МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет по лабораторной работе №14 по дисциплине:

основы программной инженерии

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2021 г.

Ход работы:

1. Область видимости local

```
def add_two(a):
    x = 2
    return a + x
```

2. Область видимости Enclosing

```
def add_four(a):
    x = 2
    def add_some():
        print("x = " + str(x))
        return a + x
    return add_some()
```

3. Пример использования функции mul():

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

4. Более удобный способ вызова функции mul():

```
def mul5(a):
    return mul5(5, a)
```

5. Результат использования замыкания

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

6. Пример использования функции с локальными и глобальными переменными

```
def funl(a):
    x = a * 3

    def fun2(b):
        nonlocal x
        return b + x

    return fun2
```

7. Выполнение свойства замыкания

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

Выполнение индивидуального задания (вариант 13):

Используя замыкания функций, объявите 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 выведите на экран.

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

def convert(type: str):
    def activate(nums: str):
        if type == 'list':
            return list(map(int, nums.split(' ')))
        if type == 'tuple':
            return tuple(map(int, nums.split(' ')))

    return activate


if __name__ == '__main__':
    print(f"List: {convert('list')('1 2 3 4 5 6')}\n"
          f"Tuple: {convert('tuple')('1 2 3 4 5 6')}")
```

```
List: [1, 2, 3, 4, 5, 6]
Tuple: (1, 2, 3, 4, 5, 6)
```

Ответы на контрольные вопросы:

1. Что такое замыкание?

Замыкание – это функция, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции в окружающем коде и не являющиеся её параметрами.

2. Как реализованы замыкания в языке программирования Python?

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

3. Что подразумевает под собой область видимости Local?

Эту область видимости имеют переменные, которые создаются и используются внутри функций.

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 её 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)

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

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**?**

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

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

Это свойство позволяет строить иерархические структуры данных.

```
>>> union = lambda a, b: (a, b)
```

```
>>> x = union(7, 9)
>>> x
# (7, 9)
```

```
>>> y = union(5, a)
>>> y
# (5, (7, 9))


>>> union(x, y)
# ((7, 9), (5, (7, 9)))
```