

## Лабораторная работа №6

Выполнил Эсеналиев Арсен

ИВТ-б-о-21-1

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

### 1. Создал общедоступный репозиторий на GitHub с MIT


## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

Owner \*

Repository name \*


 Arsen445 ▾

/ LB2.11 ✓


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

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.

---

Create repository

2. Выполнил клонирование созданного репозитория.

```
C:\Users\GG_Force>d;  
"d" не является внутренней или внешней  
командой, исполняемой программой или пакетным файлом.  
  
C:\Users\GG_Force>d:  
  
D:\>cd REP6  
  
D:\REP6>git clone https://github.com/Arsen445/LB2.11.git  
Cloning into 'LB2.11'...  
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.  
  
D:\REP6>
```

3. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm  
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm  
  
### PyCharm ###  
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio  
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839  
  
# User-specific stuff  
.idea/**/workspace.xml  
.idea/**/tasks.xml  
.idea/**/usage.statistics.xml  
.idea/**/dictionaries  
.idea/**/shelf  
  
# AWS User-specific  
.idea/**/aws.xml  
  
# Generated files  
.idea/**/contentModel.xml  
  
# Sensitive or high-churn files  
.idea/**/dataSources/
```

4. Организовал репозиторий в соответствии с моделью ветвления git-flow.

```
D:\REP6\LB2.11>git flow init

which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/REP6/LB2.11/.git/hooks]

D:\REP6\LB2.11>
```

5. Создал проект пайчарм

Новый том (D:) > REP6 > LB2.11 >

Имя	Дата изменения	Тип	Размер
.idea	18.10.2022 1:22	Папка с файлами	
doc	03.11.2022 4:19	Папка с файлами	
.gitignore	18.10.2022 1:22	Файл "GITIGNORE"	6 КБ
LICENSE	03.11.2022 4:01	Файл	2 КБ
README.md	03.11.2022 4:01	Файл "MD"	1 КБ

6. Проработал примеры лабораторной работы.

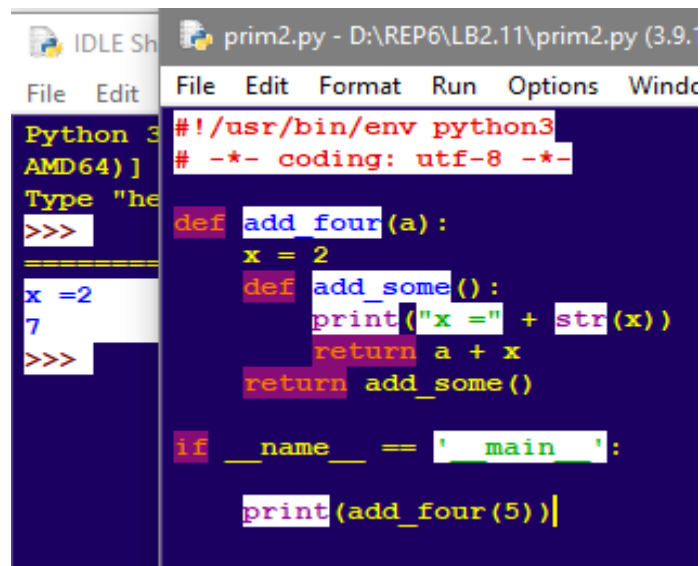
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def add_two(a):
    x = 2
    return a + x

if __name__ == '__main__':
    print(add_two(3))
    print(x)
```

===== RESTART: D:\REP6\LB2.11\prim1.py

```
5
Traceback (most recent call last):
  File "D:\REP6\LB2.11\prim1.py", line 11, in <module>
    print(x)
NameError: name 'x' is not defined
>>>
```



