# print("hello")

# "hi"

# # for jupyter""" comment anything""" this will be shown in output if there is no code after 3pairs of ' " '

# # data type numbers

# x=12

# print(type(x))

# y=12.5

# print(type(y))

# z=12j

# print(type(z))

# num=y>x

# print(type(num))

# print(num)

# # data type string

# name="prince"

# print(len(name))

# print(name[2]) # tells the alphabet at location 2

# print(name[-2]) # tells us the alphabet from back

# print(name[2:5]) # tells us a series from so and so location

# print(name.upper())

# # data types lists

# mylist = [1,2,3,'hi','hello']

# print(mylist)

# print(mylist[2:5])

# #print(mylist[3] = 7 )

# print(mylist.append(10))

# print(mylist.insert(4,100))

# print(mylist.reverse())

*#example from programming with mosh*

numbers = [1,2,3,5,6,8,9,1,2,2,2,3,5,6]

unique =[]

for number in numbers:

     if number not in unique:

         unique.append(number)

**print**(unique)

# # data type Dictonary

# courses = {1:'python',2:'java', 'third': 'c++' }

# print(courses)

# print(courses['third'])

# print(courses.get('third'))

# #print(courses['third']=="hadoop")

# #print(courses['four']=="machine learning")

# # data type tuple

# # tuple is an unorder and unchangeable but we can have a duplicate value

# animals=(10,10,20,'tiger','lion','tiger')

# print(animals[2])

# print(animals.count('tiger'))# for advance step have look at python collections: counter

# # data type sets

# #1) set in unordered lists and has no duplicate entries

# sets={10,10,20,'tiger','lion','tiger'}

# print(sets)

# #2) set does not have any indexes

# #print(sets[2])

# #data type range

# print(range(10))

# print(list(range(11)))

# # data type Mislaneous

# c=[mylist , sets]

# print(c)

# # type conversion

# g=10

# gname="prince"

# sum = str(g)+gname

# print(sum)

#x=2.9

**#print**(**round**(x))

#**print**(**abs**(-2.9))

*# import math*

*# print(math.ceil(2.9))*

*# print(math.floor(2.9))*

# # for other examples refer to pic in notes

# # python collections

# from collections import namedtuple

# # namedtuple is just like creating a table

# a=namedtuple('courses', 'pyhton, java')

# b=a('started','Started')

# c=a('completed','completed')

# print(b,c)

# # alternate mathod

# # to represent this in a list

# s=a.\_make( ['pending','pending'])

# print(s)

# #deque

# #pronounced:deck

# #an optimised list which is used to perform insertion and deletion easily

# from collections import deque

# a=['p','r','i','n','c','e']

# d=deque(a)

# print(d)

# print(d.append('python'))

# print(d.appendleft('python'))

# print(d.pop(),d)

# print(d.popleft(),d)

# #chainmap

# from collections import ChainMap

# # chainmap is a dictionary like class for creating a single view of multiple classes

# cou = {1:'python',2:'java' }

# co = {1:'python',2:'java'}

# print(ChainMap(cou,co))

# #counter

# from collections import Counter

# #counter is a dictionary subclass for cojnting hashable objects

# #used to count hashable objects in a list

# f=[2,4,4,4,6,6,6,6,6,8,8,8,8,8,8,8]

# hj = Counter(f)

# print(hj)

# print(list(hj.elements()))

# print(hj.most\_common())

# sub = {2:1,8:3 }

# print(hj.subtract(sub))

# print(hj.most\_common())

# # ordered ditionary

# from collections import OrderedDict

# #order dict is a dictionary subclass which rememers the order in which the entries were done

# d= OrderedDict()

# d[1] ='p'

# d[2] ='r'

# d[3] ='i'

# d[4] ='n'

# d[5] ='c'

# d[6] ='e'

# print(d)

# print(d.keys())

# d[1]='e'

# print(d)

# #Default dictionary

# #it is a dictionary subclass which calls a fxactory function to supply missing values

# from collections import defaultdict

# d= defaultdict(int)

# d[1]='python'

# d[2]='java'

# print(d[3])

# #Some other collections

# #userDict is a wrapper around dictionary objects for easier dictionary objects sub-classing.

# #userDict is a wrapper around list objects for easier list sub-classing.

# #userDict is a wrapper around string objects for easier string sub-classing.

# #array

# import array

# a=array.array('i',[1,2,3,4,5,6])

# print(a)

# print(a[2])

# print(len(a))

# a.append(7) #adding only 1 value at end of array

# print(a)

# a.insert(0,0) #adding only 1 value at start of array

# print(a)

# a.extend([8,9,10])#adding more than 1 value at end of array

# print(a)

# print("poping out a value",a.pop())

# print('poping out a specific value of your choice', a.pop(2))# pop function returns a value

# print(a.remove(8))# does not return value

# # array concatenation

# bb=array.array('i',[1,2,3,4,5,6])

# cc=array.array('i',[1,2,3,4,5,6])

# dd=array.array('i')

# dd=bb+cc

# print(dd)

# # can add array of same type only

# # bb1=array.array('i',[1,2,3,4,5,6])

# # cc1=array.array('d',[1,2,3,4,5,6])

# # dd1=array.array('i')

# # dd1=bb1+cc1

# # print(dd)

# # slicing an array

# # an array can be sliced using ':' symbol. this return the range of elements

# #that we have specifed by their index no

# bb2=array.array('i',[1,2,3,4,5,6])

# print(bb2[0:5])

# print(bb2[0:-4])

# #reverse of array

# print(bb2[::-1])

# #looping

# bb3=array.array('i',[1,2,3,4,5,6,7,8,9,10,11,12])

# for x in bb3:

# print(x)

# for x in bb3[0:-3]:

# print(x)

# #while loop

# temp=0

# while temp<bb3[3]:

# print(bb3[temp])

# temp=temp+1 #temp+=1

# tem=0

# while tem<bb3[2]:

# print(bb3[temp])

# tem+=1

# #Hash table or HashMap

# # it is a type of data structure ythat maps to iits value pairs

# #It implements abstract array data type

# #understand with help of dictionaries

# # creating dictionaries

# #method-1

# my\_dict={'dave':'001','ava':'002','joe':'003'}

# print(my\_dict)

# type(my\_dict)

# #method 2

# new\_dict=dict()

# print(new\_dict)

# type(new\_dict)

# new\_dict2=dict(dave='001',ava='002')

# print(new\_dict2)

# #nested dictionries

# emp\_details={'employee':{'dave':{'id':'001','salary':'2000','designation':'team lead'},

# 'ava':{'id':'002','salary':'3000','designation':'leadar'}}}

# print(emp\_details)

# # performing operations on hash tables

# print(my\_dict['dave']) #acessing data

# print(my\_dict.keys()) #all the keys in dict will be returned

# print(my\_dict.values()) #all the values in dict will be returned

# print(my\_dict.get('ava')) #all the speciic values in dict will be returned

# for x in my\_dict: # getting content with help of loop

# print(x)

# for x in my\_dict.values():

# print(x)

# for x,y in my\_dict.items():

# print(x,y)

# #updating content in dictionary

# my\_dict['dave']='004'

# my\_dict['chirs']='003'

# print(my\_dict)

# #deleting content from dictionary

# print(my\_dict.pop('ava'))

# print(my\_dict.popitem())# removes last item

# del my\_dict['dave']

# print(my\_dict)

# #converting a dictionary in a dataframe

# #dataFrame is a 2-d data structure that consists of columns of various types, ver similarly to python dictionary and can be converted to pandas data framework

