AI Assignment 3

Arjun Temura
2020497
CSB 2020

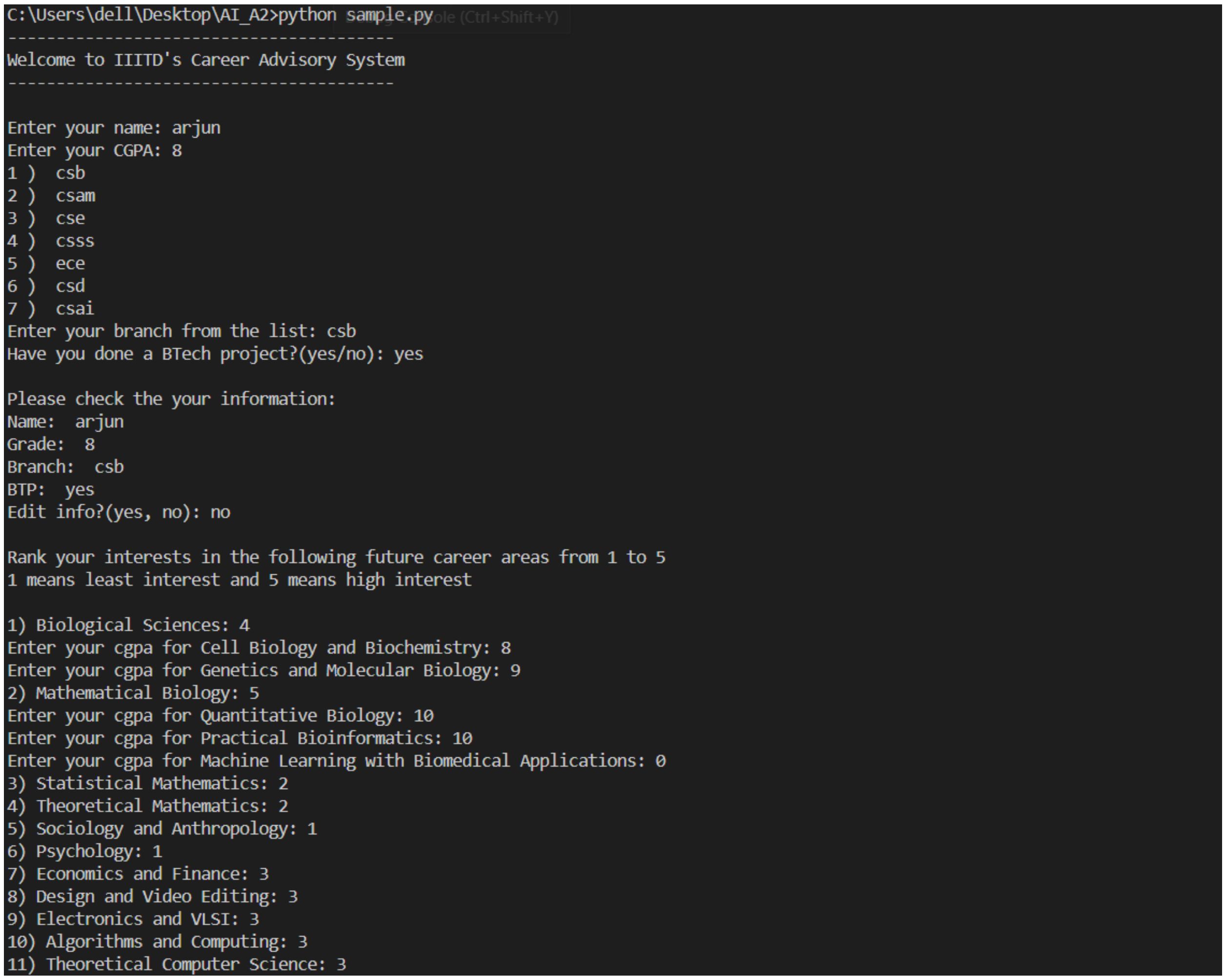
Problem Statement:
Write a python program using durable-rules module with forward-chaining rules for career advisory system for a graduating student of IIITD based on courses done, grades got and interest areas. Make your own rules and test it out with facts.

