Lab 2: Python Basics II

INFSCI 0201
Intermediate Programming (Spring 2025)

Grading and Submission

You are to write a complete Python program that meets the requirements outlined in the Lab 2 Tasks section. Push your code to GitHub in the same way you've pushed previous Labs:

- Your code should be in a file named "setoperations.py" and in a folder named "lab02"
    - So, the path to your code should be: "labs/lab02/setoperations.py"

The lab has the following tests:

1. make_set is implemented correctly
2. is_set is implemented correctly
3. union is implemented correctly
4. intersection is implemented correctly

A reminder: violating the class style guide counts as a test failure. The autotester should give you comprehensive feedback. You can submit and test as early and often as you'd like.

Background

A "set" in math is, technically speaking, a collection of *stuff*. That seems like a pretty unimportant definition but it serves the logical basis for most of our contemporary understanding of mathematics. One important feature of sets is they don't contain duplicate elements, and for the most part, that's their primary difference from a **list** in Python. For all practical purposes, though, if an array contains no duplicates, we'll call that a set.

Lab 2 Tasks

You are to write the following four functions:

- ***make _set(data)***
  - Takes a list of integers as an input and turns it into a set (i.e. a version of this array that contains no duplicates).
  - Returns the input data as a list with no duplicates
  - You CANNOT use the built-in Python set() conversion for this - you need to write your own conversion logic for removing duplicates
  - For example:
    - make_set([1, 2, 3, 4, 4, 5]) returns [1, 2, 3, 4, 5]
- ***is_set(data)***
  - Returns true if the list given as input is a set (i.e. it is a list that contains no duplicates)
  - You CANNOT use the built-in Python set() conversion for this!
  - For example:
    - is_set([1, 2, 3, 4, 5]) returns **True**
    - is_set([5, 5]) returns **False**
    - is_set([]) returns **True**
    - is_set(None) returns **False**
- ***union(setA, setB)***
  - If either setA or setB is not a set, return an empty list
  - If both setA and setB are valid sets, return the set that is the union of setA and setB (i.e. it contains every element in either setA or setB)
    - union([1,2], [2,3]) returns [1,2,3]
    - union([], [2,3]) returns [2,3]
    - union([1,1,1], [2,3]) returns an empty list, because the first parameter IS NOT a valid set
- ***intersection(setA, setB)***
  - If either setA or setB is not a set, return an empty list
  - If both setA and setB are valid sets, return the set that is the intersection of setA and setB (i.e. it contains every element that is in both setA and setB)
    - intersection([1,2], [2,3]) returns [2]
    - intersection([], [2,3]) returns []
    - intersection([1,1,1], [2,]) returns an empty list, because the first parameter IS NOT a valid set

PLEASE NOTE: You may not use set() or any other Python data structures or functions that may trivialize this lab. You can only use normal lists! Using an alternative WILL result in a test failure.