Ονοματεπώνυμο: Γεώργιος Τσίρης
Αριθμός Μητρώου: 1115201700173

- ***addition.py:*** The add function returns the sum of two given numbers a and b. Disclaimer: The presented code is given by the instructions of this task.

- ***buyLotsOfFruit.py:*** The buyLotsOfFruit function returns the total cost of a given order. For each fruit in the order, checks if it is among the available ones. If it is, it multiplies its cost per pound by the number of pounds and accumulatively computes the total cost. If during that procedure an unavailable fruit is encountered, then an error message is getting printed and the return value is going to be None.

- ***shopSmart.py:*** The shopSmart function returns the fruit shop, among the given ones, that is the cheapest to choose according to a given order. At first, it checks that there are available fruit shops and also that the order is not empty. In those cases, error messages are printed accordingly and the return value is None. Else, the proccess of actually finding the best option begins by initiallizing the minimum cost as the price of the order if it would be served at the first availiable fruit shop. Then check that same price in every fruit shop respectively. If a smaller price is encoutered, then both the best option and the minimum cost are getting updated. At the end, the cheapest shop for that particular order is found.

- ***stack.py:*** This .py file includes both the implementation of the class Stack and the requested showcase. Abstract data type stack is implemented using built-in lists. Class stack has all the five requested attributes (initialize, is_empty, is_full, push, pop). The __main__ is calling another function to get the job done, for simplicity reasons (eg. In __main__ in order to exit before the end of the function I should use sys.exit() instead of return. This common practise also makes it easier to call this function from other scripts, in case I want to use it in a future project etc). A string that contains only '(', ')', '[', ']', '{', '}' is balanced when after an openning character the next closing character is of the same type and also at the end there must be no character remaining unmatched. To achieve that, we store every oppening character in the stack, expecting the next closing character to match. If they don't match, the string is not balanced. If we make to the end of the string without getting in a situation that we need to pop from an empty stack and we don't have remaining unmatched character(s) after the whole procedure, then the string is balanced.