AI EX 3

CRYPT ARITHMETIC

AIM: To solve the following puzzle by assigning numeral (0-9) in such a way that each letter is assigned unique digit which satisfy the following addition. Constraints : No two letters have the same value.(The constraints of arithmetic).

PROCEDURE:

BASE + BALL = GAMES

- Assuming numbers can't start with 0, G is 1 because two four-digit numbers can't sum to 20000 or more.
- SE+LL=ES or 1ES.
- If it is ES, then LL must be a multiple of 9 because SE and ES are always congruent mod 9. But LL is a multiple of 11, so it would have to be 99, which is impossible.
- So SE+LL=1ES. LL must be congruent to 100 mod 9. The only multiple of 11 that work is 55, so L is 5.
- SE+55=1ES. This is possible when E+5=S. The possibilities for ES are 27, 38, or 49.
- BA+BA+1=1AM. B must be at least 5 because B+B (possibly +1 from a carry) is at least 10.
- If A is less than 5, then A+A+1 does not carry, a and A must be even. Inversely, if A is greater than 5, it must be odd. The possibilities for A are 0, 2, 4, 7, or 9.
- 0 does not work because M would have to be 1.
- 2 and 7 don't work because M would have to be 5.
- 9 doesn't work because M would also have to be 9.

- So A is 4, M is 9, and B is 7. This leaves 38 as the only possibility for ES. The
- full equation is:

  7483

  + 7455

  14938

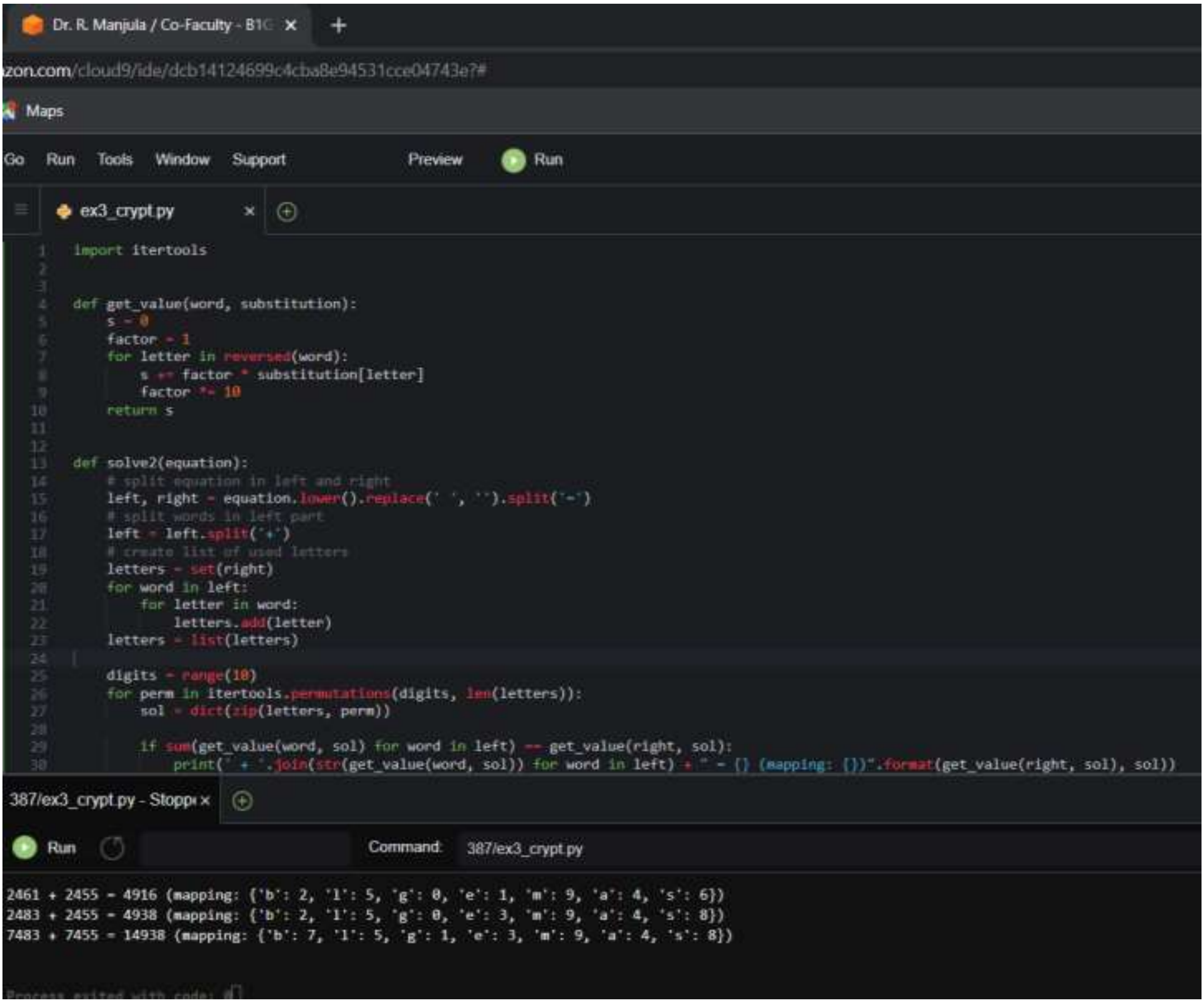
CODE:

```
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):
    # split equation in left and right
    left, right = equation.lower().replace(' ', '').split('=')
    # split words in left part
    left = left.split('+')
    # create list of used letters
    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('base + ball = games')
```

RESULT:

The Constraint Satisfaction Problem was implemented successfully where the possible solutions were displayed based on user input.

OUTPUT:


Dr. R. Manjula / Co-Faculty - B1C
zon.com/cloud9/ide/dcb14124699c4cba8e94531cce04743e?#
Maps
Go
Run
Tools
Window
Support
Preview
Run
ex3_crypt.py
387/ex3_crypt.py - Stopp
Run

Command: 387/ex3_crypt.py

```python
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):
    # split equation in left and right
    left, right = equation.lower().replace(' ', '').split('=')
    # split words in left part
    left = left.split('+')
    # create list of used letters
    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))
```

```
2461 + 2455 = 4916 (mapping: {'b': 2, 'l': 5, 'g': 0, 'e': 1, 'm': 9, 'a': 4, 's': 6})
2483 + 2455 = 4938 (mapping: {'b': 2, 'l': 5, 'g': 0, 'e': 3, 'm': 9, 'a': 4, 's': 8})
7483 + 7455 = 14938 (mapping: {'b': 7, 'l': 5, 'g': 1, 'e': 3, 'm': 9, 'a': 4, 's': 8})

Process exited with code: 0
```

RA1911003010387

CHITRALEKHA.CH