Group membes:Chetan Sharma,Santosh,Sandesh,Shubam
Roll Numbers:10,9,11,12
Reg No:11810452
Project Number:3

Part 2:

Report of Project:Time Table generator for Faculty Using Rule Based Approach

#----MODULES WE USED ARE MENTIONED THERE

```
import prettytable
import random
```

#USE OF PRETTY TABLE:Pretty table module basically used to create tabular structure in a command Line.So there is no need to use Tkinter or other graphical modules of python.
Link of DOC Ptable:https://ptable.readthedocs.io/en/latest/

#USE OF RANDOM:Random Module basically used to get random number by specifying a range for it.
In this project it is used to shuffle the elements or teachers in Teachers LIST.

#Taking Input of Teachers

```
no_of_teachers=int(input("Enter number of teachers:"))
```

```
# VARIABLES AND DATA STRUCTURES USED
```

```
flag=no_of_teachers
#Storing teachers in flag variable

subject=[]
teachers=[]

time=['9-10','10-11','11-12','12-1','1-2','2-3','3-4','4-5']
days=['Mon','Tue','Wed','Thur','Fri','Sat']
Mon=[]
Tue=[]
```

```
Wed=[]
Thur=[]
Fri=[]
Sat=[]


flag_list=[]
ex=[]
ex1=[]
ex2=[]
ex3=[]
ex5=[]
ex4=[]


fulldaylist=[Mon,Tue,Wed,Thur,Fri,Sat]
absenties=[]
```

#Taking Name of Teachers with respect to Subjects and Appeding in <u>Teachers LIST</u>

<u>and subject LIST</u>

```
while no_of_teachers>0:

name=input("Enter name of teacher:")
subject_name=input("Enter subject name of "+name +" :")
subject.append(subject_name)
teachers.append(name)
no_of_teachers=no_of_teachers-1
```

#Printing Teachers and Subject List

```
def display_details():

display=prettytable.PrettyTable()

display.add_column(“Teachers
Name”,teachers);display.add_column(“Subjects”,subjects);print(display)

display_details()
```

#This function switch() works like switch cases ,With this function we append the teachers in desired List of days.

If faculty want to work on monday then append in monday vica versa

```
#PART 1:Arranging Table


def switch(dy):
if dy=='Mon':
Mon.append(teachers[i])

elif dy=='Tue':
Tue.append(teachers[i])
elif dy=='Wed':
Wed.append(teachers[i])
elif dy=='Thur':
Thur.append(teachers[i])
elif dy=='Fri':
Fri.append(teachers[i])
elif dy=='Sat':
Sat.append(teachers[i])
```

#This Function handle the input of user if user enter greater than 8 hrs a day than it will give error msg about wrong step.

ie.Increment value of s=s+1 which is used below to break the loop

so that is why we name it stoper.

#Otherwise pass the result

```
def stoper(nhrs,s):
if nhrs<8 and nhrs>0:
if len(Mon)<8 and len(Mon)>0:
pass
elif len(Tue)<8 and len(Tue)>0:
pass
elif len(Wed)<8 and len(Wed)>0:
pass
elif len(Thur)<8 and len(Thur)>0:
pass
```

```
elif len(Fri)<8 and len(Fri)>0:
pass
elif len(Sat)<8 and len(Sat)>0:
pass
else:
s=s+1
return s
else:
s=s+1
return s
```

#This function helps to evaluate the length of hours of teachers in specific day.For example:

Let we have MON list and Assume TEACHER 1 comes and enter 7 long hours of working whole Monday,

then we left 8-7=1 hour for other teacher

If Other Teacher enter >1 hour than it gives ERROR message to user.

```
def length_equalizer():
for i in range(len(fulldaylist)):
if(len(fulldaylist[i])<8):
extra=8-len(fulldaylist[i])
for j in range(0,extra):
fulldaylist[i].append(0)
extra=0
```

#Random Function helps to Shuffle the lists randomly:eg we have a list of [1,2,3,4]
then it will give maximum 2x2x2x2=16 possible outcomes which enough to get unique results like:[2,3,1,4].....so on.

```
def randomer(fulldaylist_modified):
for i in range(len(fulldaylist_modified)):
random.shuffle(fulldaylist_modified[i])
```

#The most important function to give ease in designing the structure of table in single command .In this function we use PrettyTable Module to print struture of table by appending lists we made above in row or column order.

```
def printer():
length_equalizer()
fulldaylist_modified=[Mon,Tue,Wed,Thur,Fri,Sat]
flag_list=fulldaylist_modified
randomer(fulldaylist_modified)
pretty=prettytable.PrettyTable()
pretty.add_column(' ',time)
pretty.add_column('Mon',Mon)
pretty.add_column('Tue',Tue)
pretty.add_column('Wed',Wed)
pretty.add_column('Thur',Thur)
pretty.add_column('Fri',Fri)
pretty.add_column('Sat',Sat)
print(pretty)
v=0
```

