game Over.")  
# else:  
# print("You fell into a hole. Game Over.")  
# print("Welcome to the rollercoaster!")  
# height=int(input("What is your height in cm? "))  
# if height!=120:  
# print("You can ride the rollercoaster!")  
# else:  
# print("Sorry,you have to grow taller before you can ride. ")  
# n=int(input("which number do you want to check "))  
# if n%2== 1:  
# print("This is odd number ")  
# elif n%2 == 0:  
# print("This is even number")  
# else:  
# print("the number is zero")  
# nested if else statments  
# print("Welcome to the rollercoaster!")  
# height=int(input("What is your height in cm? "))  
# if height>=120:  
# print("You can ride the rollercoaster!")  
# age=int(input("What is your age?"))  
# if age<12:  
# print("Please pay $5")  
# elif age<=18:  
# print("Please pay $7")  
# else:  
# print("Please pay $12 ")  
# else:  
# print("Sorry,you have to grow taller before you can ride. ")  
# height=float(input("enter your height in m: "))  
# weight=float(input("enter your weight in kg: "))  
# bmi=round(weight / height\*\*2)  
# if bmi<18.5:  
# print(f"Your bmi is {bmi},you are underweight")  
# elif bmi<25:  
# print(f"your bmi is {bmi},you have a normal weight.")  
# elif bmi < 30:  
# print(f"your bmi is {bmi},you are overweight.")  
# elif bmi < 35:  
# print(f"your bmi is {bmi},you are obese.")  
# else:  
# print(f"your bmi is {bmi},you are clinically obese.")  
# Data analysis by using pandas  
# import pandas as pd  
  
# xyz\_web={'Day':[1,2,3,4,5,6],"Visitors":[1000,700,6000,1000,400,350],'Bounce\_Rate':[20,20,23,15,10,34]}  
# df=pd.DataFrame(xyz\_web)  
# # print(df)  
# # print(df.head(2))  
# print(df.tail(2))  
# df1=pd.DataFrame({"HPI":[80,90,70,60],"int\_rate":[2,1,2,3],"IND\_GDP":[50,45,45,67]},index=  
# [2001,2002,2003,2004])  
# df2=pd.DataFrame({"HPI":[80,90,70,60],"int\_rate":[2,1,2,3],"IND\_GDP":[50,45,45,67]},index=  
# [2005,2006,2007,2008])  
# # merge=pd.merge(df1,df2)  
# # print(merge)  
# merge=pd.merge(df1,df2,on="HPI")  
# print(merge)  
# perform joining operator  
# df1=pd.DataFrame({"int\_rate":[2,1,2,3],"IND\_GDP":[50,45,45,67]},index=  
# [2001,2002,2003,2004])  
# df2=pd.DataFrame({"Low\_Tier\_HPI":[50,45,67,34],"Unemployment":[1,3,5,6]},index=  
# [2001,2003,2004,2004])  
# joined=df1.join(df2)  
# print(joined)  
# import matplotlib.pyplot as plt  
# from matplotlib import style  
# style.use("fivethirtyeight")  
# df=pd.DataFrame({"Day":[1,2,3,4],"Visitors":[200,100,230,300],"Bounce\_Rate":[20,45,60,10]})  
# # df.set\_index("Day",inplace=True)  
# # df.plot()  
# # plt.show()  
# # change header  
# df=df.rename(columns={"Visitors":"Users"})  
# print(df)  
#how to use concatenation operator  
# df1=pd.DataFrame({"HPI":[80,90,70,60],"int\_rate":[2,1,2,3],"IND\_GDP":[50,45,45,67]},index=  
# [2001,2002,2003,2004])  
# df2=pd.DataFrame({"HPI":[80,90,70,60],"int\_rate":[2,1,2,3],"IND\_GDP":[50,45,45,67]},index=  
# [2005,2006,2007,2008])  
# concat=pd.concat([df1,df2])  
# print(concat)  
# country=pd.read\_csv('D:\Documentnust\Sem10\PHP Developer\php practice\sample.csv',index\_col=0)  
# country.to\_html('sample.html')  
# country = pd.read\_csv('D:\Documentnust\Sem10\PHP Developer\php practice\sample.csv')  
# df=country.head(5)  
# # print(df)  
# df=country.head(5)  
# # df=df.set\_index(['WO'])  
# sd=df.reindex(columns=['District','Service'])  
# db=sd.diff(axis=1)  
# db.plot(kind='bar')  
# plt.show()  
# from statistics import mean  
# from statistics import median  
# from statistics import mode  
# from statistics import variance  
# print(mean([1,1,1,1,3,4,4,5,2]))  
# print(median([1,1,1,2,2]))  
# print(mode([1,1,1,2,2]))  
# print(variance([1,1,1,2,2]))  
# lets try some examples  
# df1 = pd.DataFrame(  
# {  
# "A": ["A0", "A1", "A2", "A3"],  
# "B": ["B0", "B1", "B2", "B3"],  
# "C": ["C0", "C1", "C2", "C3"],  
# "D": ["D0", "D1", "D2", "D3"],  
# },index=[0, 1, 2, 3],  
# )  
# df2 = pd.DataFrame(  
# {  
# "A": ["A4", "A5", "A6", "A7"],  
# "B": ["B4", "B5", "B6", "B7"],  
# "C": ["C4", "C5", "C6", "C7"],  
# "D": ["D4", "D5", "D6", "D7"],},index=[4, 5, 6, 7], )  
# df3 = pd.DataFrame(  
# {  
# "A": ["A8", "A9", "A10", "A11"],  
# "B": ["B8", "B9", "B10", "B11"],  
# "C": ["C8", "C9", "C10", "C11"],  
# "D": ["D8", "D9", "D10", "D11"],  
# },  
# index=[8, 9, 10, 11],  
# )  
# print(df1)  
# print(df2)  
# print(df3)  
# frames = [df1, df2, df3]  
# result = pd.concat(frames)  
# print(result)  
# result=pd.concat(  
# objs,  
# axis=0,  
# join="outer",  
# ignore\_index=False,  
# keys=None,  
# levels=None,  
# names=None,  
# verify\_integrity=False,  
# copy=True,)  
# result = pd.concat(frames,keys=["x","y","z"])  
# print(result)  
# print(result.loc["y"])  
#------------------------------------------------------------------------------------------------------------------------------  