# #pandas in short help in organising content in the form of table

# # to install pandas pls run command in terminal as " pip install pandas"

# import pandas as pd

# df = pd.DataFrame(emp\_details['employee'])

# print(df)

# #Arithmetic operators

# #\_ are used to perform arithmetic operations between variables

# q=10

# w=20

# print(q+w)

# print(q-w)

# print(q\*w)

# print(q\*\*w) # returns exponential value

# print(q/w)

# print(w//q) # returns floor division value # returns integer value after division

# print(q//w)

# print(q%w)

# #Assingnment operators

# a=15

# a += 15

# print( a)

# a += 15

# print(a)

# a\*\*=5

# print(a)

# a -= 5

# print(a)

# a\*=2

# print(a)

# a%=5

# print(a)

# a//=5

# print(a)

# a|=2

# print(a)

# a^=3

# print(a)

# a&=4

# print(a)

# #comparison operators

# val =50

# numd =20

# compare =val==numd

# print(compare)

# if val > numd:

# print("val grater")

# elif val<numd:

# print("val is smaller")

# if val!=numd:

# print("val is not equal to num")

# if val>= numd:

# print("val might be greater or equal to numd")

# if val<=numd:

# print("val might be lesser or equal to numd")

# #Logical opertors

# #logical opertors are used to combine conditional statements

# fg=10

# print(fg>10 and fg>5)

# print(fg>8 and fg>5)

# print(fg>10 or fg>5)

# print(fg>8 or fg>5)

# print(not(fg>10 and fg>5))

# print(not(fg>8 and fg>5))

# #Identity Operators

# #1) IS # returns true if both variables are same object

# #2) IS NOT # returns true if both variables are not same object

# list1=[10,20,30]

# list2=[10,20,30]

# x=list1

# print(x)

# print(x is list1)

# print(list1 is list2)# in not same objects

# print(list1 is not list2)# although has same values, it doesn't depend

# #MEMBERShIP OPERTAORS

# #these are used to check if a sequence is present in an object or not

# print(x in list1)

# print(20 in list1)

# print(x not in list1)

# print(20 not in list1)

# #in

# #returns true if a sequence with the specified value is present in object

# #not in

# #returns true if a sequence with the specified value is not present in object

# #BITWISE OPERATORS

# #types

# # & Bitwise AND #sets each bit to 1 if both bits are 1.

# # | Bitwise OR #sets each bit to 1 if one of the bits are 1.

# # ^ Bitwise XOR #sets each bit to 1 if only one of the bits are 1.

# # ~ Bitwise NOT #inverts all bits.

# # << bitwise left shift #shift left by pushing in zeros from the right and let the leftmost bits fell off

# # >> bitwise right shift #shift right by pushing copies of the leftmost bit from the left and let the rightmost bits fell off

# # concept not cleared

# print(10 & 12)

# #he says 10 in binary is 1010 and 12 is 1100 and 1st digit matches(of 1010 & 1100) but not the second on so it becomes 1000 and if 1000 is converted into decimal it becomes 8

# print(10 | 12)

# print(10 << 12)

# print(10 >> 12)

# print(10 ^ 12)

# # not wroking

# #print(10 ~ 12)

# # LOOPS

# # while loop

# # while loops are known as indefinite or conditional loops. they will keep iterating

# #until certain contion are met. There is no guarantee ahead of time regarding how many time the loop will iterate.

# count=0

# while( count<9):

# print('while loop',count )

# count+=1

# print("bye")

# # # example 2

# # import random

# # n=20

# # to\_be\_guessed = int(n\* random.random())+1

# # guess=0

# # while guess != to\_be\_guessed:

# # guess=int(input("new number:"))

# # if guess > 0:

# # if guess > to\_be\_guessed:

# # print("number too large")

# # elif guess< to\_be\_guessed:

# # print("number too small")

# # else:

# # print("sorry that you're giving up!")

# # break

# # else:

# # print("congo. you made it!")

# # # basically this will automatically create an random no and we have guess it

# # #for loop

# # fruits = ['mango','grapes','apple']

# # for fruit in fruits:

# # print("current fruit:",fruit)

# # print("good bye")

# # #example 2

# # import math

# # numm = int(input("Number:"))

# # factorial = 1

# # if numm<0:

# # print("must be +ve")

# # elif numm==0:

# # print("factorial=1")

# # else:

# # for i in range(1,numm + 1):

# # factorial = factorial\*i

# # print(factorial)

# # #nested loop

# # #banking service

# # print('welcome to my bank')

# # restart=('y')

# # chances= 3

# # balance = 67.14

# # while chances >=0:

# # pin = int(input('pls enter your 4 digit pin:'))

# # if pin==(1234):

# # print('u have entered the pin correctly\n')

# # while restart not in ('n','no','NO','N'):

# # print("press 1 for your balance\n")

# # print("press 2 to make a withdrawl \n")

# # print("press 3 for payin\n")

# # print("press 4 to return card\n")

# # option = int(input('what would u like to choose?'))

# # if option==1:

# # print("your balance is",balance,'\n')

# # restart = input("would u like to go back?")

# # if restart in ('n','no','NO','N'):

# # print("thank u")

# # break

# # elif option == 2:

# # option2 = ('y')

# # withdrawl = float(input('how much would u like to withdraw? pls enter in 10,20,40,60,80,100'))

# # if withdrawl in [10,20,40,60,80,100]:

# # balance= balance -withdrawl

# # print("\n your balance is ",balance)

# # restart = input("would u like to go back?")

# # if restart in ('n','no','NO','N'):

# # print("thank u")

# # break

# # elif withdrawl != [10,20,40,60,80,100]:

# # print("invalid amount pls retry\n")

# # restart('y')

# # elif withdrawl ==1:

# # withdrawl = float(input("pls enter the desired amount:"))

# # elif option==3:

# # pay\_in = float(input("how much u would like to pay in?"))

# # balance= balance + pay\_in

# # print("your balance is ", balance)

# # restart = input("would u like to go back?")

# # if restart in ('n','no','NO','N'):

# # print("thank u")

# # break

# # elif option==4:

# # print("pls wait while your card is being returned...\n")

# # print("thank u for your services")

# # break

# # else:

# # print("pls enter the correct no. \n")

# # elif pin !=('1234'):

# # print('incorrect password')

# # chances = chances-1

# # if chances == 0:

# # print('\n no more tries')

# # break

# # #example 2

# # #somewhat like pythgoros thm

# # from math import sqrt

# # n = int(input("max no?"))

# # for a in range(1,n+1):

# # for b in range(a,n):

# # c\_square = a\*\*2 + b\*\*2

# # c = int(sqrt(c\_square))

# # if((c\_square - c\*\*2) == 0):

# # print(a,b,c)

# # #example 3

# # travelling = input("yes or no:")

# # while travelling == 'yes':

# # num = int(input("number of people travelling:"))

# # for num in range(1, num +1):

# # name = input("name:")

# # age = input("age:")

# # sex = input("sex:")

# # print(name)

# # print(age)

# # print(sex)

# # travelling = input("oops! forgot someone")

*# exampple 4 //from programming with mosh*

for x in **range**(4):

    for y in **range**(3):

