CSc 110 Assignment 6:
Lists

Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee.

Learning Outcomes:

When you have completed this assignment, you should understand:

- How to define and call functions that take lists as arguments and create and return lists

Getting started

1. Create a new file called `assignment6.py` and open it in your Wing editor
2. Design the functions according to the specifications in the Function Specifications section.

Submission

1. Double check your file before submission to avoid a **zero grade** for your submission for issues described in the Grading section of this document:
    1. Open and **run `assignment6.py`** in your Wing editor using the green arrow.
       You should see no errors and no output after the shell prompt.
       If there are errors, you must fix them before submitting.
       If there is output printed to the shell, find and remove any top-level print statements and/or top level function calls.
    2. At the shell prompt in Wing (>>>) type the following: **`import assignment6`**
       You should see no errors and no output after the shell prompt.
       If there are errors, you must fix them before submitting.
       If there is output printed to the shell, find and remove any top-level print statements and/or top level function calls.
    3. At the shell prompt in Wing (>>>), make calls to the required functions to ensure you have named them correctly. Ensure the function is exhibiting the expected behaviour when called and does not contain any unexpected output. You should be able to make calls to the following functions with expected input. Note: When making these calls, you must precede it with the module name ie. `assignment6.multiply_by([1, 2, 3], 14)`
        i. multiply_by
        ii. remove_multiples
        iii. remove_ends_with
        iv. get_index_of_largest
        v. does_contain_proper_divisor
        vi. are_all_less_than_threshold
2. Upload your **`assignment6.py`** containing the completed function designs to BrightSpace

Grading:

- Late submissions will be given a **zero grade**.
- The files you submit must be named **`assignment6.py`**
  The filenames must be EXACT for them to work with our grading scripts. Errors in the filenames will result in a **zero grade**.
  Example mistakes often made by students include but are not limited to: spelling errors, different case letters, space characters and incorrect extension (not using .py)
- Your function names and the order of your function arguments must match EXACTLY as specified in this document or you will be given a **zero grade**.
- Your submission must not contain any print statements that are not required in the specification or any top-level calls to functions. This unexpected code can cause the automated tester to crash and will result in a **zero grade**.
- We will do **spot-check grading** in this course. That is, all submissions are graded BUT only a subset of your code might be graded. You will not know which portions of the code will be graded, so all of your code must be complete and adhere to specifications to receive marks.
- Your code must run without errors with Python 3.9. If you tested your configuration with setup.py file this would have verified you are using Python 3.9. Code that generates errors cannot be tested and will be given a **zero grade**.

Marks will be awarded for correctness, considering:

- the function signature matches the description given (has the name and arguments EXACTLY as specified)
- the function has the expected behaviour

and for code quality according to software engineering properties such as:

- documentation in docstring format: type hints, purpose, examples
- Test coverage – examples within the function docstring cover all boundary cases of all conditions within your function
- readability
  - use of whitespace
  - splitting complex computation or long statements across multiple lines
- meaningful variable names
  - lower case, starting with an alpha-character
- proper use of constants (avoid magic numbers)
  - defined above all function definitions
  - in UPPERCASE
- use of code constructs (functions, variables, conditions) to:
  - eliminate redundant code and redundant computation
  - make complex code easier to read

Function Specifications

You cannot use list comprehensions to solve any of these problems. If you don't know what these are, don't worry about it, you don't need to know about them for this course.

1. Design the function `multiply_by` that takes a list as an argument and an additional integer argument as a multiplier. The function should **create and return a new list** that contains every element in the given list multiplied by the given multiplier. This function should assume that the given list will contain values that can be multiplied by an integer.
   This function should **not** update the list passed as an argument.

2. Design the function `remove_multiples` that takes a list of integers as an argument and an additional integer argument. The function should create and return a new list that contains only the values in the given list that are not multiples of the additional integer argument.
   Think about how you might update your `is_multiple` function you wrote in Assignment 2 and add it as a helper function to this assignment file. Make sure to document and test it though!
   This function should **not** update the list passed as an argument.

3. Design the function `remove_ends_with` that takes a list of strings as an argument and an additional string argument to match. The function should create and return a new list that contains every element in the given list that DOES NOT end with the given string to match. The function should ignore case but add the strings to the new list without changing them.
   You cannot use the Python string `endswith` function – but you can write your own function that does something similar!
   This function should **not** update the list passed as an argument.

   Examples:
   if the list is ['bat', 'ratchet', 'BCAT', 'at', 'atlas'] and the additional string argument to match is 'at' then the function should return ['ratchet', 'atlas']

   if the list is ['bat', 'ratchet', 'BCAT', 'at', 'atlas'] and the additional string argument to match is 'AT' then the function should return ['ratchet', 'atlas']

4. Design the function `get_index_of_largest` that takes a list of numbers as an argument. The function should find and return the index of the largest value in the list. The function should assume the given list has at least one element. If the list has 2 occurrences of the largest value, the function should return the index of the last occurrence.
   NOTE: You WILL NOT get marks for a solution that uses Python's `min`, `max` or `sort` functions.
   This function should **not** update the list passed as an argument.

5. Design the function `does_contain_proper_divisor` that takes a list of integers as an argument and an additional integer argument. The function should **determine whether** the given list contains any values that are a proper divisor of the additional argument.
   You are free to copy your `is_proper_divisor` function from your previous assignment – make sure it is correct though!
   This function should **not** update the list passed as an argument.

   What is the BEST return type for this function?
   What should this function return if the list is empty? If the list is empty does is contain any proper divisors?

6. Design the function `are_all_less_than_threshold` that takes a list of integers as an argument and an additional integer argument as a threshold. The function should **determine whether** the given list contains only values that are less than the given threshold.
   This function should **not** update the list passed as an argument.

   What is the BEST return type for this function?
   What should this function return if the list is empty? If the list is empty are there any values that are not less than the threshold?