

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-  
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Основы кроссплатформенного программирования**

**Отчет по лабораторной работе №6**

Тема: «Замыкания в языке Python»

Выполнил студент группы

ИВТ-б-о-21-1

Харченко Б.Р. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

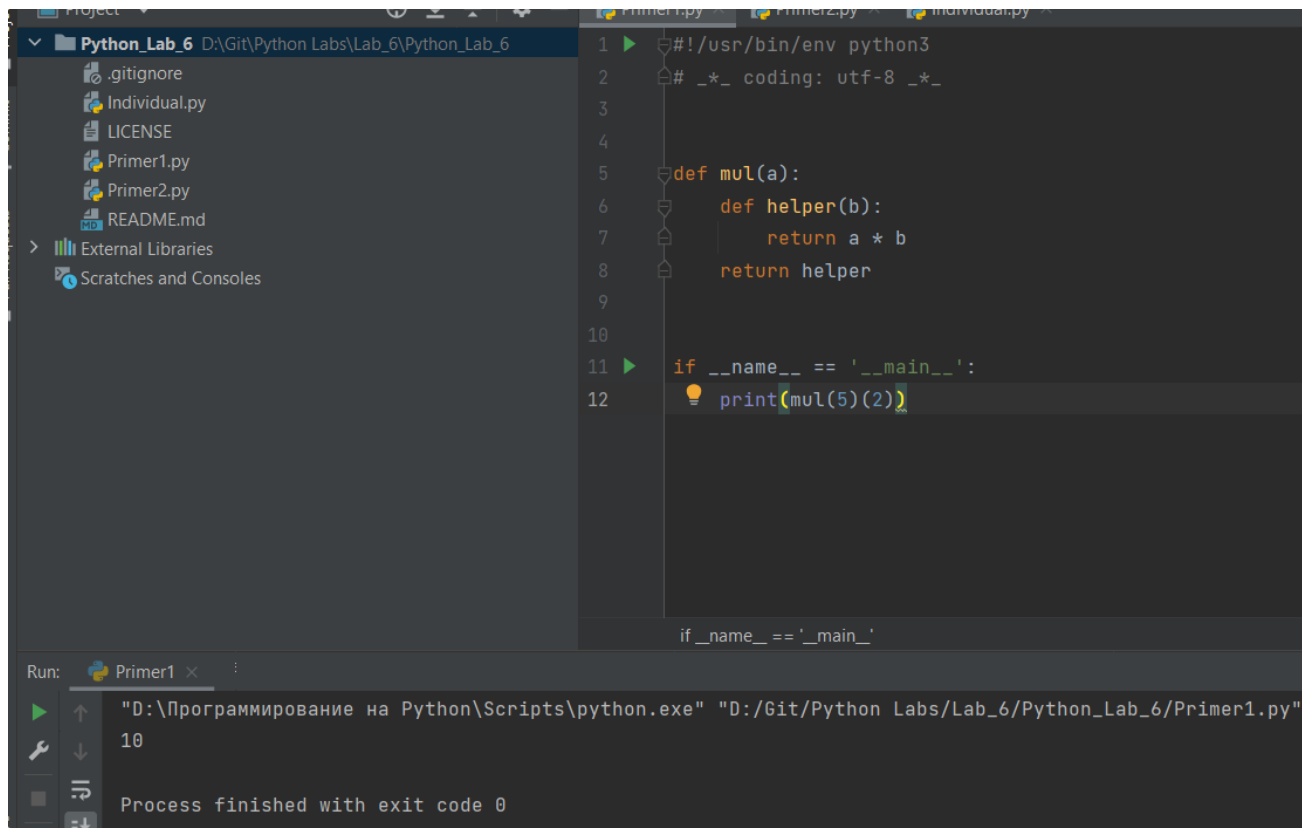
Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2022

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

## Ход работы



The screenshot shows a Python IDE with a project named 'Python\_Lab\_6'. The file explorer on the left lists files: .gitignore, Individual.py, LICENSE, Primer1.py, Primer2.py, README.md, External Libraries, and Scratches and Consoles. The main editor displays the code in 'Primer1.py':

