Lab 14: Lists + Mutability in Python

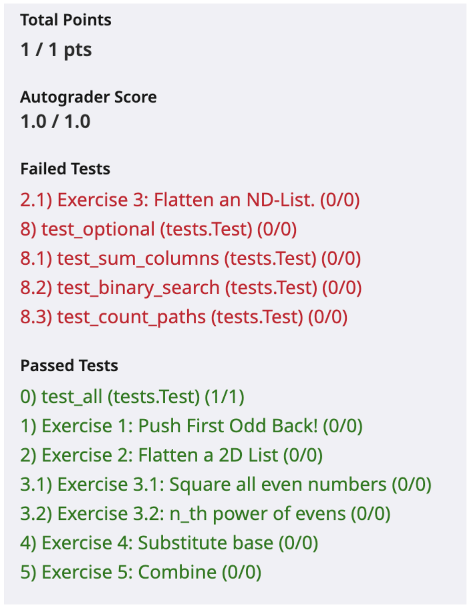
Instructions:

This worksheet serves as a guide and set of instructions to complete the lab.

- You must use the starter file, found here, to get credit for the lab.
- Additionally, here is the workbook that you can read through for further context and additional (non-required) material.
- All material was sourced from the CS10 version of The Beauty and Joy of Computing course.

Submitting:

You will need to complete **Exercises 1-5** and submit this to Gradescope.

- To receive full credit, you will need to complete the required functions, and the required functions must pass all tests from the autograder in Gradescope.
  - Note that optional exercises are included in the Gradescope autograder but do not count towards credit. See below for example.
  - 

    Total Points
    1 / 1 pts

    Autograder Score
    1.0 / 1.0

    Failed Tests
    2.1) Exercise 3: Flatten an ND-List. (0/0)
    8) test_optional (tests.Test) (0/0)
    8.1) test_sum_columns (tests.Test) (0/0)
    8.2) test_binary_search (tests.Test) (0/0)
    8.3) test_count_paths (tests.Test) (0/0)

    Passed Tests
    0) test_all (tests.Test) (1/1)
    1) Exercise 1: Push First Odd Back! (0/0)
    2) Exercise 2: Flatten a 2D List (0/0)
    3.1) Exercise 3.1: Square all even numbers (0/0)
    3.2) Exercise 3.2: n_th power of evens (0/0)
    4) Exercise 4: Substitute base (0/0)
    5) Exercise 5: Combine (0/0)
- Note that you only have *6 functions* to complete for full credit on Lab 14
  - Other functions on file are optional
- For instructions on how to submit to labs to Gradescope, please see this document.

Please note, you must use the starter file, and you must NOT edit the name of any of the required functions. Failing to do either for these will result in the autograder failing.

Objectives:

Programmers use Data Structures to help organize data in ways which are easy to access, understand, and manipulate. Lists are a common Data Structure that you have used before in Snap. Let's take what you have learned about lists and nested lists in Snap, and translate that into Python. In today's lab you will:

- Practice using data structures to access, understand, and manipulate data
- Practice writing functions that interact with different data structures in Python

Required Functions:

- Function 1: push_first_odd_back(lst)
- Function 2: flatten(lst)
- Function 3: squares_of_evens(lst)
- Function 4: nth_power_of_evens(lst, n)
- Function 5: substitute_base(string, old, new)
- Function 6: combine(items)

Important Topics mentioned in the Workbook:
For better understanding of the lab we highly recommend going through these workbook pages! Topics that are important but not required for this lab will be indicated with an asterisk**. These topics are best reviewed in order and as you complete the lab.

- Basic List Operations
- List Mutability
- List Functions
- Iterables

Exercise 1: push_first_odd_back(lst)

- Objective:
    - Create a function that places the first odd number found at the back of the input list
- Notes
    - The pop() function may be useful here
    - We are not returning anything, but rather modifying our input list
- Inputs:
    - lst = List of numbers
- Output:
    - Reports: nothing
    - *This function shouldn't return anything*
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Exercise 2: flatten(lst)

- Objective:
    - Create a function that takes a list of lists (2D) and returns all the items of each list concatenated together into one new list (1D)
- Notes
    - The append() function could be useful here
- Inputs:
    - lst = List of lists
- Output:
    - Reports: list
    - Your output list should be a 1-dimensional version of the input list
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Exercise 3.1: squares_of_evens(lst)

- Objective:
    - Create a function that takes in a list of numbers, and returns a list that contains the squares of all even numbers from the input list
- Notes
    - ***You must use list comprehension***
        - Not doing so will result in a failing autograder
- Inputs:
    - lst = List of numbers
- Output:
    - Reports: List
    - Your output should be a list containing the squares of all even numbers found from the input list
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Exercise 3.2: nth_power_of_evens(lst, n)

- Objective:

- Create a function that takes an input integer n as well as a list of numbers, and returns a list that contains the nth power of all even numbers found in the input list
- Notes
  - ***You must use list comprehension***
    - Not doing so will result in a failing autograder
- Inputs:
  - lst = List of numbers
  - n = some integer >= 0
- Output:
  - Reports: List
  - Your output should be a list containing all even numbers found from the input list to the nth power
- Examples:
  - Doctests available
    - python3 -m doctest <labfilename>.py to run autograder
      - Must be in correct parent file
  - **Requires flatten to pass local doctests!**

Exercise 4: substitute_base (string, old, new)

- Objective:
  - Create a function that takes in a string and replaces all characters that match (old) with the the character (new)
- Notes
  - Use of the replace() function is *prohibited*
  - Use list comprehension along with "".join for a clean/efficient solution
- Inputs:
  - string = some string
  - old = the character to be replaced. A single character
  - new = the character(s) to replace the 'old' character with
- Output:
  - Reports: string
  - Your output should be a string with the proper characters substituted
- Examples:
  - Doctests available
    - python3 -m doctest <labfilename>.py to run autograder
      - Must be in correct parent file

Exercise 5: combine(items)

- Objective:
  - Recreate your own version of combine in Python that takes in a list of numbers OR strings and returns the combined result.

- Notes
    - Use reduce from the functools library is *prohibited*
    - Think about what the + operator does for strings and for numbers.
    - For the most elegant solution, use recursion
        - The rough equivalent of "all but first of" in Python is [1:]
- Inputs:
    - items = List of numbers OR strings
- Output:
    - Reports: integer, float, or string
    - Your output should be the combined result of the items
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

You can always check the validity of your solutions by using the local autograder.
Remember to submit on Gradescope!