Workflow of code:

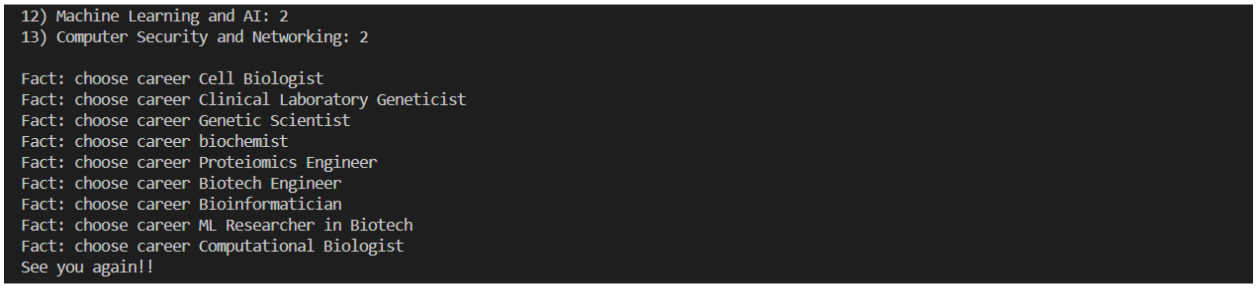
1. The **start()** presents an interactive to query the user for their name, cgpa, branch and BTP status.
2. This data is passed to **verify_data()** which asks the user for any edit required.
3. This data is passed to **interest_areas()** which asks the user about their interest in various predefined areas(stored as list_interest).
4. If a user is highly interested in an area, a fact is asserted to the **'interests' ruleset** in durable-rules module
5. User is asked about grade in relevant courses in that area using **get_marks()**
6. Using Forward Chaining rules based on **BTP status, branch and course grade**, facts are asserted to the **'courses' ruleset**.
7. This ruleset asserts facts about the ideal career path for the user which is stored in a list (fact_base).
8. The output is printed to the user.

Output Snippets:

1.

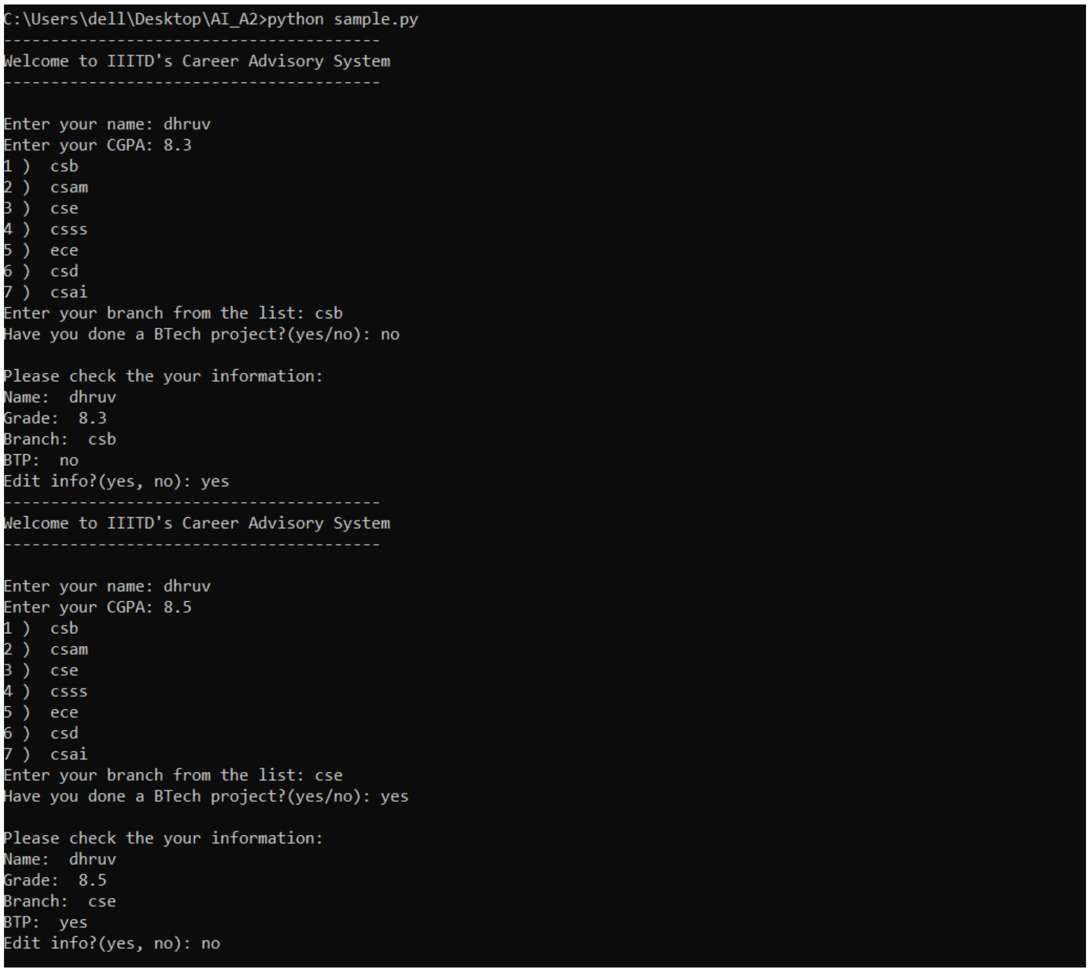


```
C:\Users\dell\Desktop\AI_A2>python sample.py
---------------------------------------
Welcome to IIITD's Career Advisory System
---------------------------------------

Enter your name: arjun
Enter your CGPA: 8
1 )  csb
2 )  csam
3 )  cse
4 )  csss
5 )  ece
6 )  csd
7 )  csai
Enter your branch from the list: csb
Have you done a BTech project?(yes/no): yes

Please check the your information:
Name:  arjun
Grade:  8
Branch:  csb
BTP:  yes
Edit info?(yes, no): no

Rank your interests in the following future career areas from 1 to 5
1 means least interest and 5 means high interest

1) Biological Sciences: 4
