

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования «СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ  
УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций  
Институт цифрового развития

ОТЧЁТ  
по лабораторной работе №2.11

Дисциплина: «Программирование на Python»

Тема: «Замыкания в языке Python»

Вариант 8

Выполнила: студентка 2  
курса, группы ИВТ-б-о-21-1  
Диченко Дина Алексеевна

Ставрополь 2022

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

## Практическая работа:

### 1. Создала репозиторий.

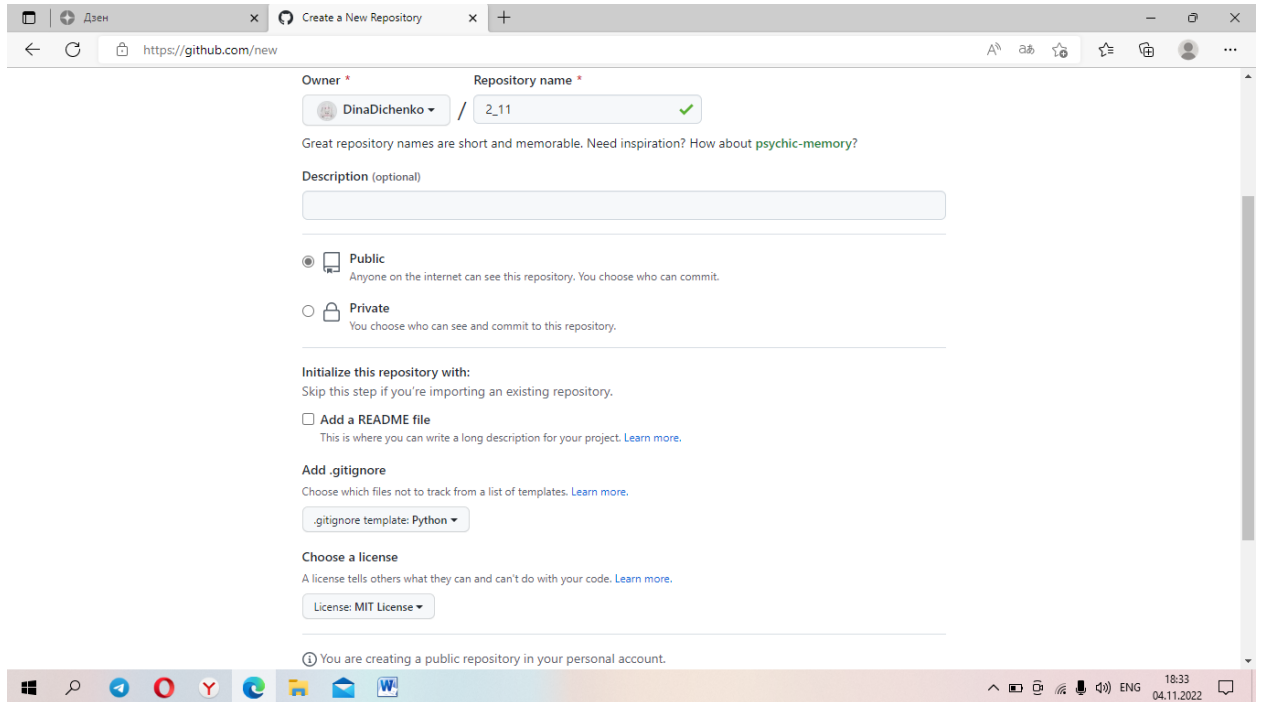


Рисунок 1. Создание репозитория

### 2. Клонировала репозиторий

```
Microsoft Windows [Version 10.0.19043.2130]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Дом\Desktop\Дина\Документы\СКОУ\2 семестр\Программирование>git clone https://github.com/DinaDichenko/2_11.git
Cloning into '2_11'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\Users\Дом\Desktop\Дина\Документы\СКОУ\2 семестр\Программирование>
```

Рисунок 2. Клонирование репозитория

### 3. Изменила файл .gitignore.

```
# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
#.idea/

# End of https://www.toptal.com/developers/gitignore/api/python,pycharm
```