# -----------------------------------------------------------pandas visvualization  
# df1 = pd.DataFrame(  
# {  
# "A": ["A0", "A1", "A2", "A3"],  
# "B": ["B0", "B1", "B2", "B3"],  
# "C": ["C0", "C1", "C2", "C3"],  
# "D": ["D0", "D1", "D2", "D3"],  
# },index=[0, 1, 2, 3],  
# )  
# df2 = pd.DataFrame(  
# {  
# "A": ["A4", "A5", "A6", "A7"],  
# "B": ["B4", "B5", "B6", "B7"],  
# "C": ["C4", "C5", "C6", "C7"],  
# "D": ["D4", "D5", "D6", "D7"],},index=[4, 5, 6, 7], )  
# df3 = pd.DataFrame(  
# {  
# "A": ["A8", "A9", "A10", "A11"],  
# "B": ["B8", "B9", "B10", "B11"],  
# "C": ["C8", "C9", "C10", "C11"],  
# "D": ["D8", "D9", "D10", "D11"],  
# },  
# index=[8, 9, 10, 11],  
# )  
# df4 = pd.DataFrame(  
# {  
# "B": ["B2", "B3", "B6", "B7"],  
# "D": ["D2", "D3", "D6", "D7"],  
# "F": ["F2", "F3", "F6", "F7"],  
# },  
# index=[2, 3, 6, 7],  
# )  
# frames = [df1, df2, df3]  
# result=pd.concat([df1,df4.reindex(df1.index)],axis=1)  
# print(result)  
# print(df4)  
# result=pd.concat([df1,df4],axis=1)  
# result=pd.concat([df1,df4],axis=1,join='inner')# for inner join that is for intersection  
# suppose we just wanted to reuse the exact index from the original DataFrame  
# result=pd.concat([df1,df4],axis=1,join='inner').reindex(df1.index)  
# result=pd.concat([df1,df4.reindex(df1.index)],axis=1)  
# print(result)  
#------------------------------------------------------------------------------------------------------------------------------------------  
# ------------------------------------------------------------->Data Cleaning and Analyzing<------------------------------------  
# df=pd.read\_csv('D:\Documentnust\Sem10\PHP Developer\PythonDeveloper\Taks\Day2\data.csv')  
# print(df.to\_string())  
# print(df)  
# print(df.loc[[0,1]])  
# system max  
# print(df.info())  
# new\_df=df.dropna()# To Remove empty values  
# df.dropna(inplace=True)# To change in origional  
# df.fillna(130,inplace=True)# replace empty by values  
#Replace NULL values in the "Calories" columns with the number 130:  
# df["Calories"].fillna(130, inplace = True)  
#A common way to replace empty cells, is to calculate  
# the mean, median or mode value of the column  
# x=df["Calories"].mean()  
# x = df["Calories"].median()  
# x = df["Calories"].mode()[0]  
# # df["Calories"].fillna(x, inplace = True)  
# df["Calories"].fillna(x, inplace = True)  
# df=pd.DataFrame(df)  
# df['Date'] = pd.to\_datetime(df['Date'])  
# df.dropna(subset=['Date'], inplace = True)  
# df.loc[160,'Duration']=0  
# df.loc[166,'Duration']='ajman' # for analyzing Wrong Format  
# df['Duration'] = pd.to\_numeric(df['Duration'])  
# df.loc[167,'Duration']='Nan'  
# df.dropna(subset=['Duration'], inplace = True)# To remove null values  
# ---------------------------------------to loop through index  
# for x in df.index:  
# if df.loc[x,'Duration'] > 120:  
# df.loc[x,'Duration']=120  
# print(df.to\_string())  
# ---------------------------------How to delete Rows-------------------------------------------------  
# for x in df.index:  
# if df.loc[x,'Duration'] >= 120:  
# df.drop(x,inplace=True)  
# print(df.to\_string())  
# ----------------------------------Invesigate and remove duplicated rows --------------------------------------------  
# print(df.duplicated())  
# print(df.drop\_duplicates(inplace=True))  
#----------------------------------------------------------------------------------------------------------  
# print(df.corr())  
# df.plot()  
# plt.show()  
# df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')  
# df["Duration"].plot(kind = 'hist')  
# plt.show()  
# import numpy as np  
# xpoints = np.array([0, 6,7,8])  
# ypoints = np.array([0, 250,260,270])  
# plt.plot(xpoints, ypoints)  
# plt.show()  
# ypoints = np.array([3, 8, 1, 10, 5, 7])  
# plt.plot(ypoints,'\*:r',ms=20,mec='r')  
# plt.show()  
# ypoints = np.array([3, 8, 1, 10])  
# plt.plot(ypoints, linestyle = 'dotted')  
# plt.show()  
# ypoints = np.array([3, 8, 1, 10])  
# plt.plot(ypoints, ls='--')  
# plt.show()  
# ypoints = np.array([3, 8, 1, 10])  
# plt.plot(ypoints, linestyle='dotted')  
# plt.show()  
# xpoint=np.array([1,8,7,9])  
# ypoiint=np.array([3,9,11,13])  
# plt.plot(xpoint,ypoiint,color='r')  
# plt.show()  
# x=np.array([23,24,25,26,27,28,29,31,32])  
# y=np.array([34,35,36,37,38,39,40,41,42])  
#day one, the age and speed of 13 cars:  
# x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])  
# y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  
# plt.scatter(x, y)  
# #day two, the age and speed of 15 cars:  
# x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])  
# y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])  
# plt.scatter(x, y)  
# plt.show()  
# another examples with color  
# day one, the age and speed of 13 cars:  
# x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])  
# y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  
# plt.scatter(x, y,color='hotpink')  
# #day two, the age and speed of 15 cars:  
# x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])  
# y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])  
# plt.scatter(x, y,color='#88c999')  
# plt.show()  
# lets take another examples of scatter plot  
# x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])  
# y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  
# colors = np.array(["red","green","blue","yellow","pink","black","orange","purple","beige","brown","gray","cyan","magenta"])  