**print**(f'({x},{y})')

# # patterns

# #pyramid

# # def pattern(n):

# # k= 2\*n-2

# # for i in range(0,n):

# # for j in range(0,k):

# # print(end=" ")

# # k = k - 1

# # for j in range(0 ,i + 1):

# # print("\* ",end="")

# # print("\r")

# # pattern(5)

# # inverse pattern

# # def pattern(n):

# # k = n-2

# # for i in range(n,-1,-1):

# # for j in range(k,0,-1):

# # print(end=" ")

# # k=k+1

# # for j in range(0,i+1):

# # print("\* ", end="")

# # print("\n")

# # pattern(10)

# #right start pattern

# # def pattern(n):

# # for i in range(0,n):

# # for j in range(0,i+1):

# # print("\* ",end=" ")

# # print("\r")

# # for i in range(n,-1,-1):

# # for j in range(0,i+1):

# # print("\* ", end=" ")

# # print("\r")

# # pattern(10)

# #left start pattern

# # def pattern(n):

# # k = 2\*n-2

# # for i in range(0,n-1):

# # for j in range(0,k):

# # print(end=" ")

# # k = k-2

# # for j in range(0,i+1):

# # print("\* ", end="")

# # print("\r")

# # k=-1

# # for i in range(n-1,-1,-1):

# # for j in range(k,-1,-1):

# # print(end=" ")

# # k = k+2

# # for j in range(0, i+1):

# # print("\* ",end="")

# # print("\r")

# # pattern(10)

# # #hourglass

# # def pattern(n):

# # k = n-2

# # for i in range(n,-1,-1):

# # for j in range(k,0,-1):

# # print(end=" ")

# # k=k+1

# # for j in range(0,i+1):

# # print("\* ", end="")

# # print("\n")

# # k= 2\*n-2

# # for i in range(0,n+1):

# # for j in range(0,k):

# # print(end=" ")

# # k = k - 1

# # for j in range(0 ,i + 1):

# # print("\* ",end="")

# # print("\n")

# # pattern(5)

# #right hand side triangle

# # def pattern(n):

# # for i in range(0,n):

# # for j in range(0,i+1):

# # print("\* ", end="")

# # print("\r")

# # pattern(10)

# #left handed triangle

# # def pattern(n):

# # k=2\*n-2

# # for i in range(0,n):

# # for j in range(0,k):

# # print(end=" ")#here space in needed

# # k=k-2

# # for j in range(0,i+1):

# # print("\* ", end="")

# # print("\r")

# # pattern(10)

# #downward pattern

# # def pattern(n):

# # for i in range(n,-1,-1):

# # for j in range(0,i+1):

# # print("\* ",end="")

# # print("\r")

# # pattern(10)

# #daimond pattern

# # def pattern(n):

# # k=2\*n-2

# # for i in range(0,n):

# # for j in range(0,k):

# # print(end=" ")

# # k=k-1

# # for j in range(0,i+1):

# # print("\* ",end="")

# # print("\r")

# # k=n-2

# # for i in range(n,-1,-1):

# # for j in range(k,0,-1):

# # print(end=" ")

# # k=k+1

# # for j in range(0,i+1):

# # print("\* ",end="")

# # print("\r")

# # pattern(10)

# #diamond star pattern

# # for i in range(5):

# # for j in range(5):

# # if i+j==2 or i-j ==2 or i+j==6 or j-i ==2:

# # print("\*",end="")

# # else:

# # print(end=" ")

# # print()

# #number pattern

# # def paatern(n):

# # x=0

# # for i in range(0,n):

# # x+=1

# # for j in range(0,i+1):

# # print(x,end=" ")

# # print("\r")

# # paatern(7)

# # #pascal's triangle

# # def pattern(n):

# # for i in range(0,n):

# # for j in range(0,i+1):

# # print(function(i,j)," ", end="")

# # print()

# # def function(n,k):

# # res=1

# # if(k> n -k):

# # k=n-k

# # for i in range(0,k):

# # res = res\*(n-1)

# # res = res//(i+1)

# # return res

# # pattern(7)

# #half pyramid with numbers

# # def pattern(n):

# # x=0

# # for i in range(0,n):

# # x=x+1

# # for j in range(0,i+1):

# # print(x, end="")

# # print("\r")

# # pattern(10)

# #diamond pattern with numbers

# # def pattern(n):

# # k=2\*n-2

# # x=0

# # for i in range(0,n):

# # x=x+1

# # for j in range(0,k):

# # print(end=" ")

# # k=k-1

# # for j in range(0,i+1):

# # print(x,end=" ")

# # print("\r")

# # k=n-2

# # x=0

# # for i in range(n,-1,-1):

# # x=x+1

# # for j in range(k,0,-1):

# # print(end=" ")

# # k=k+1

# # for j in range(0,i+1):

# # print(x,end=" ")

# # print("\r")

# # pattern(5)

# #descending order

# # def pattern(n):

# # for i in range(n,0,-1):

# # for j in range(1,i+1):

# # print(j,end=" ")

# # print("\r")

# # pattern(5)

# #bainary no pattern

# # def pattern(n):

# # k = 2\*n-2

# # for i in range(0,n):

# # for j in range(0,k):

# # print(end=" ")

# # k=k-1

# # for j in range(0,i+1):

# # print('10',end="")

# # print("\r")

# # pattern(5)

# #right alphbetical triangle

# # def pattern(n):

# # x=65

# # for i in range(0,n):

# # ch=chr(x)

# # x=x+1

# # for j in range(0,i+1):

# # print(ch,end=" ")

# # print("\r")

# # pattern(10)

# #type-2

# # def pattern(n):

# # k=2\*n-2

# # x=65

# # for i in range(0,n):

# # for j in range(0,k):

# # print(end=" ")

# # k=k-1

# # for j in range(0,i+1):

# # ch =chr(x)

# # print(ch,end=" ")

# # x+=1

# # print("\r")

# # pattern(10)

# #in specific shape

# # for i in range(7):

# # for j in range(7):

# # if j ==0 or i-j ==3 or i+j ==3:

# # print("A",end="")

# # else:

# # print(end=" ")

# # print()

# #file handling

# # file.read()

# # import os

# # file = open("D:/What I Know/python/experimental.txt",'r')

# #file.close()

# # print(file.read())

# # file.close()

# # print(file.read(5))

# # file.close()

# #read lines

# # print(file.readline())

# # file.close()

# # print(file.readline(2))

# # file.close()

# # print(file.readlines())

# # file.close()

# #looping over a file object

# # for line in file:

# # print(file.readlines())# one problem is that it skips the first line- no sol provided

# # file.close()

# #writing in an existing file

# import os

# file = open("D:/What I Know/python/experimental.txt",'w')

# file.write("just leave me alone\n")

# file.write("you moron just get away")

# file.close()

# #creation of file

# # file = open("D:/What I Know/python/experimental-creting1.txt",'x')

# # file.write("u r here again, but why")

# # file.close()

# #deleting a file

# #not working

# # if os.path.exists("D:/What I Know/python/experimental-creting1.txt"):

# # os.remove("D:/What I Know/python/exerimental-creating1.txt")

# # else:

# # print("file does not exists")

# #deleting a folder

# #not working

# #os.rmdir("new folder")

# #decorators

# # def fun1(name):

# # return f"Hello{name}"

# # def fun2(name):

# # return f"{name} , how are you doing?"

# # def fun3(fun4):

# # return fun4(' dear learner')

# # print(fun3(fun1))

# # print(fun3(fun2))

# #inner functions

# def fun():

# print("parent function")

# def fun1():

# print("1st child function")

# def fun2():

# print("2nd child function")

# fun2()

# fun1()

# fun()

# #return a function from funtion

# def fun(n):

# def fun1():

# return "prince"

# def fun2():

# return "python"

# if n==1:

# return fun1

# else:

# return fun2

# a = fun(1)

# b = fun(2)

# print(a())

# print(b())

# #deorators in python are ery powerful whch modify the behaviour of a function without modifying it permanently.

# #it basically wraps another fun and since both functions are callable, it returns callable

# #example2

# def fun1(function):

# def wrapper():

# print("hello")

# function()

# print("welcome")

# return wrapper

# def fun2():

# print("quarantined!!")

# fun2 = fun1(fun2)

# fun2()

# #diiferent method of example 2

# def fun1(function):

# def wrapper():

# print("hello")

# function()

# print("welcome")

# return wrapper

# @fun1

# def fun2():

# print("quarntined")

# fun2()

# def fun(function):

# def wrapper(\*args,\*\*kwargs):

# print("hello")

# function(\*args,\*\*kwargs)

# print("welcome")

# return wrapper

# @fun

# def funtion2(name):

# print(f"{name}")

# funtion2("prince")

# #example 3

# def function1(function):

# def wrapper(\*args,\*\*kwargs):

# print("it worked")

# return wrapper

# @function1

# def function2(name):

# print(f"{name}")

# function2("python")

# #fancy decorators

# class square:

# def \_\_init\_\_(self,side):

# self.