

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 «Основы программной инженерии»**

Выполнил:  
Магомедов Имран Борисович  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная  
инженерия», направленность  
(профиль) «Разработка и  
сопровождение программного  
обеспечения», очная форма  
обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., кандидат  
технических наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

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

Ставрополь, 2023 г.

## ТЕМА: Замыкания в языке Python

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

### МЕТОДИКА И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

**Create a new repository**

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

*Required fields are marked with an asterisk (\*).*

**Owner \*** MarcusPlay / **Repository name \*** BSE\_Python\_lab\_14

✔ BSE\_Python\_lab\_14 is available.

Great repository names are short and memorable. Need inspiration? How about **fantastic-guacamole** ?

**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:**

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

**Add .gitignore**

.gitignore template: Python

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

**Choose a license**

License: MIT License

3. Выполните клонирование созданного репозитория.

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №14
$ git clone https://github.com/MarcusPlay/BSE_Python_lab_14.git
Cloning into 'BSE_Python_lab_14'...
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.
```

4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №14/BSE_Python_lab_14 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

5. Проработать примеры лабораторной работы.

6. Выполнить индивидуальное задание. Составить программу с использованием замыканий для решения задачи. Вариант 12. 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.

```
def create_tagged_string(tag):
    def tag_string(content):
        return f"<{tag}>{content}</{tag}>"
    return tag_string

if __name__=="__main__":
    tag_input = input("Введите тег: ")
    content_input = input("Введите содержимое: ")

    tagged_string_closure = create_tagged_string(tag_input)
    result = tagged_string_closure(content_input)

    print("Результат:", result)
```

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №14/BSE_Python_lab_14 (develop)
$ python3 individual_task.py
Введите тег: strong
Введите содержимое: Magomedov
Результат: <strong>Magomedov </strong>
```

7. Зафиксируйте изменения в репозитории.

```
Imran@Kaskad MINGW64 ~/Desktop/Лабораторная работа №14/BSE_Python_lab_14 (develop)
$ git commit -m "added individual_task"
[develop e85b405] added individual_task
1 file changed, 26 insertions(+)
create mode 100644 individual_task.py
```

8. Добавьте отчет по лабораторной работе в *формате PDF* в папку *doc* репозитория. Зафиксируйте изменения.

9. Выполните слияние ветки для разработки с веткой *master/main*.

10. Отправьте сделанные изменения на сервер GitHub.

11. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

## ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ

1. Что такое замыкание?

Замыкание — это функция, которая запоминает значения в окружающем ее лексическом контексте, даже если эта функция выполняется вне этого

контекста. Она имеет доступ к переменным из внешней области видимости, где она была определена.

## 2. Как реализованы замыкания в языке программирования Python?

Замыкания в Python реализуются путем определения функции внутри другой функции и возвращения этой внутренней функции.

## 3. Что подразумевает под собой область видимости Local?

Локальная область видимости относится к переменным, определенным внутри текущей функции. Эти переменные видны только внутри этой функции.

## 4. Что подразумевает под собой область видимости Enclosing?

Окружающая область видимости связана с замыканиями. Она включает в себя переменные из внешних функций, в которых была определена текущая функция.

## 5. Что подразумевает под собой область видимости Global?

Глобальная область видимости относится к переменным, определенным на верхнем уровне скрипта или модуля. Эти переменные видны в пределах всего модуля.

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

Встроенная область видимости включает в себя встроенные функции и имена, предоставляемые Python (например, `print()`, `len()`, `str()`).

## 7. Как использовать замыкания в языке программирования Python?

Замыкания можно использовать, например, для создания функций с дополнительными параметрами, сохраненными в окружении. Пример:

```
def outer_function(x):  
    def inner_function(y):  
        return x + y  
    return inner_function  
  
closure = outer_function(10)  
result = closure(5)    # Результат: 15
```

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

Замыкания могут быть использованы для создания иерархии функций или объектов, где каждый уровень имеет доступ к своему контексту и может использовать данные из внешних уровней. Например, в создании генераторов данных или структур данных.