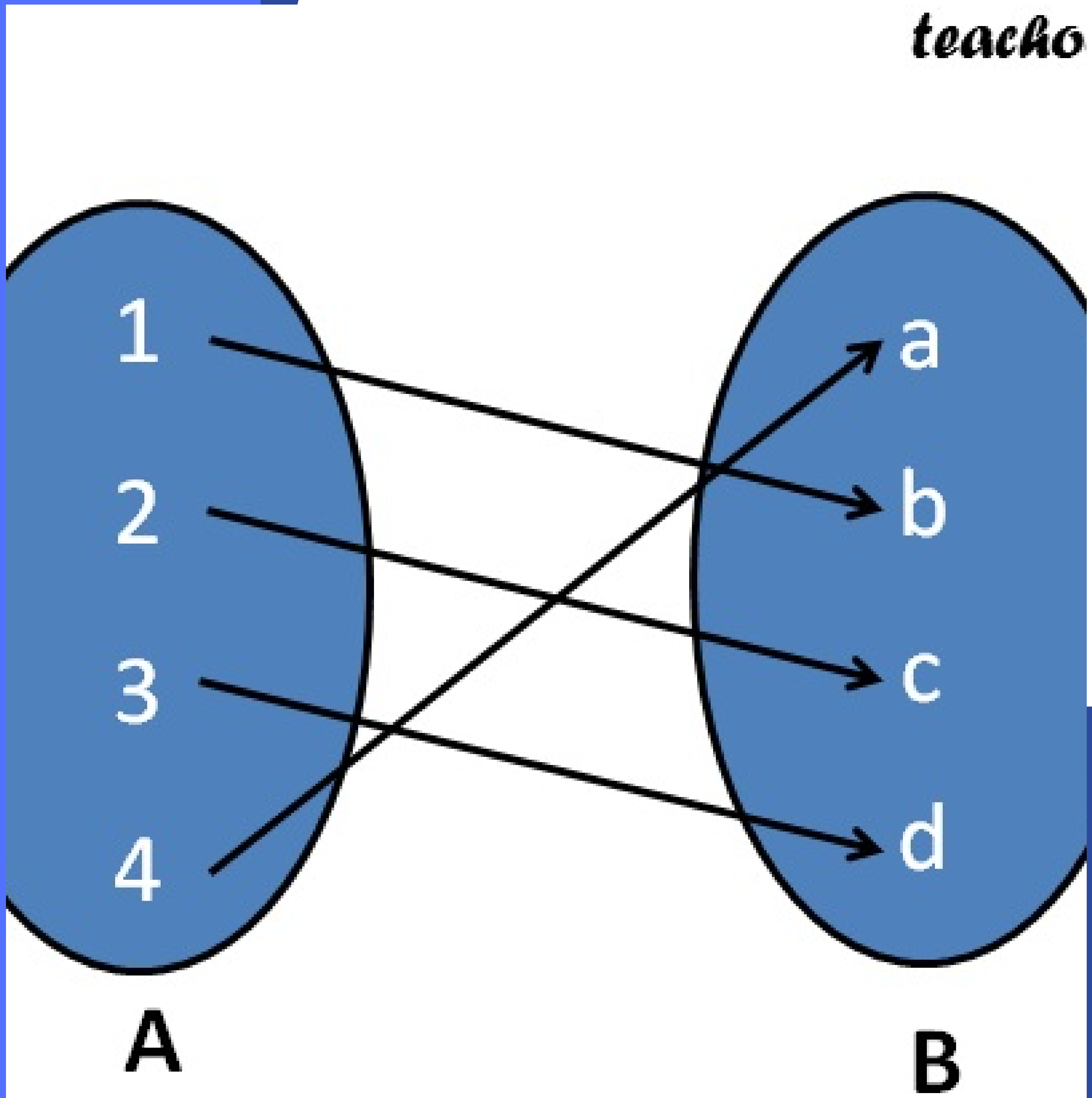
HARDIK HIRAMAN PAWAR
(1RV21CS046)

