Practice Interview

Objective

The partner assignment aims to provide participants with the opportunity to practice coding in an interview context. You will analyze your partner's Assignment 1. Moreover, code reviews are common practice in a software development team. This assignment should give you a taste of the code review process.

Group Size

Each group should have 2 people. You will be assigned a partner

Part 1:

You and your partner must share each other's Assignment 1 submission.

Part 2:

Create a Jupyter Notebook, create 6 of the following headings, and complete the following for your partner's assignment 1:

- Paraphrase the problem in your own words.

```
# Your answer here
A list of integers is given,
including some numbers within the range from 0 to n but not necessarily all numbers in this range.
The task identifies which numbers within the range [0, n]
are missing from the list. If all numbers are present, return -1.
```

- Create 1 new example that demonstrates you understand the problem. Trace/walkthrough 1 example that your partner made and explain it.

```
# Your answer here
Example.
Input: lst = [1, 3, 5, 7]
Output: [0, 2, 4, 6]
Explanation: The numbers in the range [0, 7] are 0, 1, 2, 3, 4, 5, 6, and 7. The numbers 0, 2, 4, and 6 are missing from the list.
```

- Copy the solution your partner wrote.

```
# Your answer here
def missing_num(nums: list) -> int:
    if len(nums) == 0:
        return -1
    min_num = min(nums)
    max_num = max(nums)
    miss_list = []

    for i in range(min_num, max_num):
        if i not in nums:
            miss_list.append(i)

    len(miss_list) == 0
    return -1 if len(miss_list) == 0 else miss_list
```

- Explain why their solution works in your own words.

```
# Your answer here
The provided solution aims to find the missing numbers in the range [0, n] for a given list of integers.
Here's a step-by-step explanation of why this solution works:
1. Initial Check for Empty List:

    if len(nums) == 0:
    return -1

- If the input list nums is empty,
the function immediately returns -1 because
there are no numbers to analyze or find missing values from.
2. Determine the Range:

    min_num = min(nums)
    max_num = max(nums)

- The function determines the smallest (min_num) and largest (max_num) numbers in the list nums.
However, since we are supposed to find missing numbers in the range [0, n],
this step is somewhat redundant because the range
should always start from 0 and go to the maximum number present in the list.
3. Initialize the Missing List:

    miss_list = []

4. Find Missing Numbers:

    for i in range(min_num, max_num):
    if i not in nums:
        miss_list.append(i)

- The function iterates through the range from min_num to max_num (exclusive)
and checks if each number i is present in the list nums.
- If a number i is not found in nums, it is appended to the miss_list.
5. Check for No Missing Numbers:

    len(miss_list) == 0
    return -1 if len(miss_list) == 0 else miss_list

- The function checks if the miss_list is empty.
- If there are no missing numbers, it returns -1.
- Otherwise, it returns the list of missing numbers.
```

- Explain the problem’s time and space complexity in your own words.

```
# Your answer here
Time Complexity
1. Initial Check for Empty List:
    if len(nums) == 0:
        return -1
-This step checks if the list is empty, which takes O(1) time.
2. Finding Minimum and Maximum Values:
    min_num = min(nums)
    max_num = max(nums)
- Finding the minimum value in the list takes O(n) time.
- Finding the maximum value in the list also takes O(n) time.
- Thus, this step takes a total of O(n+n)=O(2n)=O(n) time.
3. Iterating Through the Range and Checking for Missing Numbers:
    for i in range(min_num, max_num):
        if i not in nums:
            miss_list.append(i)
- The for loop iterates from min_num to max_num (exclusive), which can take up to O(n) iterations.
- For each iteration, the if i not in nums statement checks if i is in the list nums,
which takes O(n) time because checking membership in a list is a linear operation.
- Therefore, the nested operations result in O(n×n)=O(n2) time complexity for this step.
4. Checking if miss_list is Empty and Returning Result:
    len(miss_list) == 0
    return -1 if len(miss_list) == 0 else miss_list
- Checking the length of miss_list and the return statement both take O(1) time.

Overall Time Complexity: The dominant term is the nested loop which gives us O(n2) time complexity.

Space Complexity
1. Space for Minimum and Maximum Values:
    min_num = min(nums)
    max_num = max(nums)
- Storing min_num and max_num takes O(1) space each.
2. Space for miss_list:
    miss_list = []
- The miss_list can grow up to the size of the range between min_num and max_num, which is O(n).
3. Space for Iteration Variables:
    for i in range(min_num, max_num):
- The iteration variable i and the loop itself take O(1) space.

Overall Space Complexity: The dominant term is the space used by miss_list, which results in O(n) space complexity.

Summary
Time Complexity: O(n2) due to the nested loop checking each number's membership in the list.
Space Complexity: O(n) primarily due to the miss_list that stores the missing numbers.
This detailed analysis helps us understand the performance characteristics of the function
 and highlights areas where optimization might be needed,
 such as improving the membership check to reduce the overall time complexity.
```

- Critique your partner's solution, including explanation, and if there is anything that should be adjusted.

```
# Your answer here
Critique of Partner's Solution

The solution provided by my partner for the problem
of finding missing numbers in a list within the range [0, n]
has a clear logical flow but has several inefficiencies
and areas for improvement. Let's break down the critique step-by-step:

Strengths:
Initial Check for Empty List: The code correctly checks if the input list is empty and returns -1,
which is a good practice to handle edge cases.

if len(nums) == 0:
    return -1

Range Identification: The code correctly identifies the minimum and maximum values in the list, which can be helpful for certain types of ra

min_num = min(nums)
max_num = max(nums)

Weaknesses and Areas for Improvement:
Incorrect Range Calculation:

The problem specifies finding missing numbers in the range [0, n],
where n is typically the maximum value in the list.
The current solution calculates the range using both minimum and maximum values,
which is unnecessary and potentially incorrect.
Inefficient Membership Check:

The use of if i not in nums inside the loop results in an O(n2) time complexity.
This is because checking if an element is in a list is O(n)
and it is done inside a loop that runs O(n) times.

for i in range(min_num, max_num):
    if i not in nums:
        miss_list.append(i)

Unnecessary Complexity:

The code includes some redundant steps and variables.
For instance, checking the length of miss_list twice and using both minimum and maximum values to create the range.
```

Part 3:

Please write a 200 word reflection documenting your process from assignment 1, and your presentation and review experience with your partner at the bottom of the Jupyter Notebook under a new heading "Reflection." Again, export this Notebook as pdf.

Reflection

```
# Your answer here
In this assignment, my objective was to delve deeply into a data structures or algorithms question,
honing my skills to prepare for industry coding interviews
```

Evaluation Criteria

We are looking for the similar points as Assignment 1