INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| Srno. | Practical Aim | Page No. | Sign |
| 1 | Write a python program to implement ‘UNION OPERATION’ on 2 datasets. | 02 |  |
| 2 | Write a python program to implement ‘INTERSECTION OPERATION’ on 2 datasets. | 03 |  |
| 3 | Write a python program to implement “NOT ‘A’ / COMPLEMENT OF A” on  datasets. | 04 |  |
| 4 | Write a python program to implement “DE-MORGAN’S LAW”. | 05 |  |
| 5 | Write a python program to visualize “FUZZY LOGIC”. | 06 |  |

# Write a python program to implement ‘UNION OPERATION’ on 2 datasets.

A = {"X1": 0.8, "X2": 0.9, "X3": 0.6}

B = {"X1": 0.7, "X2": 0.8, "X3": 0.5} Y = {}

print("This is set A", A) print("This is set B", B)

for A\_key, B\_key in zip(A, B): A\_value = A[A\_key] B\_value = B[B\_key]

if A\_value > B\_value:

Y[A\_key] = A\_value else:

Y[B\_key] = B\_value print(Y)

# OUTPUT

This is set A {'X1': 0.8, 'X2': 0.9, 'X3': 0.6}

This is set B {'X1': 0.7, 'X2': 0.8, 'X3': 0.5}

{'X1': 0.8, 'X2': 0.9, 'X3': 0.6}

# Write a python program to implement ‘INTERSECTION OPERATION’ on 2 datasets.

A = {"X1": 0.8, "X2": 0.9, "X3": 0.6}

B = {"X1": 0.7, "X2": 0.8, "X3": 0.5} Y = {}

print("This is set A", A) print("This is set B", B)

for A\_key, B\_key in zip(A, B): A\_value = A[A\_key] B\_value = B[B\_key]

if A\_value < B\_value:

Y[A\_key] = A\_value else:

Y[B\_key] = B\_value print(Y)

# OUTPUT

This is set A {'X1': 0.8, 'X2': 0.9, 'X3': 0.6}

This is set B {'X1': 0.7, 'X2': 0.8, 'X3': 0.5}

{'X1': 0.7, 'X2': 0.8, 'X3': 0.5}

# Write a python program to implement “NOT ‘A’ / COMPLEMENT OF A” on datasets.

A = {"X1": 0.8, "X2": 0.9, "X3": 0.6}

B = {"X1": 0.7, "X2": 0.8, "X3": 0.5} Y = {}

print("This is set A", A) print("This is set B", B)

for A\_key in A:

Y[A\_key] = 1-A\_value print(Y)

# OUTPUT

This is set A {'X1': 0.8, 'X2': 0.9, 'X3': 0.6}

This is set B {'X1': 0.7, 'X2': 0.8, 'X3': 0.5}

{'X1': 0.4, 'X2': 0.4, 'X3': 0.4}

# Write a python program to implement “DE-MORGAN’S LAW”.

A = [0.9, 0.6, 0.4]

B = [0.7, 0.8, 0.5]

def fuzzy\_not(mu): return 1 - mu

def fuzzy\_AND(mu1, mu2): return min(mu1, mu2)

def fuzzy\_OR(mu1, mu2): return max(mu1, mu2)

def fuzzy\_demorgan(mu\_A, mu\_B):

return fuzzy\_not(fuzzy\_AND(mu\_A, mu\_B)) == fuzzy\_OR(fuzzy\_not(mu\_A), fuzzy\_not(mu\_B))

for mu\_A in A: for mu\_B in B:

print(f"fuzzy\_demorgan({mu\_A}, {mu\_B}) =", fuzzy\_demorgan(mu\_A, mu\_B))

# OUTPUT

fuzzy\_demorgan(0.9, 0.7) = True fuzzy\_demorgan(0.9, 0.8) = True fuzzy\_demorgan(0.9, 0.5) = True fuzzy\_demorgan(0.6, 0.7) = True fuzzy\_demorgan(0.6, 0.8) = True fuzzy\_demorgan(0.6, 0.5) = True fuzzy\_demorgan(0.4, 0.7) = True fuzzy\_demorgan(0.4, 0.8) = True fuzzy\_demorgan(0.4, 0.5) = True

# Write a python program to visualize “FUZZY LOGIC”.

!pip install scikit-fuzzy import numpy as np import skfuzzy as fuzz

from matplotlib import pyplot as plt a=np.arange(11) fa=fuzz.trimf(a,[0,5,10])

print(a) print(fa) plt.plot(a,fa)

# OUTPUT

