**Week 11 Additional Basic Exercises (dictionaries)**

**Q1: [\*]**

**Part (a)**

Define a function called get\_square\_dict() that takes in a positive integer n as its parameter. The function returns a dictionary that contains all the mappings between an integer in the range of 1 to n and its square.

* Example #1:

get\_square\_dict(4) returns {1:1, 2:4, 3:9, 4:16}

* Example #2:

get\_square\_dict(8) returns {1:1, 2:4, 3:9, 4:16, 5:25, 6:36, 7:49, 8:64}

**Part (b)**

Define a function called get\_word\_length\_dict(). This function takes in a list of strings as its parameter. You can assume that all the strings in the list are unique. The function returns a dictionary that contains mappings from each word in the list to its length.

* Example:

get\_word\_length\_dict(["IS111", "coding", "python", "lab"]) returns {"IS111":5, "coding":6, "python":6, "lab":3}

If the list passed to the function is empty, the function returns an empty dictionary.

**Part (c)**

Define a function called get\_values(). This function takes in a dictionary and a list of strings as its parameters. The dictionary stores a mapping from strings to numbers. The list contains strings. The function takes each string in the list, treats it as a key, and looks for its corresponding value in the dictionary. The function returns a list of numbers that contains the values found.

* Example #1:

get\_values({"apple":1, "orange":4, "peach":16, "mango":12, "grape":9}, ["orange", "mango", "apple", "mango"]) returns [4, 12, 1, 12]

* Example #2: get\_values({"Sunday":1.5, "Monday":4.2, "Tuesday":16.1, "Wednesday":12.8}, ["Monday", "Tuesday", "Sunday"]) returns [4.2, 16.1, 1.5]

Can you improve your function such that if a string in the given list is not a key in the given dictionary, the special value None is to be used in the returned list?

* Example: get\_values( {"apple":1, "orange":4, "peach":16, "mango":12, "grape":9}, ["orange", "mango", "banana", "cherry"]) returns [4, 12, None, None]

**Part (d)**

Define a function called add\_key\_value(). The function takes in a dictionary and two strings as its parameters. The function treats the first string as a key and the second string as a value, and tries to add this (key, value) pair to the given dictionary. However, if the key already exists in the dictionary, the function does NOT replace the old value with the new value. The function returns True if the (key, value) pair is added, or False if the key already exists in the dictionary (and hence the (key, value) pair is not added).

* Example #1:

If my\_dict is initially {"a":"apple", "o":"orange", "p":"peach"}, calling

add\_key\_value(my\_dict, "b", "banana") returns True, and my\_dict becomes {"a":"apple", "o":"orange", "p":"peach", "b":"banana"}

* Example #2:

If my\_dict is initially {"a":"apple", "o":"orange", "p":"peach"}, calling

add\_key\_value(my\_dict, "p", "pear") returns False, and my\_dict remains the same.