

**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 №14 по дисциплине:
«Основы программной инженерии»**

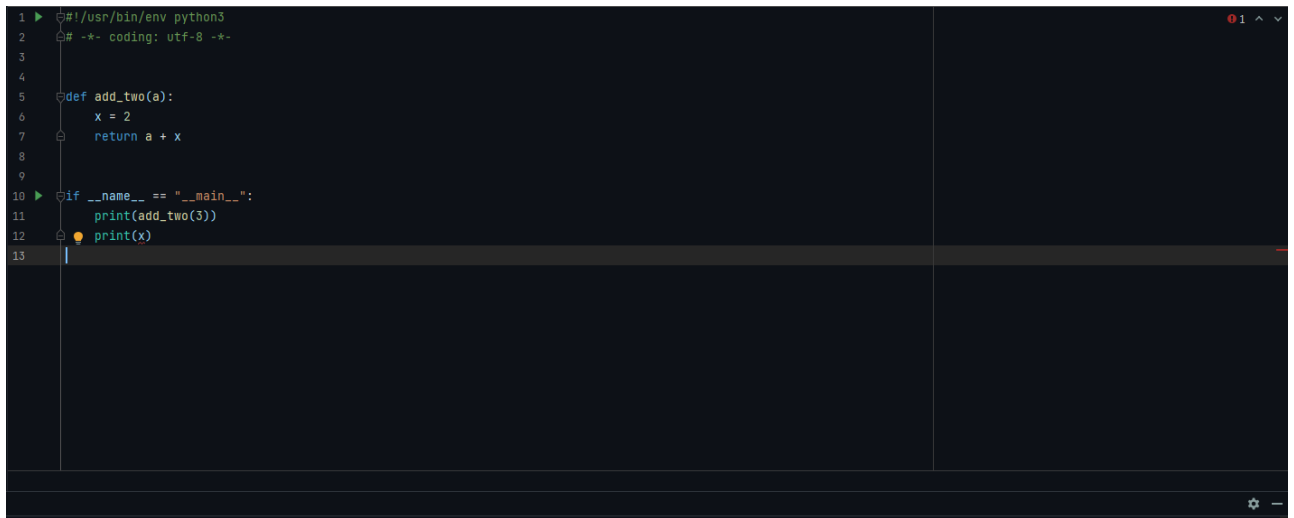
Выполнил:
Гребе Владислав
Александрович,
2 курс, группа ПИЖ-б-о-20-1,

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

Отчет защищен с оценкой_____Дата защиты_____

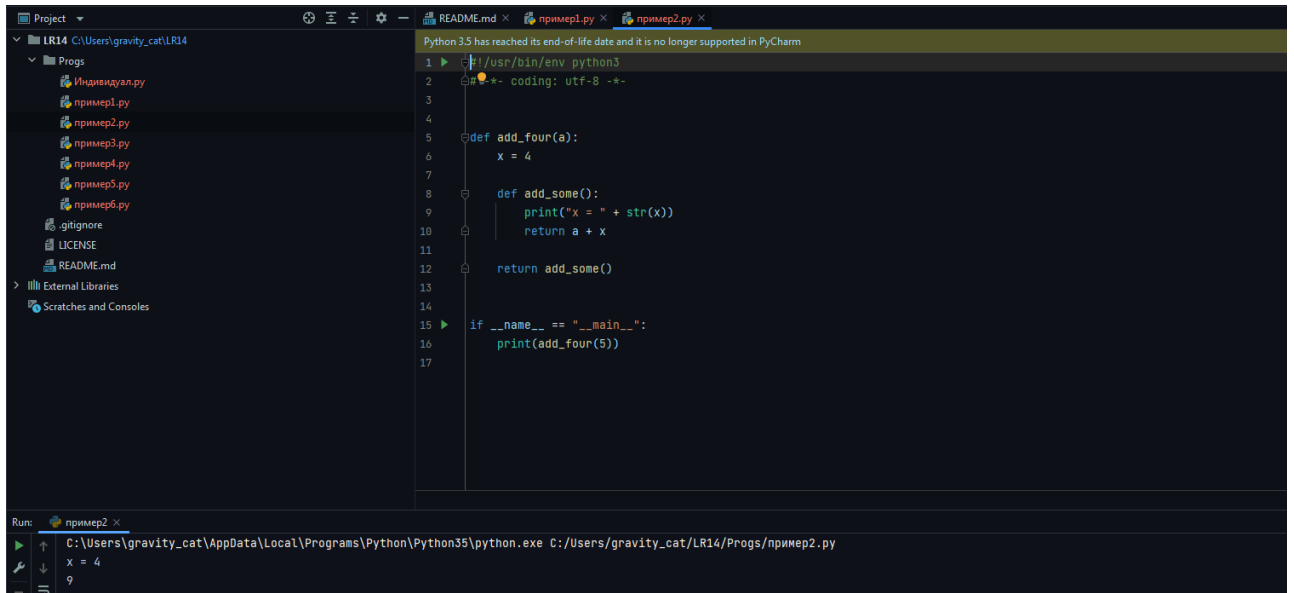
Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ:



