Experiment 6 - Implementation of Fuzzy Membership Functions using Python

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	L3: Implement a fuzzy controller system

Aim: To implement Fuzzy Membership Functions using python .

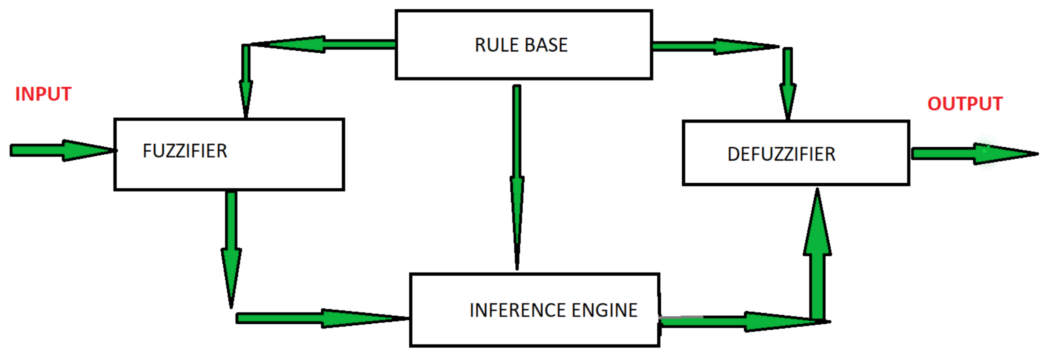
Fuzzy Logic:

The term fuzzy refers to things that are not clear or are vague. In the real world many times we encounter a situation when we can't determine whether the state is true or false, their fuzzy logic provides very valuable flexibility for reasoning. In this way, we can consider the inaccuracies and uncertainties of any situation.

Fuzzy Logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1, instead of just the traditional values of true or false. It is used to deal with imprecise or uncertain information and is a mathematical method for representing vagueness and uncertainty in decision-making.

Fuzzy Logic is based on the idea that in many cases, the concept of true or false is too restrictive, and that there are many shades of gray in between. It allows for partial truths, where a statement can be partially true or false, rather than fully true or false.

Fuzzy Logic is used in a wide range of applications, such as control systems, image processing, natural language processing, medical diagnosis, and artificial intelligence.



FUZZY LOGIC ARCHITECTURE

Membership function

Definition: A graph that defines how each point in the input space is mapped to membership value between 0 and 1. Input space is often referred to as the universe of discourse or universal set (u), which contains all the possible elements of concern in each particular application.
There are largely three types of fuzzifiers:

- Singleton fuzzifier

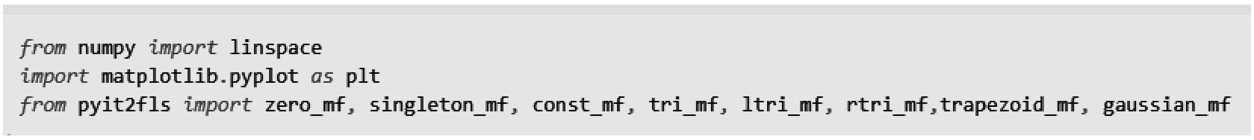
- Gaussian fuzzifier
- Trapezoidal or triangular fuzzifier

Python Library Function Used:

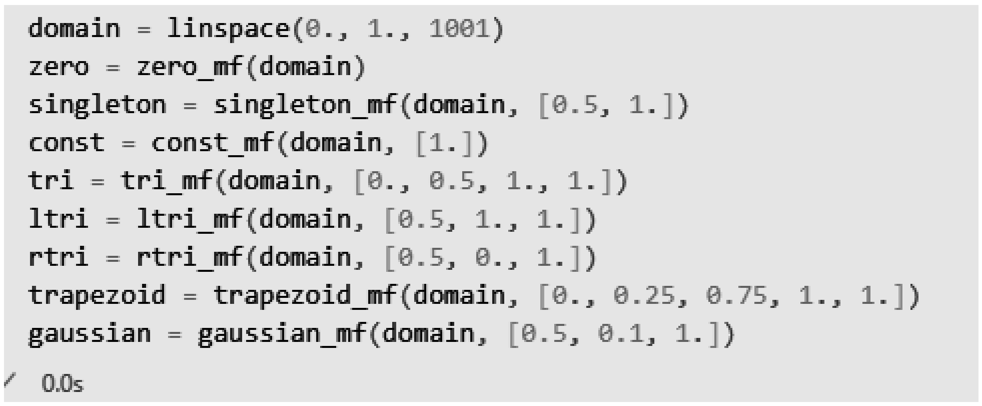
1. Matplotlib-
   Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in various formats. It provides a wide range of functions and tools for creating plots, charts, and graphs, making it a go-to choice for data visualization and scientific plotting. Matplotlib's customizable and publication-quality graphics support various plot types, including line plots, scatter plots, bar charts, histograms, and more. It offers flexibility to control every aspect of the visualization, from axis labels to colors and styles. Matplotlib can be used both as a standalone library and in conjunction with other libraries, such as NumPy and Pandas, to visualize data effectively.