Рисунок 3. Изменение файла .gitignore

#### 4. Организовала репозиторий в соответствии с git-flow.

```
Switched to branch 'develop'
M   .gitignore

C:\Users\Дом\Desktop\Дина\Документы\СКФУ\2 семестр\Программирование\2_11>git add .

C:\Users\Дом\Desktop\Дина\Документы\СКФУ\2 семестр\Программирование\2_11>git commit -m "gitignore"
[develop b2737e2] gitignore
1 file changed, 155 insertions(+), 3 deletions(-)

C:\Users\Дом\Desktop\Дина\Документы\СКФУ\2 семестр\Программирование\2_11>git flow init

Which branch should be used for bringing forth production releases?
- develop
- main
Branch name for production releases: [main] main

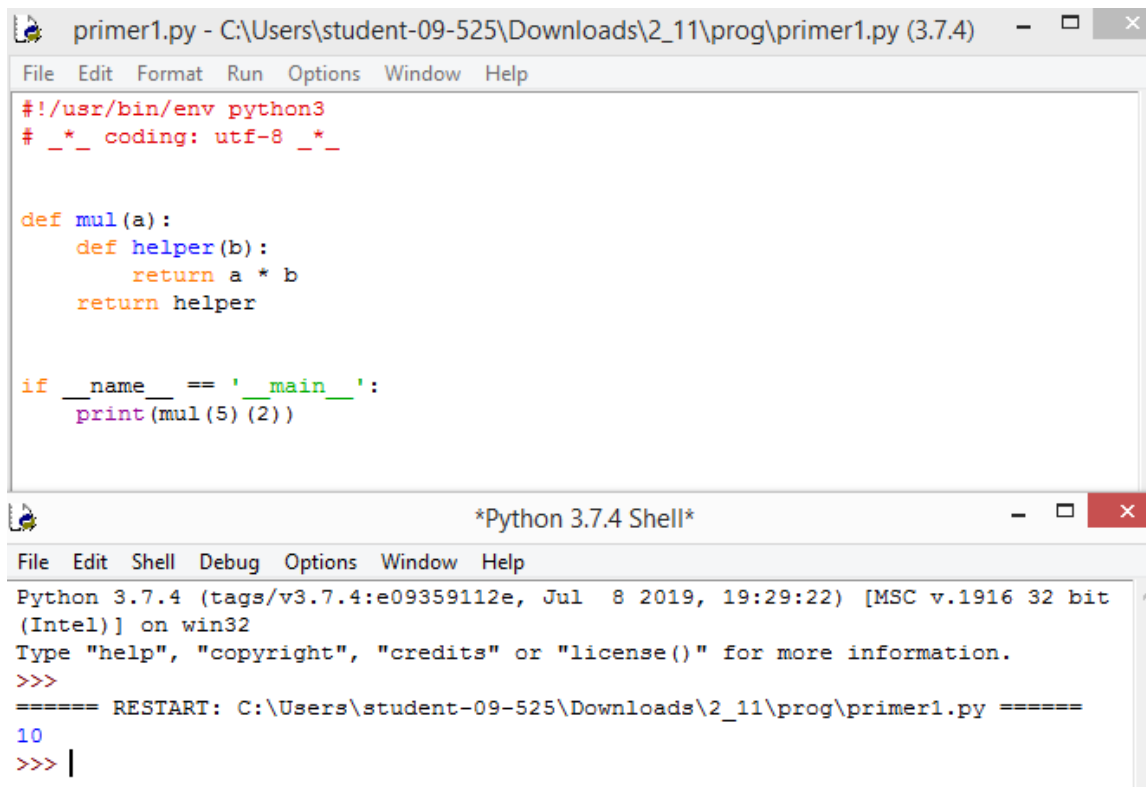
Which branch should be used for integration of the "next release"?
- develop
Branch name for "next release" development: [develop] develop

How to name your supporting branch prefixes?
Feature branches? [feature/] f
Bugfix branches? [bugfix/] b
Release branches? [release/] r
Hotfix branches? [hotfix/] h
Support branches? [support/] s
Version tag prefix? [] v
Hooks and filters directory? [C:/Users/Дом/Desktop/Дина/Документы/СКФУ/2 семестр/Программирование/2_11/.git/hooks] h

C:\Users\Дом\Desktop\Дина\Документы\СКФУ\2 семестр\Программирование\2_11>
```

Рисунок 4. Организация репозитория в соответствии с git-flow

#### 5. Проработала примеры.



The image shows a screenshot of a Python IDE. The top window, titled 'primer1.py - C:\Users\student-09-525\Downloads\2\_11\prog\primer1.py (3.7.4)', contains the following code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