\_side = side

# @property

# def side(self):

# return self.\_side

# @side.setter

# def side(self,value):

# if value >=0:

# self.\_side = value

# else:

# print("error")

# @property

# def area(self):

# return self.\_side \*\*2

# @classmethod

# def unit\_square(cls):

# return cls(1)

# s=square(5)

# print(s.side)

# print(s.area)

# #singleton class

# import functools

# def singleton(cls):

# @functools.wraps(cls)

# def wrapper(\*args,\*\*kwargs):

# if not wrapper.instance:

# wrapper.instance = cls(\*args, \*\*kwargs)

# return wrapper.instance

# wrapper.instance = None

# return wrapper

# @singleton

# class one:

# pass

# first =one()

# second = one()

# print(first is second)

# #nesting decorators -- google it!

# #argument in a decorators

# import functools

# def repeat(num):

# def decorator\_repeat(func):

# @functools.wraps(func)

# def wrapper(\*args,\*\*kwargs):

# for \_ in range(num):

# value = func(\*args,\*\*kwargs)

# return value

# return wrapper

# return decorator\_repeat

# @repeat (num =5)

# def function(name):

# print(f"{name}")

# function("python")

# #lambda function

# # anonymous ornameless functions

# # lambda is not a name, but is a keyword

# # used for one-time use

# # i/o of other functions

# # reduce code size

# #lambda arguments:expression

# x = lambda a: a\*a

# print(x(3))

# #otherwise we have to follw traditional code i.e.

# def new(a):

# return a\*a

# print(new(3))

# # anonymous functions within user definef funcctions

# def a(x):

# return(lambda y:x+y)

# t=a(4)

# print(t(9))

# #using lambda functions within filter,map,refuce

# #filters are used to filter the given iteralbes(lists,sets,etc) with help of another function passed

# #as an argument to test all teh elements to be true or false.

# mylist=[1,2,3,4,5,6]

# newlist=list(filter(lambda a:(a/3==2),mylist))

# print(newlist)

# #map function

# #applies a givven function to all the iterables and return a new list

# p=list(map(lambda a:(a/3!=2),mylist))

# print(p)

# #reduce function

# # applies some other function to a list of elements that are passed as parameters

# # to it and finally returns a single value

# from functools import reduce

# c=reduce(lambda a,b: a+b,[23,56,43,98,1])#it keeps on addding all the items in a list

# print(c)

# #solving algerbraic expressions using almbda

# #linear equations

# s=lambda a:a\*a

# print(s(4))

# d=lambda x,y: 3\*x+4\*y

# print(d(4,7))

# #quadratic equations

# x=lambda a,b:(a+b)\*\*2

# print(x(3,4))

# #detail for map,filter,return

# def new(a):

# return a\*a

# x=map(new,[1,2,3,4])

# print(x)

# print(list(x))

# print(tuple(x))

# def new(a,b):

# return a\*b

# x=map(new,[1,2,3,4],[2,3,4,5])

# print(x)

# print(tuple(x))

# lst=[1,2,3,4,5]

# y=list(map(lambda x: x+3,lst))

# print(y)

# #filter-details

# #filter(function, iterables)

# def new1(i):

# if i>=3:

# return i

# j=filter(new1,(1,2,3,4,5,6,7))

# print(j)

# print(tuple(j))

# #or

# z=filter(lambda x: (x>=3),(1,2,3,4,5,6,7))

# print(list(z))

# #reduce-detail

# #reduce(function,iterables)

# from functools import reduce

# def a(x,y):

# return x+y

# s=reduce(a,[1,3,4,5,6,7,7,8,8])

# print(s)

# #or

# f=reduce(lambda p,q: p\*q,[1,2,3,4,5,6,7,7])

# print(f)

# #filter within map

# c=map(lambda x:x+x,filter(lambda x:(x>=4),[2,3,4,5]))

# print(tuple(c))

# #map within filter

# d=filter(lambda x:(x>=4),map(lambda x:x+x,[2,3,4,5,6]))

# print(set(d))

# #map and filetr within reduce

# r=reduce(lambda x,y:x+y, map(lambda x:x+x,filter(lambda x:(x<=4),[1,2,3,4,5,6,7])))

# print(r)

# #generators

# #are functions that return traversable objects

# #produce items one at a time and only when required

# #are run along with "for" loops

# #advantage

# #easy to implemrnt (implemnets \_\_iter\_\_().\_\_next\_\_() automatically)

# #better memoru management and utilization

# #can be used to produce infinite item

# #can also be used to pipeline a number of operations

# #diff. bw generators and normal functions

# # make use of 'yield' keyword #make use of 'return' keyword

# #run when 'next()' method is called #run when name of method is called

# #produce items one at a time & only when required #produce all items at once

# #baisc

# def new(dict):

# for x,y in dict.items():

# yield x,y

# a={1:"hi",2:"welcome"}

# b=new(a)

# print(b)

# print(next(b))

# print(next(b))

# #print(next(b))

# def myfunc(i):

# while i<=3:

# yield i

# i+=1

# j=myfunc(2)

# print(next(j))

# print(next(j))

# #print(next(j))

# def ex():

# n=3

# yield n

# n=n\*n

# yield n

# v=ex()

# print(next(v)) #returns 1st n

# print(next(v)) #returns 2nd n

# #print(next(v))

# #generators with loops

# def ex():

# n=3

# yield n

# n=n\*n

# yield n

# v=ex()

# for x in v:

# print(x)

# #generator expressions

# #resembles list comprehensiond and like lambda functions,

# #generators expressions create anonymous generator functions.

# #difference in print keyword

# f=range(6)

# print("list comp",end=":")

# q=[x+2 for x in f]

# print(q)

# print("gen exp",end=":")

# r=(x+2 for x in f)

# print(r)

# #to print generator

# for x in r:

# print(x)

# #to find min val from genrstor

# print("gen exp",end=":")

# r=(x+2 for x in f)

# print(r)

# print(min(r))

# #use cases

# #fibonacci series

# #a sreries of number where in each number also called as the fibinacci number is the sum of the 2 preceding numbers.

# def fib():

# f,s=0,1

# while True:

# yield f

# f,s=s,f+s

# for x in fib():

# if x>50:

# break

# print(x,end= " ")

# #generating a stream of numbers

# a=range(100)

# b=(x for x in a)

# print(b)

# for y in b:

# print(y)

# #odd or even

# a=range(2,100,2)

# b=(x for x in a)

# print(b)

# for y in b:

# print(y)

# a=range(1,100,2)#excludes 100 in intervals of 2

# b=(x for x in a)

# print(b)

# for y in b:

# print(y)

# #Sine wave

# #not working

# # import numpy as np

# # from matplotlib import pyplot as plt

# # import seaborn as sb

# # def s(flip = 2):

# # x=np.linesplace(0,14,100)

# # for i in range(1,5):

# # plt.plot(x,np.sin(x + 1 \* .5)\* (7-i)\*flip)

# # sb.set()

# # s()

# # plt.show()

# #type 2

# import numpy as np

# from matplotlib import pyplot as plt

# import seaborn as sb

# def s(flip = 2):

# x =np.linspace(0,14,100)

# for i in range(1,10):

# yield(plt.plot(np.sin(x+i\*.5)\*(7-i)\*flip))

# print(sb.set())

# s=s()

# plt.show()

# print(next(s))

# print(next(s))

# print(next(s))

# #oops concept

# #Class and Objects

# #class -> A class is the blueprint from which specific objects are created

# #class variable -> a variable that is shared by all instances of class

# #instance variables -> instance varianble are unique to each instance

# #data member -> a class variabe or instance variable that holds data associated with class and its objects

# class car():

# pass

# honda = car()# these are 2 objects that belongs to a specific class

# tata = car()

# honda.modelname='city'

# honda.yearm = 2017

# honda.price = 100000

# tata.modelname='bolt'

# tata.yearm = 2016

# tata.price = 60000

# print(honda.price)

# #method 2

# class car2():

# def \_\_init\_\_(self,modelname,yearm,price):

# self.modelname = modelname

# self.yearm = yearm

# self.price = price

# def price\_inc(self):

# self.price = int(self.price\*1.15)

# honda = car2('city',2017,100000)

# tota = car2('bolt',2016,60000)

# honda.cc = 1500 # here i have created instance variable for specific object

# print(honda.\_\_dict\_\_)#this will return the whole info about an object

# print(honda.price)

# honda.price\_inc()

# print(honda.price)

# #OOPS in python

# #inheritance -> a class can inherit attributes and behaviour methods from another class,called the superclass.