2. NumPy (Numerical Python):
   NumPy is a core library for numerical and scientific computing in Python. It provides essential data structures, such as arrays and matrices, along with a collection of mathematical functions to perform operations on these arrays efficiently. NumPy is the foundation for many other scientific libraries and data analysis tools in Python. It's commonly used for tasks like data manipulation, linear algebra, statistical analysis, and numerical simulations.

Code and Observation:



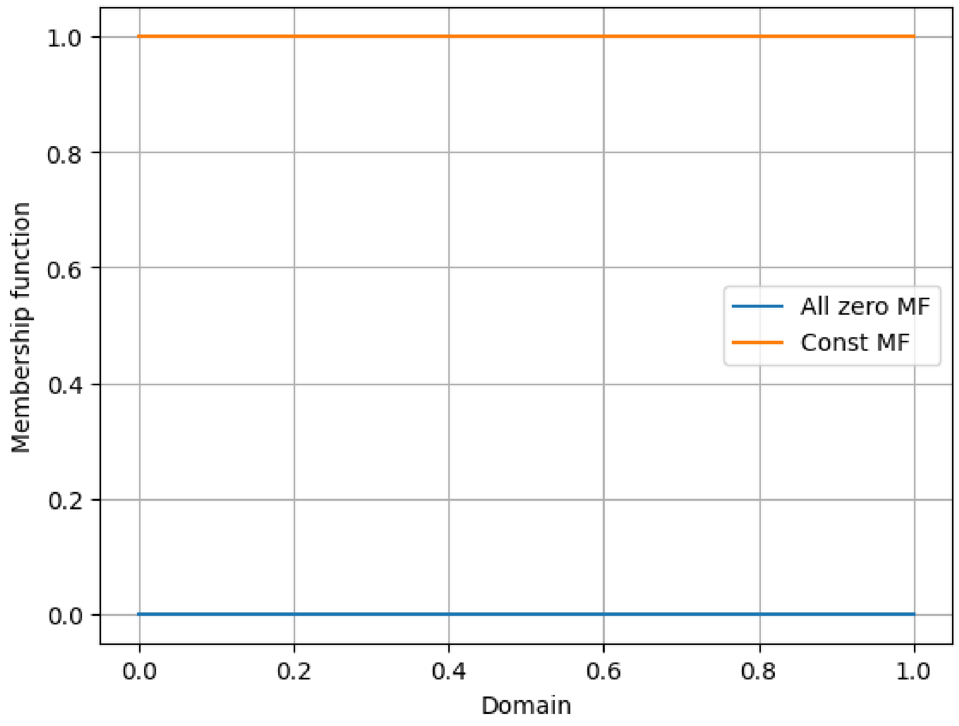
```python
from numpy import linspace
import matplotlib.pyplot as plt
from pyit2fls import zero_mf, singleton_mf, const_mf, tri_mf, ltri_mf, rtri_mf,trapezoid_mf, gaussian_mf
```



