AI Ethics and Applications
Lab - 8

Utkarsh Bhangale

20200802124

Track: Data Science

AIM:

The aim of this practical lab is to teach participants how to use the SymPy library in Python for symbolic and boolean logic manipulation. It provides hands-on experience in defining symbolic variables, creating logic expressions, and solving for unknown values. This knowledge can be applied to various fields like mathematics, engineering, computer science, and AI.

Software Required: Google Colab , install python sympy library

Prerequisite:

a. Problem Solving , Understanding python SymPy

Theory/Concept:

Symbolic and boolean logic manipulation in Python, facilitated by the SymPy library, empowers programmers to work with abstract and logical expressions, allowing them to define, compare, and solve complex problems with ease. This concept is built on the foundation of key principles:

1. **Installation and Import:** The journey begins with the installation of SymPy using the 'pip' package manager and the subsequent import of the library within the Python environment. SymPy provides a powerful toolkit for symbolic mathematics, making it a valuable asset for logic programming.

2. **Defining Symbolic Variables:** In this process, symbolic variables are created using the `sp.symbols` function, such as 'x' and 'y' in the given example. These variables serve as placeholders for unknown values within logical expressions.

3. **Creating Logic Expressions:** Logic expressions are constructed using symbolic variables and boolean operations, enabling the formation of intricate statements. For example, the expression `(x > 5) & (y < 10)` represents the logical conjunction of 'x > 5' and 'y < 10,' signifying that both conditions must hold true for the entire expression to evaluate as true.

4. **Solving for Unknown Values:** The primary goal is to find values for symbolic variables that satisfy the given logical expression. This is achieved through the 'solve' function provided by SymPy. It systematically calculates the possible values for 'x' and 'y' that make the expression true, generating a list of dictionaries containing these solutions.

5. **Accessing and Printing Solutions:** The final step involves extracting and displaying the solutions. A loop iterates through the list of dictionaries, allowing the programmer to access and print the values that satisfy the logical expression. The result is a set of solutions that fulfill the logical constraints defined in the expression.

Relative Applications:

1. Control Systems : SymPy is used in control theory to analyze and design control systems. Engineers can manipulate transfer functions, Laplace transforms, and block diagrams symbolically. This is essential for tasks like stability analysis and controller design in fields like robotics, aerospace, and automation.

2. Education : SymPy is a valuable tool for teaching and learning mathematics. It enables students to understand and visualize mathematical concepts, providing step-by-step solutions to equations, calculus problems, and algebraic expressions.

3. Data Analysis: While libraries like NumPy and pandas are commonly used for numerical data analysis, SymPy can be employed to perform symbolic data analysis, which is useful in certain specialized fields, such as cryptography and statistical analysis.

Output:

```
[ ] import sympy as sp
```

```
[ ] import sympy as sp

    x = sp.symbols('x')
    y = sp.symbols('y')

    inequality1 = x - 5
    inequality2 = 10 - y

    solution_x = sp.solveset(inequality1, domain=sp.S.Reals)
    solution_y = sp.solveset(inequality2, domain=sp.S.Reals)

    print(f"x={solution_x}, y={solution_y}")
```

```
x={5}, y={10}
```

```
[ ]
    temp=int(input("enter the temperature: "))
    weather = input("enter the weather: ")
    if temp>20:
      if weather=='sunny' or "Sunny" or"SUNNY":
          print("picnic")
    else:
      print("not go outside")
```

```
enter the temperature: 48
enter the weather: sunny
picnic
```

```
temp=int(input("enter the temperature (IN FARHEIENHIET) : "))
WBC = int(input("enter the WBC : "))
if temp>100:
  if WBC>=10000:
      print("Infection")
  else:
    print("fever")
else:
  print("Healthy")
```

```
enter the temperature (IN FARHEIENHIET) : 98
enter the WBC : 50
Healthy
```

Conclusion:

This practical has provided me with a solid foundation in logic programming using the SymPy library in Python. I can now confidently manipulate symbolic and boolean values, create and solve logic expressions, and apply these skills to real-world problems. This knowledge is essential for a wide range of applications, making this practical a valuable learning experience.