# # a class which inherits from a asuperclass is called a subclass, also called heir class or child class.

# #eg parents are the base class and children are the derived class

# #super class for method 2

# class supercar(car2):

# pass

# honda = supercar('supercity',2017,100000)

# tata = car2('bolt',2016,600000)

# honda.cc=1500

# #print(help(honda))

# print(honda.yearm)

# honda.price\_inc()

# print(honda.price)

# #superclass method 2

# class supercar2(car2):

# def \_\_init\_\_(self,modelname,yearm,price,cc):

# super.\_\_init\_\_(modelname,yearm,price)

# self.cc = cc #inheritence

# honda = supercar('supercity',2017,100000)

# tata = car2('bolt',2016,600000)

# honda.cc=1500

# #print(help(honda))

# print(honda.yearm)

# honda.price\_inc()

# print(honda.price)

# #encapsulation

# # binding data and code togetther as a single unit.

# #securing data by hiding the implementation details to user.

# #abstraction

# # hides the implementation details and only provides the functionality to users

# #you can achieve abstraction using abstract class and interfaces

# #abstract class cannot be instantiatd.

# # it can only be inherited

# #notworking

# # for abc import ABC, abstractmethod

# # class car3(ABC):

# # @abstractmethod

# # def price\_inc(self):

# # pass

# # class supercar3(car3):

# # def \_\_init\_\_(self,modelname,yearm,price,cc):

# # super.\_\_init\_\_(modelname,yearm,price)

# # self.cc=cc

# # def price\_inc(self):

# # self.price\_inc= int(self.price\*2.5)

# # honda =supercar3('city',2017,100000)

# # tata = car3('bolt',2016,600000)

# # honda.cc=1500

# # #print(help(honda))

# # print(honda.yearm)

# # honda.price\_inc()

# # print(honda.price)

# #inheritence in python

# #\_\_init\_\_() function s automatically called every time the class is used to create an object.

# class parent:

# def \_\_init\_\_(self,fname,fage):

# self.name = fname

# self.age = fage

# def view(self):

# print(self.name ,self.age)

# class child(parent):

# def \_\_init\_\_(self,fname,fage):

# parent.\_\_init\_\_(self,fname,fage)

# self.lastname = "Prince"

# def view(self):

# print(self.age,self.lastname,self.name)

# ob = child(23 , 'python')

# ob.view()

# #types of inheritence

# #1)single -> when the inheritance involves one child class and one parent class

# #2)multiple ->it invoves more than one parent class

# #3)multilevel -> the child class acts as a parent class for another child class

# #4)Hierarchical -> more than one type of inheritance

# #5)hybrid

# #eg of single inheritence

# class parent:

# def funcl(self):

# print("this is fun1")

# class child(parent):

# def func2(self):

# print("this is fun2")

# ob = child()

# ob.funcl()

# #eg of multiple inheritence

# class parent:

# def funcl(self):

# print("this is fun1")

# class parent2:

# def func3(self):

# print("this is function 3")

# class child(parent,parent2):

# def func2(self):

# print("this is fun2")

# ob = child()

# ob.funcl()

# ob.func3()

# #eg of multilevel inheritence

# class parent:

# def funcl(self):

# print("this is fun1")

# class parent2(parent):

# def func3(self):

# print("this is function 3")

# class child(parent2):

# def func2(self):

# print("this is fun2")

# ob = child()

# ob.funcl()

# ob.func3()

# #eg of heirarchical inheritence

# class parent:

# def funcl(self):

# print("this is fun1")

# class parent2(parent):

# def func3(self):

# print("this is function 3")

# class child(parent):

# def func2(self):

# print("this is fun2")

# ob = child()

# ob.funcl()

# ob1 =parent2()

# ob1.funcl()

# #eg of hybrid inheritence

# class parent:

# def funcl(self):

# print("this is fun1")

# class parent2(parent):

# def func3(self):

# print("this is function 3")

# class parent3:

# def func4(self):

# print("this is func4")

# class child(parent,parent3):

# def func2(self):

# print("this is fun2")

# ob = child()

# ob.funcl()

# ob.func4()

# #python super function

# #super funtions directly calls the parent class methods

# class parent:

# def func1(self):

# print('this is fun1')

# class child(parent):

# def func2(self):

# super().func1()

# print("this is func2")

# ob = child()

# ob.func2()

# #method overriding

# #it can be achieved to change functionally of parent class function

# class parent:

# def func1(self):

# print('this is fun1')

# class child(parent):

# def func1(self):

# print("this is func2")#over ridden method

# ob = child()

# ob.func1()

# #EXCEPTION HANDLING

# #EXCEPTION -> an exception is an event, which occurs during the execution of a program,that disrupts the normal floe of the program's instructions

# #HANDLING -> process of responding to the occurrence, during computation,of exceptional conditions requiring special processing - often changing the normal flow of progrsm execution

# #rasing an exception

# # x=10

# # if x> 5:

# #raise Exception('x should not be greater than 5. the value of x was:{}'.format(x))

# #assertion error exception

# #instead of waiting for a program to crsh midway,you can also start by making an assertion in python

# import sys

# # assert('linus' in sys.platform),"this code runs on linux only."

# #after assertion program will come to end and to continue the program use try and except blck

# def linux\_interaction():

# assert('linux' in sys.platform),"function can only run on linux systems."

# print("doing something....")

# try :

# linux\_interaction()

# except:

# print('linux function was not executed')

# #type2

# try :

# linux\_interaction()

# except AssertionError as error:

# print('linux function was not executed')

# print(error)

# try:

# with open('file.log') as file:

# read\_data = file.read()

# except:

# print('could not open file.log')

# # or

# #file not found

# # raised when a file or director is requested but doesn't exist. corresponds to errno ENOENT

# try:

# with open('file.log') as file:

# read\_data = file.read()

# except FileNotFoundError as fnf\_error:

# print(fnf\_error)

# except AssertionError as error:

# print(error)

# print('linux linux\_interaction() function was not excecuted')

# #key points

# # a try clause is executed up until the point where the first exception is encountered

# # inside the except clause, or the exception handler, you determine how the program responds to the exception

# # you can anticipate multiple exceptions and differentite how program should respond

# # avoid using bare except clauses.

# try:

# linux\_interaction()

# except AssertionError as error:

# print(error)

# else:

# print('Executing the else statement')

# #finally clause

# try:

# linux\_interaction()

# except AssertionError as error:

# print(error)

# else:

# try:

# with open('file.log') as file:

# read\_data = file.read()

# except FileNotFoundError as fnf\_error:

# print(fnf\_error)

# finally:

# print('cleaning up,irrespective of any exceptions.')

# #summary

# #• raise allows you to throw an exception at any time.

# #• assert enables you to verify if a certain condition is met and throw an exception if it isn't.

# #• In the try clause, all statements are executed until an exception is encountered.

# #• except is used to catch and handle the exception(s) that are encountered in the try clause.

# #• else lets you code sections that should run only when no exceptions are encountered in the try clause.

# #• finally enables you to execute sections of code that should always run, with or without any previously encountered exceptions.

*# from programming with mosh*

try:

    age = int(**input**('age: '))

**print**(age)

except ValueError:

