PROGRAM TITLE 3

CRIPT ARITHMETIC PROBLEM

AIM:

To write a python program to solve cript arithmetic problem.

PROCEDURE:

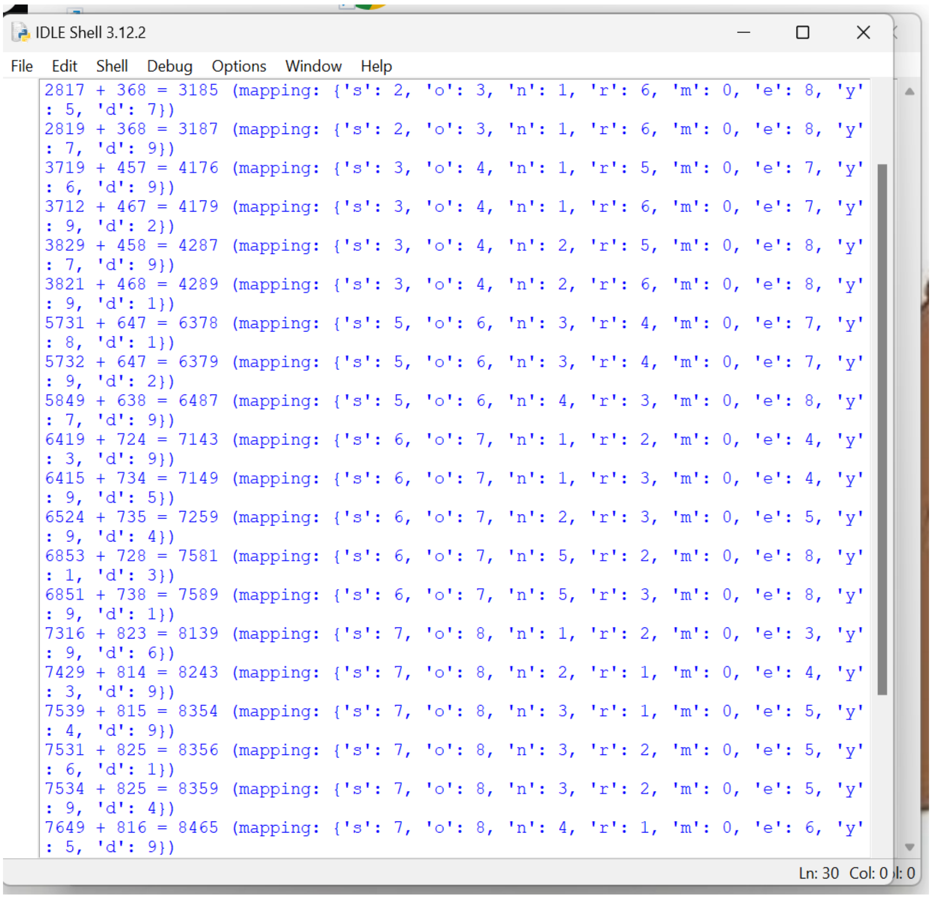
1. **Define Variables:**
   - Identify the distinct letters in the puzzle and assign them variables. For example, if you have a puzzle like "SEND + MORE = MONEY," assign variables to the letters S, E, N, D, M, O, R, Y.
2. **Generate Possible Assignments:**
   - Use a permutation algorithm to generate all possible assignments of digits to the variables, ensuring that each digit is assigned to a unique letter. You can start with a simple brute-force approach.
3. **Evaluate Constraints:**
   - Implement a function to check whether a given assignment satisfies the constraints of the puzzle. This involves substituting the assigned values into the puzzle equation and verifying that it holds true.
4. **Search for Solutions:**
   - Iterate through the generated assignments and use the constraint evaluation function to identify solutions to the puzzle. Keep track of valid solutions.
5. **Print or Output Solutions:**
   - Once solutions are found, print or output the values of the variables that satisfy the puzzle equation. If there are multiple solutions, you can choose to print all of them.