#Now this lines of code handle or call the function one after another

Here flag is number of teachers and days is a list we predefined in above

Note: s=0 is predefined and goes in stoper and return stop value in case of extra hours So that we stop the LOOP .

```
for i in range(0,flag):
for d in range(len(days)):
nhrs=int(input("Enter number of hours for " +teachers[i]+" "+ days[d] +" :"))
for x in range(0,nhrs):
switch(days[d])
s=0
o=stoper(nhrs,s)
if(o==1):
break
else:
v=v+1
```

#PART 2:

#Here v decides whether the previous results

successfully run If any Problem arise than

it will not proceeds to absent management blocks

This will first take Input of Absenties

```
#PART -2:Absenties Handling

if(v>0):
printer()
no_of_teachers_absents=int(input("Enter number of teacher be absents?:"))
```

And this rule specify teachers absenties should be >=1 Otherwise it will end the Result

```
if(no_of_teachers_absents>=1):

for i in range(0,no_of_teachers_absents):
name_of_absenties=input("Enter name of absenties:")
absenties.append(name_of_absenties)
print("Absenties are:",absenties)
```

And This rule take details of absenties and Append in absenties LIST And next step Print the absenties details

Now,If teachers on Monday has No zero lecture and teachers are not absent then

```
for s in range(len(Mon)):
if(Mon[s]!=0):
if Mon[s] not in absenties:
ex.append(Mon[s])
print(ex)
for i in range(len(absenties)):
for j in range(len(Mon)):
```

We move further and append the new list of teachers eg.ex[]

After that if any absenty is found on Monday then del that absenty and replace randomly any teacher who will come on that day.

```
if absenties[i]==Mon[j]:
del Mon[j]
if(len(ex)!=0):
Mon.insert(j,ex[random.randint(0,len(ex)-1)])
else:
Mon.insert(j,0)
# ------------------------------------------------------------



for s in range(len(Tue)):
if(Tue[s]!=0):
if Tue[s] not in absenties:
ex1.append(Tue[s])
for i in range(len(absenties)):
for j in range(len(Tue)):
if absenties[i]==Tue[j]:
del Tue[j]
if(len(ex1)!=0):
Tue.insert(j,ex1[random.randint(0,len(ex1)-1)])
else:
Tue.insert(j,0)

# ------------------------------------------------------------

for s in range(len(Wed)):
if(Wed[s]!=0):
if Wed[s] not in absenties:
ex2.append(Wed[s])
for i in range(len(absenties)):
for j in range(len(Wed)):
if absenties[i]==Wed[j]:
del Wed[j]
if(len(ex2)!=0):
Wed.insert(j,ex2[random.randint(0,len(ex2)-1)])
else:
Wed.insert(j,0)

# ------------------------------------------------------------

for s in range(len(Thur)):
if(Thur[s]!=0):
if Thur[s] not in absenties:
ex3.append(Thur[s])
for i in range(len(absenties)):
for j in range(len(Thur)):
if absenties[i]==Thur[j]:
```

Previous Steps Repeated....
For Tue to Saturday..

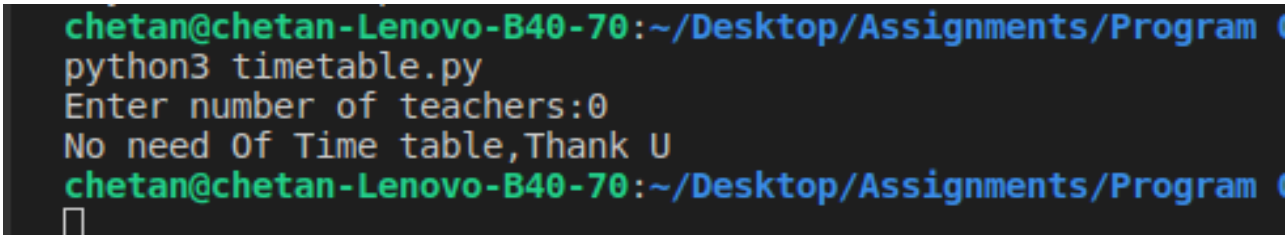
```
del Thur[j]
if(len(ex3)!=0):
Thur.insert(j,ex3[random.randint(0,len(ex3)-1)])
else:
Thur.insert(j,0)

# ----------------------------------------------------------------

for s in range(len(Fri)):
if(Fri[s]!=0):
if Fri[s] not in absenties:
ex4.append(Fri[s])
for i in range(len(absenties)):
for j in range(len(Fri)):
if absenties[i]==Fri[j]:
del Fri[j]
if(len(ex4)!=0):
Fri.insert(j,ex4[random.randint(0,len(ex4)-1)])
else:
Fri.insert(j,0)
# ----------------------------------------------------------------

for s in range(len(Sat)):
if(Sat[s]!=0):
if Sat[s] not in absenties:
ex5.append(Sat[s])
for i in range(len(absenties)):
for j in range(len(Sat)):
if absenties[i]==Sat[j]:
del Sat[j]
if(len(ex5)!=0):
Sat.insert(j,ex5[random.randint(0,len(ex5)-1)])
else:
Sat.insert(j,0)

def printer2(Mon,Tue,Wed,Thur,Fri,Sat):
pretty2=prettytable.PrettyTable()
pretty2.add_column(' ',time)
pretty2.add_column('Mon',Mon)
pretty2.add_column('Tue',Tue)
pretty2.add_column('Wed',Wed)
pretty2.add_column('Thur',Thur)
pretty2.add_column('Fri',Fri)
pretty2.add_column('Sat',Sat)
print(pretty2)
printer2(Mon,Tue,Wed,Thur,Fri,Sat)
```