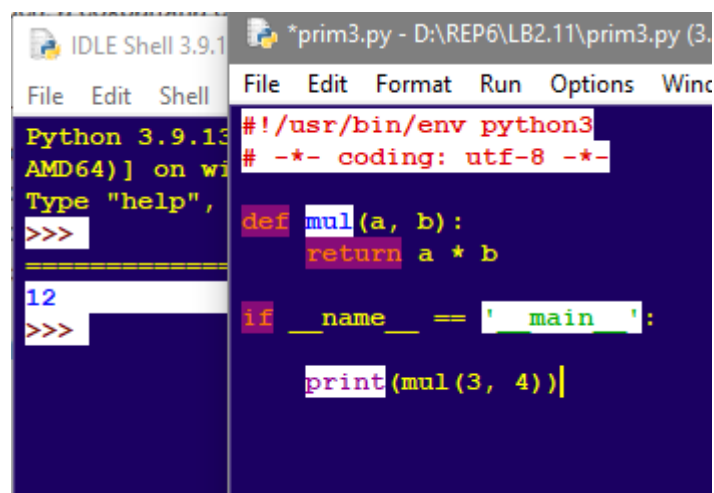
The screenshot shows the Python IDLE environment. On the left, the 'IDLE Shell' window displays the Python prompt and the execution of a script. On the right, the 'prim2.py' editor window shows the source code. The code defines a function 'add\_four' that sets a local variable 'x' to 2, defines an inner function 'add\_some' that prints 'x' and returns 'a + x', and then calls 'add\_some' and prints the result. The main block prints 'add\_four(5)'.

```
Python 3.9.13 (tags/v3.9.13:17e3093, Dec 6, 2021) [AMD64] on win32
Type "help()", "copyright()", "credits()", "license()", "list()", "quit()", or "exit()" for more help.
>>>
=====
x = 2
7
>>>
```

```
prim2.py - D:\REP6\LB2.11\prim2.py (3.9.13)
File Edit Format Run Options Window
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def add_four(a):
    x = 2
    def add_some():
        print("x =" + str(x))
        return a + x
    return add_some()

if __name__ == '__main__':
    print(add_four(5))
```



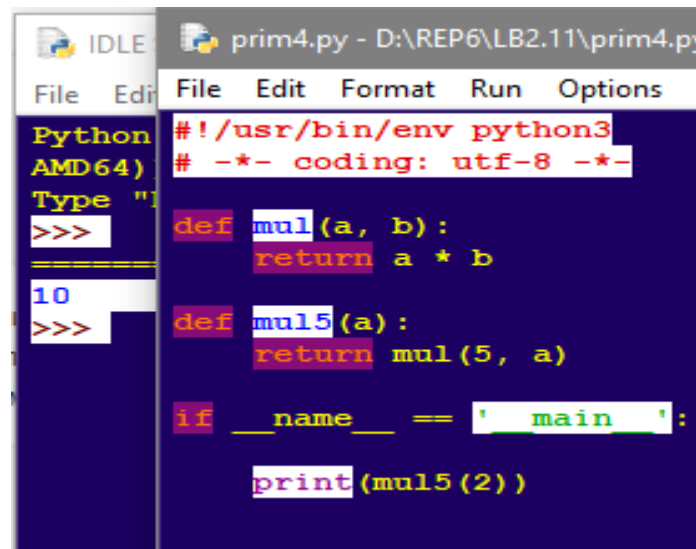
The screenshot shows the Python IDLE environment. On the left, the 'IDLE Shell' window displays the Python prompt and the execution of a script. On the right, the 'prim3.py' editor window shows the source code. The code defines a function 'mul' that returns the product of two numbers 'a' and 'b'. The main block prints 'mul(3, 4)'.

```
Python 3.9.13 (tags/v3.9.13:17e3093, Dec 6, 2021) [AMD64] on win32
Type "help()", "copyright()", "credits()", "license()", "list()", "quit()", or "exit()" for more help.
>>>
=====
12
>>>
```

```
*prim3.py - D:\REP6\LB2.11\prim3.py (3.9.13)
File Edit Format Run Options Window
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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

if __name__ == '__main__':
    print(mul(3, 4))
```



The screenshot shows the Python IDLE environment. On the left, the 'IDLE Shell' window displays the Python prompt and the execution of a script. On the right, the 'prim4.py' editor window shows the source code. The code defines a function 'mul' that returns the product of two numbers 'a' and 'b', and a function 'mul5' that returns 'mul(5, a)'. The main block prints 'mul5(2)'.

```
Python 3.9.13 (tags/v3.9.13:17e3093, Dec 6, 2021) [AMD64] on win32
Type "help()", "copyright()", "credits()", "license()", "list()", "quit()", or "exit()" for more help.
>>>
=====
10
>>>
```

```
prim4.py - D:\REP6\LB2.11\prim4.py (3.9.13)
File Edit Format Run Options Window
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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

def mul5(a):
    return mul(5, a)

if __name__ == '__main__':
    print(mul5(2))
```

