Function designs for interoperability:

TASK 1: Function `matrix_split`

The matrix split function shall be able to split a given matrix into pieces, considering a maximal size of a part.

Prototype

```
def matrix_split(matrix: list, max_size: tuple) -> dict:
  pass
```

The `maxtrix` is a 2-Dimensional List with the pattern:

Example matrix:

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

The python's matrix representation is:

```
matrix = [[1,2,3],[4,5,6][7,8,9]]

# Accessing the second element in the first row would be:
e2_r1 = matrix[1][0]
```

The limitation parameter `max_size` has the following syntax:

```
max_size = (1,2) # Means that the maximal matrix part size is 1 row and 2
columns
```

Splitting the matrix with this parameter shall result the following:

Part 1,1:

$$\begin{matrix} 1 & 2 \end{matrix}$$

Part 1,2:

$$\begin{matrix} 3 \end{matrix}$$

Part 2,1:

$$\begin{matrix} 4 & 5 \end{matrix}$$

Part 2,2:

$$6$$

Part 3,1:

$$7 \quad 8$$

Part 3,2:

$$9$$

This should lead to the following result in python:

```
result: dict = {
  (1,1): [1,2],
  (1,2): [3],
  (2,1): [4,5],
  (2,2): [6],
  (3,1): [7,8],
  (3,2): [9]
}
```

As you can see in the example, the parts are stored in a dictionary using the keys of the parts as a (row, col) tuple.

> Your task is to implement a function, that can do this splitting of the matrix with any matrix dimensions, and return the results correctly. In case of invalid Input, the appropriate exceptions has to be raised.

TASK 2: Function `matrix_collapse`

The matrix collapse function is the inverse of the `matrix_split`. It should receive any *dict* as an input, and reconstruct the matrix using the elements.

Prototype

```
def matrix_collapse(splitted_matrix: dict) -> list:
  pass
```

> The function should work for any valid input dictionary and should return a list object.

Validating and Testing

1. Implement unit tests for every important case of the inputs.

    1. Represent corner-case (critical) data samples
    2. Represent random data samples
    3. Represent invalid data samples
2. Test reversibility
    1. Using the result of the `matrix_split` the same matrix has to be reconstructed via `matrix_collapse`

```
# Reversibility check

matrix: list = [[1,2,3],[4,5,6][7,8,9]]
splitted_matrix: dict = matrix_split(matrix, (2,2)) # Or any other split
condition
reconstructed_matrix: list = matrix_collapse(splitted_matrix)

# Requirement
self.assertTrue(matrix == reconstructed_matrix)
```