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

Отчет о лабораторной работе №9

по дисциплине основы программной инженерии

Выполнил: Духно Михаил Александрович,

2 курс, группа ПИЖ-б-о-20-1,

Проверил: Доцент кафедры инфокоммуникаций, Воронкин Р.А.

Ставрополь, 2021 г

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

add_two(3)
#5

print(x)
"""
Traceback (most recent call last):
    File "<pyshell#5>", line 1, in <module>
        print(x)
NameError: name 'x' is not defined
"""
def add_four(a):
    x = 2
    def add_some()
        print('x = ' + str(x))
        return a + x
    return add_some()

add_four(5)
#x = 2
#7

x = 4
def fun()
    print(x+3)

fun()
#7
```

Рисунок 14.1 – Что такое замыкание

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

mul(3, 4)
#12

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

mul5(2)
#10

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

mul(5)(2)
#10

def fun1(a):
    x = a * 3
    def fun2(b):
        nonlocal x
        return b + x
    return fun2

test_fun = fun1(4)

test_fun(7)
#19
```

Рисунок 14.2 – Как использовать замыкания

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

import math


def func(a, b):
    def func2():
        res = math.sqrt(a**2 + b**2)
        res = f'Для значений {a}, {b} функция f({a}, {b}) = {res}'
        return res
    return func2



if __name__ == '__main__':
    a = int(input('a = '))
    b = int(input('b = '))
    print(func(a, b)())
```

Рисунок 14.3 – Код программы индивидуального задания

```
a = 5
b = 5
Для значений 5, 5 функция f(5, 5) = 7.0710678118654755
```

Рисунок 14.4 – Результат работы программы

Контрольные вопросы:

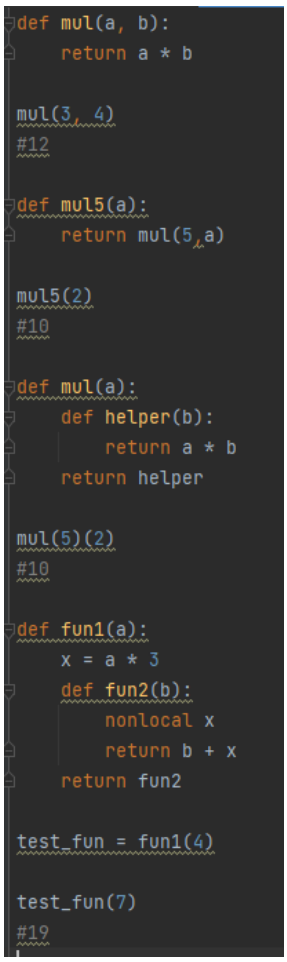
1. Что такое замыкание?

“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?



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

mul(3, 4)
#12

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

mul5(2)
#10

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

mul(5)(2)
#10

def fun1(a):
    x = a * 3
    def fun2(b):
        nonlocal x
        return b + x
    return fun2

test_fun = fun1(4)

test_fun(7)
#19
```

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 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

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

Built-in – это максимально широкая область видимости.

7. Как использовать замыкания в языке программирования Python?

Замыкания позволяют избежать использования глобальных (global) значений и обеспечивают некоторую форму сокрытия данных. Для этого также может использоваться объектно-ориентированный подход.

Если в классе необходимо реализовать небольшое количество методов (в большинстве случаев один метод), замыкания могут обеспечить альтернативное и более элегантное решение. Но когда количество атрибутов и методов становится больше, лучше реализовать класс.

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

Перейдем с уровня математики на уровень функционального программирования. Вот как определяется “свойство замыкания” в книге “Структура и интерпретация компьютерных программ” Айбельсона Х., Сассмана Д. Д. : “В общем случае, операция комбинирования объектов данных обладает свойством замыкания в том случае, если результаты соединения объектов с помощью этой операции сами могут соединяться этой же операцией”.

Это свойство позволяет строить иерархические структуры данных.

Создадим функцию *tpl()*, которая на вход принимает два аргумента и возвращает кортеж. Эта функция реализует операцию “объединения элементов в кортеж”.

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

Если мы передадим в качестве аргументов числа, то, получим простой кортеж.

```
>>> a = tpl(1, 2)
>>> a
(1, 2)
```

Эту операцию можно производить не только над числами, но и над сущностями, ей же и порожденными.

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

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

Таким образом, в нашем примере кортежи оказались замкнуты относительно операции объединения *tpl*. Вспомните аналогию с натуральными числами, замкнутыми относительно сложения.