```
from sympy import symbols, Not, Or, And, Implies, Equivalent
from sympy.logic.boolalg import to_cnf

def fol_to_cnf(fol_expr):
    """
    Converts a First-Order Logic (FOL) statement to Conjunctive Normal Form (CNF).
    Arguments:
        fol_expr: A sympy logical expression representing the FOL statement.
    Returns:
        The CNF equivalent of the input expression.
    """
    # Step 1: Eliminate equivalences (A ↔ B) using (A → B) ∧ (B → A)
    fol_expr = fol_expr.replace(Equivalent, lambda a, b: And(Implies(a, b), Implies(b, a)))

    # Step 2: Eliminate implications (A → B) using (¬A ∨ B)
    fol_expr = fol_expr.replace(Implies, lambda a, b: Or(Not(a), b))

    # Step 3: Convert to CNF
    cnf_form = to_cnf(fol_expr, simplify=True)

    return cnf_form

def main():
    # Define propositional symbols instead of first-order predicates
    P = symbols("P")
    Q = symbols("Q")
    R = symbols("R")

    # Example 1: P → Q
    fol_expr1 = Implies(P, Q)
    print("Example 1: P → Q")
    print("Original FOL Expression:")
    print(fol_expr1)

    # Convert to CNF
    cnf1 = fol_to_cnf(fol_expr1)
    print("\nCNF Form:")
    print(cnf1)

    # Example 2: (P ∨ ¬Q) → (Q ∨ R)
    fol_expr2 = Implies(Or(P, Not(Q)), Or(Q, R))
    print("\nExample 2: (P ∨ ¬Q) → (Q ∨ R)")
    print("Original FOL Expression:")
    print(fol_expr2)

    # Convert to CNF
    cnf2 = fol_to_cnf(fol_expr2)
    print("\nCNF Form:")
    print(cnf2)

if __name__ == "__main__":
    main()
```

```
Example 1: P → Q
Original FOL Expression:
Implies(P, Q)

CNF Form:
Q | ~P

Example 2: (P ∨ ¬Q) → (Q ∨ R)
Original FOL Expression:
Implies(P | ~Q, Q | R)
```

```
CNF Form:
Q | R
```