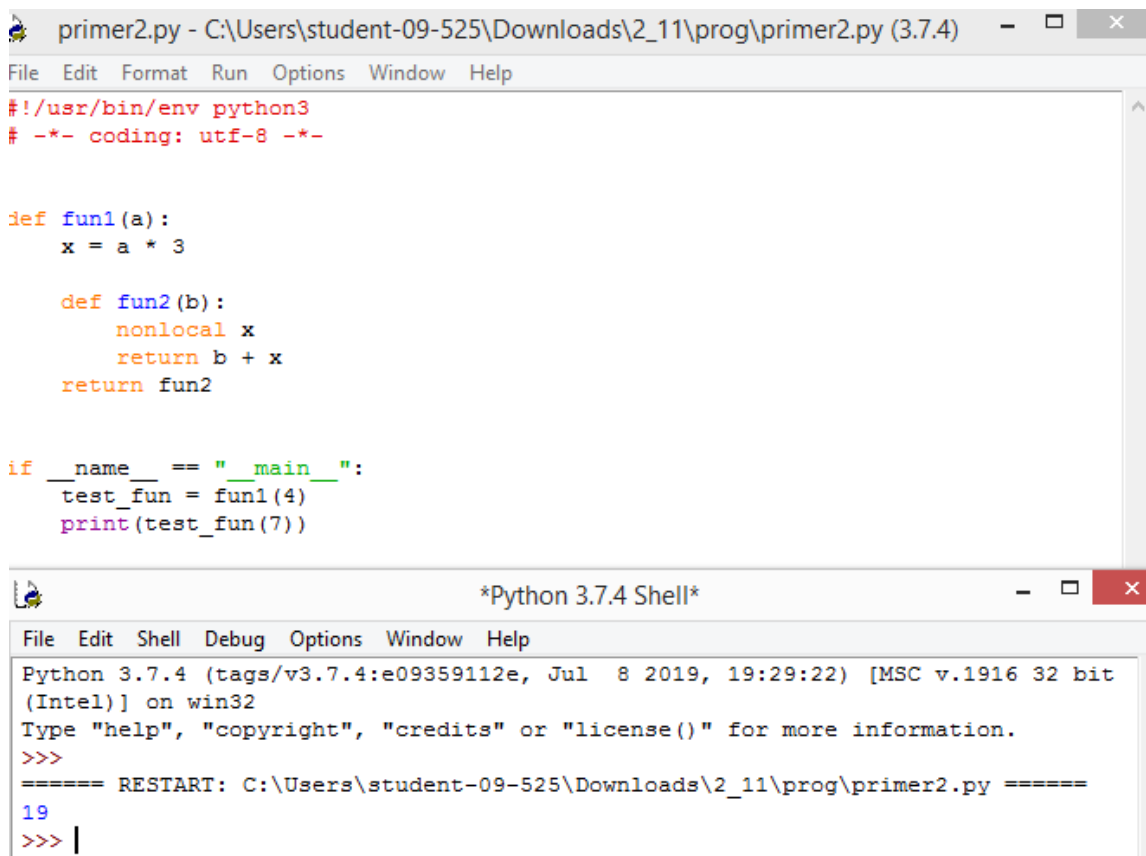
def mul(a):
    def helper(b):
        return a * b
    return helper

if __name__ == '__main__':
    print(mul(5)(2))
```

The bottom window, titled '\*Python 3.7.4 Shell\*', shows the execution output:

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student-09-525\Downloads\2_11\prog\primer1.py =====
10
>>> |
```

Рисунок 5. Результат выполнения примера 1



The image shows a screenshot of a Python IDE. The top window, titled 'primer2.py - C:\Users\student-09-525\Downloads\2\_11\prog\primer2.py (3.7.4)', contains the following code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def fun1(a):
    x = a * 3

