![Text

Description automatically generated]()

\*\*DISCLAIMER\*\* You may notice throughout this analysis that the program is actually broken in two sections, one primarily focusing on dictionaries, and the other focused on sets. This is because I ran into an error that I was unable to solve on the Dictionary side of things, but wanted to keep around so I could solve it. As a solution, I moved the rest of my project to another program to work on it, so as to leave the original in its initial condition. All that being said, let’s dive in!

This screenshot contains items 1a, 1b, 1c, and 1d, as well as attempts for 1e, 1f, and 1g (I accidentally left G unlabelled!)

For 1a, we begin with a prewritten dictionary (though we make one from zipping in section 2) and check with the type function that it is in fact a dictionary, and find that it is of a length 5 (not ten, so definitely a dictionary!)

For 1b we add data using previously written user input functions placed inside the .update() method. (Note, due to the nature of dictionaries if the user were to input the exact information from an element of the dictionary, nothing would change.)

For 1c, we update the dictionary with a new value for a key that already existed. This replaces the previous value.

For 1d I used the pop() method to remove an element from the dictionary.

Lastly, I continued to come across the error shown at the bottom of the screenclip when trying to execute two functions that were meant to tackle e, f, and g. (e was to be incorporated into functions f and g)… I recognized the format of 0x00000(numbers) from some time working with repairing and replacing operating systems, and have come across such a format for errors quite often. I’m not sure where it comes from, though I assume it’s pointing to somewhere in memory? It’s strange because Python is not throwing an error here, but rather doing something very unexpected. After much trial and error, I could not seem to uncover my mistake here, so I moved on to the second half of the project.

A picture containing text, monitor, screenshot, screen

Description automatically generatedThe functions designed to cover E, F, and G can be seen here, as well as the input functions carried over from previous work.

Text

Description automatically generated

Here we can see the body of the Sets half of the program, and my attempts to replicate the problem one requirements here in level two.

For 2a, I created a set by definition (we will build one from a list later!) and checked it using the type() function.

For 2b, I used the .add() function combined with a user input to add some simple data to the set.

2d and 2c can be found inside one another, as it is impossible to ‘change’ data in a set, as an element is defined only by its only value, not by any form of key or position, so to change it *is* to delete it. Still to create the illusion of changing, I first removed some data (2d) and then added another user input to satisfy 2c.

Problem 3 can be seen here, but will be discussed more when we can see the function it references in the last screenshot.

Problem 4 also continues on, but in this frame, I am demonstrating a set comprehension starting with a list, building another list in reference to the first, and then using a zip function to align both lists together into a single dictionary. (We will perform a similar process with Sets on the next page!)

![Graphical user interface, text

Description automatically generated]()

Here, we can see the second part of problem 4, where I used a set to rule out any repetitions of the same number, to be certain that no two people were given the same numerical assignment, zipped the corrected list to the group list, and then reformed the zip into a set of tuples.

Text

Description automatically generated

Again, we can see my user input functions from a previous project, but more importantly, the function for problem 3. This function is designed to look at all numbers that can be considered integers and rank which one is the highest. To do this was more complicated than I expected, as I had to deal with variables changing when I didn’t want them to (perhaps most notably having the length of my set change and mess with the for loop) so I had to reestablish them with secondary variables. In short, this program checks each element of a set, attempts to convert it to an integer, and then asks if it is greater or less than the previous element. If it is greater, it will replace the previous element with the current one as the new ‘winner’ and if not, the current element will be discarded. This process continues until we have a maximum value that is then printed to the user. It is also worth noting that this program destroys the set as it processes it, so a copy must be made before running it if the set is to be accessed again.