As the program is written in python, it encounters speed efficacy programs over another language, like C. An example of this is pythons match statement not providing significant, if any, time improvements over using if chains, while C switch statements have a much greater efficiency than if statements (O(1) compared to O(n)) as C's switch statement uses lookup tables (https://stackoverflow.com/questions/68476576/python-match-case-switch-performance, https://stackoverflow.com/questions/4442835/what-is-the-runtime-complexity-of-a-switch-statement). In fact, according to cpython bug #92868, structural pattern matching is slower than using if statements for a similar result.

The programs usage of for loops for any finite lists, instead of a while loop, created more optimised code as it uses less interpreted actions, and uses more native c functions. This results in faster code as the interpreter requires excess CPU cycles to turn every interpreted line of code into native kernel calls, whereas if a native c function was called multiple instruction could be run without interpretation.

The program's usage of dictionaries will scale with larger amounts of data as most operations with dictionaries have a big O of 1. However, some actions require getting the list of keys in the dictionaries which has a linear space complexity (O(n)), such as in the menu's where the items in the menu must be listed with a number index. Iterating over over the food that the person ordered is an example of O(n) time complexity, meaning that the display of user information will slow down as they order more items. However even though these items have a non constant O, it does not mean that the operation is slow, but that it will slow down as more data is added to the array.