```
# Problem selection
(hash('Frederick')%3)+1
```

```
3
```

Original question:

You are given a list containing `n` integers in the range `[0, n]`. Return a list of numbers that are missing from the range `[0, n]` of the array. If there is no missing number, return -1. Note, all the integers in the list may not be unique.

Paraphrased question:

Write a function that takes a list of random and non-unique integers between `0` and `n` as an argument, and prints any missing numbers within that range. If there are no missing numbers, the function should simply print `-1`.

Q: Create 2 new examples that demonstrate you understand the problem.

Example 1

Input: `lst = [2, 4, 7, 0]`

Output: [1, 3, 5, 6]

Example 2

Input: `lst = [0, 3, 6, 5]`

Output: [1, 2, 4]

Q: Code the solution to your assigned problem in Python (code chunk). Try to find the best time and space complexity solution!

```
# Import typing module with 'List' element
from typing import List

# Define the "missing number" function
def missing_num(nums: List) -> int:

# Create a master list with all integers between 0 and n
    nums = [0]
    n = max(lst)
    counter = 1
    for i in range (n):
      nums.append(counter)
      counter += 1

# Display the master list
    print ("Master list with all the integers:", nums)

# Create a nested loop that compares the submitted list and the master list
    for iteration in range(len(lst)):
        for n in reversed(range(len(nums))): # Iterating through "nums" in reverse order to prevent index error
            if lst[iteration] == nums[n]:
                nums.pop(n) # remove numbers found in submitted list

# Print "-1" if the two lists are identical otherwise display the missing numbers
    if not nums:
        print(-1)
    else:
     print ("Here are the missing numbers:", nums)

# Function ends and regular code with argument begins
lst = [0, 1, 2, 3, 10, 3, 7]
missing_num(lst)
```

```
Master list with all the integers: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Here are the missing numbers: [4, 5, 6, 8, 9]
```

Q: Explain why your solution works.

A: The solution works because it uses nested loops to compare the 2 lists.

Each number found in the submitted list is then "popped" from the master list.

After both iterations, only the integers missing from the submitted list are left in the modified master list

Q: Explain the problem's time and space complexity.

A:

(i) The nested `for` loops compare each element of the submitted list (`O(n)`) with each element of the master list (also `O(n)`) resulting in `O(n^2)` time complexity.

(ii) The solution requires storing all elements of the master list, resulting in a space complexity of `O(n)`.

Q: Explain the thinking to an alternative solution (no coding required, but a classmate reading this should be able to code it up based off your text).

A: Perhaps a simpler solution with less time complexity (but comparable space complexity) would involve:

1. Creating the master list as before
2. Converting both the master and submitted lists into sets
3. Subtracting the submitted list from the master list
4. Printing the resultant set (or `-1`)