Lab 11: map, filter, reduce

INFSCI 0201
Intermediate Programming (Spring 2025)

In this week's lab, your knowledge of functional programming in Python will be tested. There are a total of three tasks that test your understanding of map(), filter(), and reduce().

Submission

You are to write a complete Python program that meets the requirements outlined in the Lab 11 Tasks section.

- As with all programming assignments in this course, you must submit Lab 11 through GitHub in the git repository the instructor created for you. You must place your Lab 10 eclipse project in the folder **/labs/lab11/**
- The code for each task should be in its own separate Python file.

Testing

The main test for this lab is slightly different from earlier labs.

- First, your programs should be able to run without any issues and generate the expected output.
- Second, think about doing the task adhering to the functional programming paradigm.

Note: Style guide violations will count as test failures! Be sure you're strictly adhering to the Python coding style guide on Canvas.

Lab 11 Tasks

Task 1: **`zipmap(key_list: list, value_list: list, override = False)`**

In this task, you need to create a function `zipmap()` that takes two sequences and creates a dictionary from the elements of the first sequence to the elements of the second.
The `zipmap()` function should also have a third parameter named override, which has a default value of False. If the override parameter is True, in the case of the key_list having duplicating members, the later cases will override the earlier result. If the override parameter is False, the function will not produce any value if there are duplicates in the key_list.
If the key_list is shorter than the value_list, the function will discard the leftover members in the value_list.
If the key_list is longer than the value_list, the function will use None as a stand-in for the missing values.

Note: You should think of using the map() function to help you achieve this.

Example:

```
list_1: [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’]
list_2: [1, 2, 3, 4, 5, 6]
Result: {‘a’: 1, ‘b’: 2, ‘c’: 3, ‘d’: 4, ‘e’: 5, ‘f’: 6}

zipmap([1, 2, 3, 2], [4, 5, 6, 7], True) => {1: 4, 2: 7, 3: 6}

zipmap([1, 2, 3], [4, 5, 6, 7, 8]) => {1: 4, 2: 5, 3: 6}

zipmap([1, 3, 5, 7], [2, 4, 6]) => {1: 2, 3: 4, 5: 6, 7: None}
```

Task 2: **`group_by(<f>, target_list: list)`**

Create a function `group_by()` that takes a function and a sequence and groups the elements of the sequence based on the result of the given function `<f>`.
For instance, using the `len` function to measure the length of a string variable

group_by(len, ["hi", "dog", "me", "bad", "good"]) => {2: ["hi", "me"], 3: ["dog", "bad"], 4: ["good"]}

Task 3:

For the final task, you will be implementing the `filter()` using `reduce()`, similar to the example we showed during the class, in which we use the `reduce()` to implementing the `map()`.

Hint: Consider using an empty list [ ] as the initial value.