Assignment 1

Henry Arzumanian

In [ ]:
```
(hash('Henry')%3)+1 #output doesn't always equal 3
```

Out[ ]:
```
3
```

Paraphrase the problem in your own words

Given any list [0, n], sorted or not, find all the missing values in the list between the smallest (0) and the biggest (n) values. The function should return -1 if no missing values were found.

Create 2 new examples that demonstrate you understand the problem

Example 4

Input: lst = [0, 3]

Output: [1,2]

Example 5

Input: lst = [5, 1, 3, 2, 4, 0]

Output: -1

Code the solution to your assigned problem in Python

In [ ]:
```
from typing import List
import timeit

def missing_num(nums: List) -> List[int]:
    # To handle duplicates turn the provided list into a set
    complete_set = set(nums)

    # assign b to max value in the set
    b = max(complete_set)

    # iterate throuh range(b+1) and return only missing values
    missing_list = [num for num in range(b+1) if num not in complete_set]

    # return missing values if they exist, otherwise return -1
    if missing_list:
        return missing_list
    else:
        return -1
```

In [ ]:
```
missing_num([0,4,2,10,12])
```

```
Out[ ]: [1, 3, 5, 6, 7, 8, 9, 11]
```

Explain why your solution works

It converts the provided list to a set to eliminate duplicate numbers, finds the largest number and then inside the list comprehension generates a range of all the numbers between 0 and the max value in the set but only adds values to the output list if they are missing from the inital list. In the end, it returns the list with missing values if there are any. If there are no missing values, it outputs -1

Explain the problem's and space complexity

The problem's space complexity is O(n + k) where n is the number of values in the set and k is the number of values in the output list. I used the default range function which is lazy to decrease space complexity. The time complexity is also O(n) where n is the number of values in range(b+1) because it has to iterate through all of them. Other steps in the function have the same or lower time complexity.

Explain the thinking to an alternative solution

One of the alternative solutions would be to create 2 empty lists. To the first empty list, append only unique values from the original list and sort it. Iterate thorugh it using a for loop while checking if the difference between sequential values equals 1. If it does, keep iterating; if it doesn't, calculate the difference between the values and save it as N, use another for loop to add every number in range (1,N) to the lower number you are iterating over in the main for loop. Add these numbers to the second empty list. After this process is over, check if the second empty list is still empty. Output -1 if it is. Otherwise output the second empty list.