3RD SEM
DMS EL

CALCULATION OF NO. OF ONTO FUNCTIONS

INTRODUCTION

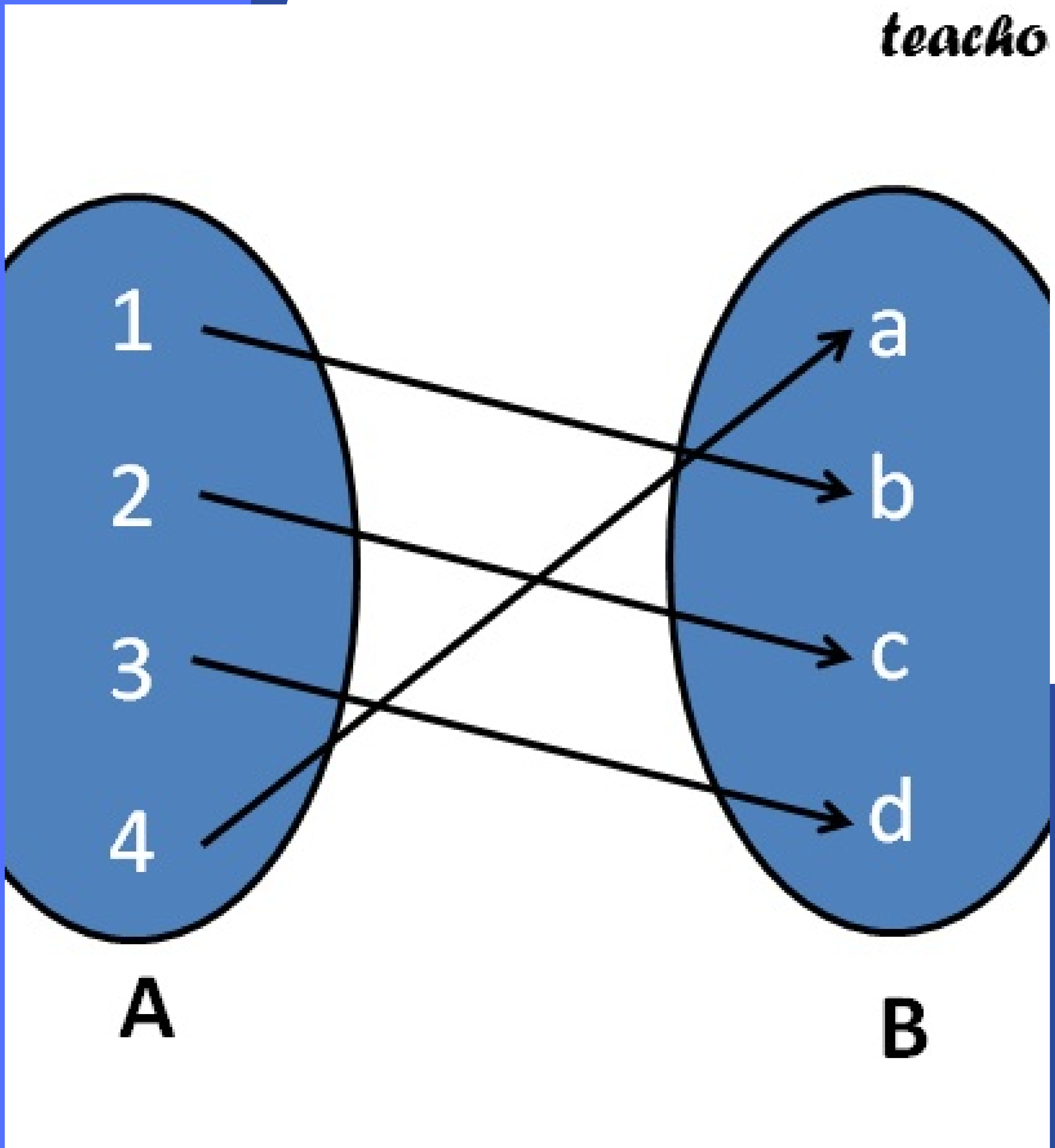
WHAT IS AN ONTO FUNCTION?



DEFINITION

A SURJECTIVE (ONTO) FUNCTION IS A FUNCTION F SUCH THAT EVERY ELEMENT Y CAN BE MAPPED FROM ELEMENT X SO THAT F(X) = Y.