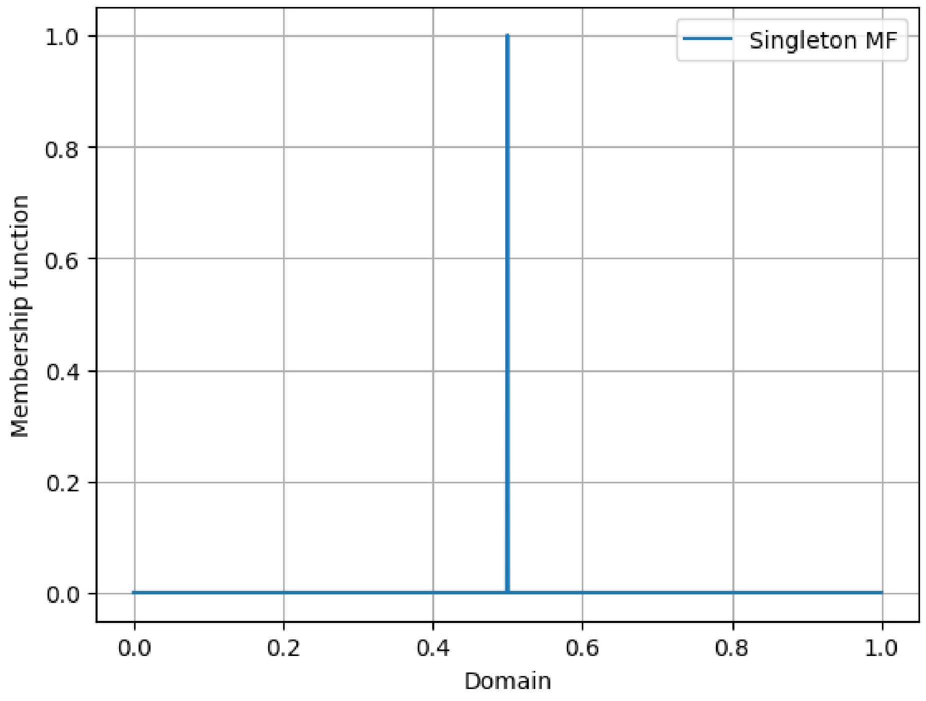
```python
domain = linspace(0., 1., 1001)
zero = zero_mf(domain)
singleton = singleton_mf(domain, [0.5, 1.])
const = const_mf(domain, [1.])
tri = tri_mf(domain, [0., 0.5, 1., 1.])
ltri = ltri_mf(domain, [0.5, 1., 1.])
rtri = rtri_mf(domain, [0.5, 0., 1.])
trapezoid = trapezoid_mf(domain, [0., 0.25, 0.75, 1., 1.])
gaussian = gaussian_mf(domain, [0.5, 0.1, 1.])
```

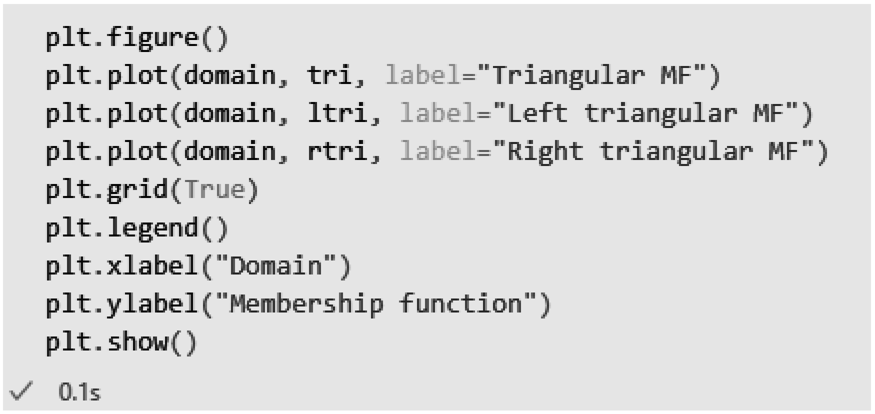
```
✓ 0.0s
```

```
plt.figure()
plt.plot(domain, zero, label="All zero MF")
plt.plot(domain, const, label="Const MF")
plt.grid(True)
plt.legend()
plt.xlabel("Domain")
plt.ylabel("Membership function")
plt.show()
```
✓ 0.1s



