МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Кафедра инфокоммуникаций

Отчет

по лабораторной работе №6

«Работа с функциями в языке Python»

по дисциплине:

«Введение в системы искусственного интеллекта»

Вариант 9

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Полещук Константин Сергеевич
_____(подпись)
Проверил:
Воронкин Роман Александрович
_____(подпись)

Ставрополь, 2022 г.

Цель работы: приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Задание №1

Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

Задание 2.6: использовать словарь, содержащий следующие ключи: название начального пункта маршрута; название конечного пункта маршрута; номер маршрута. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по номерам маршрутов; вывод на экран информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

```
import os
#Список маршрутов
from os import sep

list_route = []

def input_data():
    #Ввод данных с клавиатуры
    start_point = input("Введите начальный пункт маршрута: ")
    end_point = input("Введите конечный пункт маршрута: ")
    number_route = int(input("Введите номер маршрута: "))

    print("Маршрут добавлен!\n")
    input("Нажмите 'Enter' для продолжения")
    #Создание словаря "Маршрут"
    route = {
        'start_point': start_point,
        'end_point': end_point,
        'number_route': number_route
    }

    #Добавление маршрута в список
    global list_route
    list_route.append(route)
```

Рисунок 1.1 – Листинг программы

```
30  def output_data():
31      if len(list_route) == 0:
32          print("Ваш список пуст! Сначала добавьте маршруты в список!\n")
33          input("Нажмите 'Enter' для продолжения")
34      else:
35          #Сортировка списка по номеру маршрутов
36          list_route.sort(key=lambda item: item.get('number_route'))
37
38          #Вывод
39          for i, route in enumerate(list_route, 1):
40              print(i,".",
41                  " Начальный пункт: ",route.get('start_point', ''),";",
42                  " Конечный пункт: ",route.get('end_point', ''),";",
43                  " Номер маршрута: ",route.get('number_route', 0),";",
44                  sep=''
45                  )
46          print()
47          input("Нажмите 'Enter' для продолжения")
```

Рисунок 1.2 – Листинг программы

```
49  while True:
50      os.system('cls')
51      print("Заполнить список >> [1]")
52      print("Вывод списка >> [2]")
53      print("Выход >> [3]")
54
55      command = int(input(">>"))
56
57      if command == 1:
58          input_data()
59
60      elif command == 2:
61          output_data()
62
63      elif command == 3:
64          break
65      else:
66          print(f"Неизветсная команда: {command}\n")
67          input("Нажмите 'Enter' для продолжения")
```

Рисунок 1.3 – Листинг программы

Заполнение данных и вывод результата в консоль приведены ниже (Рисунки 2.1-2.3):