# plt.scatter(x, y, c=colors)  
# plt.show()  
# use of cmap  
# x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])  
# y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  
# colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])  
# plt.scatter(x, y, c=colors, cmap='viridis')  
# plt.colorbar()  
# plt.show()  
# x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])  
# y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  
# colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])  
# plt.scatter(x, y, c=colors, cmap='BuGn')  
# plt.colorbar()  
# plt.show()  
# x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])  
# y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  
# colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])  
# plt.scatter(x, y, c=colors, cmap='Purples')  
# plt.colorbar()  
# plt.show()  
# sizes of colors  
# x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])  
# y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  
# sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])  
# #You can adjust the transparency of the dots with the alpha argument.  
# plt.scatter(x, y, s=sizes,alpha=0.5)  
# plt.show()  
# x=np.random.randint(100,size=(100))  
# y=np.random.randint(100,size=(100))  
# colors=np.random.randint(100,size=(100))  
# sizes=10\*np.random.randint(100,size=(100))  
# plt.scatter(x,y,c=colors,s=sizes,cmap='nipy\_spectral',alpha=0.5)  
# plt.colorbar()  
# plt.suptitle("Ajman engineer Data analyst")  
# plt.show()  
# x=np.random.randint(100,size=(100))  
# y=np.random.randint(100,size=(100))  
# colors=np.random.randint(100,size=(100))  
# sizes=10\*np.random.randint(100,size=(100))  
# plt.scatter(x,y,c=colors,s=sizes,alpha=0.5,cmap='nipy\_spectral')  
# plt.colorbar()  
# plt.suptitle("Ajman Engineer")  
# plt.show()  
# x=np.array(["A","B","C","D"])  
# y=np.array([3,8,1,10])  
# x=["Apples","Bannana"]  
# y=[400,350]  
# plt.bar(x,y)  
# x=["Apples","Bannana","Mangoes"]  
# y=[400,350,300]  
# plt.barh(x,y)  
# plt.show()  
# x=["Apples","Bannana","Mangoes"]  
# y=[400,350,300]  
# plt.bar(x,y,color="red",width=0.1)  
# plt.show()  
# x=np.random.normal(170,10,250)  
# # print(x)  
# plt.hist(x)  
# plt.show()  
# y=np.array([35,25,25,15])  
# mylabels=["Apples","bananas","cherries","plates"]  
# plt.pie(y,labels=mylabels)  
# plt.show()  
# y=np.array([35,25,25,15])  
# mylabels=["Apples","bananas","cherries","plates"]  
# myexplodes=[0.2,0,0,0]  
# plt.pie(y,labels=mylabels,startangle=90,explode=myexplodes)  
# plt.legend()  
# plt.show()  
# y=np.array([35,25,25,15])  
# mylabels=["Apples","bananas","cherries","plates"]  
# myexplodes=[0.2,0,0,0]  
# plt.pie(y,labels=mylabels,explode=myexplodes,shadow=True)  
# plt.legend()  
# plt.show()  
# y=np.array([35,25,30,15])  
# mylabels=["Ajman","kalim","Irfan","Salam"]  
# mycolors=["hotpink","c","m","y"]  
# myexplodes=[0.2,0,0,0]  
# plt.pie(y,labels=mylabels,explode=myexplodes,shadow=True)  
# plt.legend()  
# plt.show()  
# y=np.array([35,25,30,15])  
# mylabels=["Ajman","kalim","Irfan","Salam"]  
# mycolors=["hotpink","c","m","y"]  
# myexplodes=[0.1,0,0,0]  
# plt.pie(y,labels=mylabels,explode=myexplodes,shadow=True,startangle=120)  
# plt.legend(title="Four Engineers")  
# plt.suptitle("Engineer Family")  
# plt.show()*from scipy import constants  
*# print(constants.liter)  
# print(constants.pi)  
# print(dir(constants))  
# print(constants.yotta) #1e+24  
# print(constants.zetta) #1e+21  
# print(constants.exa) #1e+18  
# print(constants.peta) #1000000000000000.0  
# print(constants.tera) #1000000000000.0  
# print(constants.giga) #1000000000.0  
# print(constants.mega) #1000000.0  
# print(constants.kilo) #1000.0  
# print(constants.hecto) #100.0  
# print(constants.deka) #10.0  
# print(constants.deci) #0.1  
# print(constants.centi) #0.01  
# print(constants.milli) #0.001  
# print(constants.micro) #1e-06  
# print(constants.nano) #1e-09  
# print(constants.pico) #1e-12  
# print(constants.femto) #1e-15  
# print(constants.atto) #1e-18  
# print(constants.zepto) #1e-21  
# print(constants.kibi) #1024  
# print(constants.mebi) #1048576  
# print(constants.gibi) #1073741824  
# print(constants.tebi) #1099511627776  
# print(constants.pebi) #1125899906842624  
# print(constants.exbi) #1152921504606846976  
# print(constants.zebi) #1180591620717411303424  
# print(constants.yobi)  
# Return the specified mass  
# print(constants.gram) #0.001  
# print(constants.metric\_ton) #1000.0  
# print(constants.grain) #6.479891e-05  
# print(constants.lb) #0.45359236999999997  
# print(constants.pound) #0.45359236999999997  
# print(constants.oz) #0.028349523124999998  
# print(constants.ounce) #0.028349523124999998  
# print(constants.stone) #6.3502931799999995  
# print(constants.long\_ton) #1016.0469088  
# print(constants.short\_ton) #907.1847399999999  
# print(constants.troy\_ounce) #0.031103476799999998  
# print(constants.troy\_pound) #0.37324172159999996  
# print(constants.carat) #0.0002  
# print(constants.atomic\_mass) #1.66053904e-27  
# print(constants.m\_u) #1.66053904e-27  
# print(constants.u) #1.66053904e-27  
# print(constants.gram)  
# print(constants.pound)  
# print(constants.lb)  
# print(constants.degree)  
# print(constants.arcmin)  
# print(constants.arcsecond)  
# print(constants.week)#604800  
# print(constants.year)#31536000.0  
# print(constants.inch) #0.0254  
# print(constants.foot) #0.30479999999999996  
# print(constants.yard) #0.9143999999999999  
# print(constants.mile) #1609.3439999999998  
# print(constants.mil) #2.5399999999999997e-05  
# print(constants.pt) #0.00035277777777777776  
# print(constants.point) #0.00035277777777777776  
# print(constants.survey\_foot) #0.3048006096012192  
# print(constants.survey\_mile) #1609.3472186944373  
# print(constants.nautical\_mile) #1852.0  
# print(constants.fermi) #1e-15  
# print(constants.angstrom) #1e-10  
# print(constants.micron) #1e-06  
# print(constants.au) #149597870691.0  
# print(constants.astronomical\_unit) #149597870691.0  
# print(constants.light\_year) #9460730472580800.0  
# print(constants.parsec)  
# print(constants.atm)#101325.0  
# print(constants.hectare)  