**print**('Invalid value')

# #MODULE

# #used to break down the code to separate parts

# import egofmodule as mod

# #from egofmodule import add

# a=10

# b=5

# add = mod.add(a,b)

# print(add)

# sub = mod.sub(a,b)

# print(sub)

# #builtin modules

# import egofmodule

# print(dir(egofmodule))

# import sys

# a=sys.builtin\_module\_names

# print(a)

# # python modules search path

# import sys

# print(sys.path)

# import math

# print(math.factorial(5))

# import random

# print(random.randrange(0,50))

# import datetime

# print(datetime.date.today())

# # from matplotlib import pyplot as plt

# # plt.bar ( [0.25,1.25,2.25,3.25,4.25],[50,40,70,80,20],

# # label="bmw",width=.5)

# # plt.bar ( [.75,1.75,2.75,3.75,4.75],[80,20,20,50,60],

# # label="audi",color='r',width=.5)

# # plt.legend()

# # plt.xlabel('Days')

# # plt.ylabel('Distance (kms)')

# # plt.title('Information')

# # plt.show()

# # # piechart

# # days=[1,2,3,4,5]

# # sleeping=[7,8,6,11,7]

# # eating =[2,3,4,3,2]

# # working=[7,8,6,11,7]

# # playing=[8,5,7,8,13]

# # slices =[7,2,2,13]

# # activities = ['sleeping', 'eating', 'working', 'playing']

# # cold = ['c','m','r','b']

# # plt.pie(slice,

# # lables=activities,

# # colors=cols,

# # startangle=90,

# # shadow=true,

# # explode=(0,0.1,0,0),

# # autopct='%1.1f %%' )

# # plt.title('pie plot')

# # plt.show()

# #time and date

# #list of functions and discription

# # time() returns the number of seconds

# # ctime() returns the current date and time

# #sleep() stops execution of a thread for given duration

# # locsltime() returns the date and time in time.struct\_time format

# # gmtime() returns time.struct\_time in UTC format

# # mktime() returns the seconds passed since epoch pas output

# # asctime() returns a string representing the same time

# import time

# print(time.time())

# print(time.ctime(time.time()))

# print(help(time.time))

# print(time.localtime())

# #attributes of struct\_timeclass

# # tm\_year 0000,2019,9999

# #tm\_mon 1-12

# #tm\_mday 1-31

# # tm\_hour 0-23

# # tm\_min 0-59

# #tm\_sec 0-61

# #tm\_wday 0-6, monday as 0

# #tm\_yday 1-366

# #tm\_isdst 0,1,-1

# # a=time.localtime()

# # b=time.mktime(a)

# # print(b)

# # c=time.asctime(a)

# # print(c)

# # #print(help(time.strftime))

# # print(a)

# # x=time.strftime("%m/%d/%y")

# # print(x)

# # y="08 august 2019"

# # s=time.strptime(y,"%d %B %Y")

# # print(s)

# #datetime module

# #datetime() datetime constuctor

# #datetime.today() returns the current date and date

# # datetime.now() returns the curret date and time

# #date() take year,month and day as parameter and creates the corresponding date

# #time() takes hour, min ,sec , microsecond and tzinfo as parameter and creates the correspnding date

# #datetime.fromstamp() converts seconds to return the corresponding date and time

# # timedelta() it is the difference between different dates and times(durations)

# import datetime

# x=datetime.datetime(2019,6,7,4,30,54,678)

# print(x)

# print(datetime.datetime.now())

# print(datetime.datetime.now().year)

# print(datetime.datetime.now().month)

# print(datetime.datetime.now().hour)

# print(datetime.date(2019,7,5))

# print(datetime.time(3,45,23))

# d=datetime.timedelta(days=20)

# f=datetime.timedelta(days=30)

# g=f-d

# print(g)

# print(type(g))

# #NUMPY

# #numy is the core library for scientific computing in python

# #it provoides a high profermence multidimensional array onject,and tools for working woth these arrays

# import numpy as np

# a= np.array([(1,2,3),(4,5,6)])#2-d arraay

# print(a)

# #nump vs list

# #less memory

# #fast

# #convenient

# import time

# import sys

# s = range(1000)

# print(sys.getsizeof(5)\*len(s))#memory occupied by list

# d= np.arange(1000)

# print(d.size\*d.itemsize)#memory occupied by numpy array

# #time difference

# size = 100000

# l1 = range(size)

# l2 = range(size)

# a1 = np.arange(size)

# a2 = np.arange(size)

# start = time.time()

# result = [(x,y) for x,y in zip(l1,l2)]

# print((time.time()-start)\*1000)

# start = time.time()

# result =a1+a2

# print((time.time()-start)\*1000)

# #few operations

# a= np.array([(1,2,3,4),(3,4,5,6)])

# b= np.array([(1,2,3,4),(3,4,5,6)])

# print(a.ndim)#type of array

# print(a.itemsize)#byte size of each elements

# print(a.dtype)##data type of elements

# print(a.size)#total no of items in array

# print(a.shape)#rows and coloumns

# print(a.reshape(4,2))

# print(a[0,2])#accessing specific item using the arra length

# print(a[0:,3])#coloumn means all the rows including zero and in that row print index three # slicing

# a1= np.array([(1,2,3,4),(3,4,5,6),(7,8,9,11)])

# print(a1)

# print(a1[0:2,3])#coloumn means all the rows including zero but less than index 2 and in that row print index three # slicing

# print(a1[0:,3])#coloumn means all the rows including zero and in that row print index three # slicing

# a2=np.linspace(1,3,5)

# print(a2)#print five values which are equall placed between one and three

# a3= np.array([1,2,3])

# print(a3.max())

# print(a3.min())

# print(a3.sum())

# #axis 0 is all rows

# #axis 1 is all coloumn

# print(a1.sum(axis=0))

# print(a1.sum(axis=1))

# print(np.sqrt(a1))

# print(np.std(a1))#standard variation ie how much it varies from mean value

# j=np.array([1,2,3])

# k=np.array([1,2,3])

# print(j+k)

# print(j-k)

# print(j\*k)

# print(j/k)

# print(np.vstack((a,b)))#veritical stacking #appends array in veritcal mode

# print(np.hstack((a,b)))#horizontal stacking #appends array in horizontal mode

# a=np.array([(1,2,3),(4,5,6)])

# print(a.ravel()) #it combines all the items in array

# #numpy special funtions

# import matplotlib.pyplot as plt

# x= np.arange(0,3\*np.pi , 0.1)

# y= np.sin(x)

# plt.plot(x,y)

# #plt.show()

# x= np.arange(0,3\*np.pi , 0.1)

# y= np.cos(x)

# plt.plot(x,y)

# #plt.show()

# x= np.arange(0,3\*np.pi , 0.1)

# y= np.tan(x)

# plt.plot(x,y)

# #plt.show()

# ar = np.array([1,2,3])

# print(np.exp(ar))#e^x

# print(np.log(ar))#log x

# print(np.log10(ar))#log base 10

# #SciPY

# #Scipy is python library used to solve scientific and mathematical problems

# #built on numPy

# #allows manipulation and visualizing

# #numpy vs scipy

# #numpy and scipy used for mathematical and numerical analysis

# #numpy contains arraydata and basic operations

# #scipy consists of all the numerical code

# #scipy contains fully-featured versiond of mathemetical and scientific functions

# #scipy packages(name - description)

# #cluster clustering algorithms

# #constants physical and mathematical constants

# #fftpack fast fourier transform routines

# #integrate integration and ordinary differential equation solvers

# #interpolate interpolation and smoothing splines

# #io input and output

# #linalg linear algebra

# #ndimage n-dimensional image processing

# #odr orthogonal distance regression

# #optimize optimization and root-finding routines

# #signal signal processing

# #sparse sparse matrices and associated routines

# #spatial spatial data structures and algorithms

# #special special functions

# #stats statical distribution and functions

# #basic functions

# #help() returns info about any function,keywords,class etc

# # info() returns info about any function, keyword, class etc

# #source() returns the source code only for objects written in python

# from scipy import cluster

# #print(help(cluster))

# import scipy

# # scipy.info(cluster)

# # scipy.source(cluster)

# #special function

# from scipy import special

# a= special.exp10(2)

# print(a)

# b= special.exp2(3)

# print(b)

# c=special.sindg(90)

# print(c)

# d= special.cosdg(90)

# print(d)

# #integration functions

# #1) general integrations the quad fun calculatesvthe integral of a fnction which has one variale

# #2) double integration the dblquad function calculates doubles integral of a function which has 2 variables

# from scipy import integrate

# #print(help(integrate.quad))

# i=scipy.integrate.quad(lambda x:special.exp10(x),0,1)

# print(i)

# e=lambda x,y:x\*y\*\*2

# f=lambda x:1

# g = lambda x:-1

# print(integrate.dblquad(e,0,2,f,g))

# #fourier transformations

# #fourier analysis is a method that deals with expressing a function as a sum of periodic components and recoverinfng singnal fromthose components

# #the fft and ifft functions can be used to return the discrete fourier tranform of a real or complex sequence.

# from scipy.fftpack import fft,ifft

# import numpy as np

# x=np.array([1,2,3,4])

# y=fft(x)#fourier transformed

# y1=ifft(x)#inverse of fourier transform

# print(y)

# print(y1)

# #Linear algebra

# from scipy import linalg

# a=np.array([[1,2],[3,4]])

# b=linalg.inv(a)#inverse of matrix

# print(b)

# #interpolation functions

# #INTERPOLATION REFERS TO CONSTRUCTING new data pointswithin set of known data points. the scip.interpolate consists of spine functionsand classes, one-diimensional and multi-dimensional(univariate and multivariatr ) interpolation classes,etc.

