Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЁТ

по лабораторной работе №1

ИССЛЕДОВАНИЕ БАЗОВЫХ ФУНКЦИЙ ЯЗЫКА PYTHON

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверил:

Бондарев В. Н.

Севастополь

2024

# ЦЕЛЬ РАБОТЫ

Изучение технологии подготовки и выполнения программ на языке Python, исследование свойств функций языка Python, используемых при обработке последовательностей, формирование навыков определения классов языка Python.

# ЗАДАНИЕ

* 1. Изучить основы языка Python, структуры данных и методы обработки списков, кортежей, множеств, словарей, классы и объекты Python, среду программирования на языке Python. Проверить выполнение приведённых примеров в среде программирования;
  2. Выполнить задания 1-5 в интерактивном режиме, используя возможности IPython;
  3. Определить в соответствии с заданием 6 функции и методы для решения 3-х сформулированных задач. Использовать для редактирования функций и выполнения кода интегрированную среду PyCharm. Зафиксировать результаты выполнения функций во всех необходимых режимах. Выполнить с помощью autograder.py автооценивание. При обнаружении ошибок отредактировать код.

# ХОД РАБОТЫ

## Строки

**Задание:** используя команды dir и help, изучить следующие методы строкового типа: ‘format’, ‘strip’, ‘lstrip’, ‘rstrip’, ‘capitalize’, ‘title’, ‘count’, ‘index’, ‘rindex’, ‘startswith’, ‘endswith’, ‘replace’, ‘split’, ‘rsplit’, ‘join’, ‘partition’, ‘rpartition’. Разработать примеры вызова указанных методов.

Метод *format* нужен для форматирования строки и подстановки в неё значений из переменных – места подстановки обозначаются фигурными скобками *{}* (рисунок 1).

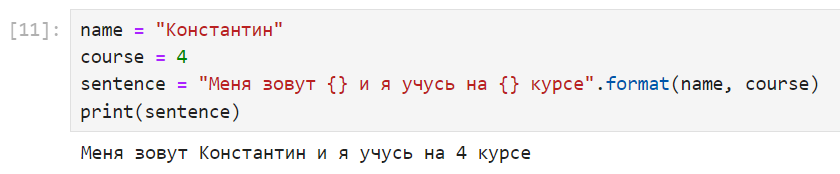


Рисунок 1 – Пример вызова метода format

Метод *strip* удаляет пробелы (по умолчанию) или любые другие указанные символы с начала и конца строки (рисунок 2).

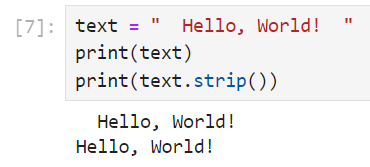


Рисунок 2 – Пример вызова метода strip

Методы *lstrip* и *rstrip* являются аналогами метода *strip*, однако удаляют символы только с левой или правой стороны соответственно (рисунок 3).

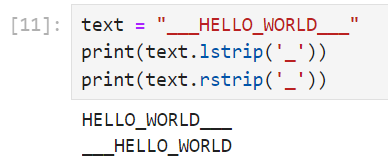


Рисунок 3 – Пример вызова методов lstrip и rstrip

Метод *capitalize* преобразует первый символ строки в верхний регистр, остальные − в нижний(рисунок 4).

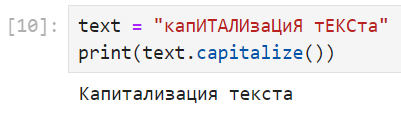


Рисунок 4 – Пример вызова метода capitalize

Метод *title* преобразует первые символы каждого слова в строке в верхний регистр(рисунок 5).

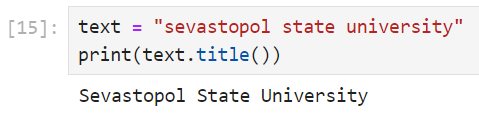


Рисунок 5 – Пример вызова метода title

Метод *count* возвращает количество вхождений указанной подстроки в строку(рисунок 6).

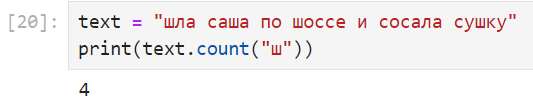


Рисунок 6 – Пример вызова метода count для строки

Метод *index* возвращает индекс первого вхождения подстроки и вызывает ошибку, если подстрока не найдена(рисунок 7).