# print(constants.liter) #0.001  
# print(constants.litre) #0.001  
# print(constants.gallon) #0.0037854117839999997  
# print(constants.gallon\_US) #0.0037854117839999997  
# print(constants.gallon\_imp) #0.00454609  
# print(constants.fluid\_ounce) #2.9573529562499998e-05  
# print(constants.fluid\_ounce\_US) #2.9573529562499998e-05  
# print(constants.fluid\_ounce\_imp) #2.84130625e-05  
# print(constants.barrel) #0.15898729492799998  
# print(constants.bbl)  
# print(constants.kmh) #0.2777777777777778  
# print(constants.mph) #0.44703999999999994  
# print(constants.mach) #340.5  
# print(constants.speed\_of\_sound) #340.5  
# print(constants.knot) #0.5144444444444445  
# print(constants.zero\_Celsius) #273.15  
# print(constants.degree\_Fahrenheit) #0.5555555555555556  
# print(constants.eV) #1.6021766208e-19  
# print(constants.electron\_volt) #1.6021766208e-19  
# print(constants.calorie) #4.184  
# print(constants.calorie\_th) #4.184  
# print(constants.calorie\_IT) #4.1868  
# print(constants.erg) #1e-07  
# print(constants.Btu) #1055.05585262  
# print(constants.Btu\_IT) #1055.05585262  
# print(constants.Btu\_th) #1054.3502644888888  
# print(constants.ton\_TNT) #4184000000.0  
# print(constants.Btu)  
# print(constants.hp) #745.6998715822701  
# print(constants.horsepower)#745.6998715822701  
# # print(constants.dyn)  
# print(constants.dyn) #1e-05  
# print(constants.dyne) #1e-05  
# print(constants.lbf) #4.4482216152605  
# print(constants.pound\_force) #4.4482216152605  
# print(constants.kgf) #9.80665  
# print(constants.kilogram\_force) #9.80665  
# from scipy.optimize import root  
# from math import cos  
# def equ(x):  
# return x+cos(x)  
# myroot=root(equ,0)  
# print(myroot.x)  
# print(myroot)  
# from scipy.optimize import minimize  
# def equ(x):  
# return x\*\*2+x+2  
# mymin=minimize(equ,0,method='BFGS')  
# print(mymin)  
# from scipy.sparse import csr\_matrix  
# arr=np.array([0,0,0,0,0,1,1,2,2,3,3])  
# arr = np.array([[0, 0, 0], [0, 0, 1], [1, 0, 2]])  
# # print(csr\_matrix(arr).data)  
# # print(csr\_matrix(arr).count\_nonzero())  
# mat=csr\_matrix(arr)  
# # mat.eliminate\_zeros()  
# mat.sum\_duplicates()  
# print(mat)  
# arr = np.array([[0, 0, 0], [0, 0, 1], [1, 0, 2]])  
# newarr = csr\_matrix(arr).tocsc()  
# print(newarr)  
# from scipy.sparse.csgraph import connected\_components  
# from scipy.sparse import csr\_matrix  
# arr = np.array([  
# [0, 1, 2],  
# [1, 0, 0],  
# [2, 0, 0]  
# ])  
# newarr = csr\_matrix(arr)  
# print(connected\_components(newarr))  
# from scipy.sparse.csgraph import connected\_components  
# from scipy.sparse.csgraph import dijkstra  
# from scipy.sparse import csr\_matrix  
# arr = np.array([  
# [0, 1, 2],  
# [1, 0, 0],  
# [2, 0, 0]  
# ])  
# newarr = csr\_matrix(arr)  
# print(dijkstra(newarr,return\_predecessors=True,indices=0))  
# from scipy.sparse.csgraph import connected\_components  
# from scipy.sparse.csgraph import floyd\_warshall  
# from scipy.sparse import csr\_matrix  
# arr = np.array([  
# [0, 1, 2],  
# [1, 0, 0],  
# [2, 0, 0]  
# ])  
# newarr = csr\_matrix(arr)  
# print(floyd\_warshall(newarr,return\_predecessors=True,indices=0))  
# from scipy.sparse.csgraph import connected\_components  
# from scipy.sparse.csgraph import bellman\_ford  
# from scipy.sparse import csr\_matrix  
# arr = np.array([  
# [0, -1, 2],  
# [1, 0, 0],  
# [2, 0, 0]  
# ])  
# newarr = csr\_matrix(arr)  
# print(bellman\_ford(newarr, return\_predecessors=True, indices=0))  
# import numpy as np  
# from scipy.sparse.csgraph import depth\_first\_order  
# from scipy.sparse import csr\_matrix  
# arr = np.array([  
# [0, 1, 0, 1],  
# [1, 1, 1, 1],  
# [2, 1, 1, 0],  
# [0, 1, 0, 1]  
# ])  
# newarr = csr\_matrix(arr)  
# print(depth\_first\_order(newarr, 1))  
# import numpy as np  
# from scipy.sparse.csgraph import breadth\_first\_order  
# from scipy.sparse import csr\_matrix  
# arr = np.array([  
# [0, 1, 0, 1],  
# [1, 1, 1, 1],  
# [2, 1, 1, 0],  
# [0, 1, 0, 1]  
# ])  
# newarr = csr\_matrix(arr)  
# print(breadth\_first\_order(newarr, 1))  
# import numpy as np  
# from scipy.spatial import Delaunay  
# import matplotlib.pyplot as plt  
# points=np.array([[2,4],[3,4],[3,0],[2,2],[4,1]])  
# simplices=Delaunay(points).simplices  
# plt.triplot(points[:, 0], points[:, 1], simplices)  
# plt.scatter(points[:, 0], points[:, 1], color='r')  
# plt.show()  
# import numpy as np  
# from scipy.spatial import ConvexHull  
# import matplotlib.pyplot as plt  
# points = np.array([  
# [2, 4],  
# [3, 4],  
# [3, 0],  
# [2, 2],  
# [4, 1],  
# [1, 2],  
# [5, 0],  
# [3, 1],  
# [1, 2],  
# [0, 2]  
# ])  
# hull = ConvexHull(points)  
# hull\_points = hull.simplices  
# plt.scatter(points[:,0], points[:,1])  
# for simplex in hull\_points:  
# plt.plot(points[simplex,0], points[simplex,1], 'k-')  
# plt.show()  
# from scipy.spatial import KDTree  
# points = [(1, -1), (2, 3), (-2, 3), (2, -3)]  
# kdtree = KDTree(points)  
# res = kdtree.query((1, 1))  
# print(res)  
#----------------------------------------------------------Functions ------------------------------------  
# def myfunc():  
# print("This is my first functions ")  
# myfunc()  
# def myfunc(firstname):  
# print("first name"+" "+ firstname+" "+ "Refrences")  
# myfunc("Jamil")  
# myfunc("Haris")  
# myfunc("Waqar")  
# def myfunc(fname,mname,lname):  
# print(f"{fname} {mname} {lname}")  
# myfunc("waqar","ullah","khan")  
# myfunc("waqas","ullah","khan")  
# myfunc("zubair","ullah","khan")  
# myfunc("zanan","ullah","khan")  
# no of arguments  
# def myfunc(\*kids):  
# print("This is my last child"+ " "+ kids[2])  
# myfunc("Noman","Zeeshan","Waqar","Zubair")  
# def myfun(child1,child2,child3):  
# print("One of my favourite child is "+ child3)  
# myfun(child1="jamil",child2="Zubair",child3="Khanan")  
# def myfun(\*\*kids):  
# print("His last Name is "+ kids["lname"])  
# myfun(fnmae="waqas",lname="khan")  
#-------------------------------passing arguments  
# def myfun(country="Norway"):  
# print("My country name is "+ country)  
# myfun("Swedan")  
# myfun("Nigeria")  
# myfun("England")  
# myfun()  
# myfun("Westindies")  
#-----------------------------------------------Passing a List as an Argument  