Enter your cgpa for Cell Biology and Biochemistry: 8
Enter your cgpa for Genetics and Molecular Biology: 9
2) Mathematical Biology: 5
Enter your cgpa for Quantitative Biology: 10
Enter your cgpa for Practical Bioinformatics: 10
Enter your cgpa for Machine Learning with Biomedical Applications: 0
3) Statistical Mathematics: 2
4) Theoretical Mathematics: 2
5) Sociology and Anthropology: 1
6) Psychology: 1
7) Economics and Finance: 3
8) Design and Video Editing: 3
9) Electronics and VLSI: 3
10) Algorithms and Computing: 3
11) Theoretical Computer Science: 3
```

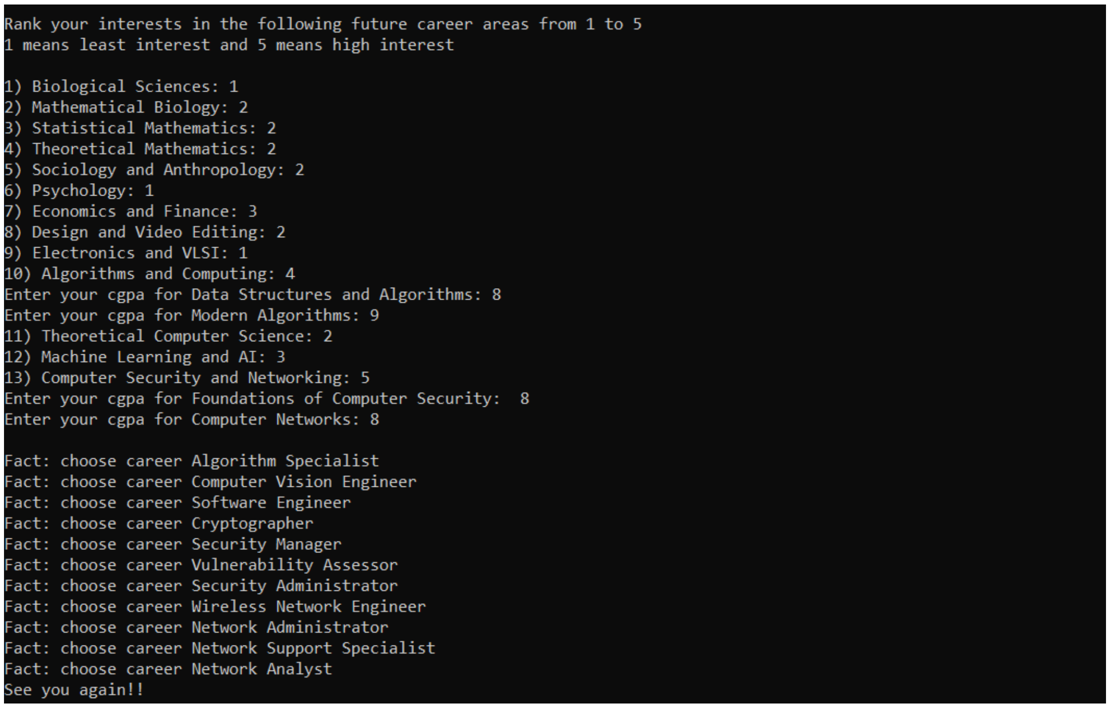


```
12) Machine Learning and AI: 2
13) Computer Security and Networking: 2

Fact: choose career Cell Biologist
Fact: choose career Clinical Laboratory Geneticist
Fact: choose career Genetic Scientist
Fact: choose career biochemist
Fact: choose career Proteiomics Engineer
Fact: choose career Biotech Engineer
Fact: choose career Bioinformatician
Fact: choose career ML Researcher in Biotech
Fact: choose career Computational Biologist
See you again!!
```

2.



```
C:\Users\dell\Desktop\AI_A2>python sample.py
----------------------------------------
Welcome to IIITD's Career Advisory System
----------------------------------------

Enter your name: dhruv
Enter your CGPA: 8.3
1 )  csb
2 )  csam
3 )  cse
4 )  csss
5 )  ece
6 )  csd
7 )  csai
Enter your branch from the list: csb
Have you done a BTech project?(yes/no): no

Please check the your information:
Name:  dhruv
Grade:  8.3
Branch:  csb
BTP:  no
Edit info?(yes, no): yes
----------------------------------------
