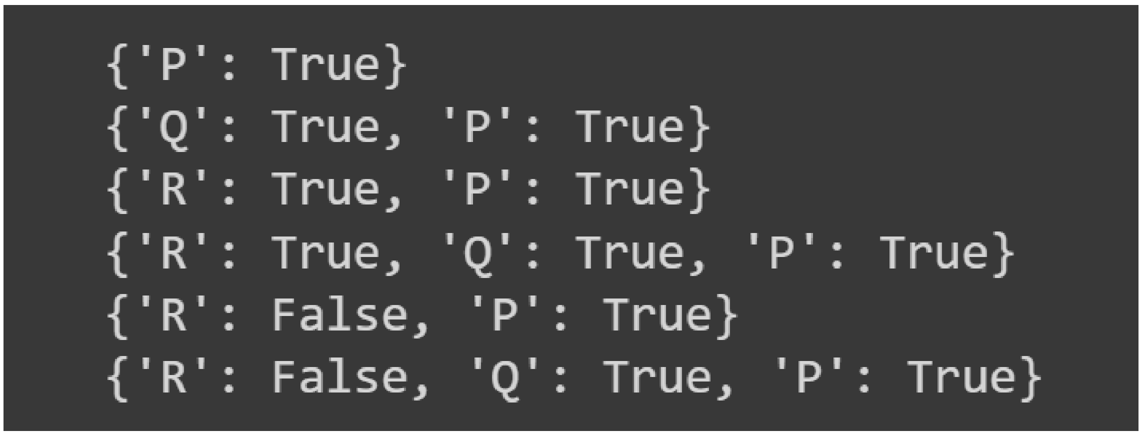
PROGRAM [8]:

```
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]:



```
{'P': True}
{'Q': True, 'P': True}
{'R': True, 'P': True}
{'R': True, 'Q': True, 'P': True}
{'R': False, 'P': True}
{'R': False, 'Q': True, 'P': True}
```

```
        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)
```