МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.11 по дисциплине основы программной инженерии

Выполнила:
Лобанова Елизавета Евгеньевна,
2 курс, группа ПИЖ-б-о-20-1

Проверил:
Доцент кафедры прикладной математики и компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2021 г.

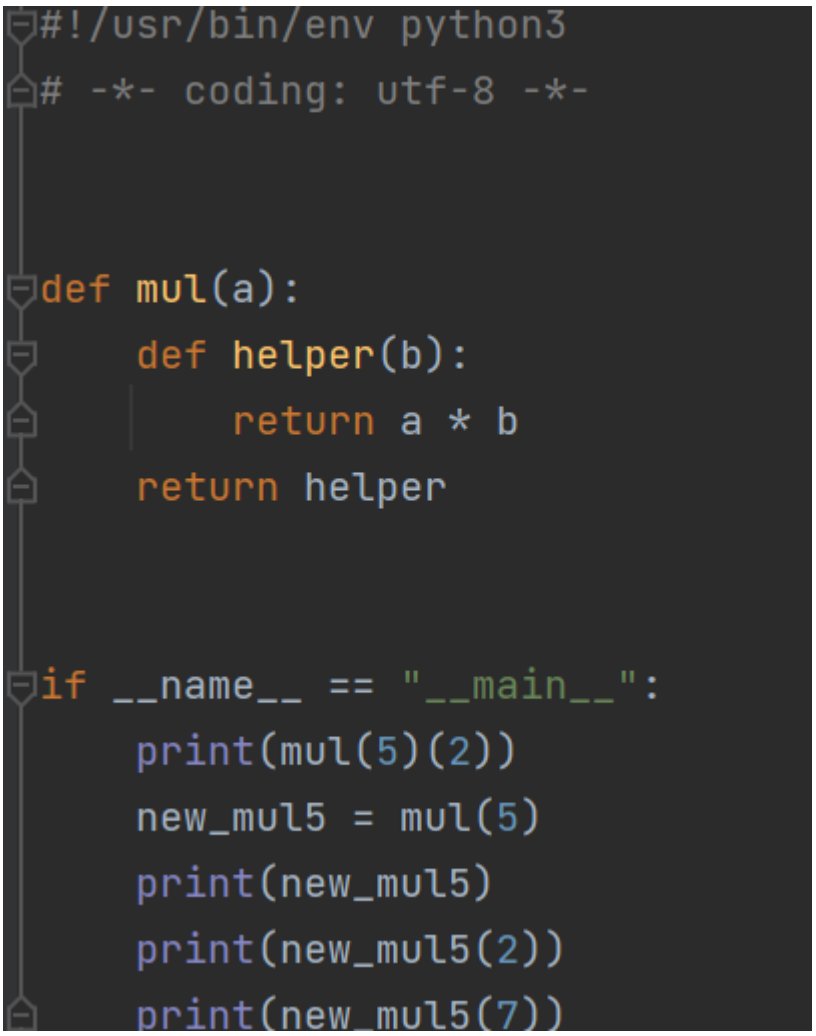
Ход работы

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


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


if __name__ == "__main__":
    print(add_two(3))
    print(x)
```

Пример 1

```
5


Process finished with exit code 1
```

Результат 1 программы (1)



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


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


if __name__ == "__main__":
    print(mul(5)(2))
    new_mul5 = mul(5)
    print(new_mul5)
    print(new_mul5(2))
    print(new_mul5(7))
```

Пример 2

```
10
<function mul.<locals>.helper at 0x000001E91358C280>
10
35

Process finished with exit code 0
```

