**I created the following modules for a homework assignment:**

## reverse\_number\_in\_list(number\_list:list)-> list

**This function will be given a list of numbers your job is to reverse all the numbers in the list and return a list with the reversed numbers. If a number ends with 0 you need to remove all the trailing zeros before reversing the number. An example of reversing numbers with trailing zeros: 10 -> 1, 590 -> 95. None of the numbers in the number\_list will be less than 1.**

**Example:**

**number\_list = [13, 45, 690, 57]**

**output = [31, 54, 96, 75]**

## tails\_same(number\_list:list) -> bool

**This function should return true if the value at the beginning and the end of the list are equal. False otherwise.**

**Example:**

**number\_list = [1, 239, 949, 0, 84, 0, 1]**

**output: True**

**number\_list = [1, 239, 949, 0, 84, 0, 13]**

**output: False**

## remove\_char(str\_list:list, char:str) -> list

**This function will be given a list of strings and a character. You must remove all occurrences of the character from each string in the list. The function should return the list of strings with the character removed.**

**Example:**

**str\_list = ['adndj', 'adjdlaa', 'aa', 'djoe']**

**char: a**

**output = ['dndj', 'djdl', '', 'djoe']**

## return\_growing\_num\_list(max:int) -> list

**This function will be given a single number, it should return a list of strings of numbers. Each string in the list will only contain a single number repeated an arbitrary amount of times. The number each string will contain will be equal to the current string's index+1. The number in the string should be repeated the same number of times as the string's index+1. Each number in the string should be separated by a space. This list should stop when its size equals the max number specified.**

**Example:**

**max = 3**

**output = ['1', '2 2', '3 3 3']**

**max = 4**

**output = ['1', '2 2', '3 3 3', '4 4 4 4']**

## find\_color(colors:list, values:list) -> list

**The function will have two parameters. The first parameter is a set of strings known as Colors. A second parameter is a list of tuple-2 known as Values. Colors will contain a set of randomly selected colors. Values will contain a list of tuples of size 2. Each tuple will contain color (str) and a number (int). The function should look at each tuple in Values. For each tuple, add the number (the second value in the tuple) to a list of numbers if the color in the tuple (the first value in the tuple) is in Colors. In other words, find all tuples that have a color in the Colors and add the tuples numbers to a list. Finally, the function should return the list of numbers collected in the order they are found in the values list.**

**Example:**

**Colors: {'black', 'pink', 'yellow'}**

**List: [('green', 100), ('yellow', 13), ('red', 6)]**

**Expected: [13]**

**Colors: {'yellow'}**

**List: [('black', 54), ('pink', 5)]**

**Expected: []**

**Colors: {'black', 'blue', 'yellow'}**

**List: [('yellow', 29), ('yellow', 19), ('black', 31), ('yellow', 67), ('green', 44)]**

**Expected: [29, 19, 31, 67]**

## dict\_contains\_keys(items:set, example\_dict:dict)->bool

## 

**This function will have two parameters. The first is a set of numbers known as Number Set. The second is a dictionary known as Dictionary. Dictionary will have keys as integers and values as letters. The functions should return true if at least one of the numbers in the Number set is a key in Dictionary. It should return false otherwise.**

**Example:**

**Items: {8, 10, 5}**

**Example: {9: 'F', 10: 'X'}**

**Expected: True**

**Items: {8, 9, 5, 1}**

**Example: {6: 'i', 5: 'Y', 1: 'N', 0: 'E'}**

**Expected: True**

**Items: {9, 10, 4}**

**Example: {5: 'i', 3: 'o', 1: 'N'}**

**Expected: False**

## concatenate\_dict(dict\_list:list)->dict

**This function will be given a single parameter known as the Dictionary List. Your job is to combine all the dictionaries found in the dictionary list into a single dictionary and return it. There are two rules for adding values to the dictionary:**

1. **You must add key-value pairs to the dictionary in the same order they are found in the Dictionary List.**
2. **If the key already exists, it cannot be overwritten. In other words, if two or more dictionaries have the same key, the key to be added cannot be overwritten by the subsequent dictionaries.**