Welcome to IIITD's Career Advisory System
----------------------------------------

Enter your name: dhruv
Enter your CGPA: 8.5
1 )  csb
2 )  csam
3 )  cse
4 )  csss
5 )  ece
6 )  csd
7 )  csai
Enter your branch from the list: cse
Have you done a BTech project?(yes/no): yes

Please check the your information:
Name:  dhruv
Grade:  8.5
Branch:  cse
BTP:  yes
Edit info?(yes, no): no
```



```
Rank your interests in the following future career areas from 1 to 5
1 means least interest and 5 means high interest

1) Biological Sciences: 1
2) Mathematical Biology: 2
3) Statistical Mathematics: 2
4) Theoretical Mathematics: 2
5) Sociology and Anthropology: 2
6) Psychology: 1
7) Economics and Finance: 3
8) Design and Video Editing: 2
9) Electronics and VLSI: 1
10) Algorithms and Computing: 4
Enter your cgpa for Data Structures and Algorithms: 8
Enter your cgpa for Modern Algorithms: 9
11) Theoretical Computer Science: 2
12) Machine Learning and AI: 3
13) Computer Security and Networking: 5
Enter your cgpa for Foundations of Computer Security:  8
Enter your cgpa for Computer Networks: 8

Fact: choose career Algorithm Specialist
Fact: choose career Computer Vision Engineer
Fact: choose career Software Engineer
Fact: choose career Cryptographer
Fact: choose career Security Manager
Fact: choose career Vulnerability Assessor
Fact: choose career Security Administrator
Fact: choose career Wireless Network Engineer
Fact: choose career Network Administrator
Fact: choose career Network Support Specialist
Fact: choose career Network Analyst
See you again!!
