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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЁТ

по лабораторной работе №2.11

Дисциплина: «Программирование на Python»
Тема: «Замыкания в языке Python»

Выполнил:
Епифанов Алексей Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств вычислительной
техники и автоматизированных систем
», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

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

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.

2. Выполнил индивидуальное задание вариант 10: Используя замыкания функций, объявите внутреннюю функцию, которая принимает в качестве аргумента список целых чисел и удаляет из него все четные или нечетные значения в зависимости от значения параметра type . Если type равен «even», то удаляются четные значения, иначе – нечетные. По умолчанию type должно принимать значение «even». Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
def del_items(type_param='even'):
    """
    Выбирается тип элементов для удаления.
    Можно выбрать 'even' - удалятся четные;
    иначе не четные. По умолчанию стоит 'even'.
    """
    def delete(list_items):
        """
        Функция удалет некоторые элементы массива в зависимости от
type_param.
        Type_param задается во внешней функции.
        """
        match type_param:
            case 'even':
                return list(filter(lambda x: x % 2 != 0, list_items))
            case _:
                return list(filter(lambda x: x % 2 == 0, list_items))
    return delete
if __name__ == "__main__":
    start_items = [0, 1, 2, 3, 4, 5, 23, 21, 14, 72]
    print(f"\n{start_items=}\n")
    del_even = del_items()
    print(f"{del_even(start_items)=}\n")
    del_not_even = del_items('not even')
    print(f"{del_not_even(start_items)=}\n")
```

```
start_items=[0, 1, 2, 3, 4, 5, 23, 21, 14, 72]
del_even(start_items)=[1, 3, 5, 23, 21]
del_not_even(start_items)=[0, 2, 4, 14, 72]
```

Рисунок 1. Вывод программы ind

Ответы на контрольные вопросы:

1. Что такое замыкание?

Замыкание (closure) в программировании — это функция, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции в окружающем коде и не являющиеся ее параметрами.

2. Как реализованы замыкания в языке программирования Python?

Замыкания (closures) в языке программирования Python реализованы путем того, что вложенные функции имеют доступ к переменным внешней функции, даже после того, как внешняя функция завершила свою работу.

3. Что подразумевает под собой область видимости Local?

Эту область видимости имеют переменные, которые создаются и используются внутри функций. Доступ к таким переменным снаружи функции невозможен.

4. Что подразумевает под собой область видимости Enclosing?

Суть данной области видимости в том, что внутри функции могут быть вложенные функции и локальные переменные, так вот локальная переменная функции для ее вложенной функции находится в enclosing области видимости.

5. Что подразумевает под собой область видимости Global?

Переменные области видимости global – это глобальные переменные уровня модуля (модуль – это файл с расширением .py).

Доступ к ним можно получить из любой функции, объявленной в данном модуле. Но если этот модуль импортируется в какой- то другой модуль, то из него уже не будет доступа к ним.

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

Уровень Python интерпретатора. В рамках этой области видимости находятся функции open, len и т. п., также туда входят исключения. Эти сущности доступны в любом модуле Python и не требуют предварительного импорта. Built-in – это максимально широкая область видимости.

7. Как использовать замыкания в языке программирования Python?

Пример:

```
def mul(a):
    def helper(b):
        return a * b
    return helper

new_mul5 = mul(5)

new_mul5(2)
```

В данном примере вызывая new_mul5(2), мы фактически обращаемся к функции helper(), которая находится внутри mul(). Переменная a, является локальной для mul(), и имеет область enclosing в helper(). Несмотря на то, что mul() завершила свою работу, переменная a не уничтожается, т.к. на нее сохраняется ссылка во внутренней функции, которая была возвращена в качестве результата.

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

Пример:

```
tpl = lambda a, b: (a, b)
a = tpl(1, 2)
b = tpl(3, a)
c = tpl(a, b)
```

В этом примере в функцию tpl подаются сущности, ей же и порожденные, и в результате «c» будет равно ((1, 2), (3, (1, 2)))

Вывод: в результате выполнения работы были приобретены навыки по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.