7. Зафиксируйте сделанные изменения в репозитории.(после создания веток не запустил, поэтому не работало)

```

D:\REP6\LB2.11>git push --all
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 92.88 kiB | 7.14 MiB/s, done.
Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Arsen445/LB2.11.git
   30b5e5e..c836547  develop -> develop

D:\REP6\LB2.11>git status
On branch develop
nothing to commit, working tree clean

D:\REP6\LB2.11>_

```

## 8. Решите 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.

File Edit Format Run Options Window Help	File Edit Shell Debug Options Window Help
#!/usr/bin/env python3 # -*- coding: utf-8 -*-	Вы знаете, какая фигура нужна? введите 1 или 0
def first_n(type = 0): print("Введите стороны фигуры: ") a = int(input()) b = int(input())	0 ... 6.0 >>>
print("Вы знаете, какая фигура нужна?\n" + "введите у или n") agr = str(input()) if agr == "y": print("Треугольник или квадрат?\n" + "Введите 0, если треуго") type = int(input()) else: print("...")	===== RESTART: D:\REP6\LB2.11 Введите стороны фигуры: 3 4 Вы знаете, какая фигура нужна? введите у или n
def second_n(a, b): if type == 0: return(0.5 * a * b) else: return(a * b) return second_n(a, b)	Треугольник или квадрат? Введите 0, если треугольник и 1, если квадрат.. 1 12 >>>
if __name__ == '__main__': y = first_n() print(y)	===== RESTART: D:\REP6\LB2.11 Введите стороны фигуры: 3 4 Вы знаете, какая фигура нужна? введите у или n
	Треугольник или квадрат? Введите 0, если треугольник и 1, если квадрат.. 0 6.0 >>>
	===== RESTART: D:\REP6\LB2.11 Введите стороны фигуры: 3 4 Вы знаете, какая фигура нужна? введите у или n
	n ... 6.0 >>>

9. Зафиксируйте сделанные изменения в репозитории.

```
nothing added to commit but untracked files present (use "git add" to track)
D:\REP6\LB2.11>git add --all
D:\REP6\LB2.11>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   ind.py
D:\REP6\LB2.11>git commit -m "e1"
[develop f888dbd] e1
1 file changed, 31 insertions(+)
create mode 100644 ind.py
D:\REP6\LB2.11>git push --all
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 632 bytes | 632.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Arsen445/LB2.11.git
   c836547..f888dbd  develop -> develop
D:\REP6\LB2.11>
```

10. Выполните слияние ветки для разработки с веткой main/master.

```
D:\REP6\LB2.11>git checkout main
Unlink of file 'doc/лб6.docx' failed. Should I try again? (y/n) y
Unlink of file 'doc/лб6.docx' failed. Should I try again? (y/n) n
warning: unable to unlink 'doc/лб6.docx': invalid argument
Updating files: 100% (13/13), done.
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
D:\REP6\LB2.11>git status
On branch main
Your branch is up to date with 'origin/main'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        doc/
nothing added to commit but untracked files present (use "git add" to track)
D:\REP6\LB2.11>git add --all
D:\REP6\LB2.11>git status
On branch main
Your branch is up to date with 'origin/main'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   "doc/\320\233\320\2216.docx"
D:\REP6\LB2.11>git merge develop
Updating 71773ee..f888dbd
Fast-forward
 .idea/.name | 1 +
 .idea/LB2.8.iml | 8 ++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/modules.xml | 8 ++++++
 .idea/vcs.xml | 6 ++++++
 "doc/~$ \320\233\320\2216.docx" | Bin 0 -> 162 bytes
 "doc/\320\233\320\2216.docx" | Bin 0 -> 566932 bytes
 ind.py | 31 ++++++++++++++++++++++++++++++++++++++
 prim1.py | 15 ++++++++
 prim2.py | 13 ++++++++
 prim3.py | 9 ++++++

```

## 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?

### 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?

```
>>> 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?

```
>>> tp1 = lambda a, b: (a, b)
```

Если мы передадим в качестве аргументов числа, то, получим простой кортеж.

```
>>> a = tp1(1, 2)
>>> a
(1, 2)
```

Эту операцию можно производить не только над числами, но и над сущностями, ей же и порожденными.

```
>>> b = tp1(3, a)
>>> b
(3, (1, 2))

>>> c = tp1(a, b)
>>> c
((1, 2), (3, (1, 2)))
```