```

Source code:

```
from durable.lang import *


fact_base=[] #to store facts formed after Forward chaining

#########################################################
#Function to query marks for a course from the user
#########################################################

def get_marks(s):
    st="Enter your cgpa for "+s
    v=int(input(st))
    return v

########################################
# ruleset with interests
########################################

with ruleset('interests'):
    # will be triggered by 'interests' facts

    @when_all((m.area == 'Biological Sciences') )
    def bsc(c):
        x=get_marks("Cell Biology and Biochemistry: ")
        y=get_marks("Genetics and Molecular Biology: ")
        #assert a general fact based on area of interest
        c.assert_fact({'subject':'choose','predicate':'career', 'object':'Biochemist'})
        if(x>7 and c.m.branch=='csb'):
           #assert fact based on marks(> 7 cgpa) in a course, project in the area and branch
of graduating student
          c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(y>7):
            #assert fact based on marks(>7 cgpa) an project
          c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})


    @when_all(m.area == 'Mathematical Biology')
    def mbio(c):
        x=get_marks("Quantitative Biology: ")
        y=get_marks("Practical Bioinformatics: ")
        z=get_marks("Machine Learning with Biomedical Applications: ")
        if(x>6 and c.m.branch=='csb'):
            c.assert_fact('course_marks',{'field':c.m.area,'courseno':'0',
'did_project':c.m.did_project})
        if(y>6):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})
        c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'2',
'did_project':c.m.did_project})


    @when_all(m.area == 'Statistical Mathematics')
    def sm(c):
        x=get_marks("Linear Algebra: ")
        y=get_marks("Probability and Statistics: ")
        c.assert_fact({'subject':'choose','predicate':'career', 'object':'Statistician'})
        if(x>7):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(y>7):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})

    @when_all(m.area == 'Theoretical Mathematics')
    def tm(c):
        x=get_marks("Graph Theory: ")
        y=get_marks("Real Analysis: ")
        if(x>7 & c.m.branch=='csam'):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})

        c.assert_fact('course_marks', { 'field': c.m.area , 'courseno':'1',
'did_project':c.m.did_project})

    @when_all(m.area == 'Sociology and Anthropology')
    def sa(c):
        x=get_marks("Science, Technology and Society: ")
        y=get_marks("Information Technology and Society: ")
        if(x>7):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(y>7 and c.m.branch=='csss'):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})

    @when_all(m.area == 'Psychology')
    def psy(c):
        x=get_marks("Cognitive Motor Behaviour: ")
        y=get_marks("Neuroscience of Decision Making: ")
        c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})

    @when_all(m.area == 'Economics and Finance')
    def ef(c):
        x=get_marks("Money and Banking: ")
        y=get_marks("Econometrics: ")
        if((x>6) and (y>6) and c.m.branch==('csss' or 'cse' or 'csb'))  :
         c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})

    @when_all(m.area == 'Design and Video Editing')
    def dve(c):
        x=get_marks("Introduction to Human Computer Interaction: ")
        y=get_marks("Design of Interactive Systems: ")
        z=get_marks("Prototyping Interactive Systems: ")
        if(x>6 and y>6):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(x>6 and z>6 and c.m.branch=='csd'):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})

    @when_all(m.area == 'Electronics and VLSI')
    def ele(c):
        x=get_marks("Basic Electronics: ")
        y=get_marks("Signals and Systems: ")
        z=get_marks("VLSI and Embedded Systems: ")
        if(x>7 and y>7 and c.m.branch=='ece'):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(x>7 and z>7 and c.m.branch=='ece'):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})


    @when_all(m.area == 'Algorithms and Computing')
    def acomp(c):
        x=get_marks("Data Structures and Algorithms: ")
        y=get_marks("Modern Algorithms: ")
        if(x>7 and y>7 and c.m.branch==('cse' or 'csam' or 'csai')):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})


    @when_all(m.area == 'Theoretical Computer Science' )
    def tcs(c):
        x=get_marks("Operating Systems: ")
        y=get_marks("Theory of Computation: ")
        c.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Software Engineer'
})
        if(x>7):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(y>7):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})

    @when_all(m.area == 'Machine Learning and AI' )
    def mlai(c):
        x=get_marks("Machine Learning: ")
        y=get_marks("Artificial Intelligence: ")
        z=get_marks("Data Mining: ")
        c.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'ML Engineer' })

        if(x>6 and y>6):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(z>6):
            c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})
```

```
    @when_all((m.area == 'Computer Security and Networking'))
    def csn(c):
        x=get_marks("Foundations of Computer Security:  ")
        y=get_marks("Computer Networks: ")
