

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Замыкания в языке Python»

ОТЧЕТ
по лабораторной работе №14
дисциплины
«Основы программной инженерии»

Выполнил:

Сотников Андрей Александрович
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная
инженерия», направленность
(профиль) «Разработка и
сопровождение программного
обеспечения», очная форма
обучения

(подпись)

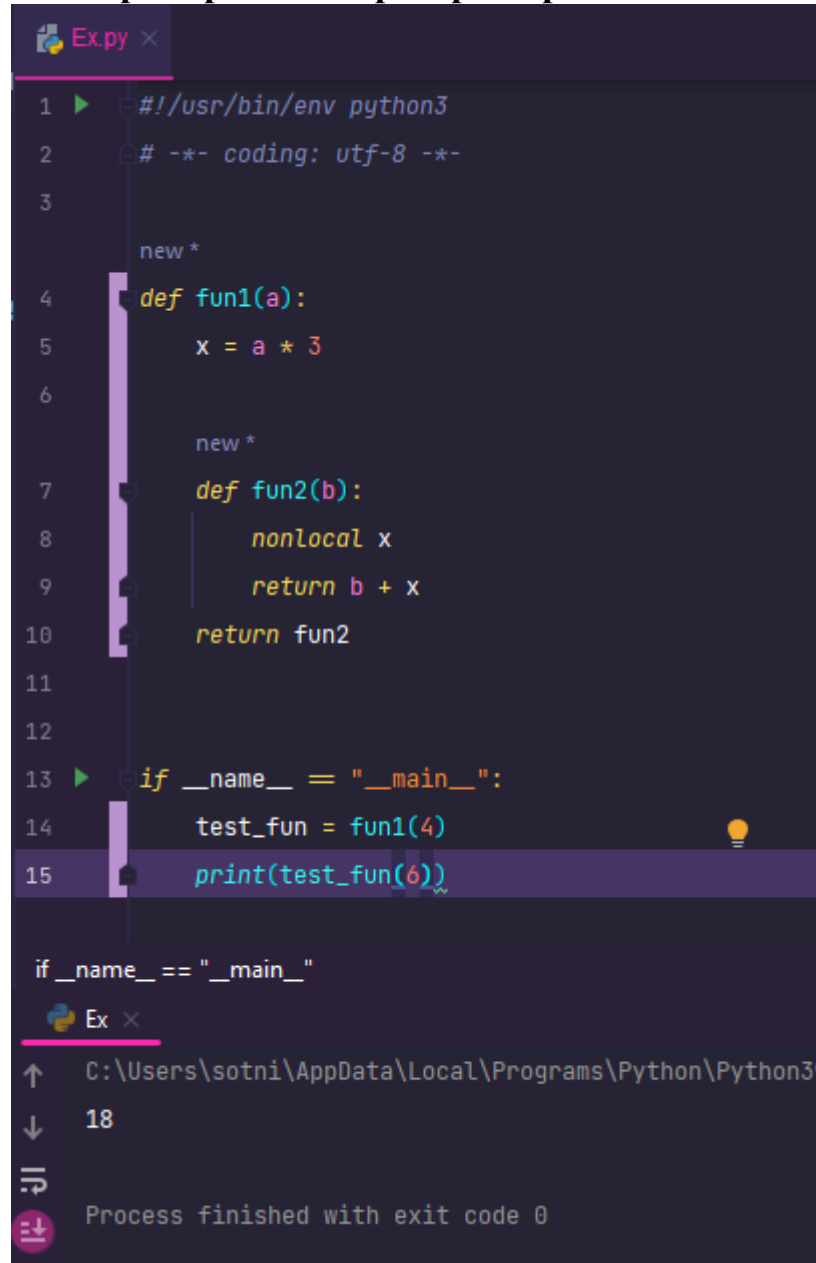
Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

Проработка примера из лабораторной работы:



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  new *
5  def fun1(a):
6      x = a * 3
7
8      new *
9      def fun2(b):
10         nonlocal x
11         return b + x
12     return fun2
13
14 if __name__ == "__main__":
15     test_fun = fun1(4)
16     print(test_fun(6))
```

if __name__ == "__main__"

Ex x

↑ C:\Users\sotni\AppData\Local\Programs\Python\Python3

↓ 18

Process finished with exit code 0

Рисунок 1 – Код и результат работы примера

Индивидуальное задание: 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.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  AndrejMirrox *
5  def del1(type="even"):
6      AndrejMirrox *
7      def del2(b):
8          if type == 'even':
9              b = [num for idx, num in enumerate(b) if num % 2 != 0]
10             return b
11         else:
12             b = [num for idx, num in enumerate(b) if num % 2 == 0]
13             return b
14         return del2
15
16 if __name__ == "__main__":
17     list = list(map(int, input("Введите список: ").split()))
18     com = input("Введите параметр функции: ")
19     print(f"Тест: {del1(com)(list)}")
```

Individual x

C:\Users\sotni\AppData\Local\Programs\Python\Python39\python.exe C:\labor-14\PyChar

Введите список: 4 1 2 9 13 7 6 8 5 1

Введите параметр функции: even

Тест: [1, 9, 13, 7, 5, 1]

Рисунок 2 – Код и результат работы индивидуального задания

Контрольные вопросы

1. Что такое замыкание?

Для начала обратимся к википедии: “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.” Перед тем как перейти к рассмотрению примеров реализации замыканий на Python, для начал вспомним тему “область видимости переменных”. Обычно, по области видимости, переменные делят на глобальные и локальные. Глобальные существуют в течении всего времени выполнения программы, а локальные создаются внутри методов, функций и прочих блоках кода, при этом, после выхода из такого блока переменная удаляется из памяти.

2. Как реализованы замыкания в языке программирования Python?

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

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 Built-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?

```
>>> def fun1(a):  
    x = a * 3  
    def fun2(b):  
        nonlocal x  
        return b + x  
    return fun2  
  
>>> test_fun = fun1(4)  
  
>>> test_fun(7)  
19
```

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

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”. Это свойство позволяет строить иерархические структуры данных. Покажем это на примере кортежей в Python.