LAB REPORT COVER PAGE

Student ID:	**23081055**
Student Name:	**Sudip Bhomjan**
Course Name:	**Artificial Intelligence**
Course Code:	**CSC261**
Semester:	**4th**
Instructor/Examiner/Lecture:	**Mr. Saroj Maharjan**
Date of submission:	**28/08/2024**
Evaluator's Comment:	**……**
Evaluator's Signature:	**……**

Write a Rule Base System in Python for the following rule systems:

1. Weather Forecasting

Rule 1: If sky is cloudy and there is no wind, then it might rain.

Rule 2: If temperature is below 0°C and the sky is clear, then it might snow.

Rule 3:If temperature is above 30°C and there is no wind, then it might be a hot day.

Rule 4: If sky is clear and there is wind, then it might be a pleasant day.

<u>Objectives:</u> To design and implement rule-based systems that predict outcomes and automate decisions based on specific input conditions.

#Python Code

```
def weather(forecasts):
    if "cloudy" in forecasts and "no wind" in forecasts:
        return "It might rain."

    if forecasts < 0 in forecasts and "sky is clear" in forecasts:
        return "It might snow."

    if forecasts > 30  in forecasts and "no wind" in forecasts:
        return "It might be a hot day."

    if "sky is clear" in forecasts and "is wind" in forecasts:
        return "It might be a pleasant day."

    return "Not clear criteria"

def main():
    forecasts =[]
    print("Forecasts:")
    print("cloudy")
    print("no wind")
    print("temperature below 0")
    print("sky is clear")
```

```
    print("temperature above 30")
    print("is wind \n")
    print("Enter your forecast: (Type 'done' if finished)")
    while True:
        forecast =input("Forecast: ")
        if forecast == "done":
            break
        forecasts.append(forecast)

    print(weather(forecasts))
main()
```

Output:

```
PS C:\Users\sudip> & C:/Users/sudip/AppData/Loca
e/weather_forecasting.py"
Available Forecasts:
cloudy
no wind
temperature below 0
sky is clear
temperature above 30
is wind

Enter your forecast: (Type 'done' if finished)
Forecast: cloudy
Forecast: no wind
Forecast: done
It might rain.
PS C:\Users\sudip>
```

2. Eligibility for a Loan

Rule 1:If applicant's age is between 18 and 65 and they have a stable income, then they are eligible for a loan.

Rule 2:If applicant has a credit score above 700, then they are eligible for a loan.

Rule 3:If applicant has a criminal record, then they are not eligible for a loan.

Rule 4:If applicant has defaulted on a loan before, then they are not eligible for a loan.

#Python Code

```
#Loan Eligibility
def eligibility(detail,age):
```

```
        if "Criminal record" in detail:
            return "Not eligible for a loan."
        if "Loan before" in detail:
           return "Not eligible for a loan."
        if  "Credit Score above 700" in detail:
            return "Eligible for Loan"
        if age >= 18 and age <= 65 and "Stable income" in detail:
            return "Eligible for Loan"
        return "Not clear criteria"

def main():
    details =[]
    age = None
    print(" Eligibility for Loan:")
    print("Age between 18 & 65")
    print("Stable Income")
    print("Credit Score above 700")
    print("Criminal Record")
    print("Loan Before")
    print("Enter applicant's details (type 'done' when finished):")
    while True:
        detail = input("Details: ")
        if detail == "done":
            break
        try:
            temp_value = int(detail)
            age = temp_value
        except ValueError:
            details.append(detail)
    print(eligibility(details,age))
main()
```

Output:

```
PS C:\Users\sudip> & C:/Users/sudip/AppData/Local/Micro
AB/.vscode/Loan_Eligibility.py"
 Eligibility for Loan:
Age between 18 & 65
Stable income
Credit Score above 700
Criminal Record
Loan Before
Enter applicant's details (type 'done' when finished):
Details: 30
Details: Stable income
Details: done
Eligible for Loan
PS C:\Users\sudip>
```

3. Simple Decision Making

Rule 1: If the time is between 6 AM and 8 AM and it's a weekday, then it's time to go to work.

Rule 2: If the time is between 12 PM and 1 PM, then it's time for lunch.

Rule 3: If the time is between 9 PM and 10 PM, then it's time to go to bed.

Rule 4: If it's the weekend and the weather is sunny, then go for a walk.