# import matplotlib.pyplot as plt

# from scipy import interpolate

# x = np.arange(5, 20)

# y= np.exp(x/3.0)

# f=interpolate.interp1d(x,y)

# x1 = np.arange(6, 12)

# y1 = f(x1) # use interpolation function returned by `interp1d`

# plt.plot(x, y, 'o', x1, y1,'--')

# #plt.show()

# #data anyalysis with python

# #pandas

# #pandas is a software libraary written for python programming language for data maniulationand analysis

# #features

# #tabular data with hetrogeneously-typed columns

# #orderd and unorderd time series data.

# #arbitary matrix adta with row and column labels

# #any other form of observational / statistical data sets. the data actually need not be loaded at all to be place into a pandas data structure

# import pandas as pd

# xyz\_web = {'day':[1,2,3,4,5,6],"visitors":[1000,600,7000,1000,400,350],'bounce\_rate':[10,20,23,15,10,34]}

# df = pd.DataFrame(xyz\_web)

# print(df)

# #other operations

# #slicing dataframes

# print(df.head(2))

# print(df.tail(2))

# #joining and merging

# 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)

# merge2 = pd.merge(df1,df2,on = "HPI")

# print(merge2)

# df3 = pd.DataFrame({"INt\_rate":[2,1,2,3],"ind\_gdp":[50,45,45,67]},

# index = [2001,2002,2003,2004])

# df4 = pd.DataFrame({"low\_tier\_HPI":[80,90,70,60,45],"Unemployment":[2,1,2,3,65]},

# index = [2001,2002,2003,2004,2005])

# merge = df4.join(df3)

# print(merge)

# #changing the coloumn headers

# #changing the index

# import matplotlib.pyplot as plt

# from matplotlib import style

# style.use("fivethirtyeight")

# df5 = pd.DataFrame({"day":[1,2,3,4], "visitors":[200,100,230,300], "Bounce\_rate":[20,45,60,10]})

# df5.set\_index("day",inplace=True)

# #df5.plot()

# #plt.show()

# print(df5)

# df6 = df.rename(columns={"visitors":"users"})

# print(df6)

# #concatenation adding a coloumn

# 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)

# #data conversions

# #data munging = converting one type of file to other

# # import pandas as pd

# # print(help(pd))

# #not working just google code provided is for csv type

# # country = pd.read\_csv('D:\What I Know\WebDevelopment\node-js\project\ReriddenText.txt', index\_col = 0)

# # country.to\_html('edu.html')

# # import pandas as pd

# # import matplotlib.pyplot as plt

# # from matplotlib import style

# # style.use('fivethirtyeight')

# # country = pd.read\_csv('D:\What I Know\WebDevelopment\node-js\project\ReriddenText.txt', index\_col = 0)

# # df = country.head(5)

# # df = df.set\_index(["country code"])

# # sd = df.reindex(columns=['2011','2012'])

# # db = sd.diff(axis=1)

# # db.plot(kind = 'bar')

# # plt.show()

# from statistics import mean

# print(mean([1,2,2,2,1,3,4,1,5]))

# from statistics import median

# print(median([1,1,1,2,2]))

# from statistics import mode # mode = most repeateditem

# print(mode([1,1,1,2,2]))

# from statistics import variance

# print(variance([1,1,1,2,2]))

# #just info #pydoop is a python interface to haddop that allows youo to write mapreduce applications and interact with HDFS in pure python

# #matplotlib

# #type1

# from matplotlib import pyplot as plt7

# plt7.plot([1,2,3],[4,5,1])

# plt7.title('info')

# plt7.ylabel('Y axis')

# plt7.xlabel('X axis')

# #plt7.show()

# #adding style to our graph

# from matplotlib import pyplot as plt

# from matplotlib import style

# style.use('ggplot')

# x=[5,8,10]

# y=[12,16,6]

# x2=[6,9,11]

# y2=[6,15,7]

# plt.plot(x,y,'g',label='line one',linewidth=5)

# plt.plot(x2,y2,'c',label='line two',linewidth=5)

# plt.title('epic info')

# plt.ylabel('Y axis')

# plt.xlabel('X axis')

# plt.legend()

# plt.grid(True,color='k')

# #plt.show()

# #bar graph

# import matplotlib.pyplot as plt

# plt.bar([1,3,5,7,9],[5,2,7,8,2],label="Example one")

# plt.bar([2,4,6,8,10],[8,6,2,5,6],label="example two", color='g')

# plt.legend()

# plt.xlabel('bar number')

# plt.ylabel('bar height')

# plt.title('my plot yo!')

# #plt.show()

# #histogram

# import matplotlib.pyplot as plt

# population\_ages=[22,55,62,45,21,22,34,42,42,4,99,102,110,120,121,122,130,111,1115,112,80,75,65,54,44,43,42,48]

# bins=[0,10,20,30,40,50,60,70,80,90,100,110,120,130]

# plt.hist(population\_ages,bins,histtype='bar',rwidth=0.8)

# plt.xlabel('x')

# plt.xlabel('x')

# plt.title('histogram')

# plt.legend()

# #plt.show()

# #scatter plot

# import matplotlib.pyplot as plt

# x=[1,2,3,4,5,6,7,8]

# y=[5,2,4,2,1,4,5,2]

# plt.scatter(x,y,label='skitscat',color='k',s=25,marker="o")

# plt.xlabel('x')

# plt.xlabel('x')

# plt.title('scatter plot')

# plt.legend()

# #plt.show()

# #stack plot

# import matplotlib.pyplot as plt

# days=[1,2,3,4,5]

# sleeping = [7,8,6,11,7]

# eating = [2,3,4,3,2]

# working = [7,8,7,2,2]

# playing = [8,5,7,8,13]

# plt.plot([],[],color='m',label="sleeping",linewidth=5)

# plt.plot([],[],color='c',label="eating",linewidth=5)

# plt.plot([],[],color='r',label="working",linewidth=5)

# plt.plot([],[],color='k',label="playing",linewidth=5)

# plt.stackplot(days,sleeping,eating,working,playing,colors=['m','c','r','k'])

# plt.xlabel('x')

# plt.xlabel('x')

# plt.title('stack plot')

# plt.legend()

# #plt.show()

# #pie chart

# import matplotlib.pyplot as plt

# slice = [7,2,2,13]

# activities = ['sleeping','eating','working','playing']

# cols = ['c','m','r','b']

# plt.pie(slice,labels=activities,colors=cols,startangle=90,shadow=True,explode=(0,0.1,0,0),

# autopct='%1.1f%%')

# plt.title('pie plot')

# #plt.show()

# #mutilple plots

# import numpy as np

# import matplotlib.pyplot as plt

# def f(t):

# return np.exp(-t)\*np.cos(2\*np.pi\*t)

# t1 = np.arange(0.0,5.0,0.1)

# t2 = np.arange(0.0,5.0,0.02)

# plt.subplot(211)#plt.subplot(221)

# plt.plot(t1,f(t1),'bo' ,t2 ,f(t2))

# plt.subplot(212)#plt.subplot(222)

# plt.plot(t2,np.cos(2\*np.pi\*t2))

# #plt.show()

# #seaborn

# #used for data visualization & based on metplotlib

# #seaborn llows the creation of statisticl graphic

# #functions

# #allows comparison bw multiple variables

# #supports multi-plot grids

# #available univarite and bivariate visualizations

# #availibility of different colour plaettes

# #estimates and plots linear regression automatically

# #seaborn vs matplotlib

# #seaborn fixes 2 shortcoming of matplotlib

# # matplotlib settings are difficult to figure out. seaborn comes with nummerous customized themes and high-levl interfaces.

# # matplotlib doesn't serve well when it comes to dealing with dataframes, while seaborn functions actually work on dtaframes