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def mul(a):
6     def helper(b):
7         return a * b
8     return helper
9
10
11 if __name__ == '__main__':
12     print(mul(5)(2))
```

The bottom panel shows the output of running 'Primer1.py':

```
Run: Primer1 x
"D:\Программирование на Python\Scripts\python.exe" "D:/Git/Python Labs/Lab_6/Python_Lab_6/Primer1.py"
10
Process finished with exit code 0
```

Рисунок 1 – Результат выполнения примера №1

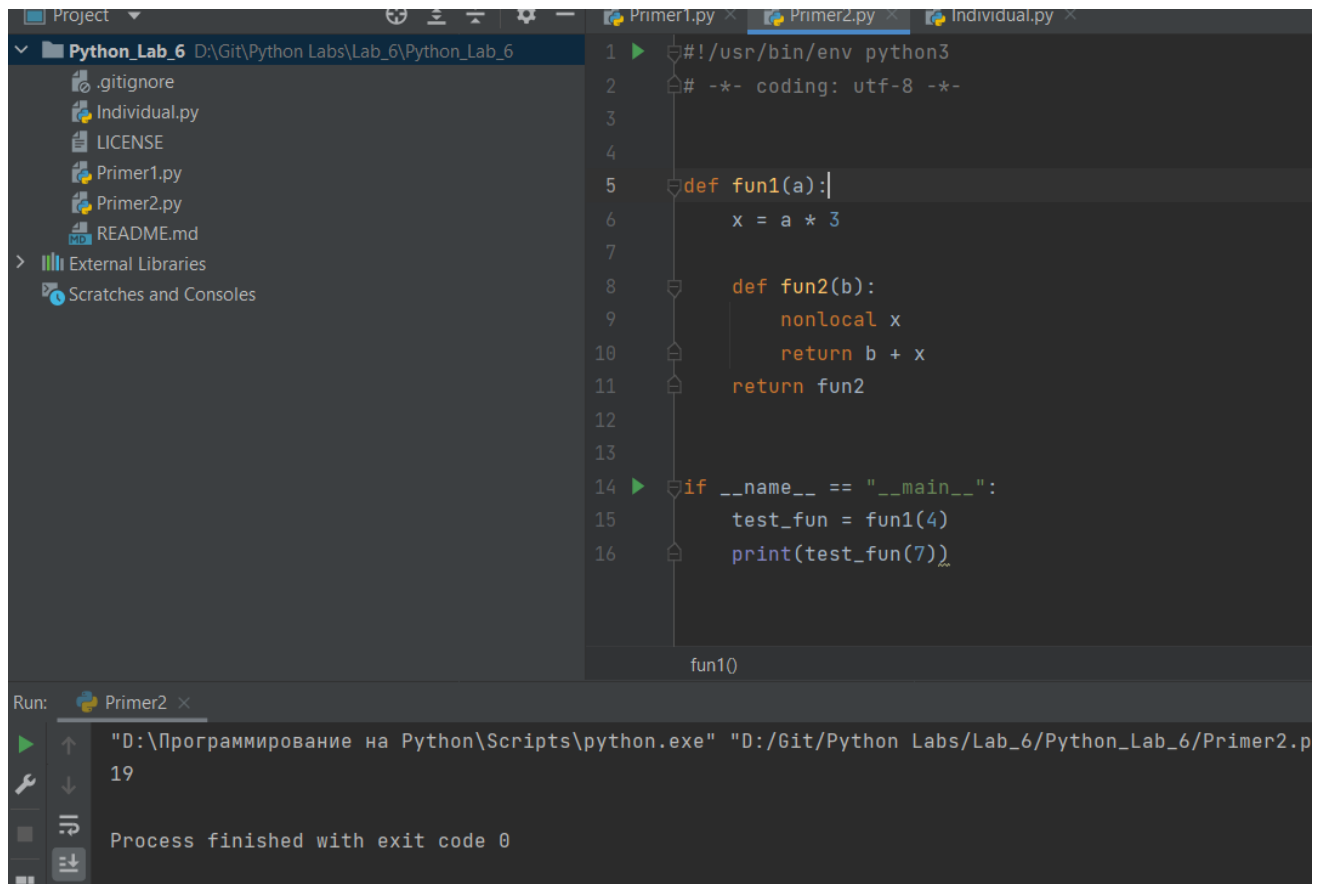


Рисунок 2 – Результат выполнения примера №2

## 1. Индивидуальное задание. Вариант – 24.

Используя замыкания функций, объявите внутреннюю функцию, которая из переданного ей списка строк формирует многострочную строку вида:

<ol>

<li>строка\_1</li>

...

<li>строка\_N</li>

</ol>

и возвращает ее. Где строка1, строка2, ... - это строки из переданного функции списка.

Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.

The screenshot shows a Python IDE with a project named 'Python\_Lab\_6'. The file explorer on the left lists files: .gitignore, Individual.py, LICENSE, Primer1.py, Primer2.py, and README.md. The main editor displays the code in 'Individual.py':

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def strok():
6     def ol(spisok):
7         li = '</li>\n<li>'.join(spisok)
8         return f"<ol>\n<li>{li}</li>\n</ol>"
9     return ol
10
11
12 if __name__ == '__main__':
13     my_list = [
14         'строка_1',
15         'строка_2',
16         'строка_3',
17         'строка_4'
18     ]
19
20 if __name__ == '__main__':
```

