

Propositional logic

```
import itertools

# Function to evaluate an expression

def evaluate_expression(a, b, c, expression):

    # Use eval() to evaluate the logical expression

    return eval(expression)


# Function to generate the truth table and evaluate a logical expression

def truth_table_and_evaluation(kb, query):

    # All possible combinations of truth values for a, b, and c

    truth_values = [True, False]

    combinations = list(itertools.product(truth_values, repeat=3))

    # Reverse the combinations to start from the bottom (False -> True)

    combinations.reverse()

    # Header for the full truth table

    print(f"{'a':<5} {'b':<5} {'c':<5} {'KB':<20}{'Query':<20}")

    # Evaluate the expressions for each combination

    for combination in combinations:

        a, b, c = combination
```

```

# Evaluate the knowledge base (KB) and query expressions

kb_result = evaluate_expression(a, b, c, kb)

query_result = evaluate_expression(a, b, c, query)


# Replace True/False with string "True"/"False"

kb_result_str = "True" if kb_result else "False"

query_result_str = "True" if query_result else "False"


# Convert boolean values of a, b, c to "True"/"False"

a_str = "True" if a else "False"

b_str = "True" if b else "False"

c_str = "True" if c else "False"


# Print the results for the knowledge base and the query

print(f'{a_str:<5} {b_str:<5} {c_str:<5} {kb_result_str:<20} {query_result_str:<20}')


# Additional output for combinations where both KB and query are true

print("\nCombinations where both KB and Query are True:")

print(f'{a_str:<5} {b_str:<5} {c_str:<5} {kb_result_str:<20} {query_result_str:<20}')


# Print only the rows where both KB and Query are True

for combination in combinations:

    a, b, c = combination

```

```
# Evaluate the knowledge base (KB) and query expressions
```

```
kb_result = evaluate_expression(a, b, c, kb)
```

```
query_result = evaluate_expression(a, b, c, query)
```

```
# If both KB and query are True, print the combination
```

```
if kb_result and query_result:
```

```
    a_str = "True" if a else "False"
```

```
    b_str = "True" if b else "False"
```

```
    c_str = "True" if c else "False"
```

```
    kb_result_str = "True" if kb_result else "False"
```

```
    query_result_str = "True" if query_result else "False"
```

```
    print(f'{a_str:<5} {b_str:<5} {c_str:<5} {kb_result_str:<20} {query_result_str:<20}')
```

```
# Define the logical expressions as strings
```

```
kb = "(a or c) and (b or not c)" # Knowledge Base
```

```
query = "a or b" # Query to evaluate
```

```
# Generate the truth table and evaluate the knowledge base and query
```

```
truth_table_and_evaluation(kb, query)
```

a	b	c	KB	Query
False	False	False	False	False
False	False	True	False	False
False	True	False	False	True
False	True	True	True	True
True	False	False	True	True
True	False	True	False	True
True	True	False	True	True
True	True	True	True	True

Combinations where both KB and Query are True:

a	b	c	KB	Query
False	True	True	True	True
True	False	False	True	True
True	True	False	True	True
True	True	True	True	True