**Implementation of Constraint Satisfaction Problem**

**Aim:**

To implement the Constraint Satisfaction Problem based on the given constraints.

Manual Procedure:

1. From Column 5, M=1, since it is only carry-over possible from sum of 2 single digit number in column 4.
2. To produce a carry from column 4 to column 5 'S + M' is atleast 9 so 'S=8or9' so 'S+M=9or10' & so 'O = 0 or 1'. But 'M=1', so 'O = 0'.
3. If there is carry from Column 3 to 4 then 'E=9' & so 'N=0'. But 'O = 0' so there is no carry & 'S=9' & 'c3=0'.
4. If there is no carry from column 2 to 3 then 'E=N' which is impossible, therefore there is carry & 'N=E+1' & 'c2=1'.
5. If there is carry from column 1 to 2 then 'N+R=E mod 10' & 'N=E+1' so 'E+1+R=E mod 10', so 'R=9' but 'S=9', so there must be carry from column 1 to 2. Therefore 'c1=1' & 'R=8'.
6. To produce carry 'c1=1' from column 1 to 2, we must have 'D+E=10+Y' as Y cannot be 0/1 so D+E is atleast 12. As D is atmost 7 & E is atleast 5 (D cannot be 8 or 9 as it is already assigned). N is atmost 7 & 'N=E+1' so 'E=5or6'.
7. If E were 6 & D+E atleast 12 then D would be 7, but 'N=E+1' & N would also be 7 which is impossible. Therefore 'E=5' &  
   'N=6'.
8. D+E is atleast 12 for that we get 'D=7' & 'Y=2'.

**CODE:**

**import** itertools

**def** get\_value(word, substitution):

    s **=** 0

    factor **=** 1

**for** letter **in** reversed(word):

        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):

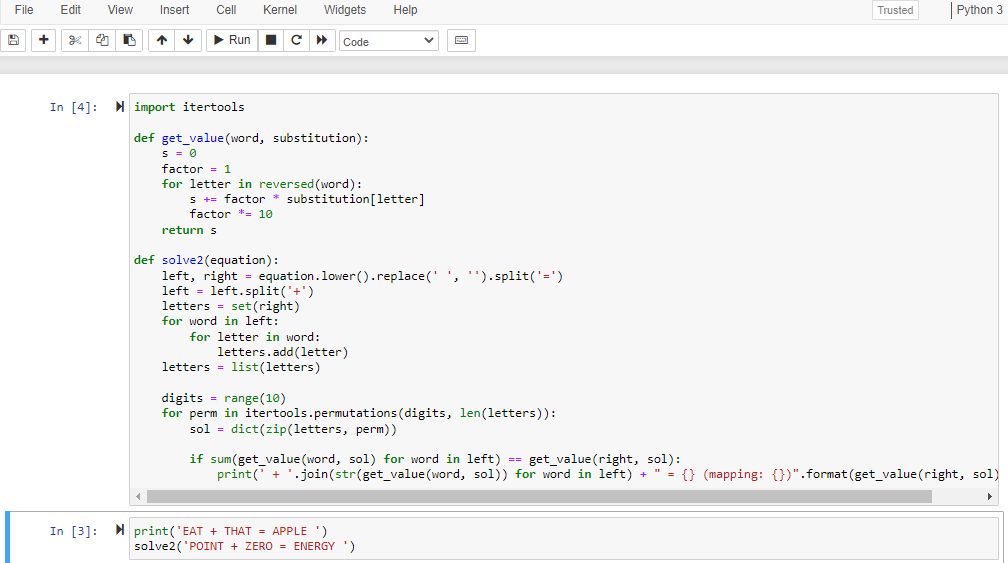
            print(' + '**.**join(str(get\_value(word, sol)) **for** word **in** left) **+** " = {} (mapping: {})"**.**format(get\_value(right, sol), sol))

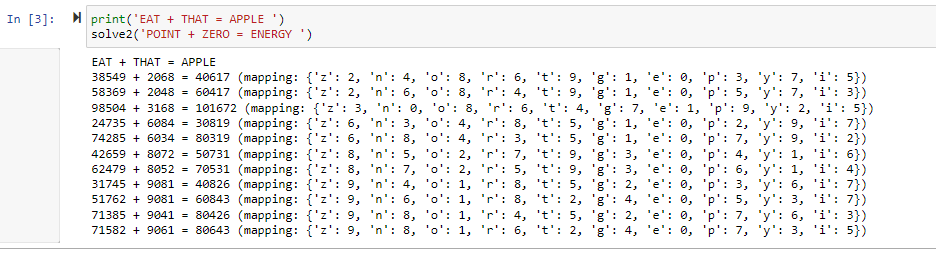
In [4]:

print('EAT + THAT = APPLE ')

solve2('POINT + ZERO = ENERGY ')

**OUTPUT:**

****

****