# def myfun(food):  
# for x in food:  
# print(x)  
# fruits=["banana","Mangoes","Cherries","Apple"]  
# myfun(fruits)  
# ------------------------------------------------Return values-----------------  
# def myfun(x):  
# return 5\*x  
# print(myfun(5))  
# print(myfun(7))  
# print(myfun(9))  
# def myf():  
# pass  
# ----------------------------Recursion--  
# def tri\_recursion(k):  
# if(k > 0):  
# result = k + tri\_recursion(k - 1)  
# print(result)  
# else:  
# result = 0  
# return result  
# print("\n\nRecursion Example Results")  
# tri\_recursion(6)  
# def recurs(k):  
# if(k>2):  
# result=k+recurs(k-1)  
# print(result)  
# else:  
# result=0  
# return result  
# print("\n\n resurcive example is ")  
# recurs(7)  
#---------------------lambda functions------------  
# x=lambda a:a\*10  
# print(x(5))  
# x=lambda a,b:a\*\*b  
# print(x(2,4))  
# x=lambda a,b,c:a+b+c  
# print(x(5,7,9))  
# def myfun(n):  
# return lambda a:a\*n  
# mydoubler=myfun(3)  
# print(mydoubler(11))  
# def myfunc(n):  
# return lambda a: a \* n  
# mydoubler = myfunc(2)  
# mytripler = myfunc(3)  
# print(mydoubler(11))  
# print(mytripler(11))  
#------------------Array----------------------------  
# cars=["Volvo","Japani","Galvanoes"]  
# # print(cars[0])  
# # cars[0]="Toyotta"  
# # for x in cars:  
# # print(x)  
# # print(len(x))  
# cars.append("Honda")  
# for x in cars:  
# print(x)  
# cars.append("Honda")  
# cars=["Cars","Toyotta","Yotta","Honda"]  
# cars.pop(1)  
# print(cars)  
# cars.remove("Volvo")  
# print(cars.reverse())  
# cars.reverse()  
# class myclass  
#--------------------------Class functions---------------------  
# class myclass:  
# x=5  
# p1=myclass()  
# print(p1.x)  
# class Person:  
# def \_\_init\_\_(self, name, age):  
# self.name = name  
# self.age = age  
# p1 = Person("John", 36)  
# print(p1.name)  
# print(p1.age)  
#----------------------------------------class function  
# class Person:  
# def \_\_init\_\_(self, name, age):  
# self.name = name  
# self.age = age  
# p1 = Person("John", 36)  
# # print(p1.name)  
# # print(p1.age)  
# print(p1)  
# class Person:  
# def \_\_init\_\_(self, name, age):  
# self.name = name  
# self.age = age  
# def \_\_str\_\_(self):  
# return f"{self.name}({self.age})"  
# p1 = Person("John", 36)  
# print(p1)  
# class Person:  
# def \_\_init\_\_(self, name, age):  
# self.name = name  
# self.age = age  
# def myfunc(self):  
# print("Hello my name is " + self.name)  
# p1 = Person("John", 36)  
# # p1.myfunc()  
# class Person:  
# def \_\_init\_\_(self, name, age):  
# self.name = name  
# self.age = age  
# def \_\_str\_\_(self):  
# return f"{self.name}({self.age})"  
# p1 = Person("John", 36)  
# print(p1)  
# class Person:  
# def \_\_init\_\_(self, name, age):  
# self.name = name  
# self.age = age  
# def myfunc(self):  
# print("Hello my name is " + self.name)  
# p1 = Person("John", 36)  
# p1.myfunc()  
# class Person:  
# def \_\_init\_\_(mysillyobject, name, age):  
# mysillyobject.name = name  
# mysillyobject.age = age  
# def myfunc(abc):  
# print("Hello my name is " + abc.name)  
# p1 = Person("John", 36)  
# p1.myfunc()  
# class Wazir:  
# def \_\_int\_\_(self,name,age):  
# self.name=name  
# self.age=age  
# def myfun(self):  
# print("This is the man"+self.name)  
# p1 = Wazir('kalim',55)  
# p1.myfun()  
# class Person:  
# def \_\_init\_\_(mysillyobject, name, age):  
# mysillyobject.name = name  
# mysillyobject.age = age  
# def myfunc(abc):  
# print("Hello my name is " + abc.name)  
# p1 = Person("John", 36)  
# # p1.age=40  
# # print(p1.age)  
# del p1.age  
# # print(p1.age)  
#-----------------------------------Inheritance -----------------  
# class Personne:  
# def \_\_init\_\_(self, nom, prenom):  
# print("Appel de la méthode \_\_init\_\_")  
# self.nom = nom  
# self.prenom = prenom  
# def \_\_new\_\_(cls, nom, prenom):  
# print("Appel de la méthode \_\_new\_\_ de la classe {}".format(cls))  
# return object.\_\_new\_\_(cls) # don't pass extra arguments here!  
# personne = Personne("Doe", "John")  
# print(personne.nom)  
# print(personne.prenom)  
# class Person:  
# def \_\_int\_\_(self,fname,lname):  
# self.fname=fname  
# self.lname=lname  
# def \_\_new\_\_(cls, fname, lname):  
# print("apple de la method\_\_new\_\_de {}".format(cls))  
# return object.\_\_new\_\_(cls)  
# person = Person("Doe","khan")  
# # print(person.fname)  
# class Person:  
# def \_\_init\_\_(self, fname, lname):  
# self.firstname = fname  
# self.lastname = lname  
# def printname(self):  
# print(self.firstname, self.lastname)  
# class Student(Person):  
# def \_\_init\_\_(self, fname, lname):  
# Person.\_\_init\_\_(self, fname, lname)  
# #Use the Person class to create an object, and then execute the printname method:  
# # x = Person("John", "Doe")  
# # x.printname()  
# x=Student("mike","khan")  
# x.printname()  
#----------------super()  
# class Person:  
# def \_\_init\_\_(self, fname, lname):  
# self.firstname = fname  
# self.lastname = lname  
# def printname(self):  
# print(self.firstname, self.lastname)  
# class Student(Person):  
# def \_\_init\_\_(self, fname, lname):  
# super().\_\_init\_\_( fname, lname)  
# #Use the Person class to create an object, and then execute the printname method:  
# # x = Person("John", "Doe")  
# # x.printname()  
# x=Student("mike","khan")  
# x.printname()  
# ------------add property  
# class Person:  
# def \_\_init\_\_(self, fname, lname):  
# self.firstname = fname  
# self.lastname = lname  
# def printname(self):  
# print(self.firstname, self.lastname)  
# class Student(Person):  
# def \_\_init\_\_(self, fname, lname,year,marks):  
# super().