FOL – Unification

def unify(x, y, substitutions=None):  
    if substitutions is None:  
        substitutions = {}  
  
    # If both are identical  
    if x == y:  
        return substitutions  
  
    # If x is a variable  
    if isinstance(x, str) and x.islower():  
        return unify\_var(x, y, substitutions)  
  
    # If y is a variable  
    if isinstance(y, str) and y.islower():  
        return unify\_var(y, x, substitutions)  
  
    # If both are compound expressions (like lists or tuples)  
    if isinstance(x, tuple) and isinstance(y, tuple):  
        if x[0] != y[0] or len(x) != len(y):  
            return None  
        for a, b in zip(x[1:], y[1:]):  
            substitutions = unify(a, b, substitutions)  
            if substitutions is None:  
                return None  
        return substitutions  
  
    return None  
  
  
def unify\_var(var, x, substitutions):  
    if var in substitutions:  
        return unify(substitutions[var], x, substitutions)  
    elif x in substitutions:  
        return unify(var, substitutions[x], substitutions)  
    elif occurs\_check(var, x, substitutions):  
        return None  
    else:  
        substitutions[var] = x  
        return substitutions  
  
  
def occurs\_check(var, x, substitutions):  
    if var == x:  
        return True  
    elif isinstance(x, tuple):  
        return any(occurs\_check(var, arg, substitutions) for arg in x[1:])  
    elif isinstance(x, str) and x in substitutions:  
        return occurs\_check(var, substitutions[x], substitutions)  
    return False  
  
  
# Example  
expr1 = ("Eats", "x", "Apple")  
expr2 = ("Eats", "Riya", "y")  
  
result = unify(expr1, expr2)  
print("Unification:", result)  
  
Output:

  
  
FOL Forward Chaining

def forward\_chaining(KB, query):  
    inferred = set()  
    new\_inferred = True  
  
    while new\_inferred:  
        new\_inferred = False  
        for rule in KB:  
            premises, conclusion = rule  
            if all(p in inferred or p in KB for p in premises) and conclusion not in inferred:  
                inferred.add(conclusion)  
                new\_inferred = True  
                if conclusion == query:  
                    return True  
    return False  
  
  
# Example Knowledge Base  
KB = [  
    (["American(Robert)", "Weapon(x)", "Sells(Robert, x, A)", "Hostile(A)"], "Criminal(Robert)"),  
    (["Missile(x)"], "Weapon(x)"),  
    (["Owns(A, x)", "Missile(x)"], "Sells(Robert, x, A)"),  
    (["Enemy(A, America)"], "Hostile(A)")  
]  
  
facts = {  
    "American(Robert)",  
    "Enemy(A, America)",  
    "Owns(A, T1)",  
    "Missile(T1)"  
}  
  
# Add base facts to KB  
for fact in facts:  
    KB.append(([], fact))  
  
# Query  
query = "Criminal(Robert)"  
print("Is Robert a criminal?", forward\_chaining(KB, query))

Output:

