

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**«Замыкания в Python»**

**Отчет по лабораторной работе № 2.11  
по дисциплине «Программирование на Python»**

Выполнил студент группы ИВТ-б-о-21-1

Шайдеров Дмитрий Викторович.

«31 » октября 2022г.

Подпись студента \_\_\_\_\_

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

Проверил 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.

**Порядок выполнения работы:**

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

Owner \* Repository name \*

dshayderov / lw\_2.11

Great repository names are short and memorable. Need inspiration? How about [fictional-telegram](#)?

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python

Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

```
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3/lw_2.10 (main)
$ cd "C:\My projects\3"

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3
$ git clone https://github.com/dshayderov/lw_2.11.git
Cloning into 'lw_2.11'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3
$ cd "C:\My projects\3\lw_2.11"

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3/lw_2.11 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3/lw_2.11 (develop)
$ |

```

Рисунок 3 - Ветвление по модели git-flow

4. Создайте проект PyCharm в папке репозитория.

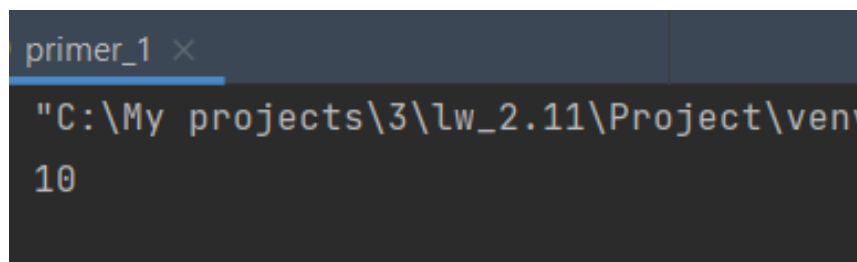
Этот компьютер > OS (C:) > My projects > 3 > lw\_2.11 >

| Имя        | Дата изменения   | Тип              | Размер |
|------------|------------------|------------------|--------|
| Project    | 10.11.2022 18:42 | Папка с файлами  |        |
| .gitignore | 10.11.2022 18:38 | Файл "GITIGNORE" | 2 КБ   |
| LICENSE    | 10.11.2022 18:38 | Файл             | 2 КБ   |
| README.md  | 10.11.2022 18:38 | Файл "MD"        | 1 КБ   |

Рисунок 4 - Проект PyCharm

5. Проработайте примеры лабораторной работы. Создайте для него отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

Функция `mul()` умножает два числа и возвращает полученный результат.



```

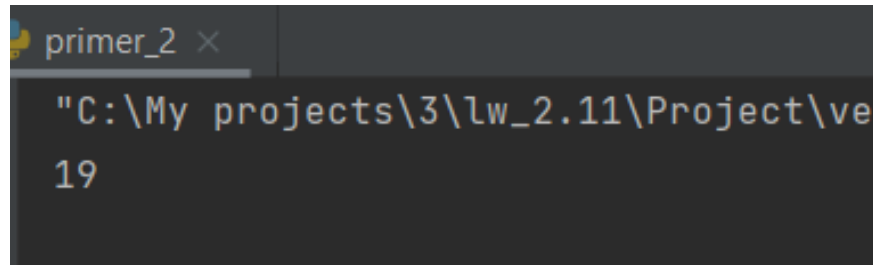
primer_1 x
"C:\My projects\3\lw_2.11\Project\venv
10

```

Рисунок 5 - Результат выполнения примера 1

В функции `fun1()` объявлена локальная переменная `x`, значение которой определяется аргументом `a`. В функции `fun2()` используются эта же переменная `x`, `nonlocal` указывает на то, что эта переменная не является локальной, следовательно, ее значение будет взято из ближайшей области видимости, в которой существует переменная с таким же именем. В нашем случае – это область `enclosing`, в которой этой переменной `x` присваивается

значение  $a * 3$ . Также как и в предыдущем случае, на переменную  $x$  после вызова `fun1(4)`, сохраняется ссылка, поэтому она не уничтожается.

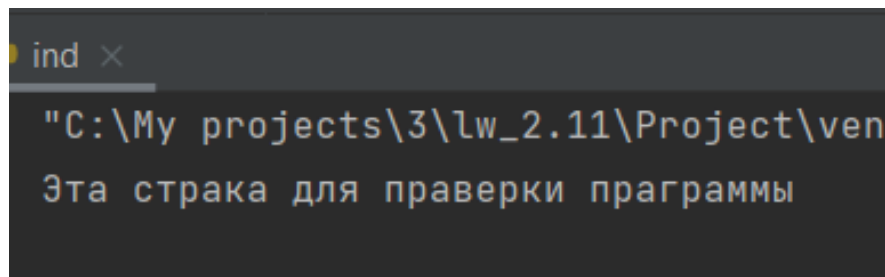


```
primer_2 x
"C:\My projects\3\lw_2.11\Project\ven
19
```

Рисунок 6 - Результат выполнения примера 2

#### 6. Решите индивидуальное задание. (Вариант 26/6)

Используя замыкания функций, объявите внутреннюю функцию, которая бы все повторяющиеся символы заменяла одним другим указанным символом. Какие повторяющиеся символы искать и на что заменять, определяются параметрами внешней функции. Внутренней функции передается только строка для преобразования. Преобразованная (сформированная) строка должна возвращаться внутренней функцией. Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.



```
ind x
"C:\My projects\3\lw_2.11\Project\ven
Эта строка для проверки праграммы
```

Рисунок 7 - Результат выполнения 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?

Замыкания в Python реализованы посредством вложенных 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?**

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?**

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?**

Мы имеем дело с замыканием в Python, когда вложенная функция ссылается на значение из локальной области видимости объемлющей функции.

Критерии, которые должны быть выполнены для создания замыкания в Python, изложены в следующих пунктах:

1. У нас должна быть вложенная функция (функция внутри функции).
2. Вложенная функция должна ссылаться на значение, определенное в объемлющей функции.
3. Объемлющая функция должна возвращать вложенную функцию.

## **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.

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