- Range = Co-Domain
- *Every element of the function's codomain is the image of atleast one element of its domain.*



FORMULA

IF A SET A HAS M ELEMENTS AND SET B HAS N ELEMENTS, THEN THE NUMBER OF ONTO FUNCTIONS FROM A TO B = NM – NC1(N-1)M + NC2(N-2)M – NC3(N-3)M+….- NCN-1 (1)M
(M >= N ONLY)

- Range = Co-Domain
- *Every element of the function's codomain is the image of atleast one element of its domain.*

CODE

PYTHON PROGRAM EXPLANATION

1

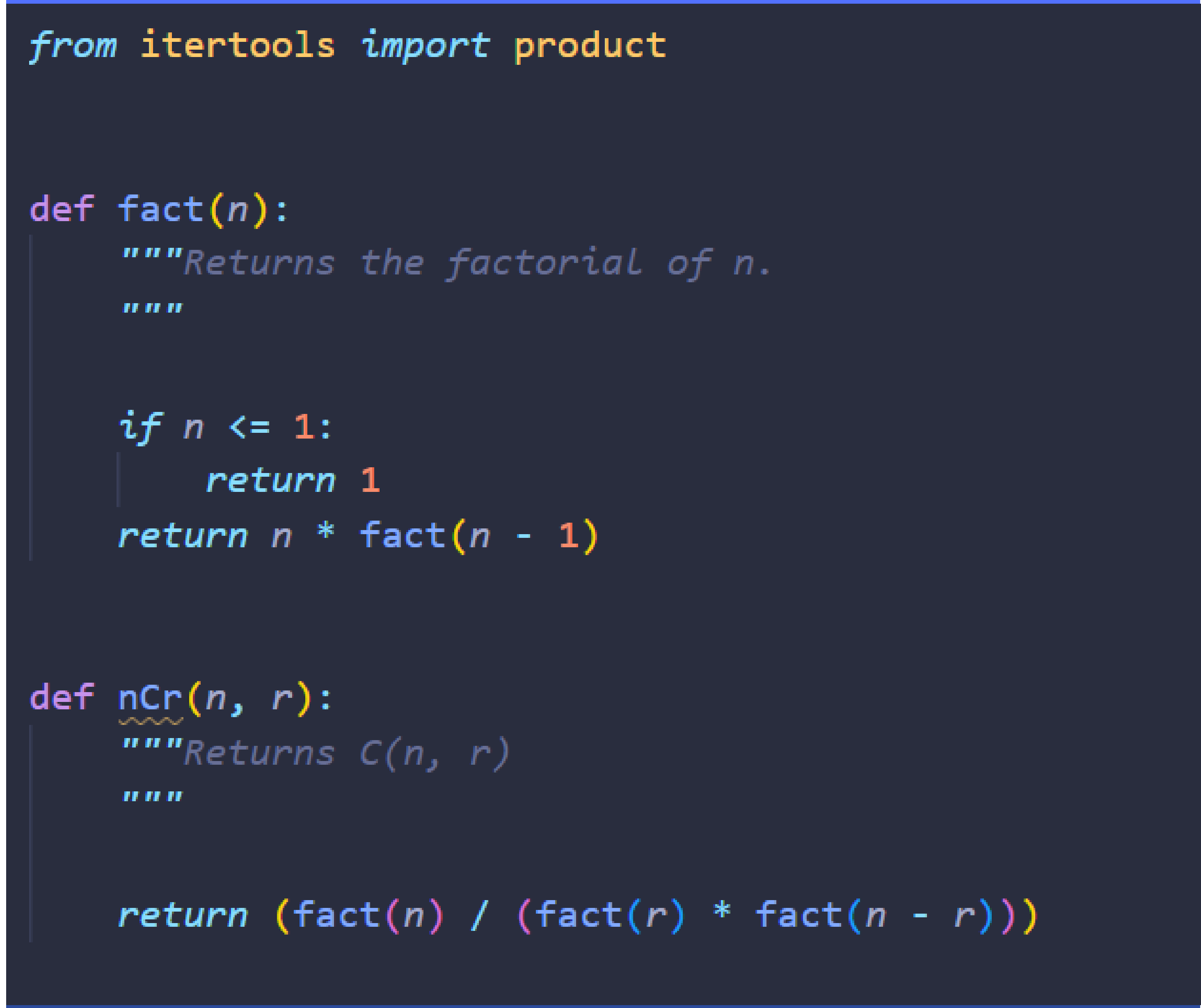
ITERTOOLS LIBRARY

To generate combinations with repetitions for the set B.

