

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ
ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Дисциплина: «Языки программирования»

Отчет по лабораторной работе №9

Замыкания в языке Python

Выполнил студент группы ИТС-б-о-21-1

Усиков Данил

« » _____ 20__ г.

Подпись студента _____

Проверил: Доцент, к.т.н, доцент

кафедры инфокоммуникаций

Воронкин Р. А.

(подпись)

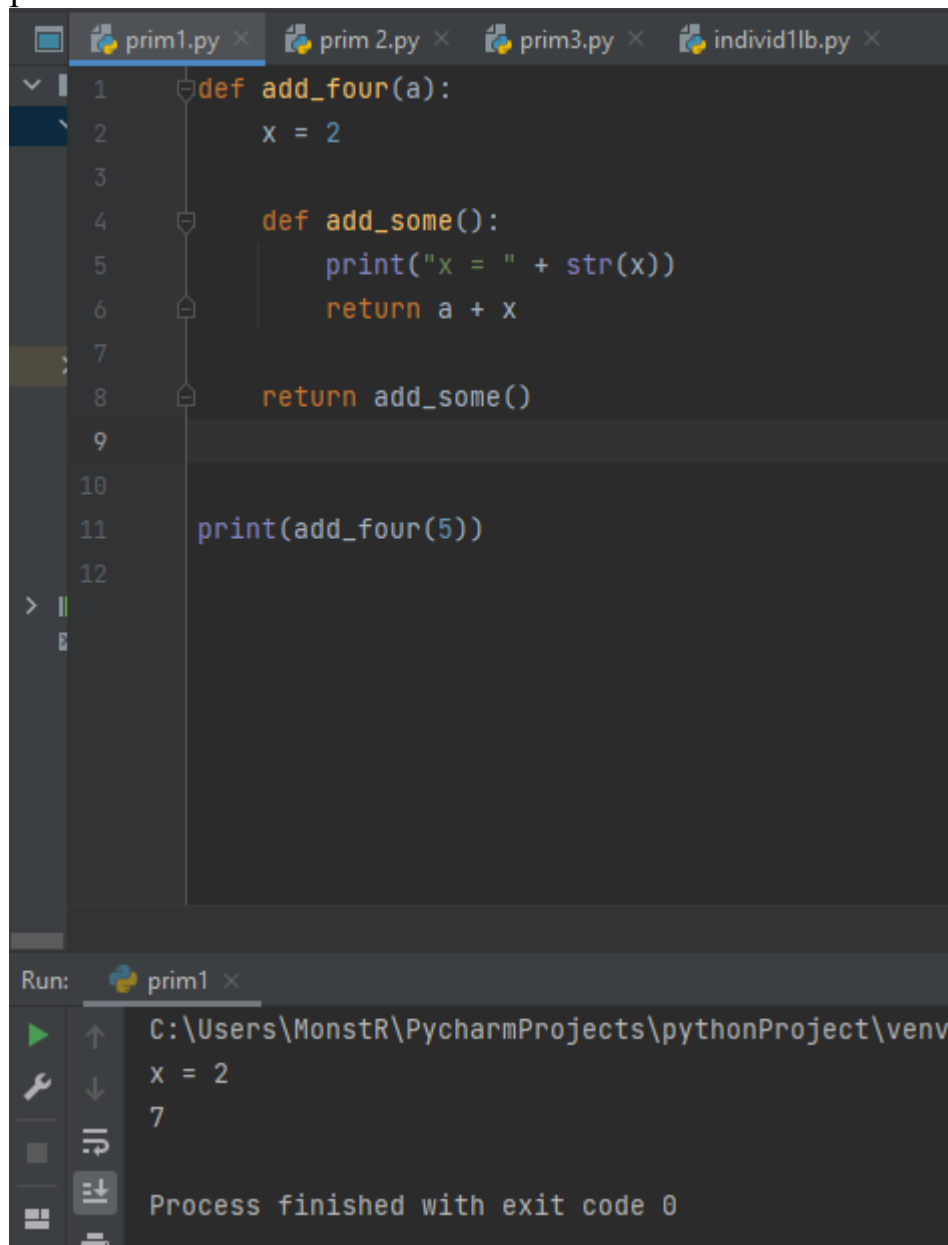
Ставрополь 2022

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

Ссылка на репозиторий - <https://github.com/danilusikov0913/YPlr1>

Ход работы:

