assignment_2 - Part 3

October 17, 2025

0.1 Paraphrase the problem in your own words.

the questions asks for checking how the brakets are closed, if they are the same type, opened and closed, it returns True and if they are not closed or the open and closed are not matched, it returns false

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

one example would be wrong order of opening and closing; "({)}"" –> False ### Partener's example: - assert not is_valid_brackets('({[') #False: this example has three open brakets: (, {, and [that are not closed, expected to be Flase as the partner showed in the assignment.

0.3 Copy the solution your partner wrote

ef is_valid_brackets(s: str) -> bool: q = [] close = {'(':')', '[':']', '{':'}'}

```
for c in s:
    if c == '(' or c == '[' or c == '{':
        q.append(close[c])
    else:
        n = len(q) -1

        if n >= 0 and c == q[n]:
            q.pop()
        else:
            return False

if len(q) > 0:
    return False

return True
```

0.4 Explain why their solution works in your own words.

In this code, the partner defined a function called is_valid_brakets, this function takes a string s and return boolian (True or False). Using a dictionary, it defines all the possible closed bracket types or It uses a dictionary map each opening bracket to its corresponding closing bracket Then with a loop it goes through the string, one charcter by character and when it find a opening braket, it stakes their corresponding closing brakets in a new list Later, when it encounters a closing bracket, it checks if that bracket matches the top item of the stack, If it matches, it removes, if not it returns False At the end it checks the length and if the stack is empty

0.5 Explain the problem's time and space complexity in your own words.

In worse case senario, the code needs to go over all the input data so it is O(n), but it can shortly gives False asnwer in shorter time as well. The same for the space complexity, in worse case it is O(n) since it should save a list of n inputs.

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

The solution and explanations were correct and well explained. If I want to suggest some improvement they are replacing "if c == '(' or c == '[' or c == '{':" with if c in '([{'. Another improvement would be instead of introducing n, to directing the last element of q, just say q[-1] Last, as mentioed by the partner in the solution, the loop in the fucntion can be replaced by a recursive funcrion, to shorten the code.

0.7 Reflection

What I learned from Assignment 1 is that thinking step by step. The most important part is planning the algorithm and try to improve it step by step. At first, I was confused and made a lot of mistakes, but by going back and checking each line carefully, I started to see where my logic was wrong and how to fix it. Correcting my algorithm and improving it by trying to improve and shorten the code, helped to learn new syntax, and ways to make it work the way I want.

From my partner's solution, I learned that there are many different ways to solve the same problem. The way they solved it looked very unique to me. I wouldn't thought about using dictionaries for this question, and since I am very new to coding, it was really interesting to see that approach. It showed me that programming can be creative, not just technical. At first, critiquing my partner's code felt intimidating because I dont feel confident in python. But after spending more time looking at their code, trying to make sense of it line by line, I started to notice small things that could be improved and it gave a bit moore confidence.