2

FACTORIAL & C(N, R) FUNCTIONS

To find the number of onto functions from set A to set B.



```python
from itertools import product


def fact(n):
    """Returns the factorial of n.
    """

    if n <= 1:
        return 1
    return n * fact(n - 1)


def nCr(n, r):
    """Returns C(n, r)
    """

    return (fact(n) / (fact(r) * fact(n - r)))
```

1

CHECKING M AND N

First checks if the no. of elements in set A (=m) is greater than equal to no. of elements in set B (=n), i.e., ***m>=n.***

2

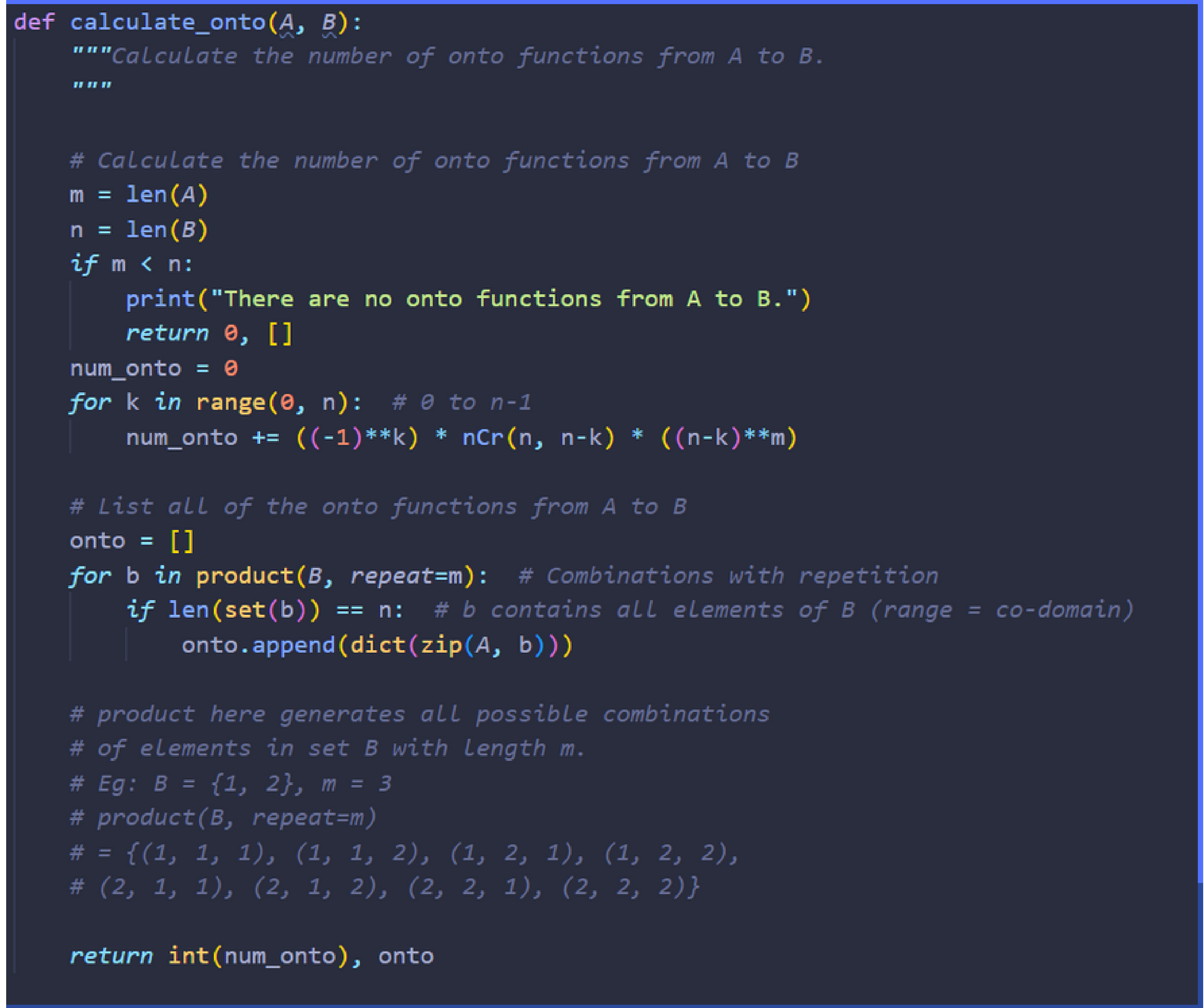
COMPUTING NO. OF ONTO FUNCTIONS

The no. of onto functions is computed by incorporating the formula showed before and storing the value in ***num_onto*** *variable.*