Пример №1:



The screenshot displays the PyCharm IDE interface. The top pane shows a Python script named `prim1.py` with the following code:

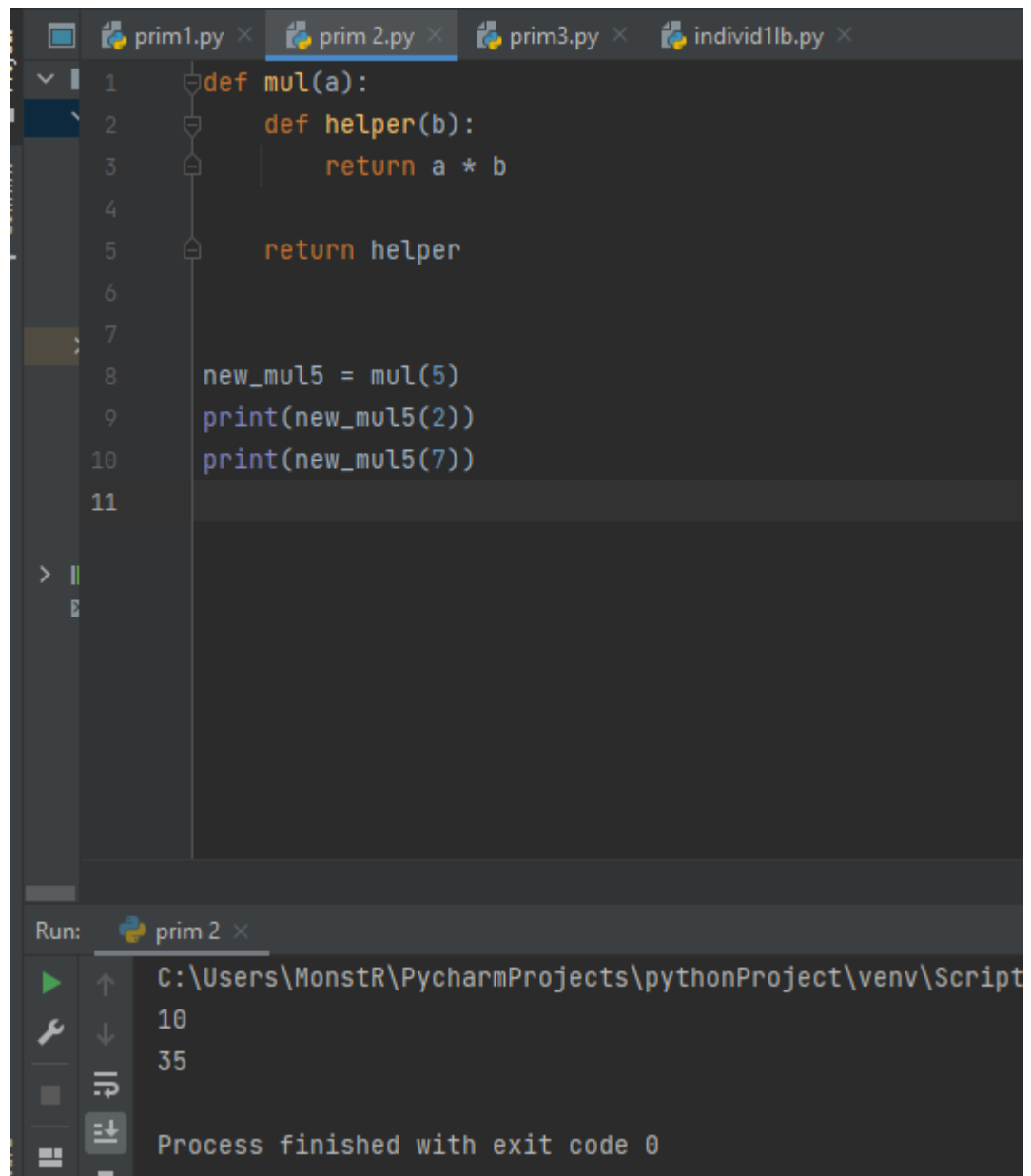
```
1 def add_four(a):
2     x = 2
3
4     def add_some():
5         print("x = " + str(x))
6         return a + x
7
8     return add_some()
9
10
11 print(add_four(5))
12
```