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def add_two(a):
6     x = 2
7     return a + x
8
9
10 if __name__ == "__main__":
11     print(add_two(3))
12     print(x)
13
```

Рисунок 14.1 – Пример №1



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def add_four(a):
6     x = 4
7
8     def add_some():
9         print("x = " + str(x))
10        return a + x
11
12    return add_some()
13
14
15 if __name__ == "__main__":
16     print(add_four(5))
17
```

Run: `пример2.py`

`C:\Users\gravity_cat\AppData\Local\Programs\Python\Python35\python.exe C:/Users/gravity_cat/LR14/Progs/пример2.py`

`x = 4`
`9`

Рисунок 14.2 – Пример №2

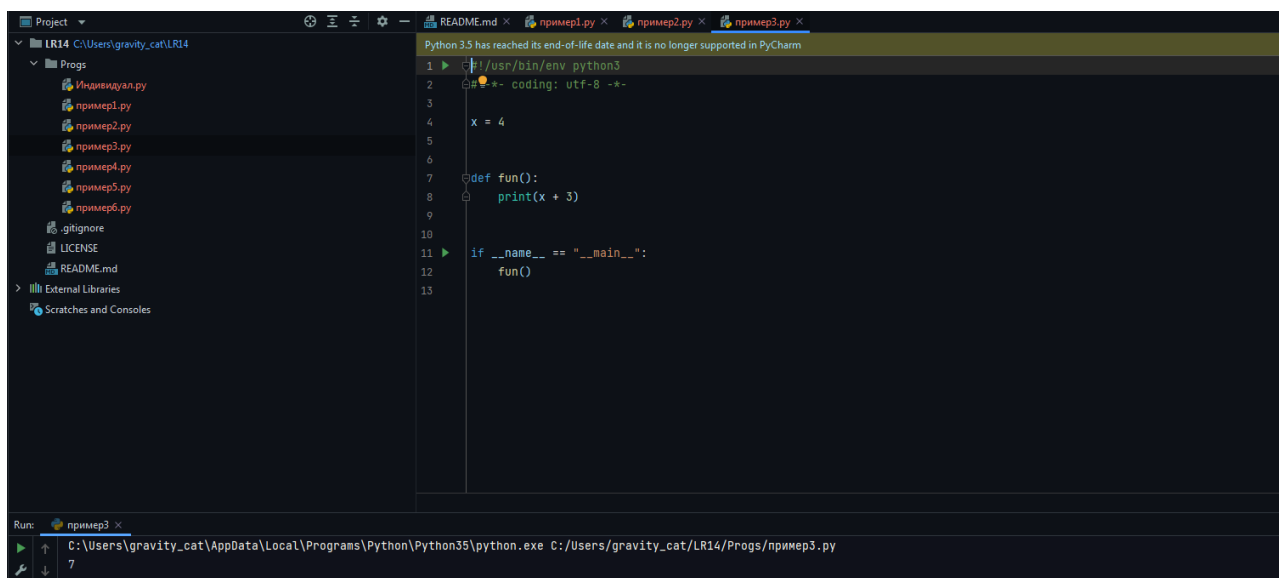


Рисунок 14.3 – Пример №3

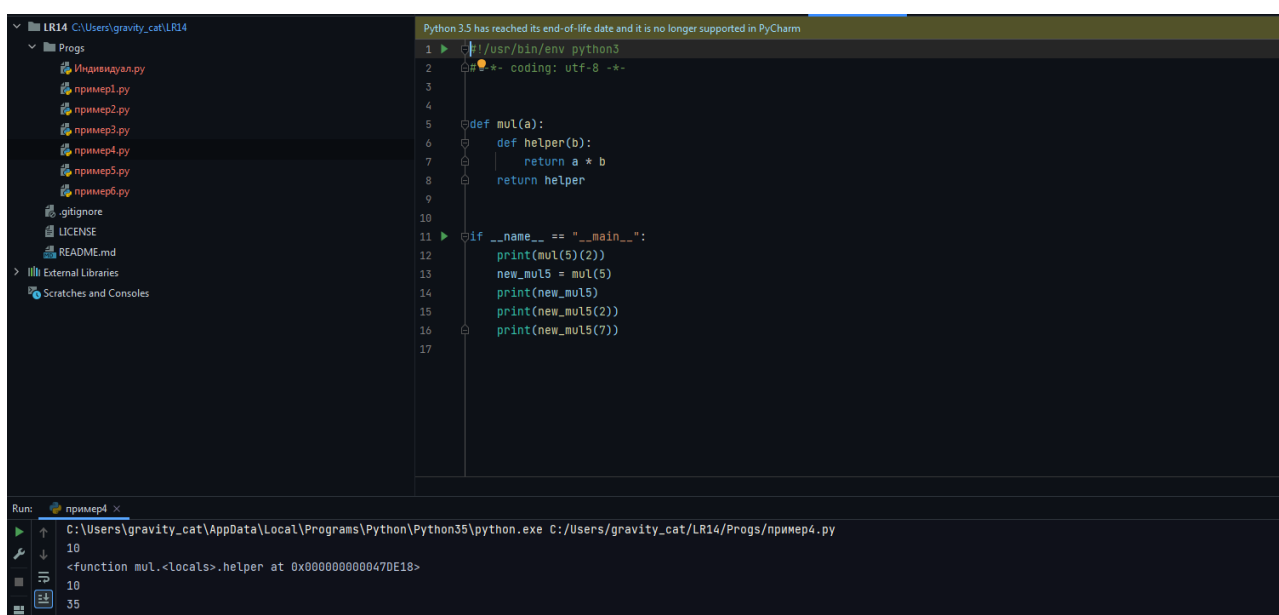


Рисунок 14.4 – Пример №4

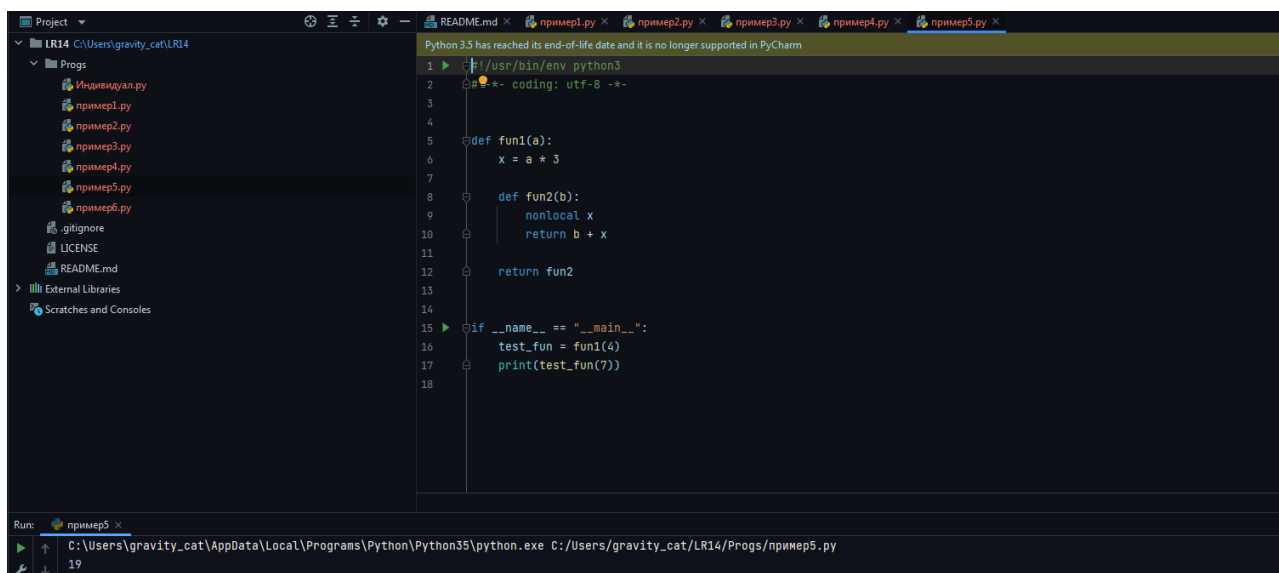


Рисунок 14.5 – Пример №5

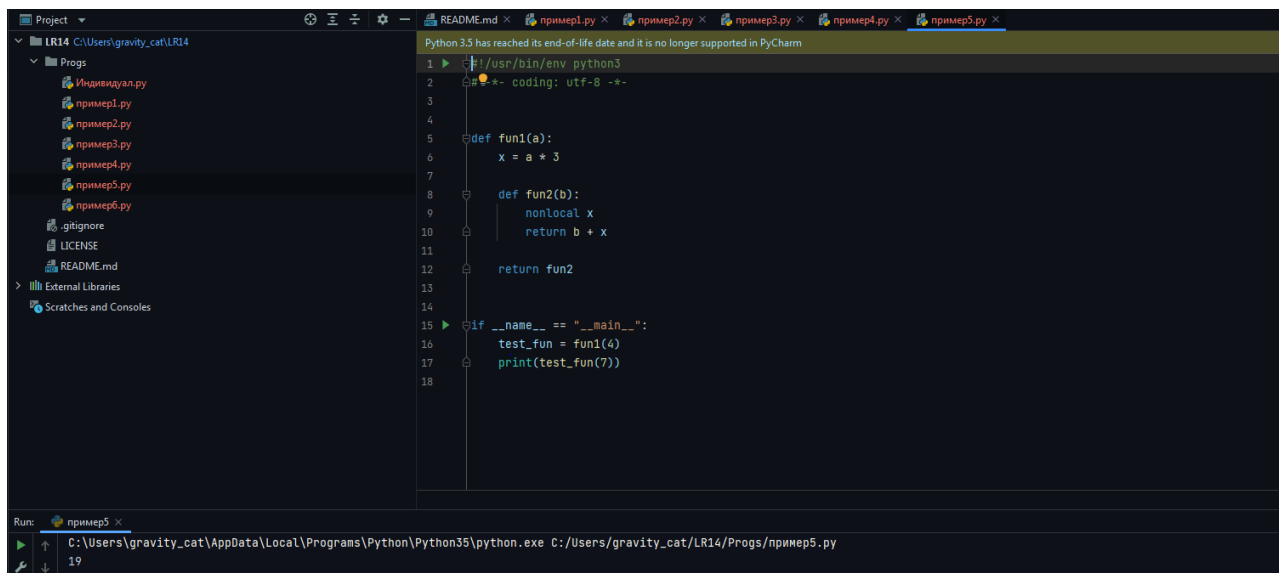
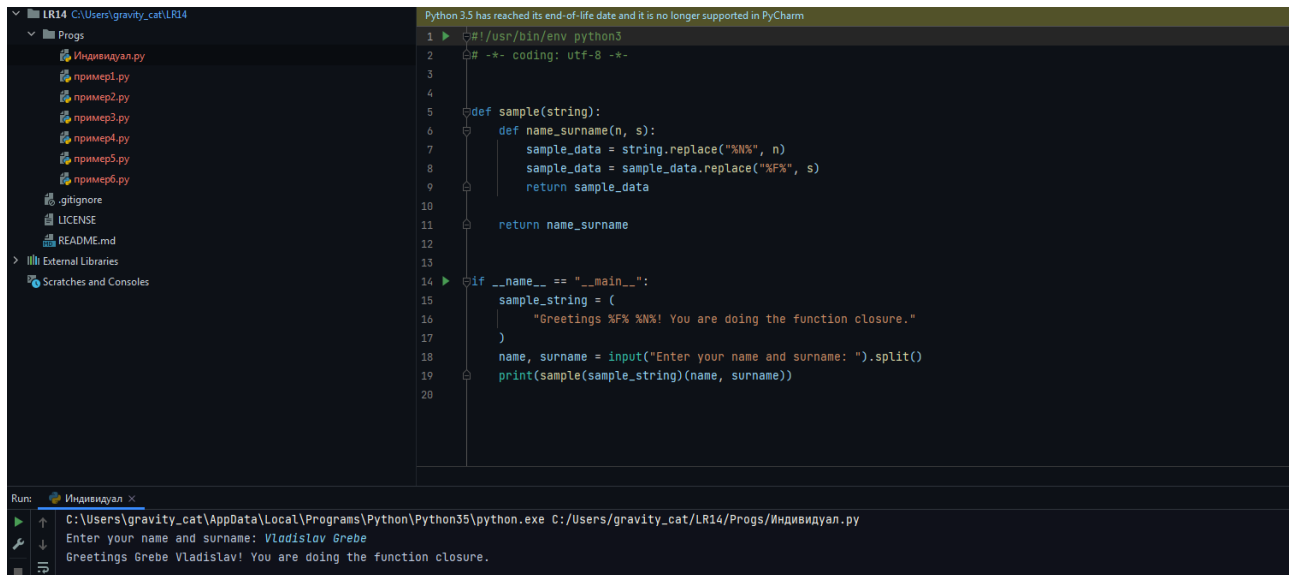