Now,Use Second Printer2 for new Pretty table sturcture...

```
else:
print("Please,try to write the number of working hours in a day from 0 to till 8 hrs")
```

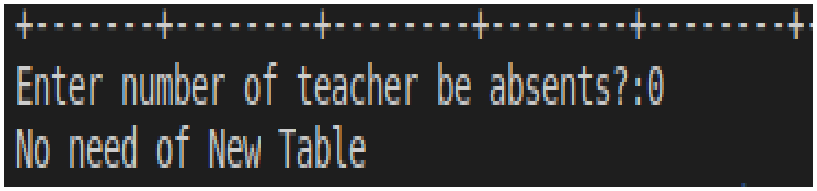
*********TEST CASES AND THEIR RESULTS WITH SCREENSHOTS**********

Case 1:When we enter no of teachers <=4 then



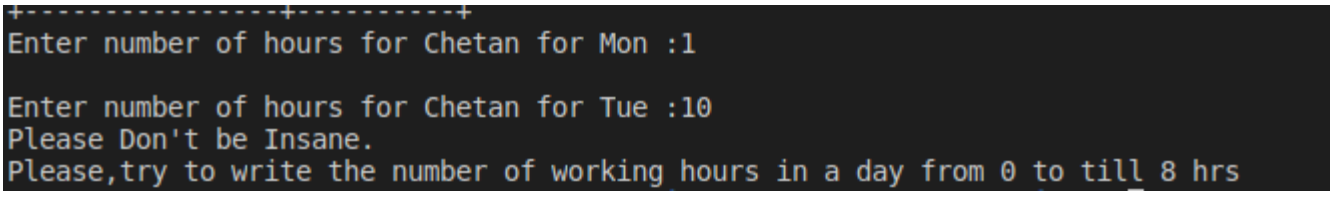
PASS

Case 2:When we enter no of absents teachers 0 or less



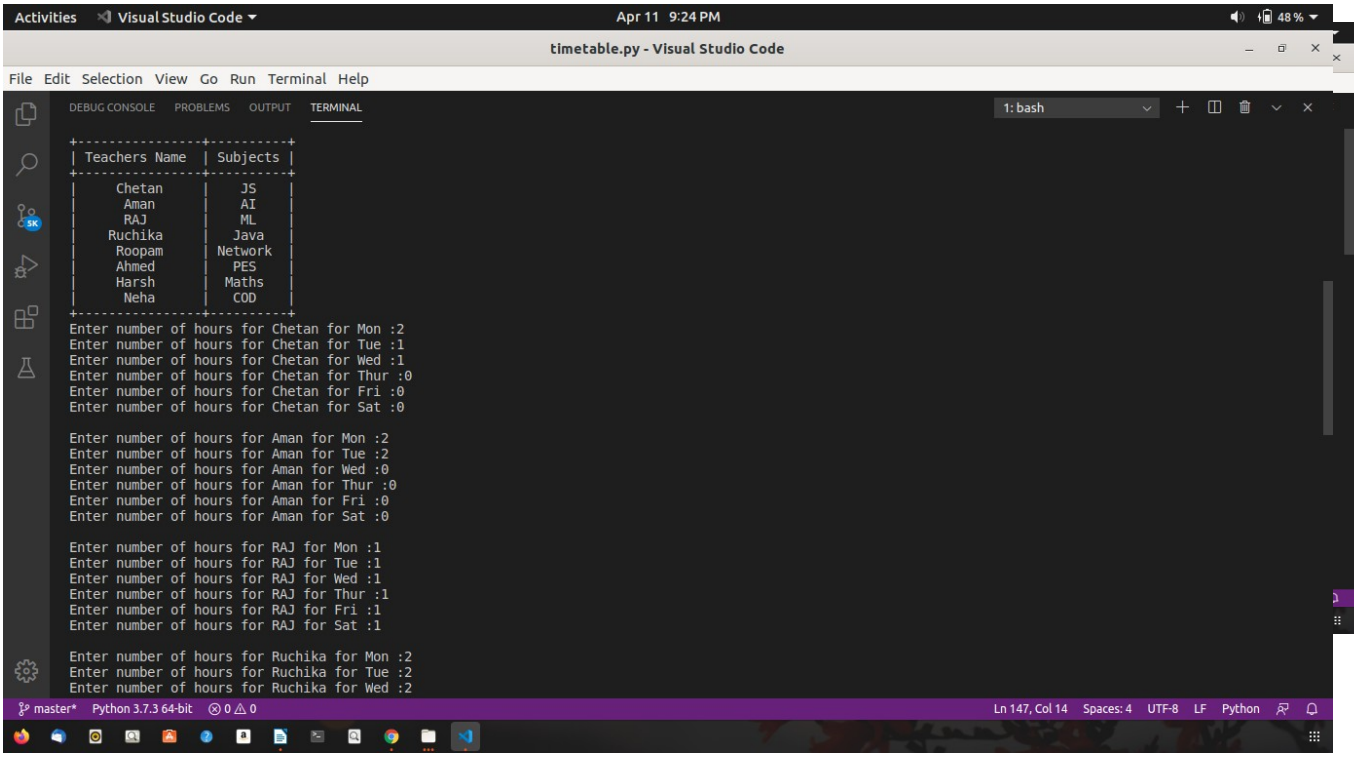
PASS

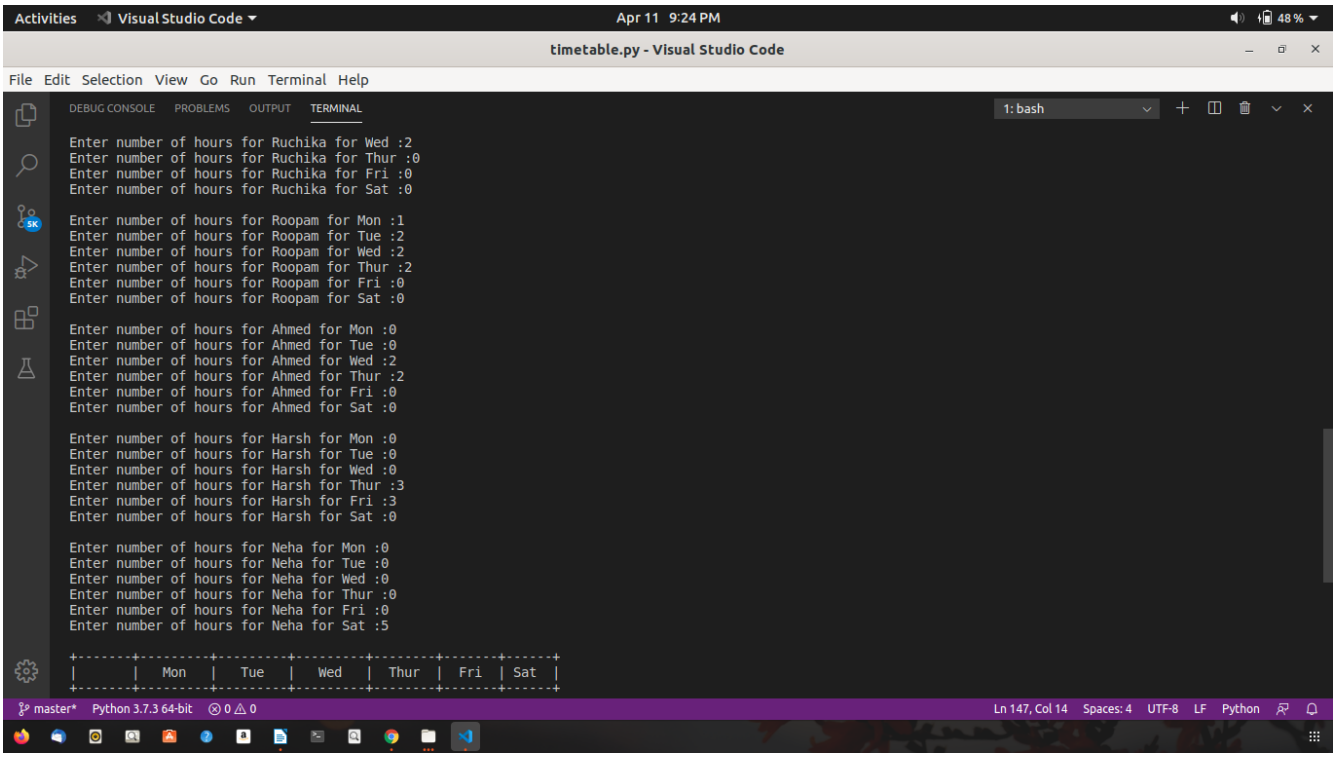
Case 3:When we enter more than working hours for each teacher