The bottom pane shows the output of the program execution. The output is as follows:

```
Run: prim1 x
C:\Users\Monstr\PycharmProjects\pythonProject\venv
x = 2
7
Process finished with exit code 0
```

Рисунок 1. Код и результат выполнения программой примера 1

Пример №2:



The screenshot shows the PyCharm IDE with four tabs: `prim1.py`, `prim 2.py` (active), `prim3.py`, and `individ1lb.py`. The active tab contains the following Python code:

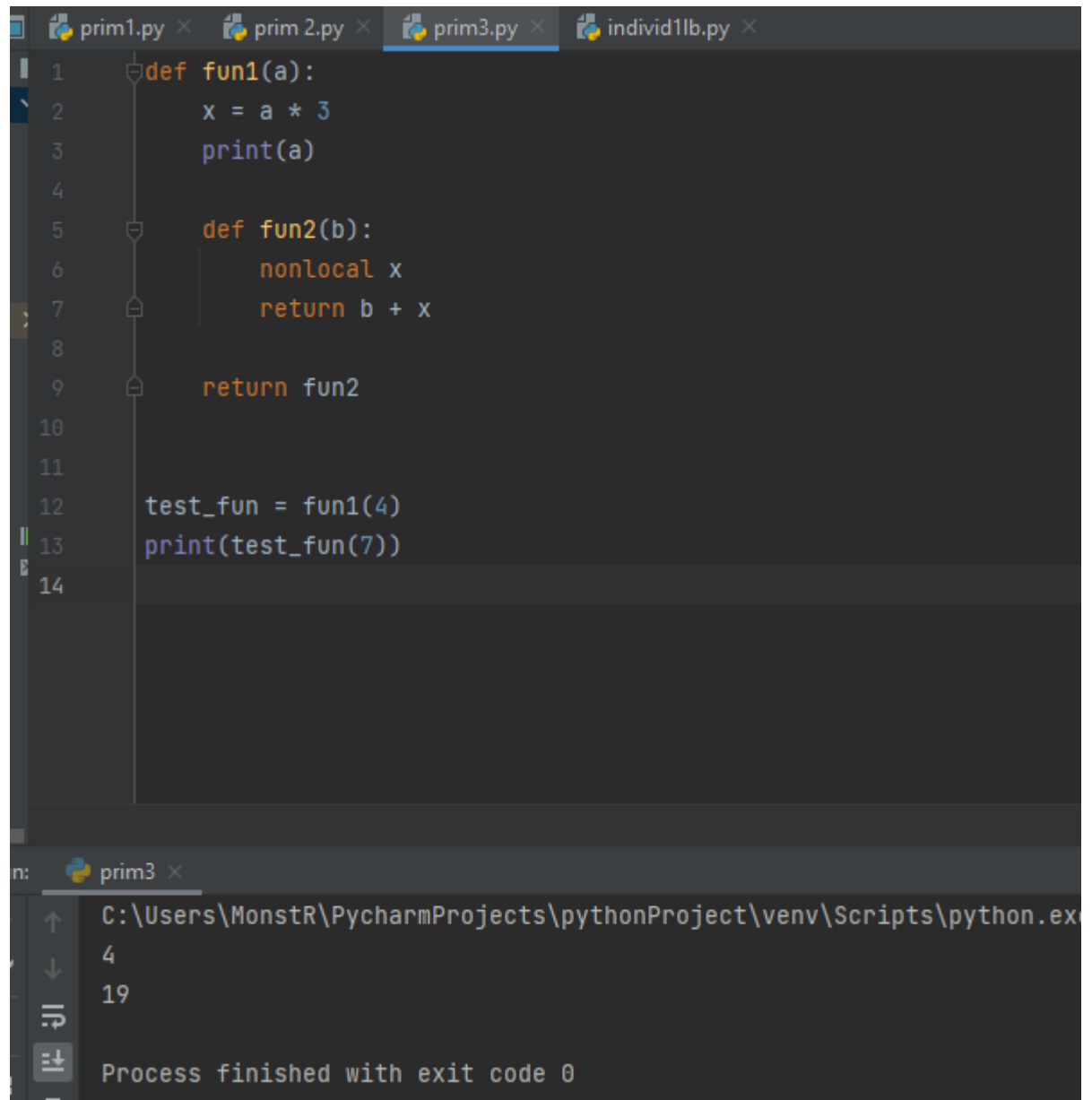
```
1 def mul(a):  
2     def helper(b):  
3         return a * b  
4  
5     return helper  
6  
7  
8 new_mul5 = mul(5)  
9 print(new_mul5(2))  
10 print(new_mul5(7))  
11
```