# import numpy as np

# import pandas as pd

# import matplotlib.pyplot as plt

# import seaborn as sns

# #inbuit examples files , online available at github.com/mwaskom/seaborn-data

# a = sns.load\_dataset("flights")

# print(a)

# print(sns.relplot(x="passengers",y="month",data=a))

# print(sns.relplot(x="passengers",y="year",data=a))

# b = sns.load\_dataset("tips")

# print(sns.relplot(x="time",y="tip",data=b,kind="line"))

# print(sns.catplot(x="day",y="total\_bill",data=b))

# print(sns.catplot(x="day",y="total\_bill",data=b,kind="violin"))

# c=np.random.normal(loc = 5, size = 100,scale = 2)

# print(sns.distplot(c))

# #no code for multivariate

# #multiplot grids

# # a=sns.load\_dataset("iris")

# # b=sns.FacetGrid(a, col="species")

# # print(b.map(plt.hist,"sepal\_length"))

# #pair grid

# a=sns.load\_dataset("flights")

# b=sns.PairGrid(a)

# b.map(plt.scatter)

# #to change colour of pallets

# sns.set(style="darkgrid")

# a=sns.load\_dataset("flights")

# b=sns.PairGrid(a)

# b.map(plt.scatter)

# #box plots

# sns.set(style="white",color\_codes=True)

# a=sns.load\_dataset('tips')

# print(sns.boxplot(x="day",y="total\_bill",data=a))

# #to remove any axis

# sns.set(style="white",color\_codes=True)

# a=sns.load\_dataset('tips')

# print(sns.boxplot(x="day",y="total\_bill",data=a))

# sns.despine(offset=10,trim=True)#x-axis is removed

# #available colour options

# c=sns.color\_palette()

# print(sns.palplot(c))

# #open cv

# #it is a librar of python designed to solve computer vision problems

# import cv2

# img = cv2.imread("D:\\prince\\Pictures\\DSC\_8022 (2).jpg",1)

# # one stands for color image

# img2 = cv2.imread("D:\\prince\\Pictures\\DSC\_8022 (2).jpg",0)

# # zero stands for black and white image

# print(img)

# print(img2)

# print(type(img))

# print(type(img2))

# print(img.shape)# at end of results 3 stands for 3 channel image

# print(img2.shape)

# #cv2.imshow("hi",img)

# #cv2.imshow("hi",img2)

# #cv2.waitKey(0)

# ##cv2.waitKey(2000)

# #cv2.destroyAllWindows()

# #Resizing images

# resized = cv2.resize(img,(600,600))

# #resize\_image = cv2.resize(img,(int(img.shape[1]/2),int(img.shape[0]/2)))

# cv2.imshow("hi",resized)

# cv2.waitKey(2000)

# cv2.destroyAllWindows()

# # Face detection

# #create a cascadeclassifier object, it will contain the features of face

# face\_cascade = cv2.CascadeClassifier("C:\\Python38\\Lib\\xml\\haarcascade\_frontalface\_default.xml")

# #reading the image as it is

# img = cv2.imread("D:\\prince\\Pictures\\DSC\_8022 (2).jpg")

# #reading the image as gray scale image

# gray\_img = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

# #serach to co-ordinates of the imageMultiscale is the method to search for the face rectangel co-ordinates

# #scale factor will decrease the shape value by 5%, until the face is found, smaller this value, the greater is the accuracy

# faces = face\_cascade.detectMultiScale(gray\_img,scaleFactor=1.05,minNeighbors=5)

# #detect

# print(faces)#tells the co ordinartes

# for x,y,w,h in faces:

# #x = x-co ordinate

# #y = y- co ordinate

# #w = width

# #h = height

# img = cv2.rectangle(img,(x,y), (x+w,y+h),(0,255,0),3)

# #rectangle - method to create face rectangle

# #(0,255,0) - rgb value of the rectangle outline

# # 3 width of rectangle

# cv2.imshow("gray",img)

# cv2.waitKey()

# cv2.destroyAllWindows()

# # import cv2, time

# # vedio = cv2.VideoCapture(0)# this will read the first frame/image of the vedio

# #0 in vedio capture is for primary camera, 1 for secondary camera

# # check,frame = vedio.read()

# # # check - it is bool data type, returns true if python is able to read the capture object

# # # frame - it is numy array, it represents the first image that vedio captures

# # print(check)# it returns true if python is able ot read camera

# # print(frame)# it will return the the 3-d array

# # time.sleep(3)

# # cv2.imshow('capturing',frame)

# # cv2.waitKey(0)

# # vedio.release()

# # cv2.destroyAllWindows()

# #how to capture the vedio,, instead of first image/frame of the vedio

# #not working

# # import cv2,time

# # vedio = cv2.VideoCapture(0)

# # a = 1

# # while True:

# # a = a+1

# # check,frame = vedio.read()

# # print(frame)

# # gray = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

# # cv2.waitKey(1) #this will generate a new frame after every 1 milliseconds

# # key = cv2.waitKey(10000)

# # if key == ord('q'):

# # #once we press q the window will destroy

# # break

# # print(a)# this will print the number of frames

# # vedio.release()

# # cv2.destroyAllWindows()

# # motion detector

# # import cv2,time,pandas

# # from datetime import datetime

# # first\_frame = None

# # status\_list = [None,None]

# # times = []

# # df=pandas.DataFrame(columns=["start","end"])

# # vedio = cv2.VideoCapture(0)# create a vesdiom capture object to record vedio using web cam

# # while True:

# # check,frame = vedio.read()

# # status = 0

# # gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)#convert the frame color to gray scale

# # gray = cv2.GaussianBlur(gray,(21,21),0)# convert the gray scale frame to guassianblur

# # if first\_frame is None: #used to store first image/frame of vedio

# # first\_frame = gray

# # continue

# # delta\_frame

# #pyhton web-development django

# #terminal = django-admin startproject django-project

# #to run server = python manage.py runserver

# # to return control to terminal press ctrl+c

# # to start a new app = python manage.py startapp DemoApp

# # to create a connection add urls.py file in same directory

# # and paste

# # from django.urls import path

# # from. import views #or the name of index file

# # urlpatterns = [

# # path('', views.hi,name='home-page'),

# # ]

# #in urls.py ile of django project

# #from django.urls import include

# #and add path with app's folder name

# #path('',include('DemoApp.urls'))

# #now create new directory with name templates

# #and inside templates create new directory with same name as of DemoApp's folder name

# #create a html file inside demoapp and do minimal coding

# #go to apps.py(app folder) and cop the class name

# #go to settings.py(project folder) and in class INSTALLED\_APPS paste is in form of appFolderName.apps.classname

# #in my case it looks like 'DemoApp.apps.DemoappConfig',

# #now in views (app folder) change return HttpResponse('<h1>This is my home age</h1>')

# # to return render(request,'appfolder name/index.html')

# # web scraping

# #one need to fetch some info from the website. to so so, copy and paste the data display by the website which is very tedious job that may take many hours or sometime days to complete

# # webscraping is a technique emplyed to extract large amount of data from websites wherby the data is extracted and saved to local file in your computer

from bs4 import BeautifulSoup as soup

from urllib.request import urlopen as uReq

my\_url = 'https://www.flipkart.com/search?q=iphone&otracker=start&as-showb=on&as=off'

uCliet = uReq(my\_url)

page\_html = uCliet.read()

uCliet.close()

page\_soup = soup(page\_html, "html.parser")

containers = page\_soup.findAll("div",{"class": "bhgxx2 col-12-12"}) # div of item 1

print(len(containers))

print(soup.prettify(containers[0]))

container = containers[0]

#print(container.div.img["alt"])

price = container.findAll("div",{"class":"col col-5-12 \_2o7WAb"})

#print(price[0].text)

rating = container.findAll("div",{"class":"niH0FQ"})

#print(rating[0].text)

# filename="webscrapping.csv"

# f=open(filename,"w")

# headers="product name,pricing,ratings\n"

# f.write(headers)

# for container in containers:

# products\_name = containers.div.img["alt"]

# product\_price = container.findAll("div",{"class":"col col-5-12 \_2o7WAb"})

# price = price\_container[0].text.strip()

# rating\_container = container.findAll("div",{"class":"niH0FQ"})

# rating = rating\_container[0].text

# print("product\_name:" + products\_name)

# print("price:" + price)

# print("rating:" + rating)

# #string prasing

# trim\_price = ''.join(price.split(','))

# rm\_rupee = trim\_price.split("$")

# add\_rs\_price = "rs." + rm\_rupee[1]

# split\_price = add\_rs\_price.split('E')

# final\_price = split\_price[0]

# split\_rating = rating.split(" ")

# final\_rating = split\_rating[0]

# print(products\_name.replace(",", "|") + "," + final\_price + "," + final\_rating + "\n")

# f.write(products\_name.replace(",", "|") + "," + final\_price + "," + final\_rating + "\n")

# f.close()

*# functions*

*#  forom programming with mosh*

def **greet**(first\_name,last\_name):

**print**(f'Hi {first\_name} {last\_name}!')

**print**("welcome abroad")

**print**("start")

greet("jhon" ,"marie")

**print**("Finish")

*# emoji\_converter+reusable\_function.py*

*#  from programming with mosh*

*# don't know how it can be used as reuseable function*

def **emoji**(message):

    words = message.split(" ")

    emojis = {

        ":)": "",

        ":(": ""

    }

    output = ""

    for word in words:

        output += emojis.get(word,word) + " "

message = **input**(">")

**print**(emoji(message))