    def fun2(b):
        nonlocal x
        return b + x
    return fun2

if __name__ == '__main__':
    test_fun = fun1(4)
    print(test_fun(7))
```

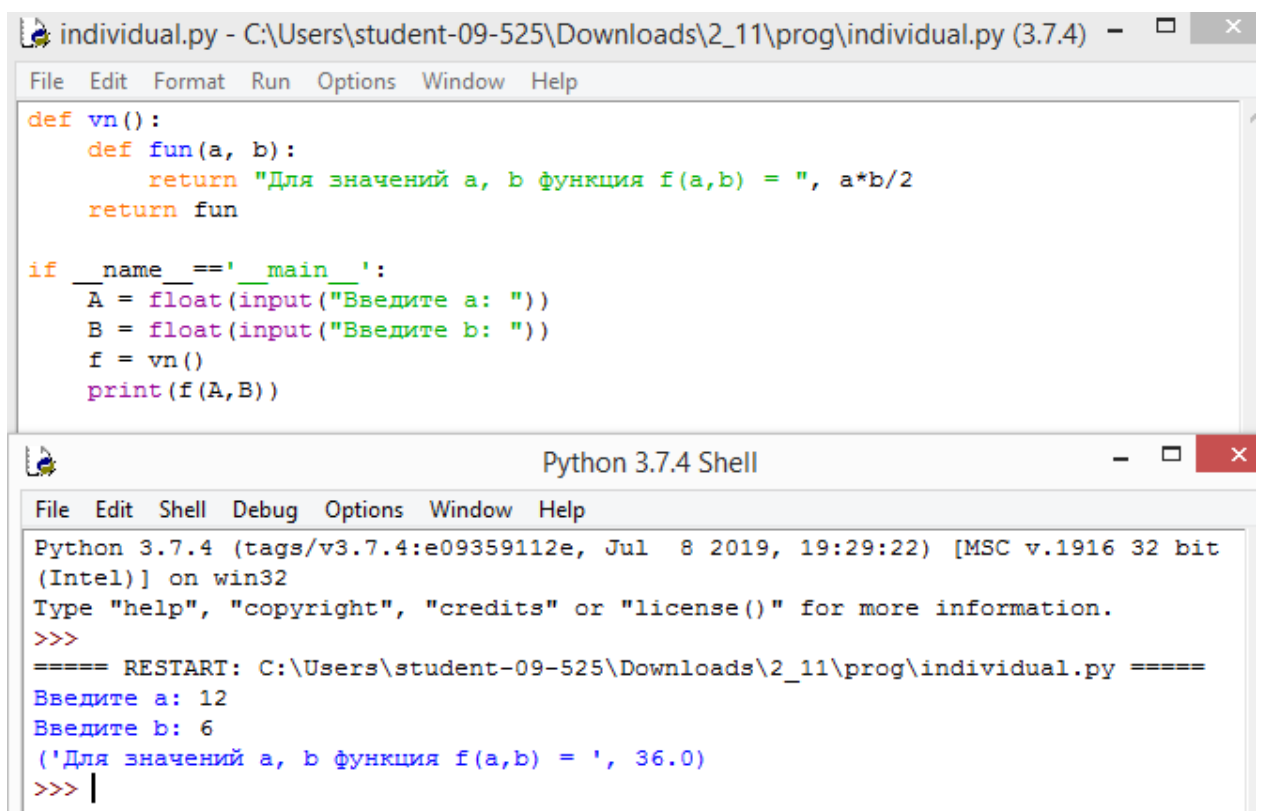
The bottom window, titled '\*Python 3.7.4 Shell\*', shows the execution output:

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student-09-525\Downloads\2_11\prog\primer2.py =====
19
>>> |
```

Рисунок 5. Результат выполнения примера 2

## 6. Выполнила индивидуальное задание.

Используя замыкания функций, объявите внутреннюю функцию, которая принимает два параметра  $a$  ,  $b$  , а затем, возвращает строку в формате: «Для значений  $a$ ,  $b$  функция  $f(a,b) = \langle \text{число} \rangle$ » где число – это вычисленное значение функции  $f$  . Ссылка на  $f$  передается как аргумент внешней функции. Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы. Функцию  $f$  придумайте самостоятельно (она должна что то делать с двумя параметрами  $a$  ,  $b$  и возвращать результат).



```
def vn():
    def fun(a, b):
        return "Для значений a, b функция f(a,b) = ", a*b/2
    return fun

if __name__ == '__main__':
    A = float(input("Введите a: "))
    B = float(input("Введите b: "))
    f = vn()
    print(f(A,B))
```

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student-09-525\Downloads\2_11\prog\individual.py =====
Введите a: 12
Введите b: 6
('Для значений a, b функция f(a,b) = ', 36.0)
>>> |
```

Рисунок 6. 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?

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.

Что касается Python, то тут выделяют четыре области видимости для переменных (с вашего позволения я буду использовать английские термины).

### 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?

Замыкания помогают сделать программу более гибкой и удобной.

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

Сразу хочу сказать, что “свойство замыкания” – это не то замыкание, которое мы разобрали выше. Начнем разбор данного термина с математической точки зрения, а точнее с алгебраической. Предметом алгебры является изучение алгебраических структур – множеств с определенными на них операциями. Под множеством обычно понимается совокупность определенных объектов. Наиболее простым примером числового множества, является множество натуральных чисел. Оно содержит следующие числа: 1, 2, 3, ... и т.д. до бесконечности. Иногда, к этому множеству относят число ноль, но мы не будем этого делать.

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