\_\_init\_\_( fname, lname)  
# self.graduationyear=year  
# self.scores=marks  
# def welcome(self):  
# print("Welcome",self.firstname ," ",self.lastname,"To the class of ",self.graduationyear," with Marks",self.scores)  
# #Use the Person class to create an object, and then execute the printname method:  
# # x = Person("John", "Doe")  
# # x.printname()  
# #Add a year parameter, and pass the correct year when creating objects:  
# x=Student("mike","khan",2019,990)  
# # x.printname()  
# # print(x.graduationyear)  
# # print(x.scores)  
# x.welcome()  
# ---------------------------------------------Gradution Result card program-----------  
# class Person:  
# def \_\_init\_\_(self, fname, lname):  
# self.firstname = fname  
# self.lastname = lname  
# def printname(self):  
# print(self.firstname, self.lastname)  
# class Student(Person):  
# def \_\_init\_\_(self, fname, lname,batch,degreename,universityname,startyear,  
# year,marks,totalmark,cgpa,totalcgp,crdhour,totalhr):  
# super().\_\_init\_\_( fname, lname)  
# self.university=universityname  
# self.batch=batch  
# self.degreename=degreename  
# self.graduationstarting=startyear  
# self.graduationyear=year  
# self.scores=marks  
# self.Totalmarks=totalmark  
# self.Cgpa=cgpa  
# self.totalcgpa=totalcgp  
# self.credithour=crdhour  
# self.totalcredithr=totalhr  
# def welcome(self):  
# print("\t \t \t \t \t \t Welcome TO ",self.university )  
# def Engineering(self):  
# print("\n Graduation Batch : ", self.batch, "\n Graduation Degree Title : ",  
# self.degreename)  
# def Graduationperiod(self):  
# print("\n Staring Graduation ",self.graduationstarting,"\n Graduation End year ",  
# self.graduationyear)  
# def computerEng(self):  
# print("\n \n Welcome to Result OF Your Degree : ","\n First name is: ",self.firstname,"\n last name is : ",  
# self.lastname,"\n scores :",  
# self.scores,"\n totalmarks",self.Totalmarks,"\n CGPA is :",self.Cgpa  
# ,"\n Total cgpa",self.totalcgpa,"\n total credit hours completed are : ",  
# self.credithour,"\n total credit hour in semester: ",self.totalcredithr)  
# x=Student("Zeeshan","khan","DE-40","Computer Engineering","NUST",2018,2022,990,1100,3.5,4,138,140)  
# x.welcome()  
# x.Engineering()  
# x.Graduationperiod()  
# x.computerEng()  
#-------------------------------------Python iterator-------------------------  
# mytupple=("Cherry","bananana","apple")  
# x=iter(mytupple)  
# print(next(x))  
# print(next(x))  
# print(next(x))  
# y="bananananana"  
# z=iter(y)  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# print(next(z))  
# y="mamamamammamma"  
# for x in y:  
# print(x)  
# class Person:  
# def \_\_iter\_\_(self):  
# self.a=1  
# return self  
# def \_\_next\_\_(self):  
# if self.a<=20:  
# x=self.a  
# self.a+=1  
# return x  
# else:  
# raise StopIteration  
# myclass=Person()  
# myiter=iter(myclass)  
# for x in myiter:  
# print(x)  
#--------------------------------scope of python ----------------------  
# def myfun():  
# x=56  
# print(x)  
# myfun()  
# def myfun():  
# x=76  
# def innerfun():  
# print(x)  
# innerfun()  
# myfun()  
#------------global varaiables  
# x = 76  
# def myfun():  
# def innerfun():  
# print(x)  
# innerfun()  
# myfun()  
# x = 300  
# def myfunc():  
# x = 200  
# print(x)  
# myfunc()#local scope 200  
# print(x)#300 global  
# def myfun():  
# global x  
# x=300  
# print(x)  
# myfun()  
#---------------make changes towardes a global varaibles  
# x=600  
# def myfun():  
# global x  
# x=300  
# print(x)  
# myfun()  
# import mymodule as mx  
# # mymodule.Greeting("ajman")  
# a=mx.person1["age"]  
# print(a)  
# import platform  
# x=platform.system()  
# print(x)  
# import nust as ns  
# # print(ns)# to Nust graduation module  
# x=dir(ns)  
# print(x)  
# import platform  
# x=dir(platform)  
# print(x)  
# import datavis1 as s1  
# print(s1)  
# from mymodule import person1  
# print(person1["age"])  
# import datetime  
# # x=datetime.datetime.now()  
# # print(x)  
# x=datetime.datetime(2022,7,1)  
# # print(x)  
# # print(x.strptime("%B"))  
# import datetime  
# x = datetime.datetime(2018, 6, 1)  
# print(x.strftime("%B"))# month name  
# print(x.strftime("%a"))  
# print(x.strftime("%j"))  
# import datetime  
# # x=datetime.MINYEAR  
# # x=datetime.MAXYEAR  
# x=datetime.date  
# print(x)  
# from datetime import timedelta  
# delta = timedelta(  
# days=50,  
# seconds=27,  
# microseconds=10,  
# milliseconds=29000,  
# minutes=5,  
# hours=8,  
# weeks=2)  
# print(delta)  
# from datetime import timedelta  
# # delta=timedelta(days=70,seconds=1200,microseconds=8000,  
# # milliseconds=3000,minutes=70,hours=27,weeks=5)  
# d=timedelta(microseconds=-1)  
# print(d.days,d.seconds,d.microseconds)  
#----------------------------------------------math module--------  
# x=max(67,567,876,908)  
# y=min(677,567,876,908)  
# z=pow(4,3)  
# print(x)  
# print(y)  
# print(z)  
# import math  
# x=math.sqrt(64)  
# y=math.ceil(2.46)  
# z=math.floor(2.46)  
# w=math.pi  
# print(x)  
# print(y)  
# print(z)  
# print(w)  
# print(math.e)  
#--------------------------------------json module  
# import json  
# # some JSON:  
# x = '{ "name":"John", "age":30, "city":"New York"}'  
# # parse x:  
# y = json.loads(x)  
# # the result is a Python dictionary:  
# print(y["age"])  
#--------------convert from python to json  
# import json  
# # a Python object (dict):  
# x = {  
# "name": "John",  
# "age": 30,  
# "city": "New York"  
# }  
# # convert into JSON:  
# y = json.dumps(x)  
# # the result is a JSON string:  
# print(y)  
# import json  
# # print(json.dumps({"name": "John", "age": 30}))  
# # print(json.dumps(["apple", "bananas"]))  
# # print(json.dumps(("apple", "bananas")))  
# # print(json.dumps("hello"))  
# # print(json.dumps(42))  
# # print(json.dumps(31.76))  
# # print(json.dumps(True))  
# # print(json.dumps(False))  
# # print(json.dumps(None))  
# # print(json.dumps({"name":"Noman","age":45,"city":"PK","country":"Pakistan"}))  
# # print(json.dumps([4,5,7,8,9,10,12,13,14]))  
# import json  
# # x = {  
# # "name": "John",  
# # "age": 30,  
# # "married": True,  
# # "divorced": False,  
# # "children": ("Ann","Billy"),  
# # "pets": None,  
# # "cars": [  
# # {"model": "BMW 230", "mpg": 27.5},  
# # {"model": "Ford Edge", "mpg": 24.1}  
# # ]  
# # }  
# # # print(json.dumps(x,indent=4,separators=(". ", " = ")))  
# # print(json.dumps(x,indent=4,sort\_keys=True))  
#----A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern  
# import re  
##Check if the string starts with "The" and ends with "Spain":  
