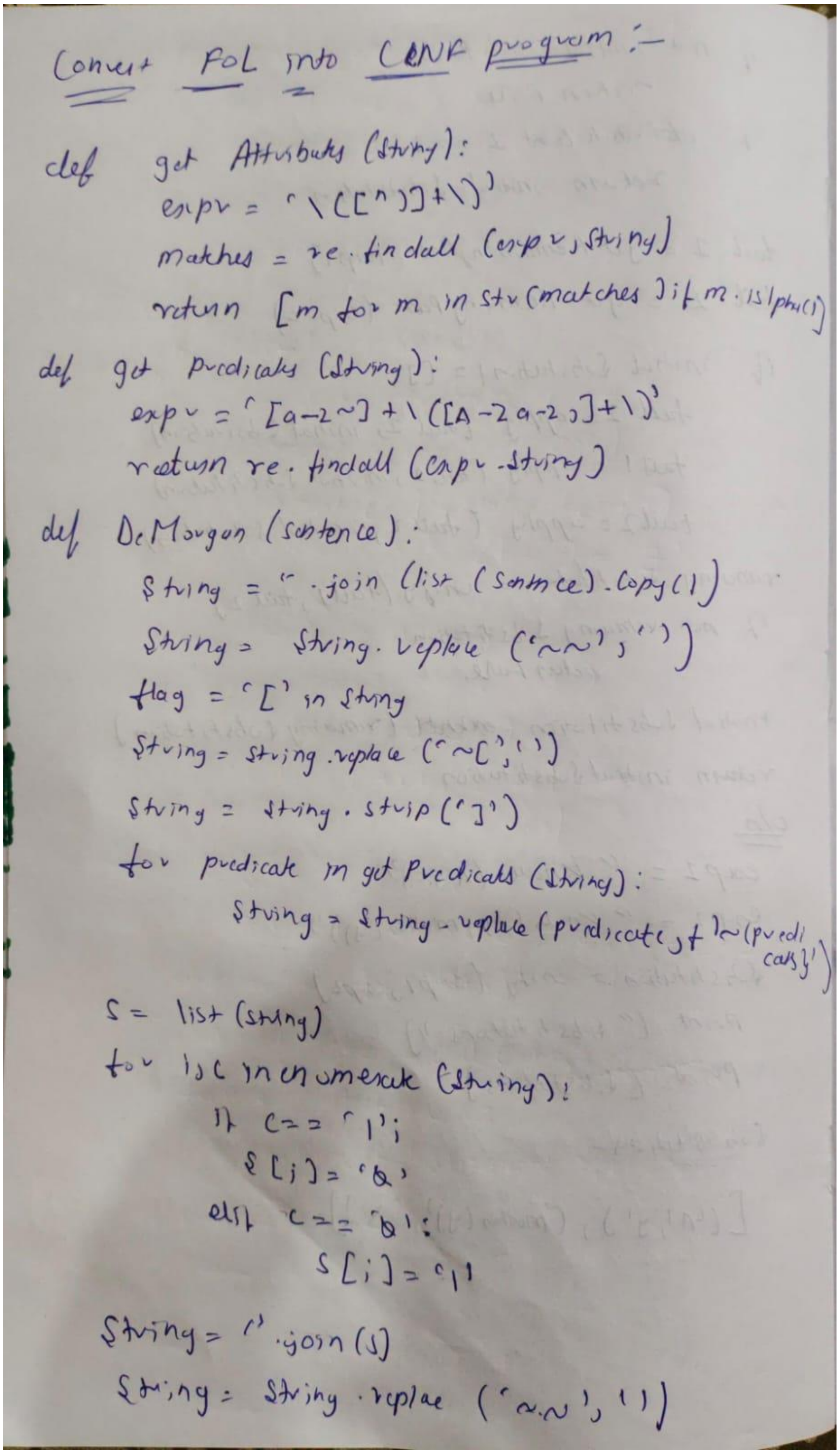
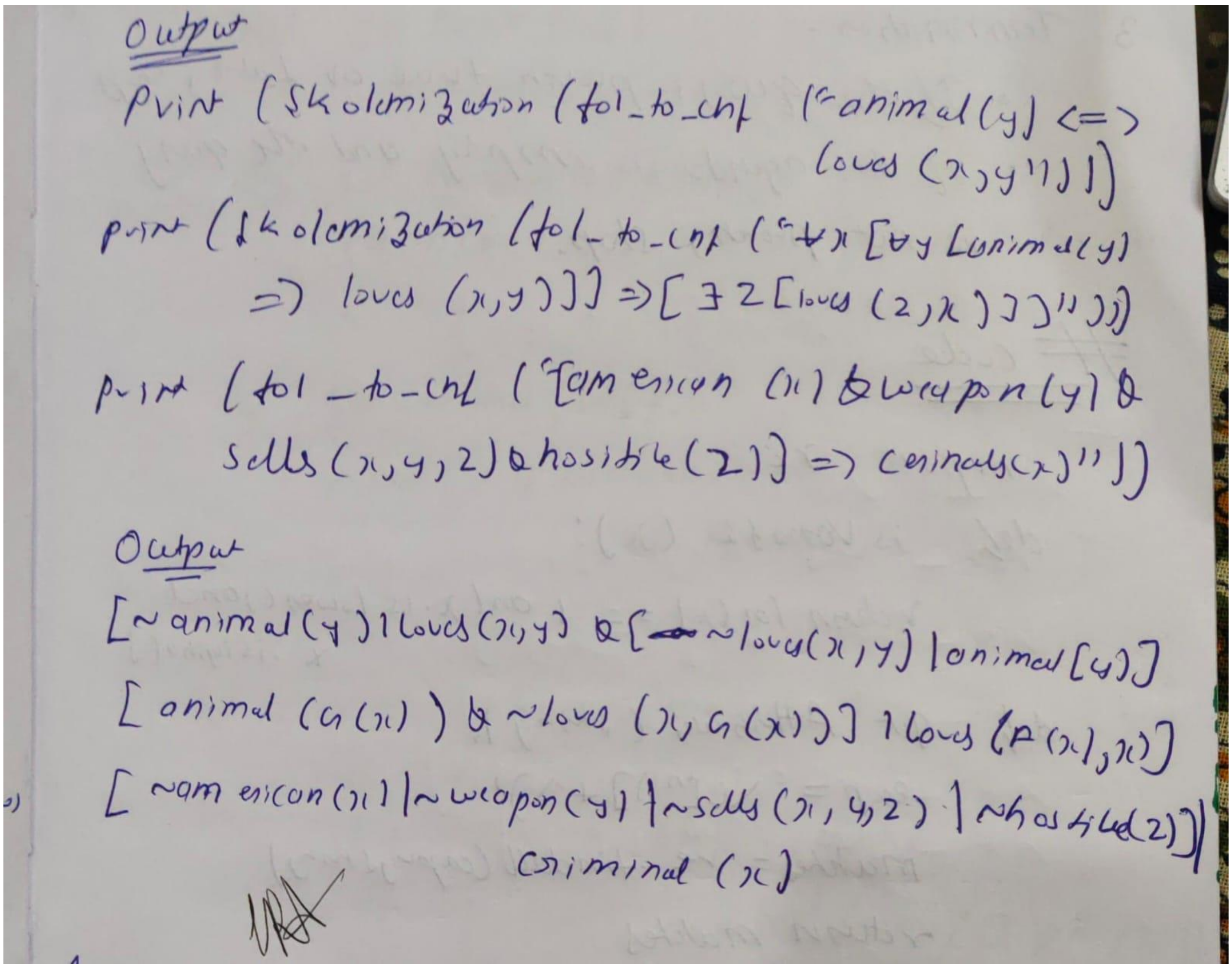


Convert FOL into CNF program :-

```
def  get Attributes (string):
        expr = '\([^)]+\)'
        matches = re.findall (expr, string)
        return [m for m in str(matches) if m.isalpha()]
def  get Predicates (String):
        expr = '[a-z~]+\([A-Za-z,]+\)'
        return re.findall (expr.string)
def  DeMorgan (sentence):
        String = ''.join (list (sentence).copy())
        String = String.replace ('~~', '')
        flag = '[' in string
        String = String.replace ('~[', '')
        String = string.strip (']')
        for predicate in get Predicates (string):
                String = String.replace (predicate, f'~(predicates}')
        S = list (string)
        for i, c in enumerate (string):
                if c == '|';
                    S[i] = 'Q'
                elif c == 'Q':
                        S[i] = '|'
        String = ''.join (s)
        String = String.replace ('~~', '')
```

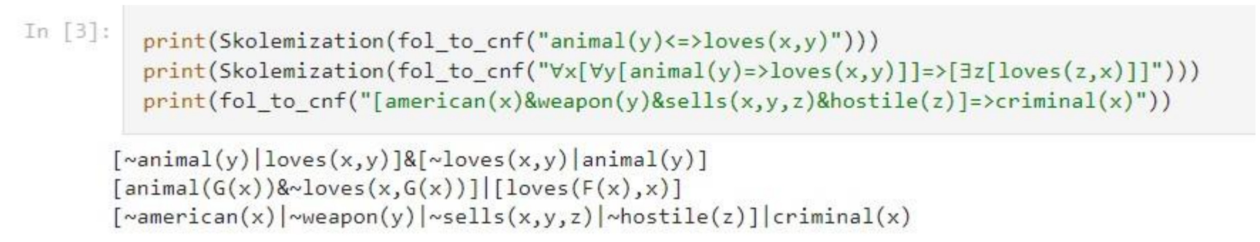
```
    return f'[{string}]' if flag else string

def Skolemization(sentence):
    skolem-constants=[f'{chr(c)}' for c in range
                                (ord('A'),ord('Z')+1)]

    Statement = ''.join(list(sentence).copy())
    matches = re.findall('[∀∃].', statement)
    for match in matches[::-1]:
        statement = statement.replace(match,'')
        Statements = re.findall('\[\[[^]]+\]]', statement)
        for s in Statements:
            Statement= Statement.replace(s,s[1:-1])

 ANTS.pop(0)}({al[0] if len(al) else match[1]}))'}
    return statement.

import re
def fol_to_cnf(fol):
    Statement = fol.replace("<=>", '-')
    while '-' in statement:
        i = Statement.index('-')
        new_statement = '[' + Statement[:i] + '=>' +
            Statement[i + 1:] + ']&[' + Statement
            [i +1:] + '=>' + Statement[:i] + ']'
        Statement = new_statement
    Statement = statement.replace("=>", "-")
    expr = '\[([^]]+)\]'
    Statement = re.findall(expr, Statement)
```

```
for i, s in enumerate (statements):
        if '[' in s & ']' not in s:
                statements [i] + = ']'

    for s in statements:
            statement = statement_replace (s, fol_to_
                                                    cnf (s))
new _statement
    while '~∀' in statement:
            i = statement .index ('~∀')
            statement = list (statement)
            statement [i ], statement [i + 1], statement
                        [i +2 ] = '∃', statement
                                        [i +2], '~'

    while '~∃' in statement:
            i = statement.index ('~∃')
            s = list (statement)
            s[i], s [i+1], s[i +2] = '∀', s[i +2], '~'
            statement = ''.join(s)
    for s in statements:
            statement = statement.replace (s, fol_to_cnf(s)
    exp = '~\[[^]]+\]'
    statements = re findall (expr, statement)
    for s in statements:
            statement = statement .replace (s, DeMorgan
                                                    (s))
    return statement
```



```
Output
Print (Skolemization (fol_to_cnf ("animal(y) <=>
                              loves (x,y")))
print (Skolemization (fol_to_cnf ("∀x [∀y [animal(y)
         => loves (x,y)]] => [∃z [loves (z,x)]]")))
print (fol_to_cnf ("[american (x) & weapon (y) &
       sells (x,y,z) & hostile (z)] => criminal(x)"))

Output
[~animal(y) | loves(x,y) & [~loves(x,y) | animal[y)]
[animal (G(x)) & ~loves (x, G(x))] | loves (F(x),x)]
[~american(x) | ~weapon(y) | ~sells (x,y,z) | ~hostile(z)]|
                  criminal (x)
```

Output:



```
In [3]: print(Skolemization(fol_to_cnf("animal(y)<=>loves(x,y)")))
        print(Skolemization(fol_to_cnf("∀x[∀y[animal(y)=>loves(x,y)]]=>[∃z[loves(z,x)]]")))
        print(fol_to_cnf("[american(x)&weapon(y)&sells(x,y,z)&hostile(z)]=>criminal(x)"))

[~animal(y)|loves(x,y)]&[~loves(x,y)|animal(y)]
[animal(G(x))&~loves(x,G(x))]|[loves(F(x),x)]
[~american(x)|~weapon(y)|~sells(x,y,z)|~hostile(z)]|criminal(x)
```