3

STORE ALL ONTO FUNCTIONS

All possible onto functions are stored in a list named ***onto*** *which is calculated using* ***product*** *function from the itertools library.*



```python
def calculate_onto(A, B):
    """Calculate the number of onto functions from A to B.
    """

    # Calculate the number of onto functions from A to B
    m = len(A)
    n = len(B)
    if m < n:
        print("There are no onto functions from A to B.")
        return 0, []
    num_onto = 0
    for k in range(0, n):  # 0 to n-1
        num_onto += ((-1)**k) * nCr(n, n-k) * ((n-k)**m)

    # List all of the onto functions from A to B
    onto = []
    for b in product(B, repeat=m):  # Combinations with repetition
        if len(set(b)) == n:  # b contains all elements of B (range = co-domain)
            onto.append(dict(zip(A, b)))

    # product here generates all possible combinations
    # of elements in set B with length m.
    # Eg: B = {1, 2}, m = 3
    # product(B, repeat=m)
    # = {(1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2),
    # (2, 1, 1), (2, 1, 2), (2, 2, 1), (2, 2, 2)}

    return int(num_onto), onto
```

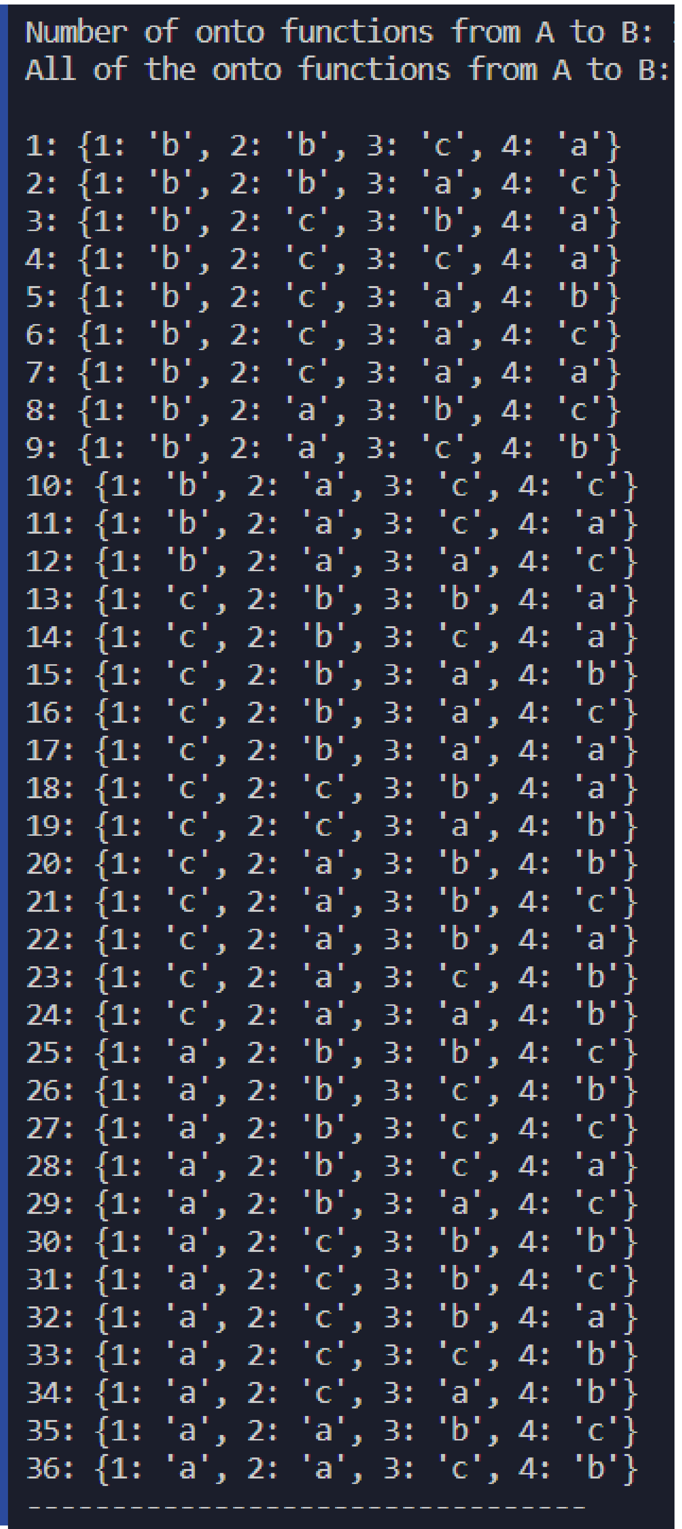
EXAMPLES

Default

FOR THE SETS:

S = {1, 2, 3, 4}

T = {'A', 'B', 'C'}



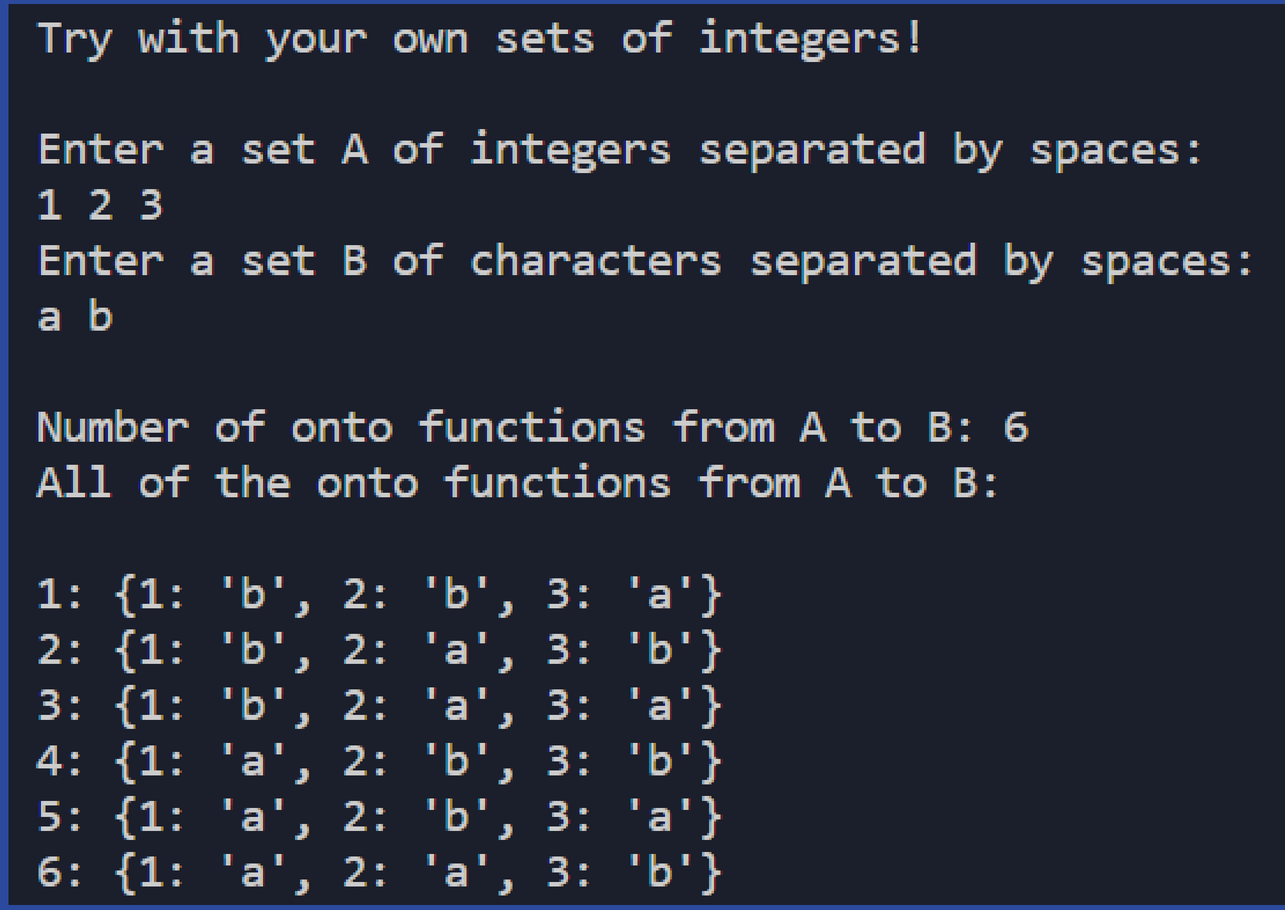
```
Number of onto functions from A to B:
All of the onto functions from A to B:

1: {1: 'b', 2: 'b', 3: 'c', 4: 'a'}
2: {1: 'b', 2: 'b', 3: 'a', 4: 'c'}
3: {1: 'b', 2: 'c', 3: 'b', 4: 'a'}
4: {1: 'b', 2: 'c', 3: 'c', 4: 'a'}
5: {1: 'b', 2: 'c', 3: 'a', 4: 'b'}
6: {1: 'b', 2: 'c', 3: 'a', 4: 'c'}
7: {1: 'b', 2: 'c', 3: 'a', 4: 'a'}
8: {1: 'b', 2: 'a', 3: 'b', 4: 'c'}
9: {1: 'b', 2: 'a', 3: 'c', 4: 'b'}
10: {1: 'b', 2: 'a', 3: 'c', 4: 'c'}
11: {1: 'b', 2: 'a', 3: 'c', 4: 'a'}
12: {1: 'b', 2: 'a', 3: 'a', 4: 'c'}
13: {1: 'c', 2: 'b', 3: 'b', 4: 'a'}
14: {1: 'c', 2: 'b', 3: 'c', 4: 'a'}
15: {1: 'c', 2: 'b', 3: 'a', 4: 'b'}
16: {1: 'c', 2: 'b', 3: 'a', 4: 'c'}