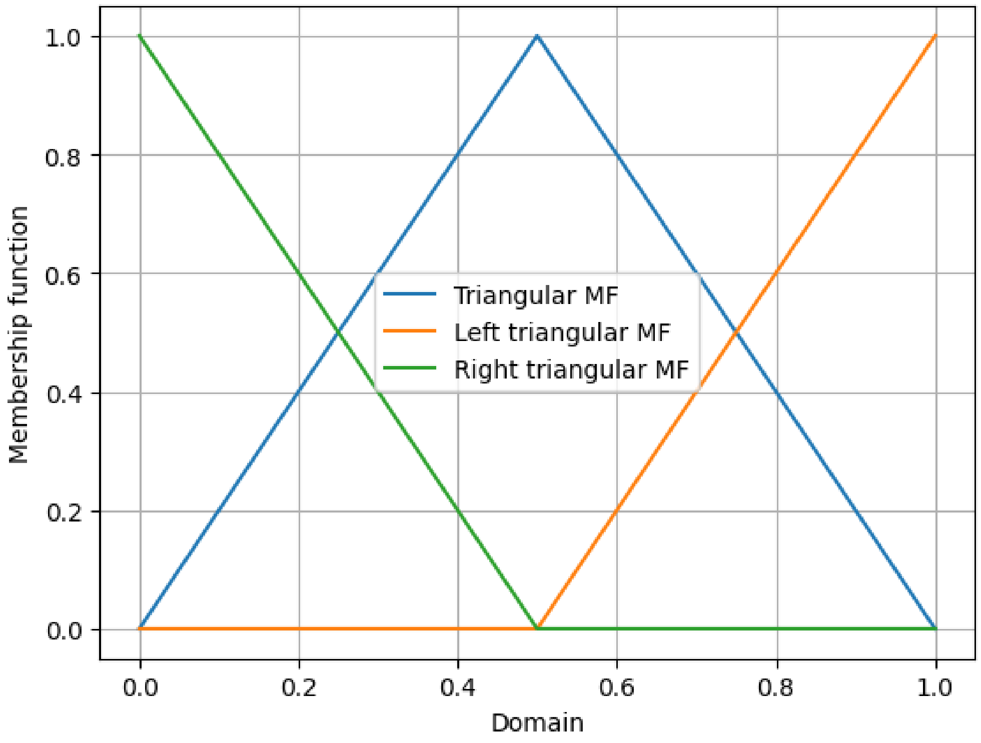
```
plt.figure()
plt.plot(domain, singleton, label="Singleton MF")
plt.grid(True)
plt.legend()
plt.xlabel("Domain")
plt.ylabel("Membership function")
plt.show()
```
✓ 0.1s

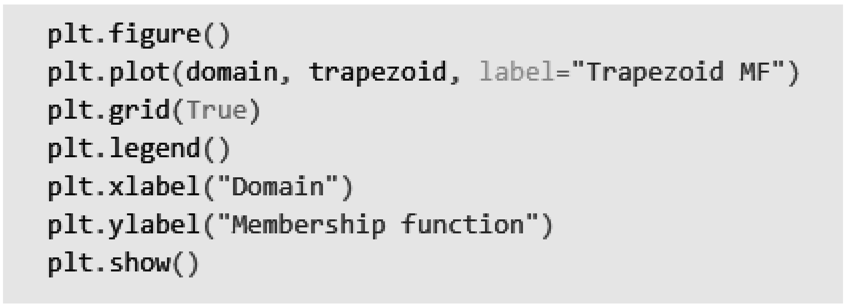




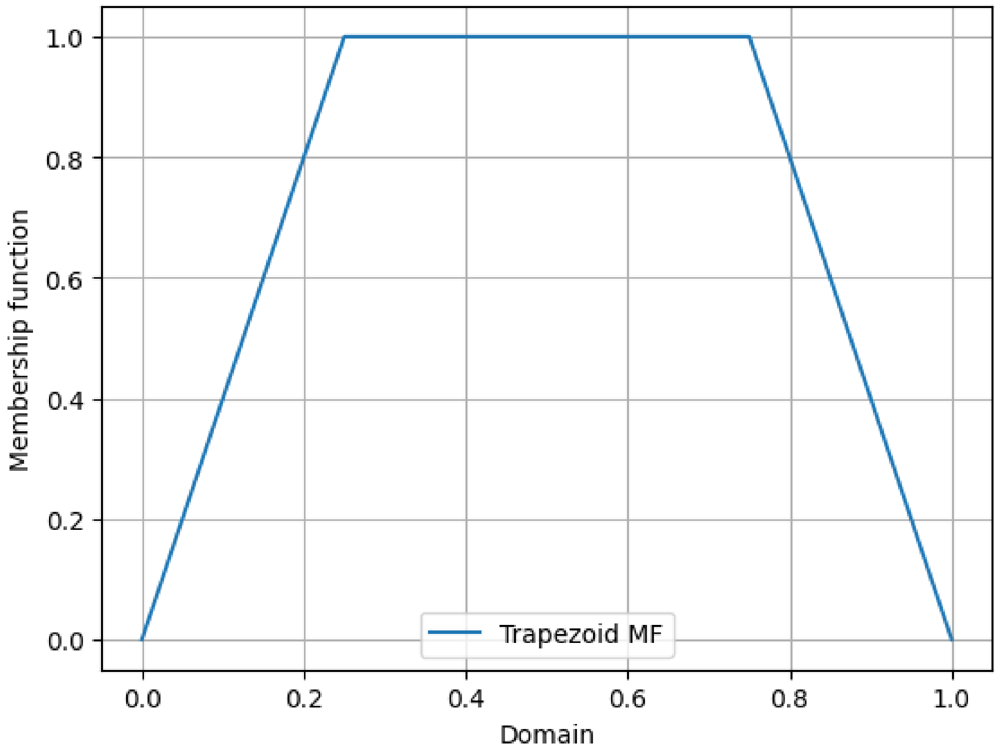
```
plt.figure()
plt.plot(domain, tri, label="Triangular MF")
plt.plot(domain, ltri, label="Left triangular MF")
plt.plot(domain, rtri, label="Right triangular MF")
plt.grid(True)
plt.legend()
plt.xlabel("Domain")
plt.ylabel("Membership function")
plt.show()
```