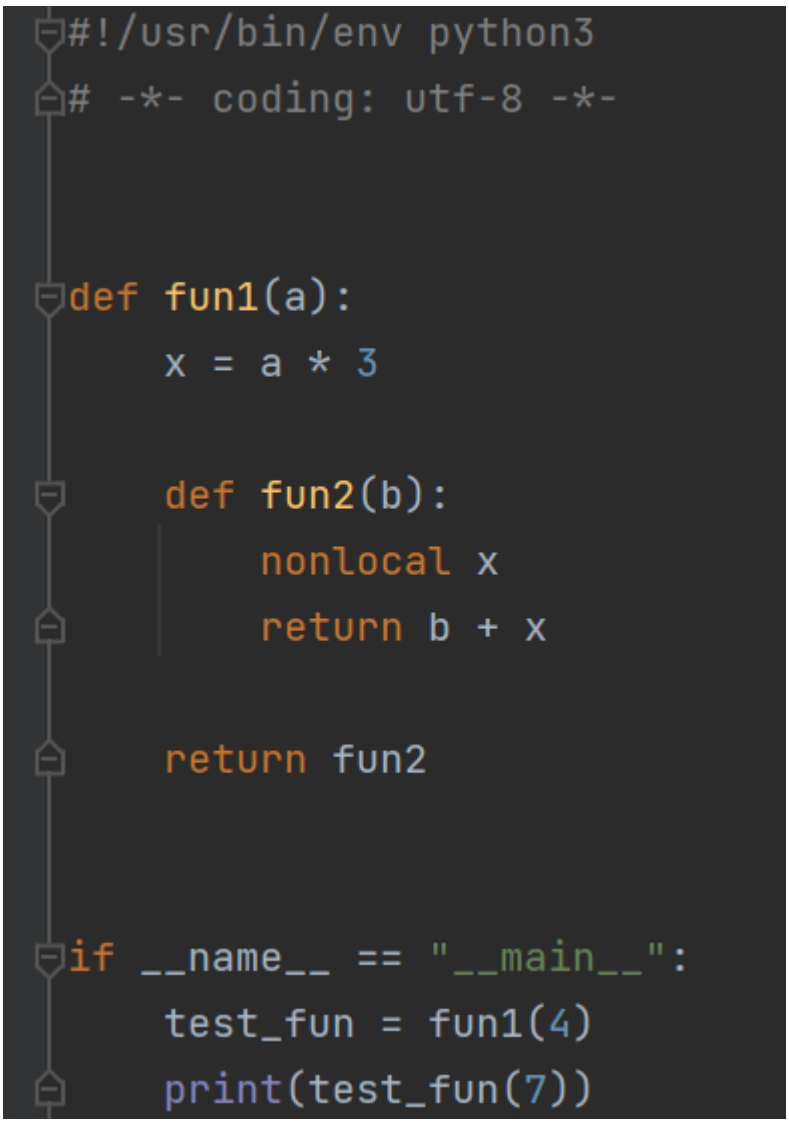
Результат 2 программы (1)

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

x = 4


def fun():
    print(x + 3)


if __name__ == "__main__":
    fun()
```

Пример 3

```
7
5
4

Process finished with exit code 0
```

Результат 3 программы (1)



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


def fun1(a):
    x = a * 3

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

    return fun2


if __name__ == "__main__":
    test_fun = fun1(4)
    print(test_fun(7))
```

Пример 4

```
19

Process finished with exit code 0
```

Результат 4 программы (1)

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

if __name__ == "__main__":
    tpl = lambda a, b: (a, b)
    a = tpl(1, 2)
    print(a)
    b = tpl(3, a)
    print(b)
    c = tpl(a, b)
    print(c)
```

Пример 5

```
(1, 2)
(3, (1, 2))
((1, 2), (3, (1, 2)))

Process finished with exit code 0
```

Результат 5 программы (1)

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

def fun1(tag):
    def fun2(s):
        return f'<{tag}>{s}</{tag}>'
    return fun2


if __name__ == "__main__":
    tag = input("Введите тег: ").split()
    s = input("Введите строку: ").split()
    print(fun1(tag)(s))
```

Индивидуальное задание 1

```
Введите тег: h1
Введите строку: var
<['h1']>['var']</['h1']>

Process finished with exit code 0
```

Результат индивидуального задания 1 (1)

Ответы на 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)
```

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 объектноориентированный 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.