17: {1: 'c', 2: 'b', 3: 'a', 4: 'a'}
18: {1: 'c', 2: 'c', 3: 'b', 4: 'a'}
19: {1: 'c', 2: 'c', 3: 'a', 4: 'b'}
20: {1: 'c', 2: 'a', 3: 'b', 4: 'b'}
21: {1: 'c', 2: 'a', 3: 'b', 4: 'c'}
22: {1: 'c', 2: 'a', 3: 'b', 4: 'a'}
23: {1: 'c', 2: 'a', 3: 'c', 4: 'b'}
24: {1: 'c', 2: 'a', 3: 'a', 4: 'b'}
25: {1: 'a', 2: 'b', 3: 'b', 4: 'c'}
26: {1: 'a', 2: 'b', 3: 'c', 4: 'b'}
27: {1: 'a', 2: 'b', 3: 'c', 4: 'c'}
28: {1: 'a', 2: 'b', 3: 'c', 4: 'a'}
29: {1: 'a', 2: 'b', 3: 'a', 4: 'c'}
30: {1: 'a', 2: 'c', 3: 'b', 4: 'b'}
31: {1: 'a', 2: 'c', 3: 'b', 4: 'c'}
32: {1: 'a', 2: 'c', 3: 'b', 4: 'a'}
33: {1: 'a', 2: 'c', 3: 'c', 4: 'b'}
34: {1: 'a', 2: 'c', 3: 'a', 4: 'b'}
35: {1: 'a', 2: 'a', 3: 'b', 4: 'c'}
36: {1: 'a', 2: 'a', 3: 'c', 4: 'b'}
---------------------------------
```