        if(x>6 and c.m.branch=='cse'):
            #threshold marks (6 cgpa in Foundations of Comp. Security)
             c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'0',
'did_project':c.m.did_project})
        if(y>6 and (c.m.branch=='cse' or c.m.branch=='csam') ):
            #threshold marks (6 cgpa in Computer Networks)
             c.assert_fact('course_marks', { 'field': c.m.area, 'courseno':'1',
'did_project':c.m.did_project})

    @when_all(+m.subject)
    def output(c):
        fact_base.append('Fact: {0} {1} {2}'.format(c.m.subject, c.m.predicate, c.m.object))

#######################################
# ruleset with course marks
#######################################

with ruleset('course_marks'):

    #assert career facts based on project done, marks threshold, interests using Forward
Chaining Rules
    @when_all((m.field == 'Biological Sciences') &(m.did_project=="yes") & (m.courseno=='0'))
    def bs1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Cell Biologist' })

    @when_all((m.field == 'Biological Sciences') &(m.did_project=="yes") & (m.courseno=='1') )
    def bs2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Genetic Scientist'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Clinical
Laboratory Geneticist' })

    @when_all((m.field == 'Mathematical Biology') &(m.courseno=='0') )
    def mb1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Biotech Engineer'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Proteiomics
Engineer' })

    @when_all((m.field == 'Mathematical Biology') &(m.courseno=='1') )
    def mb2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Bioinformatician'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Proteiomics
Engineer' })

    @when_all((m.field == 'Mathematical Biology') &(m.did_project=="yes") & (m.courseno=='2'))
    def mb3(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Computational
Biologist' })
```

```
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'ML Researcher in Biotech' })

    @when_all((m.field == 'Statistical Mathematics') & (m.courseno=='0'))
    def sm1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Computational Scientist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Research Analyst' })

    @when_all((m.field == 'Statistical Mathematics') & (m.courseno=='1'))
    def sm2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Database Administrator' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Operations Research Analyst' })

    @when_all((m.field == 'Theoretical Mathematics') & (m.courseno=='0'), (m.field == 'Theoretical Mathematics') & (m.did_project=='yes'))
    def tm1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Research Mathematician' })

    @when_all((m.field == 'Theoretical Mathematics') & (m.courseno=='1'))
    def tm2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Computational Statistician' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Quantitative Researcher' })

    @when_all((m.field == 'Sociology and Anthropology') & (m.courseno=='0'))
    def sa1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Sociologist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Public Policy Researcher' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Human Resource Manager' })


    @when_all((m.field == 'Sociology and Anthropology') & (m.courseno=='1'))
    def sa2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Paleontologist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Archaeologist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Ethnologist' })

    @when_all((m.field == 'Psychology') & (m.did_project=='yes'))
    def psy(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Psychiatrists' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Neuropsychologist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Forensic Psychologist' })

    @when_all((m.field == 'Economics and Finance') & (m.did_project=='yes'))
    def eco(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Financial Risk
Analyst' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Economist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Financial Planner'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Investment
Analyst' })

    @when_all((m.field == 'Design and Video Editing') & (m.courseno=='0'))
    def dve1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Video Game
Designer' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'UI/UX Engineer' })

    @when_all((m.field == 'Design and Video Editing') & (m.courseno=='1') &
(m.did_project=='yes'))
    def dve2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'AR/VR Designer' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Design Researcher'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Web Developer' })

    @when_all((m.field == 'Electronics and VLSI') & (m.courseno=='0'))
    def ev1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Telecom Engineer'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Hardware Analyst'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Service Engineer'
})

    @when_all((m.field == 'Electronics and VLSI') & (m.courseno=='1') )
    def ev2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'VLSI Engineer' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Design
Verification Engineer' })

    @when_all((m.field == 'Algorithms and Computing') & (m.courseno=='0') &
(m.did_project=='yes') )