CODING:

```
import itertools def
get_value(word, substitution):
   s = 0
factor = 1
for letter in
```

```
reversed(wor
d):

        s += factor * substitution[letter]
factor *= 10     return s def
solve2(equation):


    left, right = equation.lower().replace(' ', ").split('=')


    left = left.split('+')


    letters = set(right)     for word in left:          for letter in
word:             letters.add(letter)     letters = list(letters)
digits = range(10)     for perm in
itertools.permutations(digits, len(letters)):
        sol = dict(zip(letters, perm))


        if sum(get_value(word, sol) for word in left) == get_value(right, sol):

            print(' + '.join(str(get_value(word, sol)) for word in left) + " = {} (mapping:
{})".format(get_value(right, sol), sol))


if __name__ == '__main__':
solve2('SEND + MORE = MONEY')
```

OUTPUT:



```
2817 + 368 = 3185 (mapping: {'s': 2, 'o': 3, 'n': 1, 'r': 6, 'm': 0, 'e': 8, 'y'
: 5, 'd': 7})
2819 + 368 = 3187 (mapping: {'s': 2, 'o': 3, 'n': 1, 'r': 6, 'm': 0, 'e': 8, 'y'
: 7, 'd': 9})
3719 + 457 = 4176 (mapping: {'s': 3, 'o': 4, 'n': 1, 'r': 5, 'm': 0, 'e': 7, 'y'
: 6, 'd': 9})
3712 + 467 = 4179 (mapping: {'s': 3, 'o': 4, 'n': 1, 'r': 6, 'm': 0, 'e': 7, 'y'
: 9, 'd': 2})
3829 + 458 = 4287 (mapping: {'s': 3, 'o': 4, 'n': 2, 'r': 5, 'm': 0, 'e': 8, 'y'
: 7, 'd': 9})
3821 + 468 = 4289 (mapping: {'s': 3, 'o': 4, 'n': 2, 'r': 6, 'm': 0, 'e': 8, 'y'
: 9, 'd': 1})
5731 + 647 = 6378 (mapping: {'s': 5, 'o': 6, 'n': 3, 'r': 4, 'm': 0, 'e': 7, 'y'
: 8, 'd': 1})
5732 + 647 = 6379 (mapping: {'s': 5, 'o': 6, 'n': 3, 'r': 4, 'm': 0, 'e': 7, 'y'
: 9, 'd': 2})
5849 + 638 = 6487 (mapping: {'s': 5, 'o': 6, 'n': 4, 'r': 3, 'm': 0, 'e': 8, 'y'
: 7, 'd': 9})
6419 + 724 = 7143 (mapping: {'s': 6, 'o': 7, 'n': 1, 'r': 2, 'm': 0, 'e': 4, 'y'
: 3, 'd': 9})
6415 + 734 = 7149 (mapping: {'s': 6, 'o': 7, 'n': 1, 'r': 3, 'm': 0, 'e': 4, 'y'
: 9, 'd': 5})
6524 + 735 = 7259 (mapping: {'s': 6, 'o': 7, 'n': 2, 'r': 3, 'm': 0, 'e': 5, 'y'
: 9, 'd': 4})
6853 + 728 = 7581 (mapping: {'s': 6, 'o': 7, 'n': 5, 'r': 2, 'm': 0, 'e': 8, 'y'
: 1, 'd': 3})
6851 + 738 = 7589 (mapping: {'s': 6, 'o': 7, 'n': 5, 'r': 3, 'm': 0, 'e': 8, 'y'
: 9, 'd': 1})
7316 + 823 = 8139 (mapping: {'s': 7, 'o': 8, 'n': 1, 'r': 2, 'm': 0, 'e': 3, 'y'
: 9, 'd': 6})
7429 + 814 = 8243 (mapping: {'s': 7, 'o': 8, 'n': 2, 'r': 1, 'm': 0, 'e': 4, 'y'
: 3, 'd': 9})
7539 + 815 = 8354 (mapping: {'s': 7, 'o': 8, 'n': 3, 'r': 1, 'm': 0, 'e': 5, 'y'
: 4, 'd': 9})
7531 + 825 = 8356 (mapping: {'s': 7, 'o': 8, 'n': 3, 'r': 2, 'm': 0, 'e': 5, 'y'
: 6, 'd': 1})
7534 + 825 = 8359 (mapping: {'s': 7, 'o': 8, 'n': 3, 'r': 2, 'm': 0, 'e': 5, 'y'
: 9, 'd': 4})
7649 + 816 = 8465 (mapping: {'s': 7, 'o': 8, 'n': 4, 'r': 1, 'm': 0, 'e': 6, 'y'
: 5, 'd': 9})
```

RESULT:

Hence the program been successfully executed and verified.