Custom

S = {1,2,3}

T = {A, B}



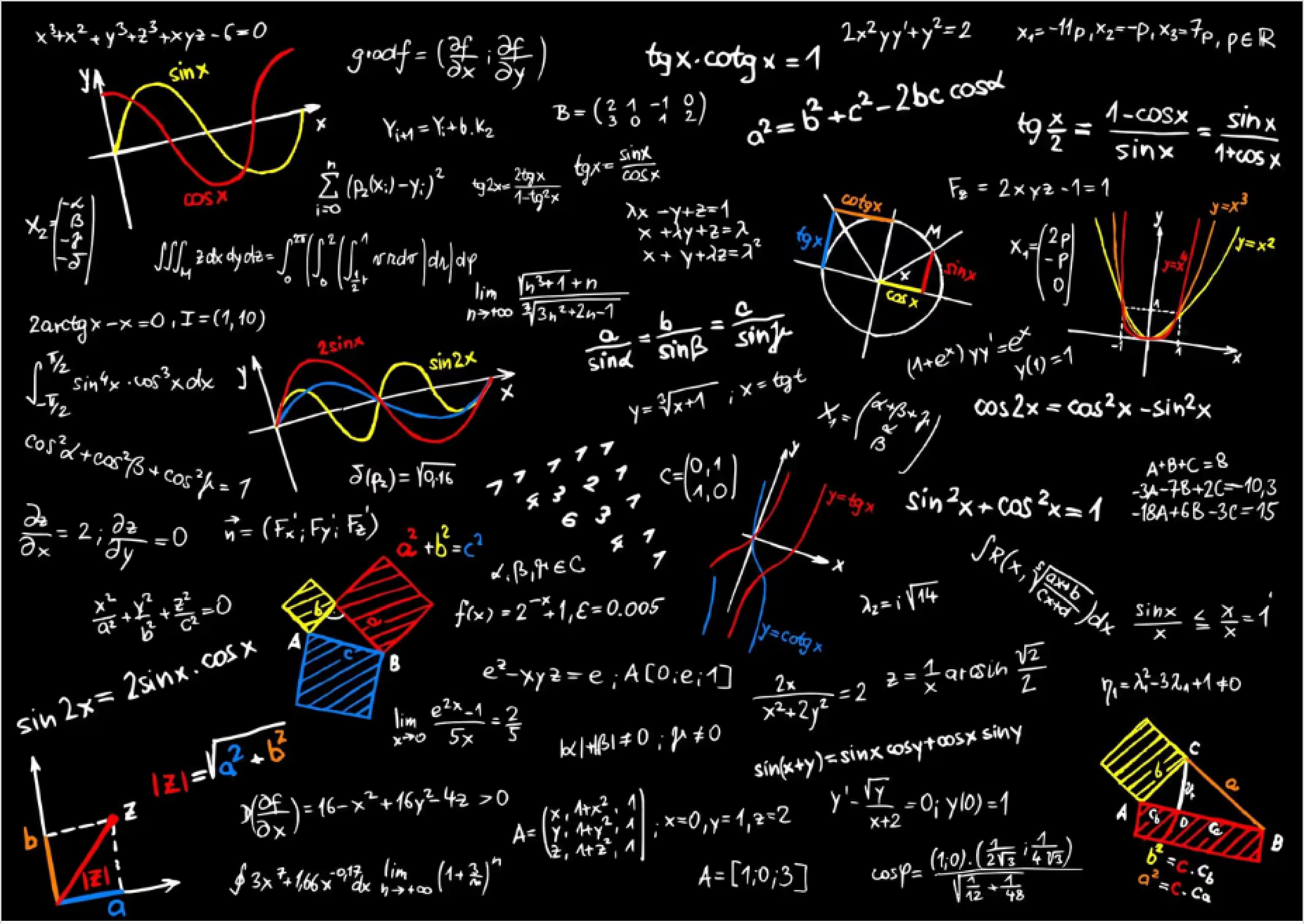
```
Try with your own sets of integers!

Enter a set A of integers separated by spaces:
1 2 3
Enter a set B of characters separated by spaces:
a b

Number of onto functions from A to B: 6
All of the onto functions from A to B:

1: {1: 'b', 2: 'b', 3: 'a'}
2: {1: 'b', 2: 'a', 3: 'b'}
3: {1: 'b', 2: 'a', 3: 'a'}
4: {1: 'a', 2: 'b', 3: 'b'}
5: {1: 'a', 2: 'b', 3: 'a'}
6: {1: 'a', 2: 'a', 3: 'b'}
```

CODEBASE LINK



Hardvan/DMS-EL-Onto-Functions

Contribute to Hardvan/DMS-EL-Onto-Functions development by creating an account on GitHub.

GitHub

LINK TO THE CODE FILE

THANK YOU