**Example:**

**Dictionary List: [{'Z': 6, 'k': 10, 'w': 3, 'I': 8, 'Y': 5}, {'Y': 1, 'Z': 4}, {'X': 2, 'L': 5}]**

**Expected: {'Z': 6, 'k': 10, 'w': 3, 'I': 8, 'Y': 5, 'X': 2, 'L': 5}**

**Dictionary List: [{'z': 0}, {'z': 7}]**

**Expected: {'z': 0}**

**Dictionary List: [{'b': 7, 'b': 10, 'A': 8, 'Z': 2, 'V': 1}]**

**Expected: {'b': 7, 'A': 8, 'Z': 2, 'V': 1}**

## unique\_values(a\_dict:dict)-> dict

**This function will receive a single dictionary parameter known as a\_dict. a\_dict will contain a single letter as a string and numbers as values. This function is supposed to search a\_dict to find all values that appear only once in the a\_dict. The function will create another dictionary named to\_ret. For all values that appear once in a\_dict the function will add the value as a key in to\_ret and set the value of the key to the key of the value in a\_dict (swap the key-value pairs, since a\_dict contains letters to numbers, to\_ret will contain Numbers to Letters). Finally, the function should return to\_ret.**

**Example:**

**a\_dict: {'X': 2, 'Y': 5, 'N': 2, 'L': 2, 'W': 1, 'G': 0, 'R': 1}**

**Expected: {5: 'Y', 0: 'G'}**

**a\_dict: {'Z': 3, 'P': 3, 'E': 2, 'G': 0, 'T': 5, 'L': 1, 'Q': 0}**

**to\_ret: {2: 'E', 5: 'T', 1: 'L'}**

**a\_dict: {'E': 3, 'X': 3}**

**to\_ret: {}**

**a\_dict: {'G': 3, 'D': 3, 'C': 4, 'Q': 1, 'H': 1, 'M': 2, 'Z': 1, 'W': 3}**

**to\_ret: {4: 'C', 2: 'M'}**

**a\_dict: {'O': 2, 'T': 1, 'L': 5, 'W': 5, 'Z': 4, 'M': 5, 'B': 4, 'D': 0, 'F': 3, 'E': 1}**

**to\_ret: {2: 'O', 0: 'D', 3: 'F'}**

## dict\_from\_string(dict\_str:str)->dict

**This function will be given a single parameter, a string representing a dictionary. Your job is to convert the string into an actual dictionary and return the dictionary. Make sure all key-value pairs in the string exist in the newly created dictionary. The string will contain only numbers or single letters as key values pairs. Make sure all letters are kept as strings and all numbers are converted to integers in the newly created dictionary.**

**Example:**

**String Input: '{9: 'V', 'G': 0, 'M': 9, 'u': 3, 2: 'o', 8: 'u', 'q': 9, 'D': 1}'**

**Expected: {9: 'V', 'G': 0, 'M': 9, 'u': 3, 2: 'o', 8: 'u', 'q': 9, 'D': 1}**

**String Input: '{10: 'D', 1: 'Z', 5: 'a'}'**

**Expected: {10: 'D', 1: 'Z', 5: 'a'}**

**String Input: '{'M': 2, 'V': 0, 3: 'x', 6: 'J', 5: 'J', 7: 'T', 8: 'P', 4: 'q', 1: 'h'}'**

**Expected: {'M': 2, 'V': 0, 3: 'x', 6: 'J', 5: 'J', 7: 'T', 8: 'P', 4: 'q', 1: 'h'}**

**String Input: '{3: 'D', 10: 'T', 7: 'm', 'u': 9, 't': 5, 6: 'Z', 'H': 10, 'B': 6}'**

**Expected: {3: 'D', 10: 'T', 7: 'm', 'u': 9, 't': 5, 6: 'Z', 'H': 10, 'B': 6}**

**String Input: '{'q': 10, 6: 'O', 'm': 6}'**

**Expected: {'q': 10, 6: 'O', 'm': 6}**

## flip\_matrix(mat:list)->list

**You will be given a single parameter a 2D list (A list with lists within it) this will look like a 2D matrix when printed out, see examples below. Your job is to flip the matrix on its horizontal axis. In other words, flip the matrix horizontally so that the bottom is at top and the top is at the bottom. Return the flipped matrix.**

**To print the matrix to the console:**

**print('\n'.join([''.join(['{:4}'.format(item) for item in row]) for row in mat]))**

**Example:**

**Matrix:**

**W R I T X**

**H D R L G**

**L K F M V**

**G I S T C**

**W N M N F**

**Expected:**

**W N M N F**

**G I S T C**

**L K F M V**

**H D R L G**

**W R I T X**

**Matrix:**

**L C**

**S P**

**Expected:**

**S P**

**L C**

**Matrix:**

**A D J**

**A Q H**

**J C I**

**Expected:**

**J C I**

**A Q H**

**A D J**