    def ac1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Software Engineer'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Computer Vision
Engineer' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Algorithm
Specialist' })

    @when_all((m.field == 'Theoretical Computer Science') & (m.courseno=='0') &
(m.did_project=='yes'))
    def tcs1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'IT Operations
Manager' })

    @when_all((m.field == 'Theoretical Computer Science') & (m.courseno=='1') &
(m.did_project=='yes'))
```

```
    def tcs2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Quantum
Computation Engineer' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Cryptography
Specialist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Computer Engineer'
})

    @when_all((m.field == 'Machine Learning and AI') & (m.courseno=='0') &
(m.did_project=='yes'))
    def ml1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Data Scientist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'AI Specialist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'NLP Scientist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'ML Cloud
Architect' })

    @when_all((m.field == 'Machine Learning and AI') & (m.courseno=='1') &
(m.did_project=='yes'))
    def ml2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Data Analyst' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Database Engineer'
})

    @when_all((m.field == 'Computer Security and Networking') & (m.courseno=='0') &
(m.did_project=='yes'))
    def csn1(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Security
Administrator' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Vulnerability
Assessor' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Security Manager'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Cryptographer' })

    @when_all((m.field == 'Computer Security and Networking') & (m.courseno=='1') )
    def csn2(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Network Analyst'
})
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Network Support
Specialist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Network
Administrator' })

    @when_all((m.field == 'Computer Security and Networking') & (m.courseno=='1') &
(m.did_project=='yes'))
    def csn3(d):
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Wireless Network
Engineer' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Network Support
Specialist' })
        d.assert_fact({ 'subject': 'choose', 'predicate':'career','object':'Telecommunication
Specialist' })

    #store the facts obtained in a fact base(list)
    @when_all(+m.subject)
    def output(d):
         fact_base.append('Fact: {0} {1} {2}'.format(d.m.subject, d.m.predicate, d.m.object))


##################################################################
#to input the interests from the user in a scale of 1 to 5
##################################################################

def interests_areas(interest_list, grade, branch , did_project):

     print("\nRank your interests in the following future career areas from 1 to 5")
     print("1 means least interest and 5 means high interest")
     print()
     for i in range(len(interest_list)):
         s=""
         st=s+str(i+1)+") "+interest_list[i]+": "
         val=int(input(st))
         if(val>=4): #if user is highly interested in an area, assert fact for that area to be
processed by the rulesets
            assert_fact('interests',{'area':interest_list[i], 'grade':grade, 'branch':branch,
'did_project':did_project})

############################################
#to verify the data input by the user
############################################

def verify_data(name, grade, branch, did_project,interest_list):
    print()
    print("Please check the your information: ")
    print("Name: ", name)
    print("Grade: ", grade)
    print("Branch: ", branch)
    print("BTP: ",did_project)
    status=input("Edit info?(yes, no): ")
    if(status=="yes"):
        start()  #if error exists, user re enters the data
    else:
        interests_areas(interest_list, grade, branch, did_project)    #else interest_areas()
is called to take in interests

###############################
#beginning of program
###############################

def start():
 #list of various areas of interest
 interest_list=["Biological Sciences","Mathematical Biology","Statistical Mathematics",
 "Theoretical Mathematics", "Sociology and Anthropology", "Psychology", "Economics and
Finance",
 "Design and Video Editing", 'Electronics and VLSI',"Algorithms and Computing", "Theoretical
Computer Science", "Machine Learning and AI",
 "Computer Security and Networking"]

 #branch data for the users
 branch_list=['csb', 'csam','cse','csss','ece','csd','csai']
 print("-"*40)
 print("Welcome to IIITD's Career Advisory System")
 print("-"*40)
 print()
 name=input("Enter your name: ")
 grade=float(input("Enter your CGPA: "))

 #prints the list of branches to the user
 for i in range(len(branch_list)):
    print(i+1, ") ",branch_list[i])

 branch=input("Enter your branch from the list: ")
 did_project=input("Have you done a BTech project?(yes/no): ")

 #send data for verification
 verify_data(name,grade, branch, did_project,interest_list)
 print()

#start here
start()

#after processing data, print the list of facts concluded by forward chaining to the user
for i in fact_base:
    print(i)
print("See you again!!")
```