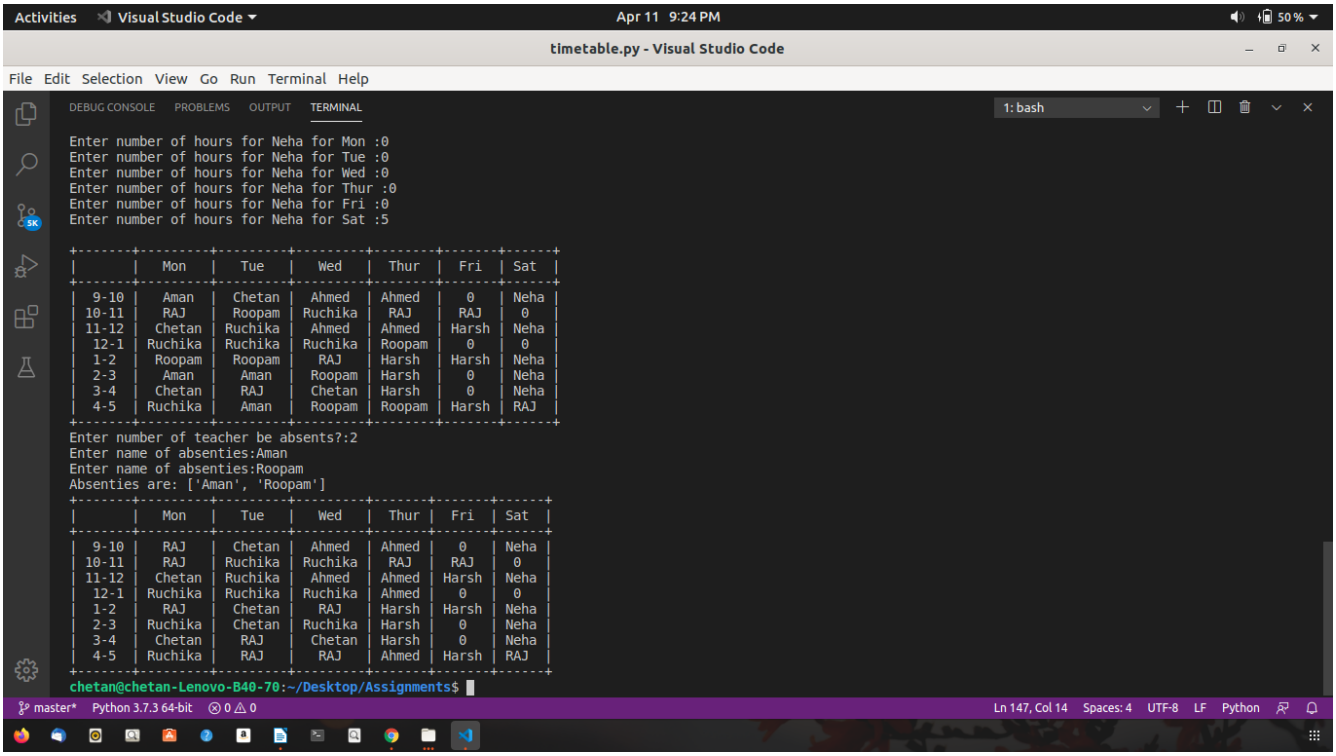


PASS

Case 4: When No of teachers absents is 2

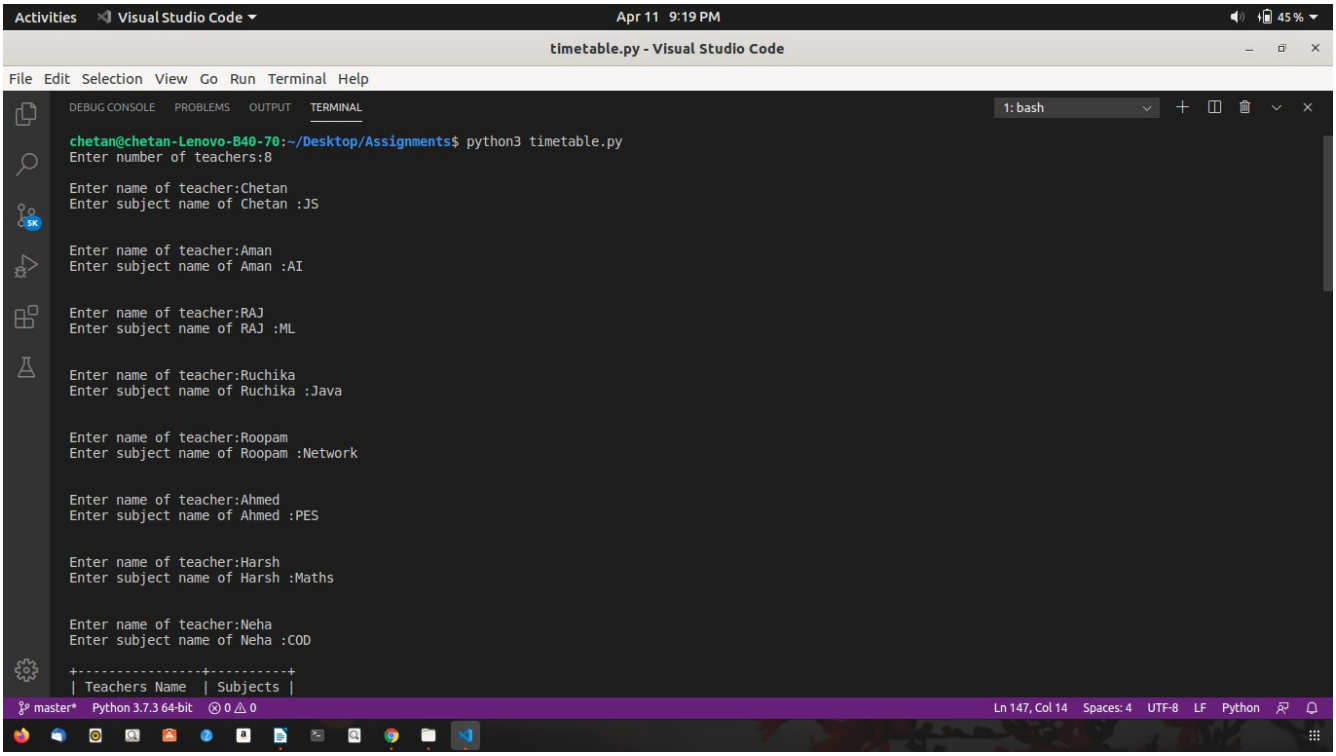


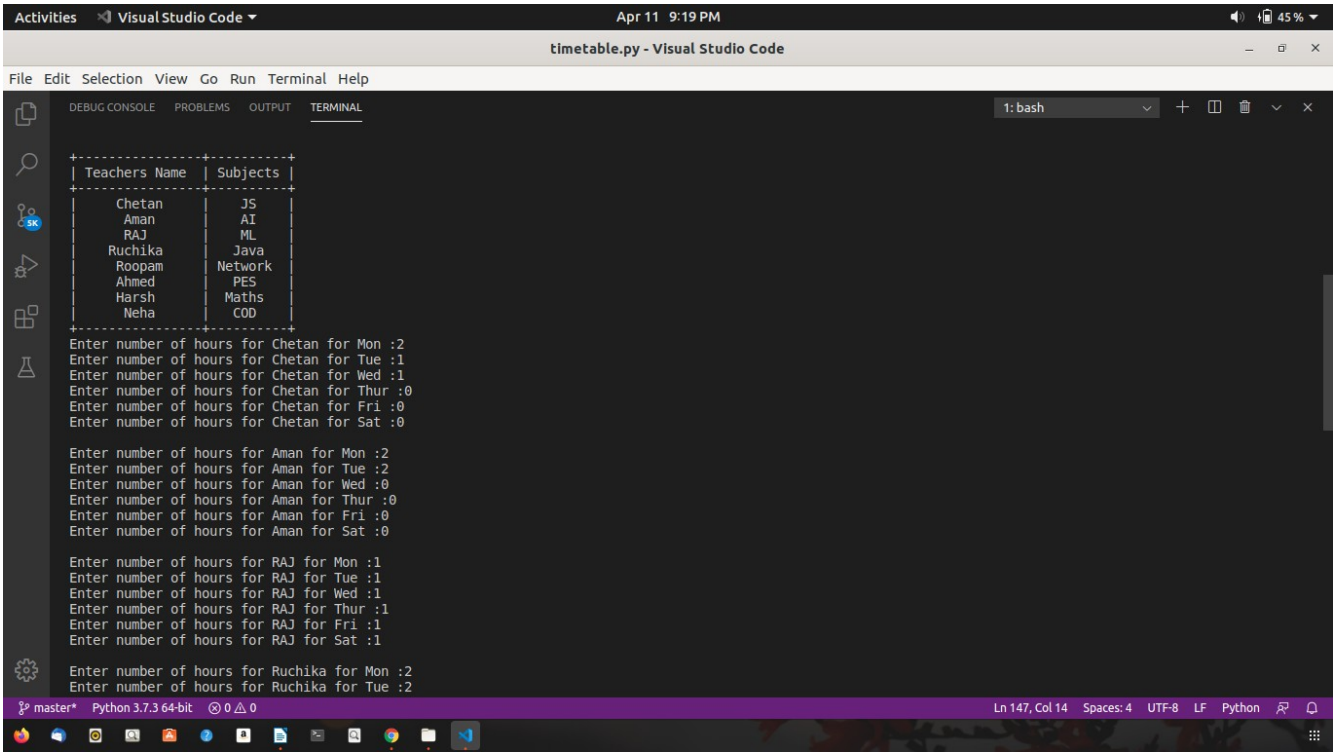


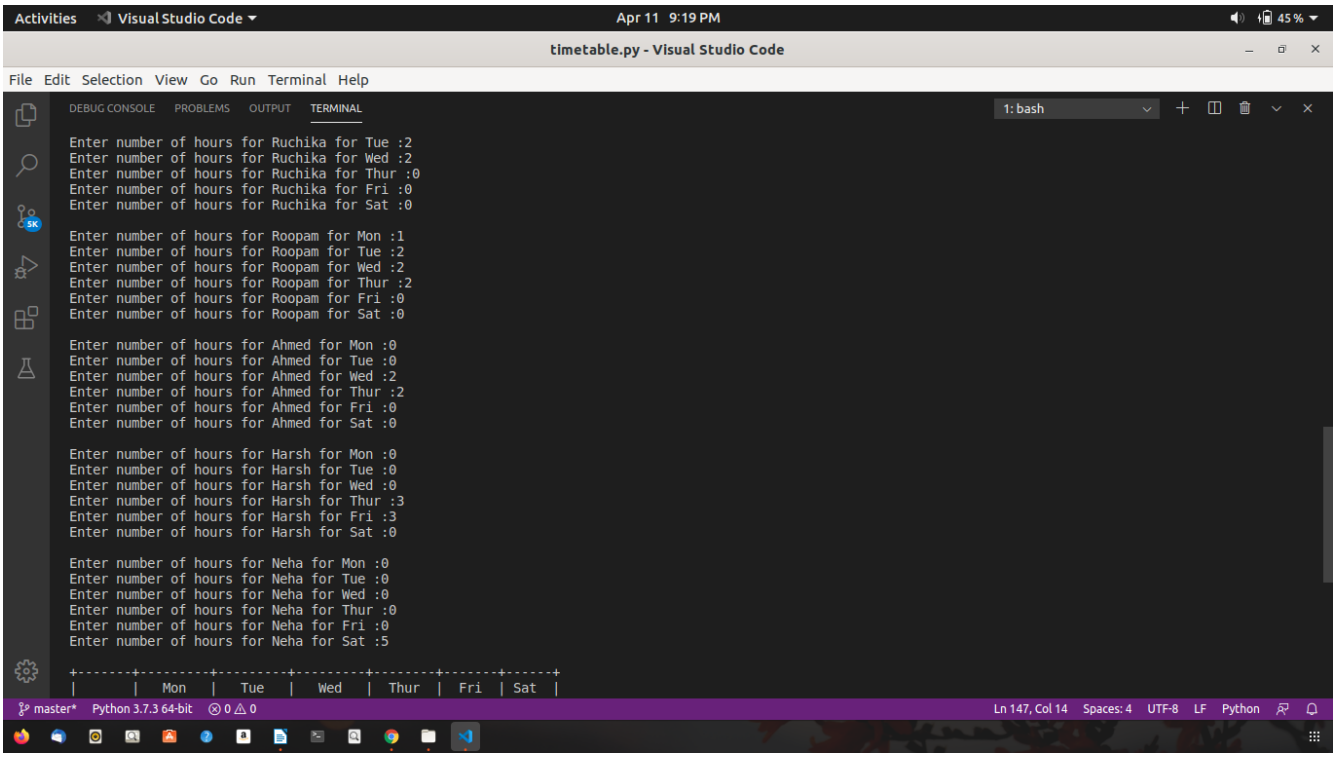


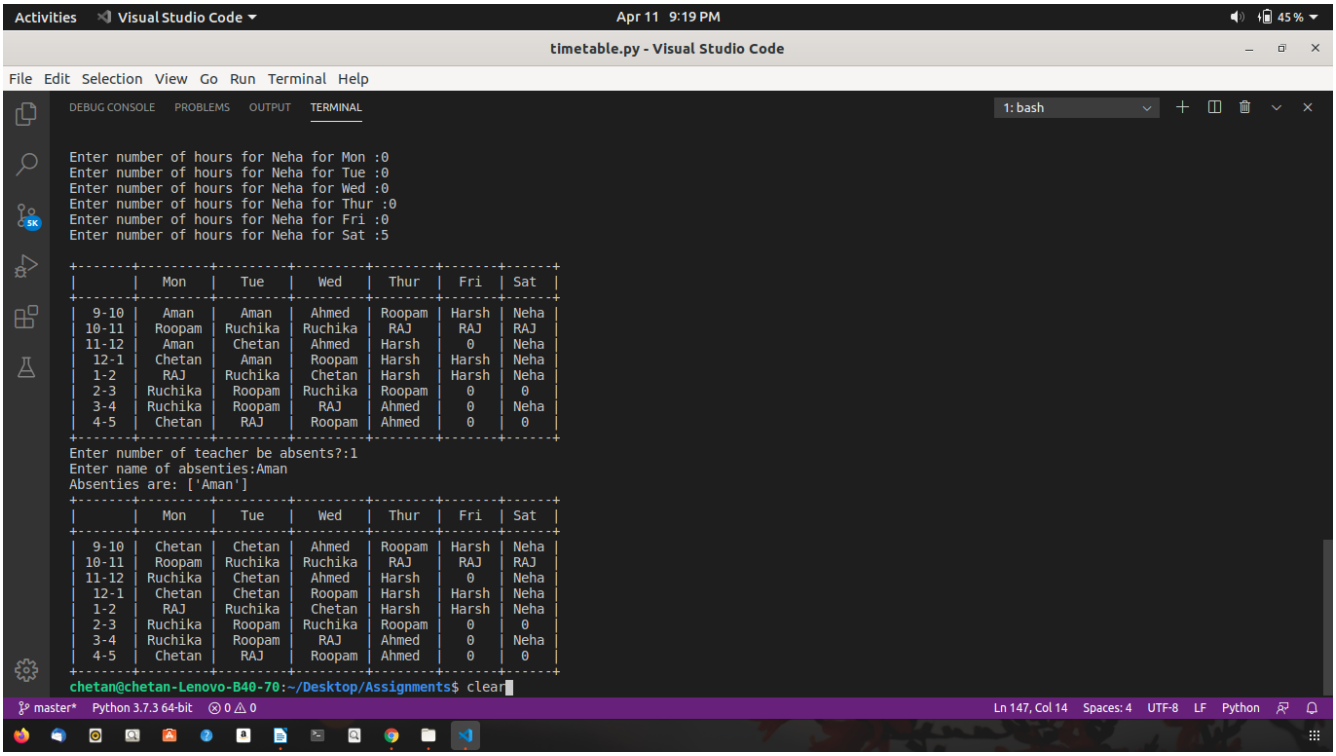
PASS

Case 5:When Number of teachers absents is 1




Activities
Visual Studio Code
Apr 11 9:19 PM
45 %
timetable.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL
1: bash
+----------------+----------+
| Teachers Name | Subjects |
+----------------+----------+
Chetan	JS
Aman	AI
RAJ	ML
Ruchika	Java
Roopam	Network
Ahmed	PES
Harsh	Maths
Neha	COD
+----------------+----------+
Enter number of hours for Chetan for Mon :2
Enter number of hours for Chetan for Tue :1
Enter number of hours for Chetan for Wed :1
Enter number of hours for Chetan for Thur :0
Enter number of hours for Chetan for Fri :0
Enter number of hours for Chetan for Sat :0
Enter number of hours for Aman for Mon :2
