

**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 № 6

Дисциплина: «Программирование на Python»

Тема: «Замыкания в языке Python»

Выполнил: студент 2 курса

группы ИТС-б-о-21-1

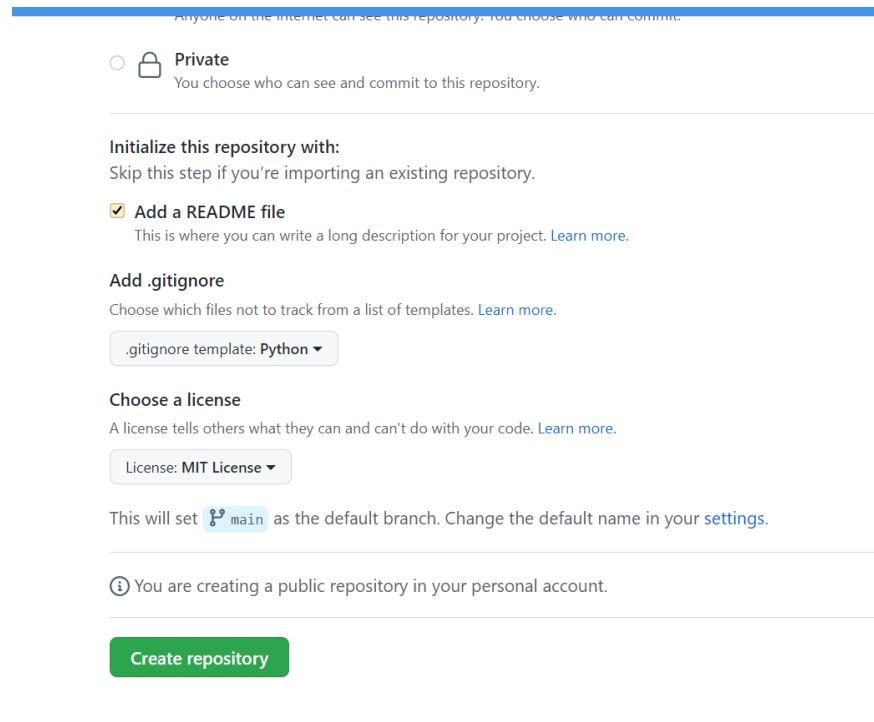
Гайибов Хасан Мамадиерович

Ставрополь 2022


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

Порядок выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



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

2. Выполните клонирование созданного репозитория.

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



1 contributor

280 lines (223 sloc) | 5.7 KB

```
1 .idea/
2 # Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
3 # Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm
4
5 ### PyCharm ###
6 # Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
7 # Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
8
9 # User-specific stuff
10 .idea/**/workspace.xml
11 .idea/**/tasks.xml
12 .idea/**/usage.statistics.xml
13 .idea/**/dictionaries
```

Рисунок 3. Дополнение файла .gitignore

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

```
C:\Users\User\Desktop\2 кypc Python\lab 10\lab-10>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/User/Desktop/2 кypc Python/lab 10/lab-10/.git/hooks]
C:\Users\User\Desktop\2 кypc Python\lab 10\lab-10>
```

Рисунок 4. Организован модель ветвления git flow

5.Проработайте пример лабораторной работы. Создайте для него отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

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

def mul5(a):
    def helper(b):
        return a * b
    return helper

if __name__ == '__main__':
    print(mul5(5)(6))
```

Рисунок 5. Пример лаб работы

6.Проработайте пример лабораторной работы. Создайте для него отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

```
#!/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))
```

Рисунок 6. Пример лаб работы

7. Индивидуальное задание

Вариант 9. Используя замыкания функций, объявите внутреннюю функцию, которая принимает в качестве аргумента коллекцию (список или кортеж) и возвращает или минимальное значение, или максимальное, в зависимости от значения параметра `type` внешней функции. Если `type` равен «`max`», то возвращается максимальное значение, иначе – минимальное. По умолчанию `type` должно принимать значение «`max`». Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.

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

def fun1(type_='max'):
    def fun2(lst):
        return eval(f'{type_}(lst)')
    print(type_)
    return fun2

a = [1, 2, 34, 54, 36, 7, 8]

max_fun = fun1()
min_fun = fun1('min')

if __name__ == '__main__':
    print(max_fun(a))
    print(min_fun(a))
```

Рисунки 7. Выполненное индивидуальное задание

8. Сделала коммит, выполнил слияние с веткой `main`, и запустил изменения в уд. репозиторий.

```
C:\Users\User\Desktop\2 курс Python\lab 14\lab14>git commit -m "new"
[main 9b969c0] new
3 files changed, 46 insertions(+)
create mode 100644 indiv 1.py
create mode 100644 primer 1.py
create mode 100644 primer 2.py

C:\Users\User\Desktop\2 курс Python\lab 14\lab14>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 885 bytes | 885.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/aikanyshkaukanbekova/lab14.git
5494437..9b969c0 main -> main
```

Рисунок 8. Сохранения

Ответы на 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?

Переменные 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 уровня интерпретатора. Для их использования не нужно импортировать модули