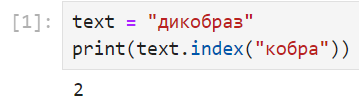


Рисунок 7 – Пример вызова метода index для строки

Метод *rindex* аналогичен методу *index* кроме того, что возвращает последнее вхождение подстроки(рисунок 8).

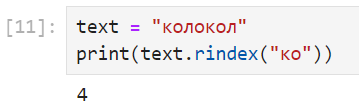


Рисунок 8 – Пример вызова метода rindex

Метод *startswith* проверяет, начинается ли строка с указанной подстроки, и возвращает логическое значение (рисунок 9).

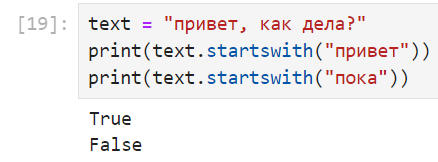


Рисунок 9 – Пример вызова метода startswith

Метод *endswith* проверяет, оканчивается ли строка указанной подстрокой, и возвращает логическое значение (рисунок 10).

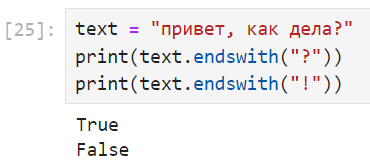


Рисунок 10 – Пример вызова метода endswith

Метод *replace* заменяет все (если не задано ограничение) вхождения подстроки на другую последовательность символов(рисунок 11).

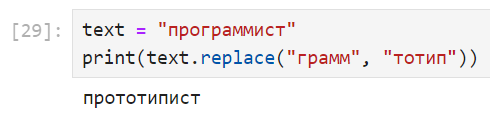


Рисунок 11 – Пример вызова метода replace

Метод *split* разбивает строку на подстроки по указанному разделителю и создаёт из них список(рисунок 12).

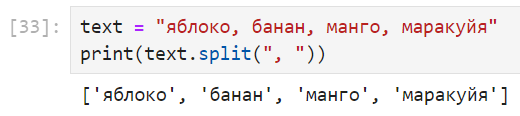


Рисунок 12 – Пример вызова метода split

Метод *rsplit* аналогичен методу *split* кроме того, что начинает разбиение с конца, что полезно при ограничении на разбиения(рисунок 13).

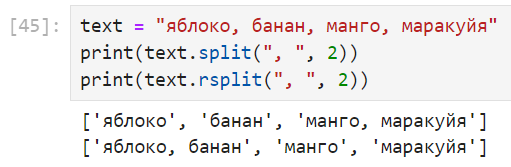


Рисунок 13 – Пример вызова метода rsplit и сравнение со split

Метод *join* использует строку в качестве разделителя элементов переданного списка, из которого формирует новую строку(рисунок 14).

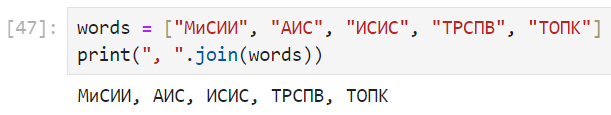


Рисунок 14 – Пример вызова метода join

Метод *partition* делит строку на три подстроки по первому вхождению указанного разделителя, включая сам разделитель(рисунок 15).

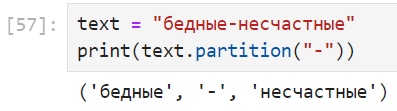


Рисунок 15 – Пример вызова метода rindex

Метод *rpartition* аналогичен методу *partition* кроме того, что ищет разделитель с конца (рисунок 16).

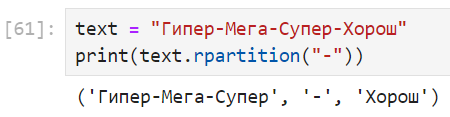


Рисунок 16 – Пример вызова метода rpartition

## Списки

**Задание:** используя команды dir и help, изучить следующие методы обработки списков: 'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort'. Разработать примеры вызова указанных методов.

Метод *append* добавляет переданный элемент в конец списка (рисунок 17).

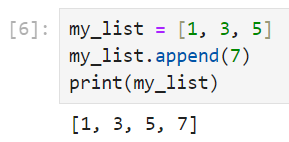


Рисунок 17 – Пример вызова метода append

Метод *count* возвращает число вхождения элемента в список (рисунок 18).

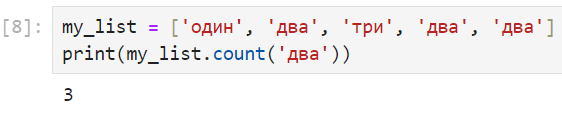


Рисунок 18 – Пример вызова метода count для списка

Метод *extend* расширяет список, добавляя в него все элементы переданного итерируемого объекта (рисунок 19).

