

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.11
дисциплины «Программирование на Python»
Вариант ____

Выполнил:
Иващенко Олег Андреевич
2 курс, группа ИВТ-б-о-22-1,
09.03.02 «Информационные и
вычислительные машины»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем»

(подпись)

Руководитель практики:
Воронкин Роман Александрович,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: «Замыкания в языке Python»

Цель: Приобретение навыков по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

Таблица 1 – Код программы example.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def outer_func(a):
    def inner_func(b):
        return a + b

    return inner_func

if __name__ == "__main__":
    temp = outer_func(10)
    result = temp(5)
    print(f"Результат: {result}")
```




Рисунок 1 – Вывод программы example.py

Индивидуальное задание. Используя замыкания функций, объявить внутреннюю функцию, которая преобразует строку из списка целых чисел, записанных через пробел, либо в список, либо в кортеж. Тип коллекции определяется параметром `type` внешней функции. Если `type = 'list'`, то используется список, иначе – кортеж. Далее на вход программы поступает две строки: первая – это значение для параметра `type`, вторая – список целых чисел, записанных через пробел. С помощью реализованного замыкания преобразовать эту строку в соответствующую коллекцию. Результат работы замыкания вывести на экран.

Таблица 2 Код программы individual.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def type_transform(selected_type):
    """
    Функция принимает строковое значение. Рекомендуется
    использовать значения 'tuple' или 'list'
    """

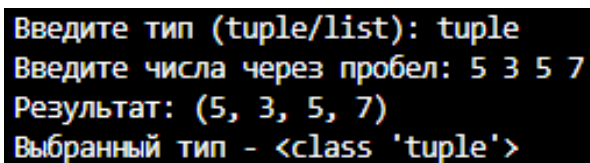
    def transform(numbers):
        """
        Функция принимает числовые значения и, в зависимости от
        указанного строкового значения внешней функции, преобразует
        введённые значения в кортеж или список. Возвращает преобразованный
        результат
        """

        collection = [int(value) for value in numbers]

        if selected_type == 'tuple':
            return tuple(collection)
        elif selected_type == 'list':
            return list(collection)

    return transform

if __name__ == "__main__":
    selected_type = type_transform(input("Введите тип (tuple/list): "))
    collection = selected_type(input("Введите числа через пробел: ").split())
    print(f"Результат: {collection}")
    print(f"Выбранный тип - {type(collection)}")
```



```
Введите тип (tuple/list): tuple
Введите числа через пробел: 5 3 5 7
Результат: (5, 3, 5, 7)
Выбранный тип - <class 'tuple'>
```

Рисунок 2.1 – Результат выполнения программы individual.py при вводе типа
tuple

```
Введите тип (tuple/list): list
Введите числа через пробел: 5 3 5 7
Результат: [5, 3, 5, 7]
Выбранный тип - <class 'list'>
```

Рисунок 2.2 – Результат выполнения программы individual.py при вводе типа list

```
Введите тип (tuple/list): int
Введите числа через пробел: 5 3 5 7
Результат: None
Выбранный тип - <class 'NoneType'>
```

Рисунок 2.3 – Результат выполнения программы individual.py при вводе любого другого типа

Контрольные вопросы

1. Что такое замыкание?

Замыкание – это функция, которая сохраняет ссылку на переменные из своей внешней области видимости, даже если эта область видимости уже не существует. Замыкание «захватывает» переменные и обеспечивает доступ к ним даже после того, как функция завершила выполнение.

2. Как реализованы замыкания в языке программирования Python?

В Python замыкания реализуются естественным образом. Функции в Python являются объектами первого класса, и они могут быть вложенными. Если функция вложена в другую функцию, и внутренняя функция ссылается на переменные внешней функции, то создаётся замыкание.

3. Что подразумевает под собой область видимости Local?

Область видимости Local (локальная) в Python относится к переменным, определённым внутри текущей функции. Эти переменные видны только внутри этой функции и не доступны за её пределами.

4. Что подразумевает под собой область видимости Enclosing?

Область видимости Enclosing (внешняя) относится к переменным, определённым в объемлющих функциях, если текущая функция является вложенной. Переменные из внешней области видимости могут быть доступны внутри вложенной функции.

5. Что подразумевает под собой область видимости Global?

Область видимости Global (глобальная) относится к переменным, определённым на уровне модуля или в глобальной области видимости. Эти переменные видны для всех функций и кода внутри модуля.

6. Что подразумевает под собой область видимости Build-in?

Область видимости Build-in (встроенная) относится к встроенным функциям и объектам, предоставляемым Python. Примеры включают функции `print()`, `len()`, `int()` и т.д.

7. Как использовать замыкания в языке программирования Python?

Замыкания можно использовать следующим образом:

```
def outer_func(x):  
    def inner_func(y):  
        return x + y  
  
    return inner_function  
  
closure = outer_function(10)  
result = closure(5)
```

Здесь функция `outer_func` принимает параметр «x» и возвращает внутреннюю функцию `inner_func`. Когда происходит вызов `outer_func(10)`, создаётся замыкание «closure», которое захватывает переменную «x» со значением 10. Затем происходит вызов `closure(5)`, и внутренняя функция `inner_func` использует захваченное значение «x», результатом чего будет 15.

8. Как замыкания могут быть использованы для построения иерархических данных?

Замыкания могут быть использованы для создания функций, которые возвращают другие функции с различными параметрами, формируя иерархию функциональности. Например, фабричные функции или декораторы могут быть реализованы с использованием замыканий для создания динамических функций.

Выводы: В процессе выполнения лабораторной работы были приобретены навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x, был разобран пример, код которого представлен в таблице 1 и в файле `example.py` данного репозитория, а так же выполнено индивидуальное задание, код которого приведён в таблице 2 и файле `individual.py` данного репозитория.