Part 3:

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.

The input will be a string of paired brackets which the function should validate whether they are paired correctly or not.

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

My example: Input= "[){}(]" output= False Partner example: Input= "{[()]}" The function adds opening brackets to a stack When it finds a closing bracket, it first check whether the stack is empty or the previous item is a matching bracket or not. If not the function returns false. If it finds a matching opening bracket, it pops it out of stack and continues to check other pairs. At the end if the stack is empty the function will return True. output= True

- Copy the solution your partner wrote.

def is_valid_brackets(s: str) -> bool:

```
    stack = []
    bracket_map = {')': '(', ']': '[', '}': '{'}

    for char in s:
        if char in bracket_map.values():  # opening brackets
            stack.append(char)
        elif char in bracket_map:  # closing brackets
            if not stack or stack[-1] != bracket_map[char]:
                return False
            stack.pop()
        else:
            # Invalid character (not required by problem, but useful in
    general)
            return False

    return not stack  # True if stack is empty
```

- Explain why their solution works in your own words.

It incorporate all the possible scenarios: The function adds opening brackets to a stack When it finds a closing bracket, it first check whether the stack is empty or the previous item is a matching bracket or not. If not the function returns false. If it finds a matching opening

bracket, it pops it out of stack and continues to check other pairs. At the end if the stack is empty the function will return True.

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

Time complexity: O(n) we have n iterations Space complexity O(2n) at worst case we will have a string of n chars + stack of n items

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

I have no critiques for him. He has done it very well.

Part 4:

Please write a 200 word reflection documenting your process from assignment 2, 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

The function turns the input list into a set to get rid of the duplicates and then sorts it. Each item is compared with the next item to see it increments by one or not. If not the range of missing numbers will be added to a output list. If output has not item the function will return -1 otherwise the output list. The experience of reviewing some one else script was amazing. It's interesting to see how others think of solutions to a problem.