You

def create_table(n: int, m: int) -> list[list]: if n > 1 and m > 1: for i, _ in enumerate(range(m)): if i > 0: table[0] += [table[0][i - 1] + create_table(n-1, m)[-1][i]] else: table = [[1]] return create_table(n-1, m) + table if n <= 1 or m <= 1: return [[1 for _ in range(m)] for _ in range(n)] what can you optimise in this function. It has to be recursive

Copilot

The function you've provided seems to be creating a table with dimensions `n` and `m`. However, it appears that the variable `table` is not defined before it's used which might cause an error. Also, the function seems to be doing unnecessary work in the recursion which can be optimized.

Here's an optimized version of your function:

```
def create_table(n: int, m: int) -> list[list]:
    if n <= 1 or m <= 1:
        return [[1 for _ in range(m)] for _ in range(n)]
    else:
        table = create_table(n-1, m)
        new_row = [1]
        for i in range(1, m):
            new_row.append(new_row[i - 1] + table[-1][i])
        table.append(new_row)
        return table
```

In this version, the function first checks if `n` or `m` is less than or equal to 1. If so, it returns a table filled with 1s. If not, it recursively creates a table of size `(n-1, m)`. Then it calculates the new row based on the last row of the table and the new row itself, and appends the new row to the table. This way, the function avoids unnecessary recursion and computations. The function remains recursive as per your requirement. Please let me know if this helps or if you have any other questions! 😊

You

now make it according to PEP8 standards

Copilot

Sure, here's your function revised according to PEP8 standards:

```
def create_table(n, m):
    """
    Create a table with dimensions n and m.

    Parameters:
    n (int): Number of rows.
    m (int): Number of columns.

    Returns:
    list[list]: A table of size n x m.
    """
    if n <= 1 or m <= 1:
        return [[1 for _ in range(m)] for _ in range(n)]
    else:
        table = create_table(n - 1, m)
        new_row = [1]
        for i in range(1, m):
            new_row.append(new_row[i - 1] + table[-1][i])
        table.append(new_row)
        return table
```

In this version, I’ve added a docstring to the function to explain what it does, its parameters, and its return value. I’ve also added spaces around the operators and after commas to improve readability, which are some of the formatting conventions recommended by PEP8. Let me know if you have any other questions! 😊