The 'Run' console at the bottom shows the command executed: "D:\Программирование на Python\Scripts\python.exe" "D:/Git/Python Labs/Lab\_6/Python\_Lab\_6/Individual.py". The output is:

```
<ol>
<li>строка_1</li>
<li>строка_2</li>
<li>строка_3</li>
```

Рисунок 3 – Результат выполнения программы

**Вывод:** в результате выполнения лабораторной работы были приобретены практические навыки и теоретические сведения по работе с замыканиями





при написании программ с помощью языка программирования Python версии 3.x..

### **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 в Python реализованы посредством манипулирования областью видимости функций.

#### **3. Что подразумевает под собой область видимости Local?**

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

```
>>> 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
```

Пример. В данной программе объявлена функция `add_two()`, которая прибавляет двойку к переданному ей числу и возвращает полученный результат. Внутри этой функции используется переменная `x`, доступ к которой снаружи невозможен. К тому же, эта переменная удаляется из памяти каждый раз (во всяком случае, должна удаляться), когда завершается `add_two()`.

#### 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.

```
>>> 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
```

#### 5. Что подразумевает под собой область видимости **Global**?

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

```
>>> x = 4  
  
>>> def fun():  
  
    print(x+3)  
  
>>> fun()
```



## **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?**

Для создания замыкания в Python, должны быть выполнены следующие пункты:

- У нас должна быть вложенная функция (функция внутри функции);
- вложенная функция должна ссылаться на значение, определенное в объемлющей функции;
- объемлющая функция должна возвращать вложенную функцию.

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

Начнем разбор данного термина с математической точки зрения, а точнее с алгебраической. Предметом алгебры является изучение алгебраических структур – множеств с определенными на них операциями. Под множеством обычно понимается совокупность определенных объектов. Наиболее простым примером числового множества, является множество натуральных чисел. Оно содержит следующие числа: 1, 2, 3, ... и т.д. до бесконечности. Иногда, к этому множеству относят число ноль, но мы не будем этого делать. Над элементами этого множества можно производить различные операции, например, сложение.

Какие бы натуральные числа мы не складывали, всегда будем получать натуральное число. С умножением точно также. Но с вычитанием и делением это условие не выполняется.

Среди натуральных чисел нет числа  $-3$ , для того, чтобы можно было использовать вычитание без ограничений, нам необходимо расширить множество натуральных чисел до множества целых чисел:

Таким образом, можно сказать, что множество натуральных чисел замкнуто относительно операции сложения – какие бы натуральные числа мы не складывали, получим натуральное число, но это множество не замкнуто относительно операции вычитания.

Теперь 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”**.