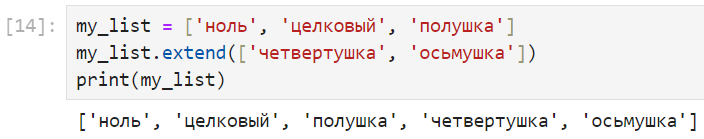


Рисунок 19 – Пример вызова метода extend

Метод *index* возвращает позицию первого вхождения переданного элемента в список (рисунок 20).

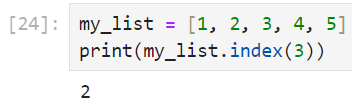


Рисунок 20 – Пример вызова метода index для списка

Метод *insert* вставляет элемент в список на заданную позицию (рисунок 21).

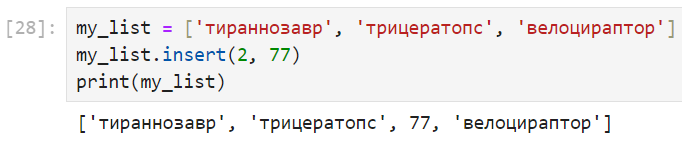


Рисунок 21 – Пример вызова метода insert для списка

Метод *pop* удаляет из списка элемент по указанному индексу (по умолчанию последний элемент) и возвращает этот элемент (рисунок 22).

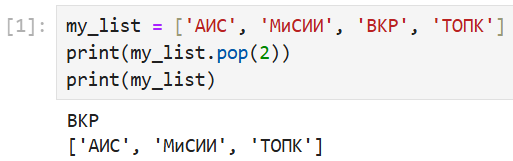


Рисунок 22 – Пример вызова метода pop для списка

Метод *remove* убирает из списка первое вхождение переданного элемента (рисунок 23).

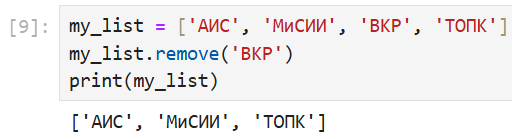


Рисунок 23 – Пример вызова метода remove для списка

Метод *reverse* переставляет элементы списка в обратном порядке, от последнего к первому (рисунок 24).

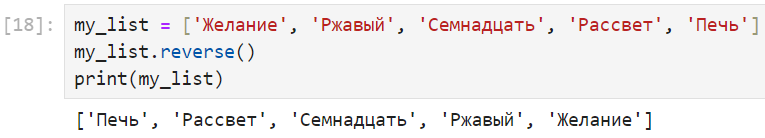


Рисунок 24 – Пример вызова метода reverse для списка

Метод *sort* организует сортировку списка (рисунок 25). По умолчанию используется сортировка по возрастанию.

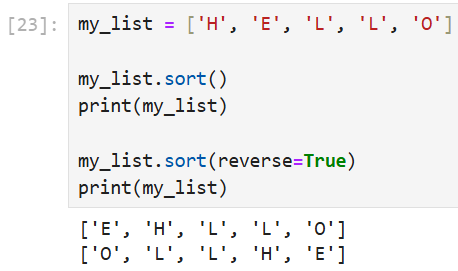


Рисунок 25 – Пример вызова метода sort для списка

## Словари

Задание: используя команды dir и help, изучить следующие методы обработки словарей: 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values'. Разработать примеры вызова указанных методов.

Метод *clear* очищает словарь, удаляя все элементы (рисунок 26).

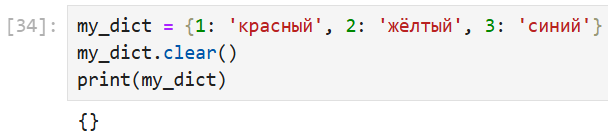


Рисунок 26 – Пример вызова метода clear для словаря

Метод *copy* производит поверхностное копирование словаря (рисунок 27).

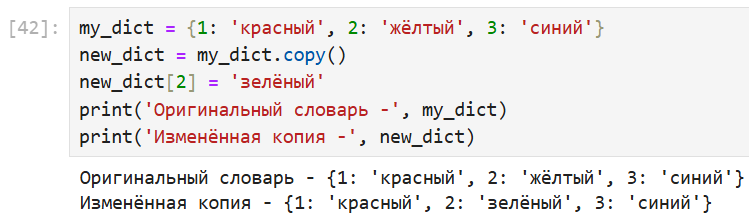


Рисунок 27 – Пример вызова метода copy для словаря

Метод *fromkeys* создаёт новый словарь из переданного списка ключей и указанного для них всех значения (рисунок 28).

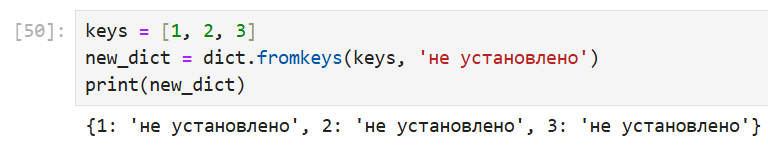


Рисунок 28 – Пример вызова метода fromkeys для словаря

Метод *get* возвращает значение словаря по указанному ключу (рисунок 29). При отсутствии ключа по умолчанию возвращает None (можно изменить), не вызывая ошибки.

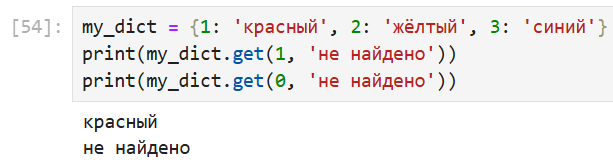


Рисунок 29 – Пример вызова метода get для словаря

Метод *items* возвращает итерируемый объект с парами ключ-значение из словаря (рисунок 30).

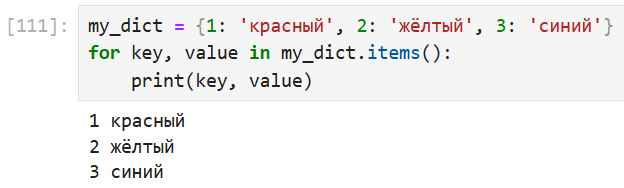


Рисунок 30 – Пример вызова метода items для словаря

Метод *keys* возвращает итерируемый объект, содержащий все ключи словаря (рисунок 31).

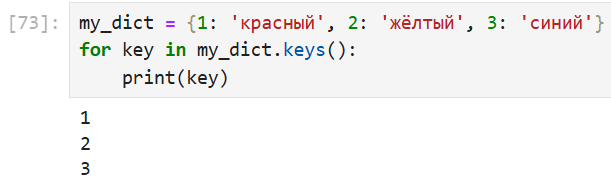


Рисунок 31 – Пример вызова метода reverse для словаря

Метод *pop* удаляет из словаря ключ и возвращает его значение, если ключ был найден (рисунок 32).

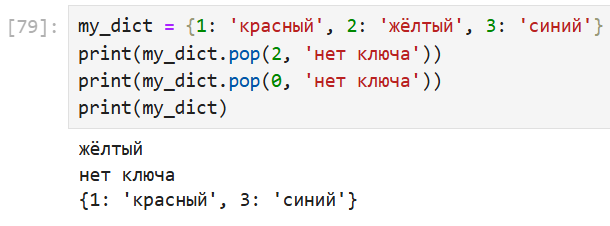


Рисунок 32 – Пример вызова метода pop для словаря

Метод *popitem* удаляет и возвращает последнюю добавленную в словарь пару ключ-значение (рисунок 33).

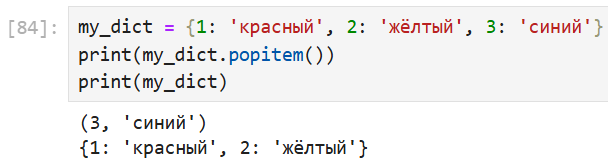


Рисунок 33 – Пример вызова метода popitem для словаря

Метод *setdefault* возвращает из словаря значение по ключу, если тот отсутствует – создаёт его и устанавливает ему значение, по умолчанию None (рисунок 34).

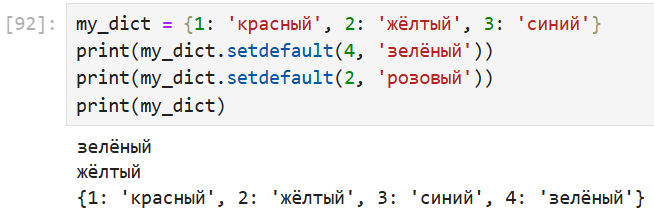


Рисунок 34 – Пример вызова метода setdefault для словаря

Метод *update* дополняет словарь новыми парами ключ-значение из другого словаря или итерируемого объекта (рисунок 35).