Enter number of hours for Aman for Tue :2
Enter number of hours for Aman for Wed :0
Enter number of hours for Aman for Thur :0
Enter number of hours for Aman for Fri :0
Enter number of hours for Aman for Sat :0
Enter number of hours for RAJ for Mon :1
Enter number of hours for RAJ for Tue :1
Enter number of hours for RAJ for Wed :1
Enter number of hours for RAJ for Thur :1
Enter number of hours for RAJ for Fri :1
Enter number of hours for RAJ for Sat :1
Enter number of hours for Ruchika for Mon :2
Enter number of hours for Ruchika for Tue :2
master* Python 3.7.3 64-bit 0 0
Ln 147, Col 14 Spaces: 4 UTF-8 LF Python


Activities
Visual Studio Code
Apr 11 9:19 PM
45 %
timetable.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL
1: bash
Enter number of hours for Ruchika for Tue :2
Enter number of hours for Ruchika for Wed :2
Enter number of hours for Ruchika for Thur :0
Enter number of hours for Ruchika for Fri :0
Enter number of hours for Ruchika for Sat :0
Enter number of hours for Roopam for Mon :1
Enter number of hours for Roopam for Tue :2
Enter number of hours for Roopam for Wed :2
Enter number of hours for Roopam for Thur :2
Enter number of hours for Roopam for Fri :0
Enter number of hours for Roopam for Sat :0
Enter number of hours for Ahmed for Mon :0
Enter number of hours for Ahmed for Tue :0
Enter number of hours for Ahmed for Wed :2
Enter number of hours for Ahmed for Thur :2
Enter number of hours for Ahmed for Fri :0
Enter number of hours for Ahmed for Sat :0
Enter number of hours for Harsh for Mon :0
Enter number of hours for Harsh for Tue :0
Enter number of hours for Harsh for Wed :0
Enter number of hours for Harsh for Thur :3
Enter number of hours for Harsh for Fri :3
Enter number of hours for Harsh for Sat :0
Enter number of hours for Neha for Mon :0
Enter number of hours for Neha for Tue :0
Enter number of hours for Neha for Wed :0
Enter number of hours for Neha for Thur :0
Enter number of hours for Neha for Fri :0
Enter number of hours for Neha for Sat :5
+-------+---------+---------+---------+--------+-------+------+
| | Mon | Tue | Wed | Thur | Fri | Sat |
master* Python 3.7.3 64-bit 0 0
Ln 147, Col 14 Spaces: 4 UTF-8 LF Python



PASS

Future Scope:
Its just a blue print logic of Our group
If we Try to work more on it then it will help the Universities and Schools to manage their time table for faculties in just a minute.

Contribution:

Algorithm Design and Implementation of Code:Chetan Kumar Sharma

Creative Idea and Reduce Work Load On Design Part by Pretty Table:Santosh Mittal

Analysis of test Cases :Sandesh Thapa Magar
Error Detection:Shubam Kanwar

Conclusion:Rule based Approach is same as Human Expertise System which helps a lot to solve human environment problems like :Faculty time table .

The program is well understanding the logics of humans.If we want to modify this program we just add same rules which match with human behaviour.