

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №14.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,

направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Рядская Мария Александровна

Проверил:

Воронкин Р. А.

Ставрополь 2023

Тема: Лабораторная работа 2.11 Замыкания в языке Python.

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


Выполнение работы:

1. Изучила теоретический материал работы.
2. Создала репозиторий на git.hub.

Создайте новый репозиторий



Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий.](#)

Обязательные поля отмечены звездочкой (*).

Владелец *	Название репозитория *
 мрядская ▾	/ lab14
	✓ lab14.

Отличные названия репозитория короткие и запоминающиеся. Нужно вдохновение? Как насчет [solid-meme?](#)

Описание (необязательно)

- ☒  **Общедоступный**
Этот репозиторий может видеть любой пользователь Интернета. Вы сами выбираете, кто может совершать коммиты.
- ☐  **Приватный**
Вы сами выбираете, кто может просматривать этот репозиторий и фиксировать его в нем.

Инициализируйте этот репозиторий с помощью:

- ☒ **Добавьте файл README**
Здесь вы можете написать подробное описание вашего проекта. [Узнайте больше о README.](#)

Добавить файл .gitignore

Рисунок 1 – создание репозитория

```
C:\>cd git1  
  
C:\git1>git clone https://github.com/mryadskaya/lab14.git  
Cloning into 'lab14'...  
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.  
  
C:\git1>cd lab14  
  
C:\git1\lab14>
```

3. Клонировала репозиторий.

```
C:\>cd git1  
  
C:\git1>git clone https://github.com/mryadskaya/lab14.git  
Cloning into 'lab14'...  
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.  
  
C:\git1>cd lab14  
  
C:\git1\lab14>
```

Рисунок 2 – клонирование репозитория 4.

4. Организовать свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\git1\lab14>git branch develop

C:\git1\lab14>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/mryadskaya/lab14/pull/new/develop
remote:
To https://github.com/mryadskaya/lab14.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\git1\lab14>git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

C:\git1\lab14>
```

Рисунок 4 – создание ветки develop

5. Проработал примеры из методички.

```
#!/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)
    test_fun(7)
    print(test_fun)
```

Рисунок 5 – пример 1

15. Используя замыкания функций, объявите внутреннюю функцию,

которая принимает в качестве параметров фамилию и имя, а затем, заносит
в

шаблон эти данные. Сам шаблон – это строка, которая передается внешней
функции и, например, может иметь такой вид: «Уважаемый %F%, %N%!
Вы

делаете работу по замыканиям функций.» Здесь %F% - это фрагмент куда
нужно

подставить фамилию, а %N% - фрагмент, куда нужно подставить имя.
(Шаблон

может быть и другим, вы это определяете сами). Здесь важно, чтобы
внутренняя

функция умела подставлять данные в шаблон, формировать новую строку и
возвращать результат. Вызовите внутреннюю функцию замыкания и
отобразите

на экране результат ее работы.

```
# -*- coding: utf-8 -*-  
  
def create_greeting_template(template):  
    def inner_function(last_name, first_name):  
        formatted_template = template.replace('%F%', last_name).replace('%N%', first_name)  
        return formatted_template  
    return inner_function  
  
if __name__ == "__main__":  
    n = input("Введите вашу фамилию: ")  
    l = input("Введите ваше имя: ")  
  
    # Создаем замыкание с шаблоном  
    greeting_template = create_greeting_template("Уважаемый %F%, %N%! Вы делаете работу по замыканиям функций.")  
  
    # Вызываем внутреннюю функцию замыкания и отображаем результат  
    result = greeting_template(n, l)  
    print(result)
```

Рисунок 6 – индивидуальное задание

```

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab9
Введите вашу фамилию: Рядская
Введите ваше имя: Мария
Уважаемый Рядская, Мария! Вы делаете работу по замыканиям функций.

Process finished with exit code 0

```

Рисунок 7 – индивидуальное задание

7. Зафиксировал все изменения в github в ветке develop и слил ветки

```

C:\git1\lab14>git add .

C:\git1\lab14>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\3
0\275\320\276\320\265.py"
    new file:   "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200.py"

C:\git1\lab14>git commit -m "схранение"
[develop f290051] схранение
 2 files changed, 32 insertions(+)
 create mode 100644 "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\3
0\275\320\276\320\265.py"
 create mode 100644 "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200.py"

C:\git1\lab14>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.00 KiB | 205.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mryadskaya/lab14.git
 73a786e..f290051 develop -> develop

C:\git1\lab14>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git1\lab14>

```

Рисунок 8 – фиксация изменений в ветку develop

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?

Замыкание (closure) в Python - это функция, которая запоминает значения в окружающей (внешней) области видимости, даже если эта область видимости больше не существует. Замыкание возникает, когда внутри функции определены вложенные функции, и вложенная функция ссылается на переменные из внешней функции.

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.

Builtin – это максимально широкая область видимости.

7. Как использовать замыкания в языке программирования Python?

Использование замыканий в Python обычно связано с созданием функций внутри других функций и возвратом этих вложенных функций. Замыкания полезны, когда вам нужно передать часть контекста (переменные) в функцию,

чтобы она могла использовать их даже после того, как внешняя функция завершила свою работу.

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

Замыкания в Python можно использовать для построения иерархических данных, таких как деревья или вложенные структуры. Замыкания позволяют сохранять состояние внешней функции внутри вложенной функции, что обеспечивает уровень иерархии.