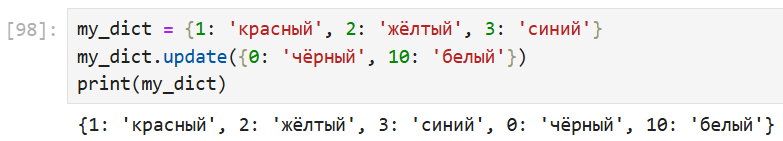


Рисунок 35 – Пример вызова метода update для словаря

Метод *values* возвращает итерируемый объект, содержащий все значения словаря (рисунок 36).

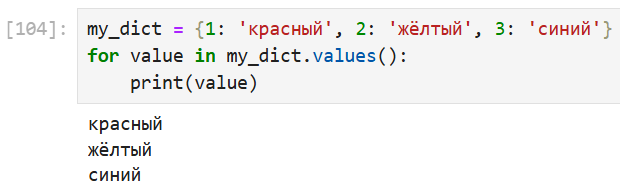


Рисунок 36 – Пример вызова метода values для словаря

## Списковое включение

**Задание:** определить списковое включение, которое из списка строк генерирует версию нового списка, состоящего из строк, длина которых больше пяти и которые записаны символами нижнего регистра (рисунок 37). Решение можно проверить, просмотрев файл listcomp2.py.

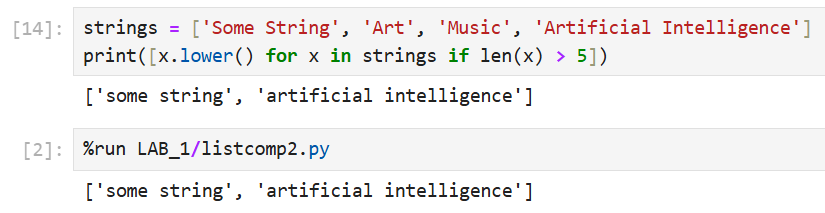


Рисунок 37 – Реализация спискового включения

## Быстрая сортировка

Задание: написать функцию быстрой сортировки на Python, используя списки. Использовать первый элемент как точку деления списка (рисунок 38). Решение можно проверить, просмотрев файл quickSort.py.

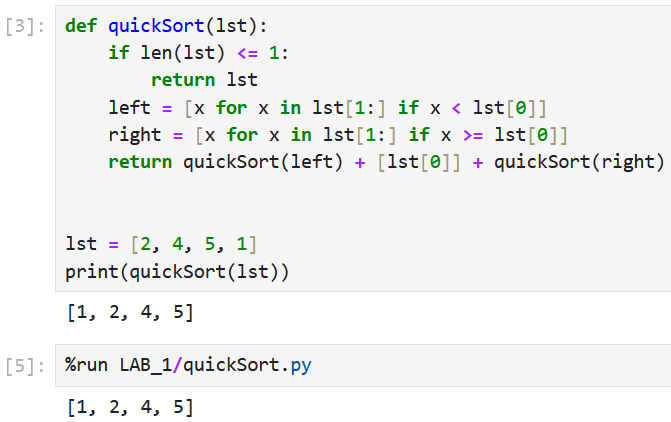


Рисунок 38 – Реализация быстрой сортировки

## Решение задач и автооценивание

**Задание 1:** был открыт файл ‘addition.py’. В нём было заполнено тело функции add, возвращающей сумму двух чисел.

Листинг 1 – Определение функции add

def add(a, b):  
 *"Возвращает сумму a и b"* print("Passed a = %s and b = %s, returning a + b = %s" % (a, b, a + b))  
 return a + b

Автооцениватель подтвердил, что функция определена верно для всех трёх примеров (рисунок 39).

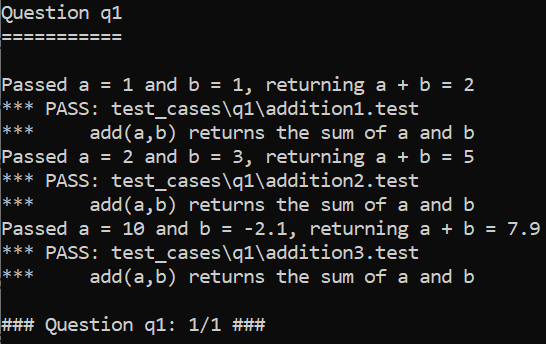


Рисунок 39 – Автооценивание задачи 1

**Задание 2:** в файле ‘buyLotsOfFruit.py’ была определена функция buyLotsOfFruit, которая подсчитывает стоимость заказа, сопоставляя его содержимое с ценником.

