class Term:

def \_\_init\_\_(self, symbol, args=None):

self.symbol = symbol

self.args = args if args else []

def \_\_str\_\_(self):

if not self.args:

return str(self.symbol)

return f"{self.symbol}({','.join(str(arg) for arg in self.args)})"

def is\_variable(self):

return isinstance(self.symbol, str) and self.symbol.isupper() and not self.args

def occurs\_check(var, term, substitution):

"""Check if variable occurs in term"""

if term.is\_variable():

if term.symbol in substitution:

return occurs\_check(var, substitution[term.symbol], substitution)

return var.symbol == term.symbol

return any(occurs\_check(var, arg, substitution) for arg in term.args)

def substitute(term, substitution):

"""Apply substitution to term"""

if term.is\_variable() and term.symbol in substitution:

return substitute(substitution[term.symbol], substitution)

if not term.args:

return term

return Term(term.symbol, [substitute(arg, substitution) for arg in term.args])

def unify(term1, term2, substitution=None, iteration=1):

"""Unify two terms with detailed iteration steps"""

if substitution is None:

substitution = {}

print(f"\nIteration {iteration}:")

print(f"Attempting to unify: {term1} and {term2}")

print(f"Current substitution: {', '.join(f'{k}->{v}' for k,v in substitution.items()) or 'empty'}")

term1 = substitute(term1, substitution)

term2 = substitute(term2, substitution)

if term1.symbol == term2.symbol and not term1.args and not term2.args:

print("Terms are identical - no substitution needed")

return substitution

if term1.is\_variable():

if occurs\_check(term1, term2, substitution):

print(f"Occurs check failed: {term1.symbol} occurs in {term2}")

return None

substitution[term1.symbol] = term2

print(f"Added substitution: {term1.symbol} -> {term2}")

return substitution

if term2.is\_variable():

if occurs\_check(term2, term1, substitution):

print(f"Occurs check failed: {term2.symbol} occurs in {term1}")

return None

substitution[term2.symbol] = term1

print(f"Added substitution: {term2.symbol} -> {term1}")

return substitution

if term1.symbol != term2.symbol or len(term1.args) != len(term2.args):

print(f"Unification failed: Different predicates or argument lengths")

return None

for arg1, arg2 in zip(term1.args, term2.args):

result = unify(arg1, arg2, substitution, iteration + 1)

if result is None:

return None

substitution = result

return substitution

def parse\_term(s):

"""Parse terms like P(X,f(Y)) or X"""

s = s.strip()

if '(' not in s:

return Term(s)

pred = s[:s.index('(')]

args\_str = s[s.index('(')+1:s.rindex(')')]

args = []

current = ''

depth = 0

for c in args\_str:

if c == '(' or c == '[':

depth += 1

elif c == ')' or c == ']':

depth -= 1

elif c == ',' and depth == 0:

args.append(parse\_term(current.strip()))

current = ''

continue

current += c

if current:

args.append(parse\_term(current.strip()))

return Term(pred, args)

def print\_examples():

print("\nExample format:")

print("1. Simple terms: P(X,Y)")

print("2. Nested terms: P(f(X),g(Y))")

print("3. Mixed terms: Knows(John,X)")

print("4. Complex nested terms: P(f(g(X)),h(Y,Z))")

print("\nNote: Use capital letters for variables (X,Y,Z) and lowercase for constants and predicates.")

def validate\_input(expr):

"""Basic validation for input expressions"""

if not expr:

return False

# Check balanced parentheses

count = 0

for char in expr:

if char == '(':

count += 1

elif char == ')':

count -= 1

if count < 0:

return False

return count == 0

def main():

while True:

print("\n=== First Order Predicate Logic Unification ===")

print("1. Start Unification")

print("2. Show Examples")

print("3. Exit")

choice = input("\nEnter your choice (1-3): ")

if choice == '1':

print("\nEnter two expressions to unify.")

print\_examples()

while True:

expr1 = input("\nEnter first expression (or 'back' to return): ")

if expr1.lower() == 'back':

break

if not validate\_input(expr1):

print("Invalid expression! Please check the format and try again.")

continue

expr2 = input("Enter second expression: ")

if not validate\_input(expr2):

print("Invalid expression! Please check the format and try again.")

continue

try:

term1 = parse\_term(expr1)

term2 = parse\_term(expr2)

print("\nUnification Process:")

result = unify(term1, term2)

print("\nFinal Result:")

if result is None:

print("Unification failed!")

else:

print("Unification successful!")

print("Final substitutions:", ', '.join(f'{k}->{v}' for k,v in result.items()))

retry = input("\nTry another unification? (y/n): ")

if retry.lower() != 'y':

break

except Exception as e:

print(f"Error processing expressions: {str(e)}")

print("Please check your input format and try again.")

elif choice == '2':

print("\n=== Example Expressions ===")

print("1. P(X,h(Y)) and P(a,f(Z))")

print("2. P(f(a),g(Y)) and P(X,X)")

print("3. Knows(John,X) and Knows(X,Elisabeth)")

print("\nPress Enter to continue...")

input()

elif choice == '3':

print("\nThank you for using the Unification Program!")

break

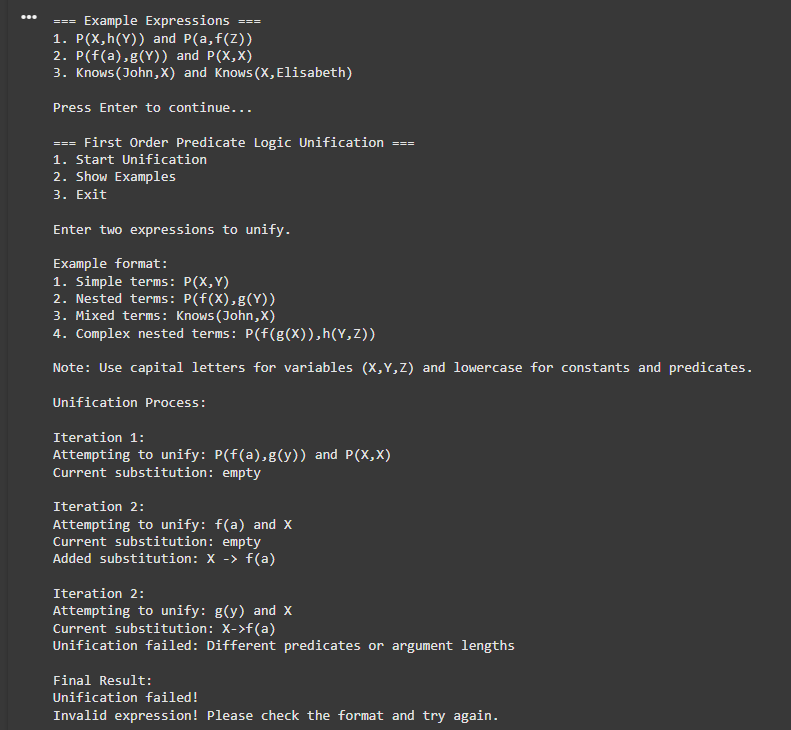
else:

print("\nInvalid choice! Please enter 1, 2, or 3.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output:



Output 2:

