

**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: «Основы программной инженерии»
Тема: «Замыкание в языке Python»**

**Выполнил:
студент 2 курса
группы Пиж-б-о-21-1
Коныжев Максим
Викторович**

Ставрополь 2022

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

1. Был создан репозиторий в Github в который были добавлены правила gitignore для работы IDE PyCharm, была выбрана лицензия MIT, сам репозиторий был клонирован на локальный сервер и был организован в соответствии с моделью ветвления git-flow.

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 ***

MaxDrill / lab2_11

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

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

Рисунок 1 – Создание репозитория

```
C:\Users\UESR\gitproj>git clone https://github.com/MaxDrill/lab2_11.git
Cloning into 'lab2_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

```
C:\Users\UESR\gitproj\lab2_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? [C:/Users/UESR/gitproj/lab2_11/.git/hooks]

C:\Users\UESR\gitproj\lab2_11>
```

Рисунок 3 – Организация репозитория с моделью ветвления git-flow

```
C:\Users\UESR\Desktop\2.1\proj\venv\Scr
example 1
19

example 2
(1, 2)
(3, (1, 2))
((1, 2), (3, (1, 2)))

Process finished with exit code 0
```

Рисунок 6 – Результат работы примера

3. Индивидуальное задание

Используя замыкания функций, объявите внутреннюю функцию, которая заключает строку *s* (*s* – строка, параметр внутренней функции) в произвольный тег, содержащийся в переменной *tag* – параметре внешней функции. Далее, на вход программы поступает две строки: первая с тегом, вторая с некоторым содержимым. Вторую строку нужно поместить в тег из первой строки с помощью реализованного замыкания. Результат выведите на экран.

Код программы:

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

def get_tag(tag):
    x = tag

    def get_str(s):
        nonlocal x
        return ":" + ".join([x, s])
    return get_str

if __name__ == '__main__':
    test_fun = get_tag("test tag")
    print(test_fun("hello"))
```

```
C:\Users\UESR\Desktop\2.1\proj\venv\
test tag: hello

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

```
C:\Users\UESR\gitproj\lab2_11>git add .
C:\Users\UESR\gitproj\lab2_11>git commit -m "Add proj"
[develop 6146497] Add proj
3 files changed, 44 insertions(+), 1 deletion(-)
create mode 100644 proj/lab14_ex.py
create mode 100644 proj/lab14_idz.py
```

Рисунок 8 – Коммит изменений

```
C:\Users\UESR\gitproj\lab2_11>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\UESR\gitproj\lab2_11>git merge develop
Updating e041fdc..6146497
Fast-forward
 README.md      |  3 +-
 proj/lab14_ex.py | 26 ++++++
 proj/lab14_idz.py | 16 ++++++
 3 files changed, 44 insertions(+), 1 deletion(-)
 create mode 100644 proj/lab14_ex.py
 create mode 100644 proj/lab14_idz.py

C:\Users\UESR\gitproj\lab2_11>
```

Рисунок 9 – Слияние веток

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

Контрольные вопросы

1. Что такое замыкание?

Замыкание – это функция, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции.

2. Как реализованы замыкания в языке программирования Python?

Замыканием в языке Python называется функция, вложенная в другую функцию и использующая переменные внешней функции.

3. Что подразумевает под собой область видимости Local?

Переменный с областью видимости Local (локальные переменные) могут быть использованы только внутри того блока кода, где она была объявлена.

4. Что подразумевает под собой область видимости Enclosing?

Для вложенных функций переменные из функции более высокого уровня имеют данную область видимости.

5. Что подразумевает под собой область видимости Global?

Область видимости Global означает, что данная переменная может быть использована (видна) во всём модуле (файле с расширением .py).

6. Что подразумевает под собой область видимости Build-in?

Это переменный уровень интерпретатора. Для их использования не нужно импортировать модули.