Листинг 2 – Определение функции buyLotsOfFruit

def buyLotsOfFruit(orderList):  
 *"""  
 orderList: Список-заказ из кортежей (fruit, numPounds)   
  
 Возвращает стоимость заказа  
 """* totalCost = 0.0  
  
 for fruit, weight in orderList:  
 if fruitPrices.get(fruit) is None:  
 print('Error: no such fruit in price list')  
 return None  
  
 totalCost += fruitPrices[fruit] \* weight  
  
 print(f"Cost of {orderList} is {totalCost:.2f}")  
 return totalCost

Автооцениватель подтвердил, что функция определена верно для всех примеров (рисунок 40).

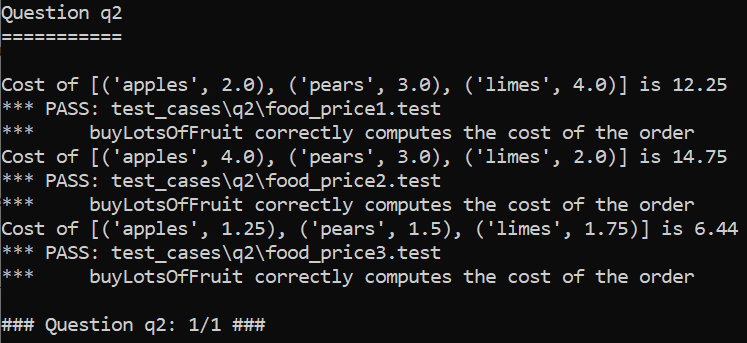


Рисунок 40 – Автооценивание задачи 2

**Задание3:** в файле ‘shopSmart.py’ была определена функция shopSmart, которая определяет магазин с наименьшей ценой для заказа, сопоставляя их ценники с его содержимым.

Листинг 3 – Определение функции shopSmart

def shopSmart(orderList, fruitShops):  
 *"""  
 Возвращает магазин с минимальной стоимостью заказа  
 orderList: Список-заказ из кортежей (fruit, numPound)   
 fruitShops: Список магазинов типа shop  
 """* minTotalCost = -1  
 bestShopName = ''  
  
 for shop in fruitShops:  
 totalCost = shop.getPriceOfOrder(orderList)  
  
 if totalCost < minTotalCost or minTotalCost < 0:  
 minTotalCost = totalCost  
 bestShopName = shop.getName()  
  
 print(f"For orders: {orderList} best shop is {bestShopName}")  
 return f"<FruitShop: {bestShopName}>"

Автооцениватель подтвердил, что функция определена верно для всех примеров (рисунок 41).

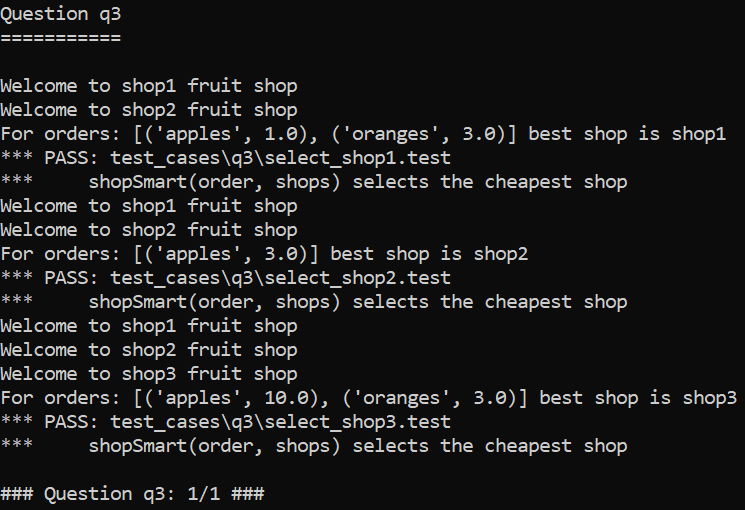


Рисунок 41 – Автооценивание для задачи 3

# ВЫВОД

В ходе работы были отработаны навыки написания прикладных программ на языке Python. Были рассмотрены и исследованы различные типы данных и методы их обработки (строки, списки, словари и др.). Изучена информация об определении классов в Python. В результате работы было написано множество примеров применения методов и свойств типов данных и несколько алгоритмов, определены функции решения задач, правильность выполнения которых была проверена с помощью автооценивателя.