#Python Code

```
from datetime import datetime

def make_decision(time, day_of_week, weather):
    current_time = datetime.strptime(time, "%H:%M").time()
        if current_time >= datetime.strptime("06:00", "%H:%M").time() and current_time <= datetime.strptime("08:00", "%H:%M").time() and day_of_week in ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]:
        return "Time to go to work."
        if current_time >= datetime.strptime("12:00", "%H:%M").time() and current_time <= datetime.strptime("13:00", "%H:%M").time():
        return "Time for lunch."
        if current_time >= datetime.strptime("21:00", "%H:%M").time() and current_time <= datetime.strptime("22:00", "%H:%M").time():
        return "Time to go to bed."
    if day_of_week in ["Saturday", "Sunday"] and weather == "sunny":
```

```
        return "Go for a walk."
    return "No specific action at this time."

def main():
    time = input("Enter the current time (HH:MM): ")
    day_of_week = input("Enter the day of the week: ").capitalize()
    weather = input("Enter the weather (sunny/cloudy/rainy/etc.): ").lower()
    decision = make_decision(time, day_of_week, weather)
    print(decision)
main()
```

Output:

```
TERMINAL
PS C:\Users\sudip> & C:/Users/sudip/AppData/Local/Mic
g.py
Enter the current time (HH:MM): 8:00
Enter the day of the week: sunday
Enter the weather (sunny/cloudy/rainy/etc.): sunny
Go for a walk.
PS C:\Users\sudip>
```

4. Traffic Light Control

Rule 1: If the light is red, then cars must stop.

Rule 2: If the light is green, then cars can go.

Rule 3: If the light is yellow, then cars must slow down and prepare to stop.

Rule 4: If the pedestrian button is pressed, then the light will turn red after a short delay.

#Python Code

```
#Traffic Light Control
def lightControl(light,pedestrianbutton):
        if "red" in light :
            return "Cars must stop."
        if "green" in light :
            return "Cars must go."
        if "yellow" in light :
```

```
        return "Cars must slow down and prepare to stop."
    if  "Yes" in pedestrianbutton :
        return "The light will turn red after a short delay."
    return "Not clear criteria"


def main():
    print("Traffic Light Control")
    print("Enter the color of  traffic light and  is the pedestrian button pressed? :")
    lights = input("Light: ")
    button = input("Pedestrain button pressed or not ? (Yes or No:)")
    print(lightControl(lights,button))
main()
```

Output:

```
TERMINAL
PS C:\Users\sudip> & C:/Users/sudip/AppData/Local/Microsoft/WindowsApps/pyt
ol.py"
Traffic Light Control
Enter the color of  traffic light and  is the pedestrian button pressed? :
Light: red
Pedestrain button pressed or not ? (Yes or No:)yes
Cars must stop.
PS C:\Users\sudip>
```

5. Smart Home Automation

Rule 1: If the temperature is below 18°C, then turn on the heater.

Rule 2: If the temperature is above 25°C, then turn on the air conditioner.

Rule 3: If it is dark outside and someone is at home, then turn on the lights.

Rule 4: If the security system is armed and a door is opened, then sound the alarm.

#Python Code:

```
#Smart Home Automation
def smart(automate,temperature):

    if "Dark Outside" in automate and "Someone is at home" in automate:
        return "Turn on the lights."

    if "Security System is armed" in automate and  "Door Opened" in automate:
        return "Sound the alarm."

    if  temperature is not None and  temperature < 18 :
```

```
        return "Turn on the heater."

    if  temperature is not None and  temperature > 25 :
        return "Turn on the air conditioner."



    return "Not clear criteria"

def main():

    automater =[]
    temperature = None


    print("Enter Enviroment details (type 'done' when finished):")
    while True:
        automate = input("Temperature or Environment Details: ")
        if automate == "done":
            break

        try:
            temp_value = int(automate)
            temperature = temp_value
        except ValueError:
            automater.append(automate)


    print(smart(automater,temperature))
main()
```

Output:

```
TERMINAL
PS C:\Users\sudip> & C:/Users/sudip/AppData/Local/Micro
.py
Enter Enviroment details (type 'done' when finished):
Temperature or Environment Details: Dark Outside
Temperature or Environment Details: Someone is at home
Temperature or Environment Details: done
Turn on the lights.
PS C:\Users\sudip>
```

Conclusion:

In this lab, various rule-based systems were successfully designed and implemented in Python. By applying logical conditions, the systems accurately predicted outcomes and automated decision-making processes for scenarios such as weather forecasting, traffic light control, and time control. This demonstrated the effectiveness of rule-based systems in handling structured decision-making tasks.