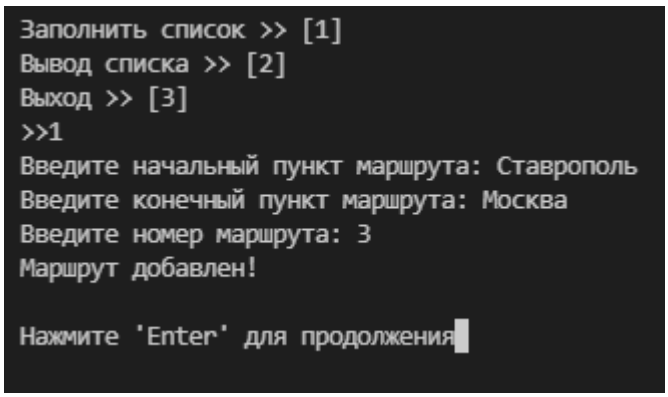

Рисунок 2.1 – Заполнение данных

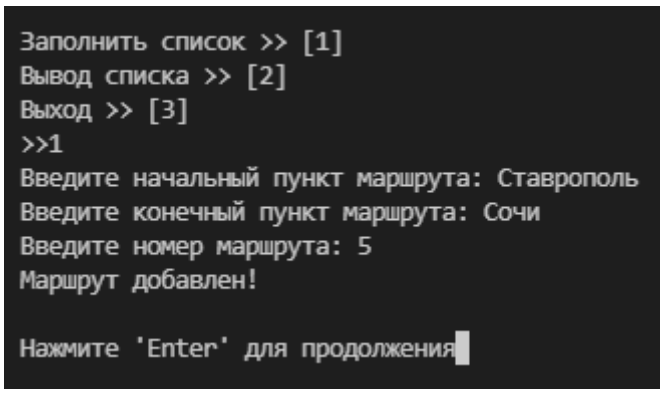

Рисунок 2.2 – Заполнение данных

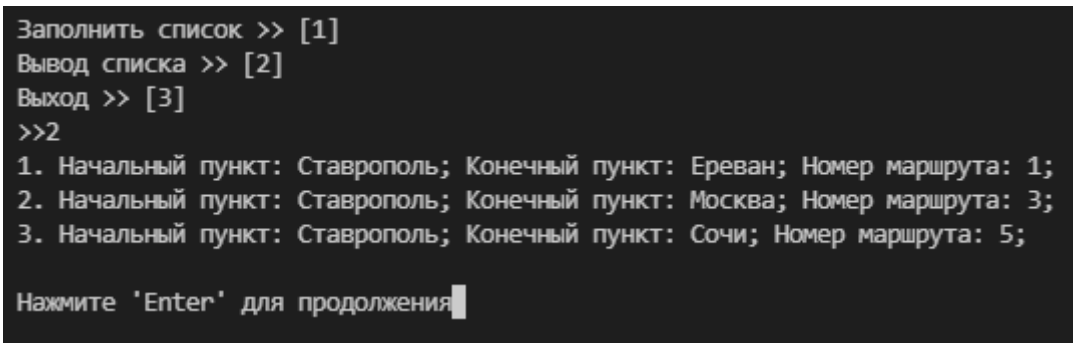

Рисунок 2.3 – Вывод результата

Файл Lab_6(2.8) с решением задачи находится на**Github**: https://github.com/Scratchykaktus/Python.git

Вывод: в процессе выполнения лабораторной работы, были приобретены навыки по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Ответы на вопросы:

1. Каково назначение функций в языке программирования Python?

Функция – это блок организованного, многократно используемого кода, который используется для выполнения конкретного задания. Функции обеспечивают лучшую модульность приложения и значительно повышают уровень повторного использования кода.

2. Каково назначение операторов def и return ?

Обратите внимание на ключевое слово def – оно необходимо для объявления функции. После def указывается имя, затем двоеточие, а следом идёт тело функции: последовательность инструкций, объединённая в один блок отступом слева.

В функции может быть использовано ключевое слово return – оно указывает какое значение передаётся программе, вызвавшей функцию. Если return не указано, то функция неявно возвращает None.

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

Доступ к глобальной переменной можно получить из всей программы.

После объявления глобальной переменной мы можем использовать его во всем коде. Например, мы можем создать функцию, которая печатает значение, содержащееся в нашей глобальной переменной name, используя следующий код:

С другой стороны, локальные переменные - это переменные, объявленные внутри функции. Известно, что эти переменные имеют локальную область видимости. Это означает, что к ним можно получить доступ только в той функции, в которой они объявлены.

4. Как вернуть несколько значений из функции Python?

Оператор return используется для возврата из функции, т.е. для прекращения её работы и выхода из неё. При этом можно также вернуть некоторое значение из функции:

Из функции можно вернуть только одно значение. Если все-таки необходимо вернуть несколько значений, то для этого можно использовать кортеж:

5. Какие существуют способы передачи значений в функцию?

По умолчанию аргументы могут передаваться в функцию Python либо по положению, либо явно по ключевому слову. Для производительности и удобочитаемости имеет смысл ограничить способ передачи аргументов.

6. Как задать значение аргументов?

Эта функция может быть вызвана несколькими способами:

давая только обязательный аргумент: ask_ok('Хочешь продолжить?')

