Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО - КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №14**

**дисциплины «Программирование на Python»**

**Вариант 23**

|  |  |  |
| --- | --- | --- |
|  | Выполнил:  Омонкулов Исомиддин Валижон угли  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | Руководитель практики:  Воронкин Р. А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024

**Тема:** Замыкания в языке Python

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

**Индивидуальное задание.** Используя замыкания функций, объявите внутреннюю функцию, которая на основе двух параметров вычисляет площадь фигуры. Какой именно фигуры: треугольника или прямоугольника, определяется параметром type внешней функции. Если type принимает значение 0, то вычисляется площадь треугольника, а иначе – прямоугольника. По умолчанию параметр type должен быть равен 0. Вычисленное значение должно возвращаться внутренней функцией. Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.

**Листинг к программе:**

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
def calculate\_area(type=0):  
 def inner\_func():  
 if type == 0:  
 a = float(input("Введите значение первой стороны треугольника: "))  
 b = float(input("Введите значение второй стороны треугольника: "))  
 c = float(input("Введите значение третьей стороны треугольника: "))  
  
 # Расчитаем площадь треугольника по формуле Герона  
 p = (a + b + c) / 2  
 area = (p \* (p - a) \* (p - b) \* (p - c)) \*\* 0.5  
  
 return area  
 else:  
 a = float(input("Введите значение первой стороны прямоугольника: "))  
 b = float(input("Введите значение второй стороны прямоугольника: "))  
  
 # Расчет площади прямоугольника  
 area = a \* b  
  
 return area  
  
 return inner\_func  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 type = int(input("Введите тип фигуры (0 - треугольник, 1 - прямоугольник): "))  
 calculate = calculate\_area(type)  
 area = calculate()  
 print("Площадь фигуры:", area)

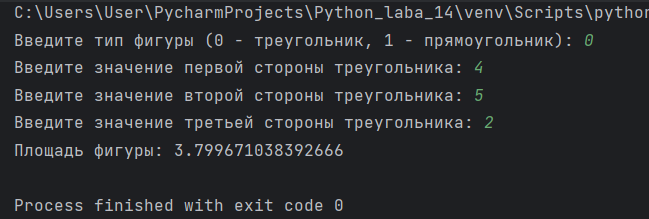


Рисунок 1. Тест №1

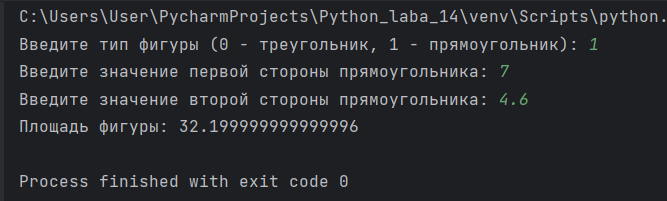


Рисунок 2. Тест №2

**Вывод:** в ходе выполнения данной лабораторной работы были приобретены навыки по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.

**Ответы на контрольные вопросы**

1. Замыкание - это функция, которая запоминает значения из окружающего её лексического контекста, даже если эта функция выполняется вне этого контекста.

2. В Python замыкания реализуются путем определения функции внутри другой функции и возвращения этой внутренней функции в качестве результата внешней функции.

3. Область видимости Local относится к переменным, определенным внутри текущей функции. Эти переменные видны только внутри этой функции.

4. Область видимости Enclosing относится к переменным, определенным в объемлющей (внешней) функции. Эти переменные видны внутри вложенной функции.

5. Область видимости Global относится к переменным, определенным на верхнем уровне модуля или файла. Эти переменные видны из любого места внутри этого модуля.

6. Область видимости Build-in относится к встроенным именам и функциям, которые предоставляются интерпретатором Python.

7. Для использования замыканий в Python нужно определить внутреннюю функцию внутри внешней функции, а затем вернуть эту внутреннюю функцию в качестве результата внешней функции. Пример использования замыкания приведен в предыдущем ответе.

8. Замыкания могут быть использованы для построения иерархических данных путем создания функций, которые возвращают другие функции. Это позволяет создавать цепочки функций с различными уровнями доступа к переменным и данным.