墨学 FIT9133 Final Review Practices

1. Write a function called checkName, let the user put an input as his name and age, then check the name's length is between 5-20, the age must be an integer. If the validation is not reached, the user should input the information again.

2. Write function to return the longest string in a list of strings

3. Write function to return the index of the longest string in a list of strings

4. Write a function longer_than which consumes a list of strings los and a non-negative integer n, and counts the number of strings in los with more than n characters in them.

5. Write a function that consumes a list of lists of integers, where each element list has the same length.

6. Suppose you are searching in the list L = [2,5,10,15,22,29].
How many iterations are required when searching for the following values in L
(a) When using Linear Search?
(b) When using Binary Search?
(i) 2
(ii) 15
(iii) 29
(iv) 0
(v) 30

6.write a Python function count_chars that consumes chars, a list of strings and returns the total number of characters in chars.

7. write the function duplicate_string that consumes a string s and a natural number n, and returns the new string containing n copies of s. Do not use the * operation.
Please add validation -> to check whether the second parameter is an integer.

For example, duplicate_string("abc", 3) => "abcabcabc"

8. write the function count_upper_case that consumes a string and returns the number of uppercase characters in the string.
For example, count_upper_case("Good Morning!") => 2

9. write the function count_case that consumes a list and returns a **dictionary** that contains the number of uppercase characters, lowercase characters, numbers, other characters in the list.
For example, count_case("abc123ABC!!!") => {“upper”:3, “lower”:3, “number”:3, “other”:3}

10. write the function spread that consumes a list of numbers, and returns the difference between the largest and smallest values in the list.
For example, spread([3,1,9,17,-4,2]) => 21,
spread([2]) => 0, spread([]) => 0.

11. write a function majority that consumes a list of booleans and determines if there are more True than False values in the list.
For example, majority([True, False, False]) => False,
majority([False, True, False, True]) => False,
majority([False, True, True, True, True]) => True

12. Write a function combine_neighbour_strings that consumes a list of strings and returns a new list with pairs of adjacent strings combined.
For example,
combine_neighbour_strings(["abc", "123", "def", "456"])
=> ["abc123", "def456"]
combine_neighbour_strings(["abc", "123", "def", "456", "q"])
=> ["abc123", "def456", "q"]

13.Write a generative solution is_palindrome that consumes a string and returns true if that string is a palindrome when spaces are ignored, and false otherwise.
A palindrome is a string that is the same forwards and backwards, for example, "hannah" and "a man a plan a canal panama" are palindromes, as are "" and "a".

14 Write a function remove_whitespace that consumes a string s and returns a new string with all "useless" whitespace removed:
- no leading whitespace
- no trailing whitespace
- only one space between words
examples:
remove-whitespace (" bears!! bears!") => "bears!! bears!"
remove-whitespace ("bears!! bears! ") => "bears!! bears!"
remove-whitespace ("bears!! bears!") => "bears!! bears!"
remove-whitespace (" ") => ""
remove-whitespace (" bears!! bears! ") => "bears!! bears!"
Note that the string strip method only removes extra whitespace from the beginning and end of a string, not "inside".

15.Complete the Python function common that consumes two dictionaries (d1 and d2, with common key and value types).
Return a list of all the keys (in increasing order) that occur in both d1 and d2 and have the same associated values in each.
For example, common({1:'1', 2:'2', 4:'4'}, {3:'3', 2:'2', 1:'1', 4:'four'}) => [1,2]

16. Definition of Movie class - note all the pieces (the class definition includes a function to complete as well)

```
class Movie:
    '''fields: title (Str),
               year (Nat),
               stars (listof Str)
    requires: year is the year the movie was released (> 1900)
              stars includes the names of the actors in the movie'''

    def __init__(self, name, date, actors):
```

```
    def __str__(self, name, date, actors):


    def __eq__(self, other):


    def add_star(self, star):
```

Sample movies

```
princess_bride = Movie("Princess Bride", 1987,
                               ["Cary Elwes", "Robin Wright", "Mandy Patinkin"])

singin_in_the_rain = Movie("Singin' in the Rain", 1952,
                           ["Gene Kelly", "Debbie Reynolds", "DonaldO'Connor"])

dirty_dancing = Movie("Dirty Dancing", 1987,
                               ["Patrick Swayze", "Jennifer Grey"])

flicks = [princess_bride, singin_in_the_rain, dirty_dancing]
```

Complete the class function add_star that consumes a Movie object and a string, star, and add the star's name to the end of the movie's stars list. In addition, the function returns the number of actors listed under stars.

Complete the function common_stars that consumes two movies and returns True if there is any actor who is in both movies, and False otherwise.

For example,

```
common_stars(Movie("Princess Bride", 1987,
                               ["Cary Elwes", "Robin Wright", "Mandy Patinkin"]),

                     Movie("Alien Nation", 1988, ["James Caan", "Mandy Patinkin"]))
=> True
```

17 A classlist is (list title students), where title is a nonempty string giving the course title, and students is a list of id numbers of students enrolled in
Write a function enrolled, that consumes a (listof classlist) and

returns a dictionary where

* the keys are the student id numbers, and

* the associated value for each key is the list of course titles the student is enrolled in.

For example,
enrolled([["FIT9133", [111, 222, 145, 345]],
["FIT9132", [145, 444, 222]],
["FIT9131", [333, 555, 111, 222]]])
=> { 111:["FIT9133", "FIT9131"], 222: ["FIT9133", "FIT9131","FIT9132"], 145: ["FIT9133", "FIT9132"],
345:["FIT9133"], 444: ["FIT9132"], 333:["FIT9131"], 555:["FIT9131"] }

```
def enrolled(all_classes):
    pass
```