Week 5 Dictionaries

Part 1 Dictionaries

In this exercise, try to come up with the answer without using IDLE/Python first. If there is an error, specify the cause and type of the error. Then type the expressions into IDLE to verify your answers. The objective is for you to understand why and how they work.

Expressions	**Output**

`a = (("A",2), ("B",3), (1,4))`
`dict_a = dict(a)`
`print(dict_a)`
`print(dict_a[2])`
 	
`b = [[1,"A"], [(2,3),4]]`
`dict_b = dict(b)`
`print(dict_b)`
`print(dict_b[(2,3)])`	

`for key in dict_b.keys():`
`  print(key)`
`for val in dict_b.values():`
`  print(val)`
`for k,v in dict_b.items():`
`  print(k, v)`	

`del dict_b[(2, 3)]`
`print(dict_b)`
`del dict_b[2]`
`print(dict_b)`	

`print(tuple(dict_a.keys()))` `print(list(dict_a.values()))`	

`dict_c = {1: {2: 3}, 4: 5}`
`dict_d = dict_c.copy()`
 	
`dict_d[4] = 9`
`dict_d[1][2] = 9`
`print(dict_d)`
`print(dict_c)`	

`del dict_c` `print(dict_c)`	

Part 2 Anagram

An anagram is a word or phrase formed by rearranging the letters of an original word or phrase, typically using all the letters of the original word or phrase exactly once. For example, "nag a ram" is an anagram for "anagram" and "eleven plus two" is an anagram for "twelve plus one".

Write a function `is_anagram(word1, word2)` that returns `True` if `word1` and `word2` are anagrams of each other, otherwise return `False`.

Part 3 T9

Old mobile phones have numerical keypads where every character is associated with exactly one number. For instance, for the keypad on the right, "a" is associated with the number 2 and " " is associated with the number 0.

We can represent a keypad in two different ways:

1. A ***list*** where each element is a string of characters that are all associated with the number which is the element's index
   `keyL = [" ", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"]`

2. A ***dictionary*** where the keys are the characters and the values are their associated numbers
   `keyD = {"a": 2, "b": 2, ..., "z": 9, " ": 0}`

Suppose there are other keypads which can support any printable character in the ASCII table, associated with any number that is a nonnegative integer.

A. Write a function `to_dict(keyL)` which takes in some keypad as a list and returns it as a dictionary.
B. Write a function `to_list(keyD)` which takes in some keypad as a dictionary and returns it as a list.

Since typing text was tedious with old mobile phones containing the above keypad, the T9 system was created as a predictive text technology for such mobile phones. The system works as follows:

- Every phrase is composed of letters.
- Every letter can be mapped to a number.
- Given a sequence of keypresses (numbers), we can predict the desired phrase.

For tasks C and D, you may assume that both `keyL` and `keyD` are already created and initialized; the keypad is specified.

C. Write a function `to_nums(phrase)` that takes in an input string and returns an integer representing the sequence of keypresses (numbers) that can be predicted as the desired phrase.
   For instance, `to_nums("i luv u")` returns `4058808` based on the above keypad.
D. Write a function `to_letters(num)` that takes in a nonnegative number and returns a list of all possible phrases that can be predicted. The phrases may appear in any order in the output list.