# txt = "The rain in Spain"  
# x=re.search("^The.\*Spain$",txt)  
# if x:  
# print("Yes! We have match!")  
# else:  
# print("No match ")  
# import re  
# # txt = "The rain in Spain"  
# # x=re.findall("[a-m]",txt)  
# # print(x)  
# txt = "The rain 56 dollars in Spain"  
# x=re.findall("\d",txt)  
# print(x)  
# import re  
# # txt = "hello planet"  
# # # x = re.findall( "^hello", txt)  
# # # x=re.findall("planet$",txt)  
# # # x=re.findall("he.\*o",txt)  
# # x=re.findall("he.+o",txt)  
# # print(x)  
# import re  
# txt = "The rain in Spain"  
# #Check if the string starts with "The":  
# x = re.findall("\AThe", txt)  
# print(x)  
# if x:  
# print("Yes, there is a match!")  
# else:  
# print("No match")  
# import re  
# txt = "8 times before 11:45 AM"  
# #Check if the string has any characters from a to z lower case, and A to Z upper case:  
# x = re.findall("[a-zA-Z]", txt)  
# print(x)  
# if x:  
# print("Yes, there is at least one match!")  
# else:  
# print("No match")  
# import re  
# txt = "The rain in Spain"  
# x = re.search("\s", txt)  
# print("The first white-space character is located in position:", x.start())  
#------split at each whitespace character  
# import re  
# txt = "The rain in Spain"  
# # x = re.split("\s", txt)  
# #Split the string only at the first occurrence:  
# # x = re.split("\s", txt, 1)  
# #The sub() function replaces the matches with the text of your choice:  
# x=re.sub("\s","9",txt,2)  
# print(x)  
# import re  
# txt = "The rain in Spain"  
# #The regular expression looks for any words  
# # that starts with an upper case "S":  
# x = re.search(r"\bS\w+", txt)  
# # print(x.span())  
# print(x.string)  
# import re  
# txt = "The rain in Spain"  
# x = re.search(r"\bS\w+", txt)  
# print(x.group())  
#---------------------------------pip python -----------------  
# import camelcase  
# c=camelcase.CamelCase()  
# txt = "hello world"  
# print(c.hump(txt))  
#---------------------------try exception handling  
# try:  
# print(x)  
# except:  
# print("An exception occured ")  
# try:  
# print(x)  
# except NameError:  
# print("X is not defined")  
# except:  
# print("something went wrong")  
# try:  
# print("Hello")  
# except :  
# print("Something went wrong ")  
# else:  
# print("Nothing went wrong")  
# try:  
# print("Hello")  
# except :  
# print("Something went wrong ")  
# finally:  
# print(" went wrong")  
# try:  
# f = open("C:\Users\amazon\Downloads\aj.txt")  
# try:  
# f.write("Lorum Ipsum")  
# except:  
# print("Something went wrong when writing to the file")  
# finally:  
# f.close()  
# except:  
# print("Something went wrong when opening the file")  
# x = -1  
# if x < 0:  
# raise Exception("Sorry, no numbers below zero")  
# x = "hello"  
# if not type(x) is bool:  
# raise TypeError("Only Bool are allowed")  
#-------------------------python string fornmating  
# price = 49  
# money=450000  
# txt = "The price is {:.2f} and money is {:.4f} dollars"  
# print(txt.format(price,money))  
# quantity = 300  
# itemno = 567  
# price = 49  
# myorder = "I want {:.2f} pieces of item number {:.2f} for {:.2f} dollars."  
# print(myorder.format(quantity, itemno, price))  
# quantity = 3  
# itemno = 567  
# price = 49  
# myorder = "I want {0} pieces of item number {1} for {2:.2f} dollars."  
# print(myorder.format(quantity, itemno, price))  
# age = 36  
# name = "John"  
# txt = "His name is {1}. {1} is {0} years old."  
# print(txt.format(age, name))  
#------------------------------file handling as an important  
# f=open("aj.txt","r")  
# # print(f.read())  
# # print(f.read(5))  
# # print(f.readline())  
# # for x in f:  
# # print(x)  
# print(f.readline())  
# f.close()  
# f=open("aj.txt","a")  
# f.write("The file has more contents")  
# f.close()  
# # to open the ffile after appending  
# f=open("aj.txt","r")  
# print(f.read())  
# f=open("aj.txt","w")  
# f.write("Woh! I have deleted the contents ")  
# f.close()  
# # after overwritting  
# f=open("aj.txt","r")  
# print(f.read())  
# f=open("mynewfile.txt","x")#create new file  
# f.write("Oh! This is new file ")  
# f.close()  
# # lets read new file  
# f=open("mynewfile.txt","r")  
# print(f.read())  
#----------------------- delete file  
# import os  
# # os.remove("mynewfile.txt")  
# if os.path.exists("mynewfile.txt"):  
# os.remove("mynewfile.txt")  
# else:  
# print("The file does not exists")  
#Remove the folder "myfolder":  
# os.rmdir("myfolder")  
#--------------------------------python excercises  
# class Vehicle:  
# def \_\_init\_\_(self, max\_speed, mileage,min\_speed):  
# self.max\_speed = max\_speed  
# self.mileage = mileage  
# self.min\_speed=min\_speed  
# def calculate(self):  
# print("This is my data of vehical \n",  
# self.max\_speed,"\n ",  
# self.mileage,"\n",self.min\_speed)  
# modelX = Vehicle(240, 18,120)  
# print(modelX.max\_speed, modelX.mileage,modelX.min\_speed)  
# modelX.calculate()  
#---------------------------------  
# class Vehicle:  
# def \_\_init\_\_(self, name, max\_speed, mileage):  
# self.name = name  
# self.max\_speed = max\_speed  
# self.mileage = mileage  
# class Bus(Vehicle):  
# pass  
# School\_bus = Bus("School Volvo", 180, 12)  
# print("Vehicle Name:", School\_bus.name, "Speed:",  
# School\_bus.max\_speed, "Mileage:", School\_bus.mileage)  
#----------------------------------------------  
# class Vehicle:  
# def \_\_init\_\_(self, name, max\_speed, mileage):  
# self.name = name  
# self.max\_speed = max\_speed  
# self.mileage = mileage  
# def seating\_capacity(self, capacity):  
# return f"The seating capacity of a {self.name} is {capacity} passengers"  
# class Bus(Vehicle):  
# def seating\_capacity(self, capacity=50):  
# return super().seating\_capacity(capacity=50)  
# school\_bus=Bus("Volvo can",180,12)  
# print(school\_bus.seating\_capacity())  
#----------------------------------excercise 5  
# class Vehicle:  
# # class attributes  
# color="white"  
# def \_\_init\_\_(self, name, max\_speed, mileage):  
# self.name = name  
# self.max\_speed = max\_speed  
# self.mileage = mileage  
# class Bus(Vehicle):  
# pass  
# class Car(Vehicle):  
# pass  
# school\_bus=Bus("Volocano",1200,12)  
# print("Bus color : \n",school\_bus.color,"\n ",school\_bus.name,  
# "\n maxspeed",school\_bus.max\_speed,"\n mileage :",  
# school\_bus.mileage)  
# car\_veh=Car("AudiQ6",1800,18)  
# print("Car color : \n",car\_veh.color,"\n name",car\_veh.name,"\n maxspeed :",  
# car\_veh.max\_speed,"\n mileage",car\_veh.mileage)  
# import vehicle as vh  
# print(vh)  
#-------------------------------------------Excercise6  