давая один из необязательных аргументов: ask_ok('OK, чтобы перезаписать файл?', 2)

или даже приводя все аргументы: ask_ok('OK, чтобы перезаписать файл?', 2, 'Вводите только "ok" или "no"!')

В этом примере также вводится in ключевое слово. Этот оператор проверяет, содержит ли последовательность определенное значение.

Если при вызове функции значение аргумента, соответствующего ключевому (именованному) параметру, не было задано, то при выполнении функции идентификатор параметра связывается с его значением по умолчанию, то есть с одним и тем же объектом.

Значения по умолчанию вычисляются в точке выполняется инструкции def, а не в точке ее вызовафункции по умолчанию?

7. Каково назначение lambda-выражений в языке Python?

Единственное преимущество использования лямбда-выражения вместо локально определенной функции состоит в том, что вам не нужно придумывать имя для функции. В отличие от лямбда-форм в других языках, где они добавляют функциональность. Лямбда-выражения Python являются лишь сокращенной записью, если вы слишком ленивы, чтобы определить функцию.

Отличие lambda-выражения от нормального определения функции:

не могут содержать return, pass, assert или raise;

не имеют имени функции, отсюда название - анонимные функции;

не могут иметь в теле более одной строки с выражением.

не поддерживает аннотации типов.

Как и обычный объект функции, лямбда-выражения поддерживают различные способы передачи аргументов:

только по позиции

по позиции или по ключу

только по ключу

*args - произвольное число позиционных аргументов

**kwargs - произвольное число именованных аргументов

Как и вложенные функции, лямбда-функции могут ссылаться на переменные из содержащей области.

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в python - достаточно важный аспект, ведь от нее порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода.

PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код.

Цель этого PEP - стандартизировать структуру строк документации: что они должны в себя включать, и как это написать (не касаясь вопроса синтаксиса строк документации). Этот PEP описывает соглашения, а не правила или синтаксис.

При нарушении этих соглашений, самое худшее, чего можно ожидать - некоторых неодобрительных взглядов. Но некоторые программы (например, docutils), знают о соглашениях, поэтому следование им даст вам лучшие результаты.

9. В чем особенность однострочных и многострочных форм строк документации?

Строки документации - строковые литералы, которые являются первым оператором в модуле, функции, классе или определении метода. Такая строка документации становится специальным атрибутом __doc__ этого объекта.

Все модули должны, как правило, иметь строки документации, и все функции и классы, экспортируемые модулем также должны иметь строки документации. Публичные методы (в том числе __init__) также должны иметь строки документации. Пакет модулей может быть документирован в __init__.py.

Для согласованности, всегда используйте """triple double quotes""" для строк документации. Используйте r"""raw triple double quotes""", если вы будете использовать обратную косую черту в строке документации.

Существует две формы строк документации: однострочная и многострочная.

Однострочники предназначены для действительно очевидных случаев. Они должны умещаться на одной строке. Используйте тройные кавычки, даже если документация умещается на одной строке. Потом будет проще её дополнить. Закрывающие кавычки на той же строке. Это смотрится лучше. Нет пустых строк перед или после документации.

Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции).

Этот тип строк документации подходит только для C функций (таких, как встроенные модули), где интроспекция не представляется возможной.

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием. Первая строка может быть использована автоматическими средствами индексации, поэтому важно, чтобы она находилась на одной строке и была отделена от остальной документации пустой строкой. Первая строка может быть на той же строке, где и открывающие кавычки, или на

следующей строке. Вся документация должна иметь такой же отступ, как кавычки на первой строке (см. пример ниже).

Вставляйте пустую строку до и после всех строк документации (однострочных или многострочных), которые документируют класс - вообще говоря, методы класса разделены друг от друга одной пустой строкой, а строка документации должна быть смещена от первого метода пустой строкой; для симметрии, поставьте пустую строку между заголовком класса и строкой документации. Строки документации функций и методов, как правило, не имеют этого требования.