Рисунок 14.6 – Пример №6

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Вариант 5



The screenshot shows the PyCharm IDE interface. On the left, the 'Progs' directory contains files named 'Индивидуал.py', 'пример1.py', 'пример2.py', 'пример3.py', 'пример4.py', 'пример5.py', and 'пример6.py'. The main editor displays a Python script with the following code:

```
1 #!/usr/bin/env python3
2 #- coding: utf-8 -*-
3
4
5 def sample(string):
6     def name_surname(n, s):
7         sample_data = string.replace("%NN%", n)
8         sample_data = sample_data.replace("%FX%", s)
9         return sample_data
10
11     return name_surname
12
13
14 if __name__ == "__main__":
15     sample_string = (
16         "Greetings %FX %NN%! You are doing the function closure."
17     )
18     name, surname = input("Enter your name and surname: ").split()
19     print(sample(sample_string)(name, surname))
20
```

At the bottom, the 'Run' console shows the execution of the script 'Индивидуал.py' using the Python 3.5 interpreter. The output is:

```
Enter your name and surname: Vladislav Grebe
Greetings Grebe Vladislav! You are doing the function closure.
```

Рисунок 14.6 - ИДЗ

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? Необходимо объявить вложенную функцию в объемлющей функции. Эта вложенная функция должна ссылаться на значение переменных, объявленных в объемлющей функции, необходимо, чтобы объемлющая функция возвращала значение вложенной функции.
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 (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? Пример:

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

Использование: `mul(5)(2)`
или `new_mul5 = mul(5)`, `new_mul5(2)`
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. Это свойство позволяет строить иерархические структуры данных. Покажем это на примере кортежей в Python: `tpl = lambda a, b: (a, b)` `a = tpl(1, 2)` `(1, 2)` `b = tpl(3, a)` `(3, (1, 2))` `c = tpl(a, b)` `((1, 2), (3, (1, 2)))`