**PROGRAM [8]:**

P = 'P'

Q = 'Q'

R = 'R'

kb = [

(P, "=>", Q),

(Q, "=>", R),

(P,),

]

def is\_true(sentence, model):

if sentence[0] == 'not':

return not is\_true(sentence[1], model)

elif sentence[0] in model:

return model[sentence[0]]

elif len(sentence) == 1:

return False

elif sentence[1] == 'and':

return is\_true(sentence[0], model) and is\_true(sentence[2], model)

elif sentence[1] == 'or':

return is\_true(sentence[0], model) or is\_true(sentence[2], model)

elif sentence[1] == '=>':

return not is\_true(sentence[0], model) or is\_true(sentence[2], model)

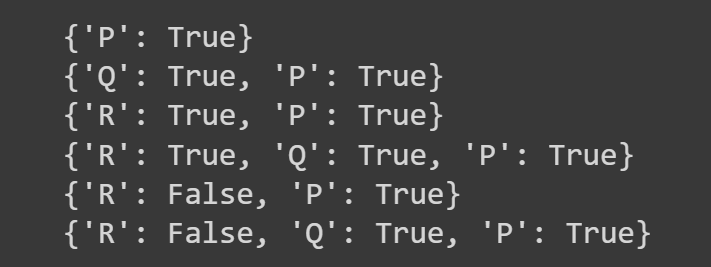
elif sentence[1] == '<=>':

return is\_true(sentence[0], model) == is\_true(sentence[2], model)

def is\_model\_satisfies\_kb(model, kb):

for sentence in kb:

**OUTPUT [8]:**



if not is\_true(sentence, model):

return False

return True

def generate\_models(symbols):

if not symbols:

return [{}]

else:

symbol = symbols[0]

rest = symbols[1:]

models = []

for model in generate\_models(rest):

models.append(model)

models.append({\*\*model, \*\*{symbol: True}})

models.append({\*\*model, \*\*{symbol: False}})

return models

symbols = [P, Q, R]

models = generate\_models(symbols)

for model in models:

if is\_model\_satisfies\_kb(model, kb):

print(model)