# class Vehicle:  
# def \_\_init\_\_(self, name, mileage, capacity,maintenence):  
# self.name = name  
# self.mileage = mileage  
# self.capacity = capacity  
# self.maintainance=maintenence  
# def fare(self):  
# return self.capacity \* 100  
# def maintaince(self):  
# return self.maintainance\*10  
# class Bus(Vehicle):  
# def fare(self):  
# amount=super().fare()  
# amount+=amount\*10/100  
# return amount  
# School\_bus = Bus("School Volvo", 12, 50,20)  
# print("Total Bus fare is:", School\_bus.fare(),  
# "\n maintainance charges are :",School\_bus.maintainance)  
#------------------------------------------Excersise 7-----------------  
# class Vehicle:  
# def \_\_init\_\_(self, name, mileage, capacity):  
# self.name = name  
# self.mileage = mileage  
# self.capacity = capacity  
# class Bus(Vehicle):  
# pass  
# School\_bus = Bus("School Volvo", 12, 50)  
# # Python's built-in type()  
# print(type(School\_bus))  
# #------------------------------------------Excercise 8----------------------  
# class Vehicle:  
# def \_\_init\_\_(self, name, mileage, capacity):  
# self.name = name  
# self.mileage = mileage  
# self.capacity = capacity  
# class Bus(Vehicle):  
# pass  
# School\_bus = Bus("School Volvo", 12, 50)  
# # Python's built-in isinstance() function  
# print(isinstance(School\_bus, Vehicle))  
#----------------------------------------------------Excercise 9---------------  
# class Vehicle:  
# def \_\_init\_\_(self, name, mileage, capacity):  
# self.name = name  
# self.mileage = mileage  
# self.capacity = capacity  
# def fare(self):  
# return self.capacity \* 100  
# class Bus(Vehicle):  
# def \_\_init\_\_(self, name, mileage, capacity=50):  
# super().\_\_init\_\_(name, mileage, capacity)  
# def fare(self):  
# fare = super().fare()  
# # this is bus so we need to add an extra 10% on full fare as a maintenance charge  
# total\_fare = fare + (fare \* 0.10)  
# return total\_fare  
# School\_bus = Bus("School Volvo", 12)  
# print("Total Bus fare is:", School\_bus.fare())  
#-------------------------------------------------------------Excercise 10  
# class Vehicle:  
# color = 'white'  
# def \_\_init\_\_(self, name='', max\_speed='', mileage=''):  
# self.name = name  
# self.max\_speed = max\_speed  
# self.mileage = mileage  
# def seatingcapacity(self):  
# print('seating capacity of {} is {}'.format(self.name, self.capacity))  
# def display(self):  
# print('Vehicle Name:{}'.format(self.name),  
# 'Max Speed:{}'.format(self.max\_speed),  
# 'Mileage:{}'.format(self.mileage),  
# 'Color:{}'.format(self.color))  
# def fare(self):  
# print("The fare for {} is {}".format(self.name, int(self.capacity) \* 100))  
# def belongs(self):  
# print(self.\_\_class\_\_.\_\_name\_\_)  
# def checkins(self):  
# print('Instance of Vehicle:{}'.format(isinstance(self, Vehicle)))  
# class Bus(Vehicle):  
# def \_\_init\_\_(self, capacity='', \*\*kwargs):  
# self.capacity = capacity  
# super().\_\_init\_\_(\*\*kwargs)  
# def fare(self):  
# print(  
# "The fare for {} is {}".format(self.name,  
# int(self.capacity) \* 100 + int((int(self.capacity) \* 100) // 10)))  
# class Car(Vehicle):  
# def \_\_init\_\_(self, capacity='', \*\*kwargs):  
# self.capacity = capacity  
# super().\_\_init\_\_(\*\*kwargs)  
# # Example Input:  
# a = {'name': 'Volvo',  
# 'max\_speed': 30,  
# 'mileage': 40,  
# 'capacity': 100}  
# b = {'name': 'Volkswagon',  
# 'max\_speed': 50,  
# 'mileage': 100,  
# 'capacity': 30}  
# c = Bus(\*\*a)  
# d = Car(\*\*b)  
# c.display()  
# c.seatingcapacity()  
# c.fare()  
# c.belongs()  
# c.checkins()  
# d.display()  
# d.seatingcapacity()  
# d.fare()  
# d.belongs()  
# d.checkins()  
#-----------------------------------------------------MYSQL Queries  
# import mysql.connector  
# from mysql.connector import Error  
# try:  
# connection = mysql.connector.connect(host='localhost',  
# database='databaseaj',  
# user='root',  
# password='')  
# if connection.is\_connected():  
# db\_Info = connection.get\_server\_info()  
# print("Connected to MySQL Server version ", db\_Info)  
# cursor = connection.cursor()  
# cursor.execute("select database();")  
# record = cursor.fetchone()  
# print("You're connected to database: ", record)  
# except Error as e:  
# print("Error while connecting to MySQL", e)  
# finally:  
# if connection.is\_connected():  
# cursor.close()  
# connection.close()  
# print("MySQL connection is closed")  
#--------------------------------------  
# import pyodbc  
# import mysql.connector  
# conn=pyodbc.connect('Driver={SQL Server};''Server=localhost\SQLEXPRESS;''Database=db2;''Trusted\_connection=yes;')  
# cursor=conn.cursor()  
# cursor.execute('Select \* from dbo.player')  
# for row in cursor:  
# print(row)  
#------------------------------dbo.sales in db2  
# import pyodbc  
# import mysql.connector  
# conn=pyodbc.connect('Driver={SQL Server};'  
# 'Server=localhost\SQLEXPRESS;''Database=db2;'  
# 'Trusted\_connection=yes;')  
# cursor=conn.cursor()  
# cursor.execute("select \* from dbo.player where name = 'dhoni'")  
# rows = cursor.fetchall()  
# for row in rows:  
# print(row)  
#---------------------------------------------  
# import pymongo  
# myclient = pymongo.MongoClient('mongodb://localhost:27017/')  
# mydb = myclient['mydatabase']  
# # print(mydb)  
# print(myclient.list\_database\_names())  
#---------------------------------Tkinter GUI Python -------------------  
# from tkinter import \*  
# from tkinter import ttk  
# root = Tk()  
# frm = ttk.Frame(root, padding=50)  
# frm.grid()  
# ttk.Label(frm, text="Hello Ajman !").grid(column=0, row=0)  
# ttk.Button(frm, text="close", command=root.destroy).grid(column=1, row=0)  
# root.mainloop()  
#------------------------------------------task2  
# from tkinter import \*  
# from tkinter import ttk  
# root=Tk()#to form window  
# frm=ttk.Frame(root,padding=60)# rame inside window  
# frm.grid()  
# ttk.Label(frm,text="Ajman khan wazir ").grid(column=0,row=0)  
# ttk.Button(frm,text="close",command=root.destroy).grid(column=1,row=0)  
# root.mainloop()  
#------------------------------------------task3  
# import tkinter as tk  
# class App(tk.Frame):  
# def \_\_init\_\_(self, master):  
# super().\_\_init\_\_(master)  
# self.pack()  
# self.entrythingy = tk.Entry()  
# self.entrythingy.pack()  
# # Create the application variable.  
# self.contents = tk.StringVar()  
# # Set it to some value.  
# self.contents.set("this is a variable")  
# # Tell the entry widget to watch this variable.  
# self.entrythingy["textvariable"] = self.contents  
# # Define a callback for when the user hits return.  
# # It prints the current value of the variable.  
# self.entrythingy.bind('<Key-Return>',  
# self.print\_contents)  
# def print\_contents(self, event):  
# print("Hi. The current entry content is:",  
# self.contents.get())  
# root = tk.Tk()  
# myapp = App(root)  
# myapp.mainloop()  
#-------------------------------------------------------------------*