Question 1 - Have you ever worked on any networking related program (whether in python or any other language). Networking means working with TCP/UDP.

Ans:- No.

Question 2 - Do you have any knowledge or experience with SIP (session initiation protocol). Again this experience does not necessarily need to be in python.

Ans:- No.

Question 3 - Do you have knowledge of databases (either NoSQL or SQL). Which ones?

Ans:- Yes. I have beginner level knowledge of “SQL”. Actually, I had “Oracle” in my syllabus of both BCA & MCA.

Question 4 - Have you ever worked with an ORM like sqlalchemy/mongoengine in python.

Ans:- No.

Question 5 - Have you ever worked with any microservices/web based frameworks in python (like Flask/Sanic/Django).

Ans:- No. I was just about to start learning django in lockdown period but unfortunately one day my laptop does not seems to get started. Since then my laptop is in the repair shop & i couldn’t learn it.

Question 6 - Have you ever used any concurrency framework like asyncio/gevents/twisted in python? If yes, which ones.

Ans:- No.

**Exercise 2**

**Agent selector**

You are given the following data for agents

agent

* is\_available
* available\_since (the time since the agent is available)
* roles (a list of roles the user has, e.g. spanish speaker, sales, support etc.)

When an issue comes in we need to present the issue to 1 or many agents based on an agent selection mode. An agent selection mode can be all available, least busy or random. In “all available mode” the issue is presented to all agents so they pick the issue if they want. In least busy the issue is presented to the agent that has been available for the longest. In random mode we randomly pick an agent. An issue also has one or many roles (sales/support e.g.). Issues are presented to agents only with matching roles.

Please write a function that takes an input the list of agents with their data, agent selection mode and returns a list of agents the issue should be presented to.

Note - We have had many people asking questions if the function needs one more argument. In the simple case no, but in a real implementation yes it will need an issues list or an existing issues queue. Please do add a queue of issues to the function if you want to implement the advanced case.

Also note that is\_available is a boolean value.

Program:-

import sys

import random

def Record\_issue():

confirmation = input("is there any issue? yes or no: ")

if(confirmation == 'y' or 'yes' or 'YES' or 'Yes'):

cust\_name = input("Enter customer name: ")

cust\_age = int(input("Enter age: "))

issue\_category = input("Enter type of issue: ")

issue\_des = input("Issue description: ")

print("issue recorded")

elif(confirmation == 'n' or 'no' or 'No' or 'NO' or ''):

sys.exit(0)

else:

return confirmation

Record\_issue()

#=============================================================================

class agent\_info:

def \_\_init\_\_(self, name, is\_available, mode, available\_since, role):

self.name = name

self.is\_available = is\_available

self.mode = mode

self.available\_since = available\_since

self.role = role

#creating some examples of agents info

#available\_since value has been count in "days"

agnt1 = agent\_info('Rohan','yes','all\_available',25,'maintainance')

agnt2 = agent\_info('Mike','yes','all\_available',5, 'Accounitng')

agnt3 = agent\_info('jiya','no','all\_available',0, 'networking')

agnt4 = agent\_info('Karan','no','all\_available',0, 'development, support')

agnt5 = agent\_info('Krisan','yes','all\_available',7, 'networking, maintainance')

print(" ")

agnt6 = agent\_info('mika','yes','least\_busy',0,'maintenance')

agnt7 = agent\_info('shiro','no','least\_busy',0,'support')

agnt8 = agent\_info('nik','yes','least\_busy',0,'networking')

agnt9 = agent\_info('ankita','no','least\_busy',0,'development, support')

agnt10 = agent\_info('steve','yes','least\_busy',0,'networking, maintainance')

#creating an empty dictionary

agents = {}

#fuction for displaying information

def all\_agntInfo(agentList):

for item in agentList:

agents[item.name] = (item.is\_available, item.mode, item.available\_since, item.role)

#calling function

all\_agntInfo([agnt1, agnt2, agnt3, agnt4, agnt5, agnt6, agnt7, agnt8, agnt9, agnt10])

#printting ouput in following format

for key, val in agents.items():

print(str(key) +":"+ str(val))

#=============================================================================

#function for comparing agents & returning lists accordingly

def getAgents(agents, mode):

mode = all\_agntInfo.is\_available

agents = all\_agntInfo

if (mode == 'all\_available'):

return(all\_agntInfo.mode == 'all\_avaliable')

elif(mode == 'least\_busy'):

return(all\_agntInfo.mode == 'least\_busy')

for agent in all\_agntInfo :

if(all\_agntInfo.is\_available == 'yes' and all\_agntInfo.role == issue\_category):

return (all\_agntInfo.role == issue\_category)

elif (mode == 'random'):

all\_agntInfo = random.choice(agents)

print(all\_agntInfo)

#calling function

getAgents(agents, mode)

Output:-