Below the editor, the Run console for `prim 2` shows the execution path and output:

```
Run: prim 2 ×  
C:\Users\MonstR\PycharmProjects\pythonProject\venv\Script  
10  
35  
Process finished with exit code 0
```

Рисунок 2. Код и результат выполнения программой примера 2

Пример №3:



The screenshot shows a PyCharm IDE with four tabs: prim1.py, prim 2.py, prim3.py (active), and individ1lb.py. The active tab contains the following Python code:

```
1 def fun1(a):  
2     x = a * 3  
3     print(a)  
4  
5     def fun2(b):  
6         nonlocal x  
7         return b + x  
8  
9     return fun2  
10  
11  
12 test_fun = fun1(4)  
13 print(test_fun(7))  
14
```

Below the code editor, the Run tool window is visible, showing the execution path and output:

```
Python 3.9.6  
C:\Users\MonstR\PycharmProjects\pythonProject\venv\Scripts\python.exe  
4  
19  
Process finished with exit code 0
```

Рисунок 3. Код и результат выполнения программой примера 3

Вариант 20(10)

Индивидуальное задание:

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.

The screenshot displays the PyCharm IDE with three open files: `individ1lb.py`, `individlb2.py`, and `lb6prim1.py`. The `individ1lb.py` file is active, showing a Python script with the following code:

```
11 def delchis(list1):
12     def delchetnechet():
13         type = input('Введите even или noteven:\n')
14         if type == 'even':
15             for x in range(len(list1)):
16                 if x % 2 == 0:
17                     list1.pop(list1.index(x))
18                     print(list1)
19             elif type == 'noteven':
20                 for x in range(len(list1)):
21                     if x % 2 != 0:
22                         list1.pop(list1.index(x))
23                         print(list1)
24             else:
25                 print('Введена не правильная команда')
26
27     return delchetnechet
28
29 f = delchis(list1=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
30 f()
31
32
33 if __name__ == '__main__':
34     main()
```

The execution console at the bottom shows the output of the program:

```
C:\Users\MonstR\PycharmProjects\pythonProject\venv\Scripts\python.exe "D:/U
Введите even или noteven:
even
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 3, 5, 6, 7, 8, 9, 10]
[1, 3, 5, 7, 8, 9, 10]
[1, 3, 5, 7, 9, 10]
[1, 3, 5, 7, 9]
```

Рисунок 3. Код и результат выполнения программой индивидуального задания

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? С помощью функций.

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?

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). Но если мы этот модуль импортируем в каком-то другом модуле, то глобальная переменная для него уже не будет переменной уровня global.

6. Что подразумевает под собой область видимости Built-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?

```
>>> new_mul5 = mul(5)

>>> new_mul5
<function mul.<locals>.helper at 0x000001A7548C1158>

>>> new_mul5(2)
10

>>> new_mul5(7)
35
```

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 ссылочно внутренней функции, которая была возвращена в качестве результата.

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

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.

Вывод: приобрели навыки по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.