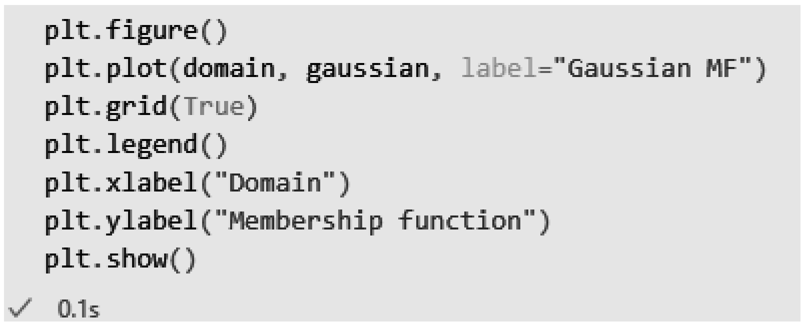
✓ 0.1s





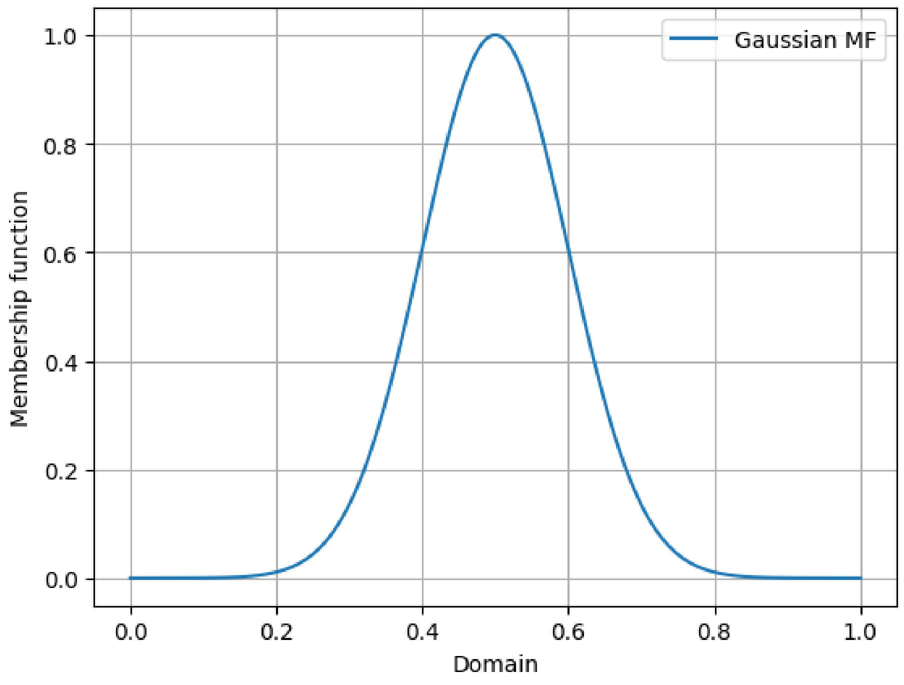
```
plt.figure()
plt.plot(domain, trapezoid, label="Trapezoid MF")
plt.grid(True)
plt.legend()
plt.xlabel("Domain")
plt.ylabel("Membership function")
plt.show()
```





```
plt.figure()
plt.plot(domain, gaussian, label="Gaussian MF")
plt.grid(True)
plt.legend()
plt.xlabel("Domain")
plt.ylabel("Membership function")
plt.show()
```

✓ 0.1s



Conclusion:
Thus we have successfully implemented Fuzzy Membership Functions using Python.