

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: «Программирование на Python»

Тема: «Замыкания в языке Python»

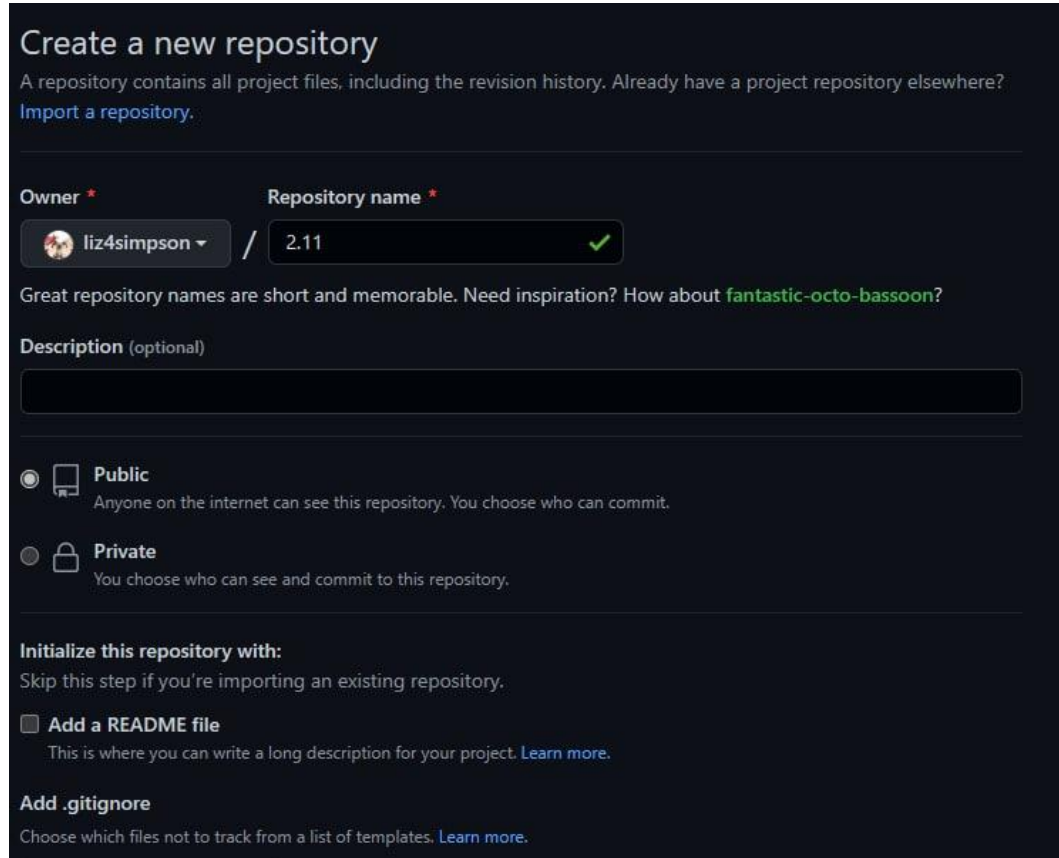
Вариант 25

Выполнила: студентка 2
курса группы ИВТ-б-о-21-1
Яковлева Елизавета
Андреевна

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

Практическая часть:

1. Создала общедоступный репозиторий на GitHub.



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 * 2.11 ✓

Great repository names are short and memorable. Need inspiration? How about [fantastic-octo-bassoon?](#)

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.](#)

Рисунок 1. Создание репозитория

2. Выполнила клонирование созданного репозитория.

```
C:\Users\Elizaveta\Desktop\git>git clone https://github.com/liz4simpson/2.10.git
Cloning into '2.10'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), 5.17 KiB | 441.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2. Клонирование репозитория

3. Дополнила файл .gitignore.

```
*.gitignore - Блокнот
Файл  Правка  Формат  Вид  Справка
# 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, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
.idea/
.idea

# 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
```

Рисунок 3. Изменение файла .gitignore

4. Организовала репозиторий в соответствии git-flow.

```
C:\Users\Elizaveta\Desktop\git\2.10>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/Elizaveta/Desktop/git/2.10/.git/hooks]
```

Рисунок 4. Организация репозитория в соответствии с git-flow

5. Проработала примеры лабораторной работы.

```
1 ▶ 1 #!/usr/bin/env python3
2 2 # -*- coding: utf-8 -*-
3
4
5 5 def mul5(a):
6 6     def helper(b):
7 7         return a * b
8 8     return helper
9
10
11 ▶ 11 if __name__ == '__main__':
12 12     print(mul5(5)(6))

if __name__ == '__main__':
prim1 x
C:\ana\2.11\Scripts\python.exe C:\Users\Elizaveta\Desktop\git\2.11\prim1.py
30
Process finished with exit code 0
```

Рисунок 5. Результат выполнения примера

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def fun1(a):
6     x = a * 3
7
8     def fun2(b):
9         nonlocal x
10        return b + x
11    return fun2
12
13
14 ▶ if __name__ == '__main__':
15     test_fun = fun1(4)
16     . . . . .
17     . . . . .
18     . . . . .
19 if __name__ == '__main__':
```

prim2 x

C:\папа\2.11\Scripts\python.exe C:\Users\E

19

Рисунок 6. Результат выполнения примера

6. Выполнила индивидуальное задание (вар-25) 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.

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 |
5 def surname(s):
6     def name(n):
7         a = 'Уважаемая, {} {}! ' \
8             'Вы делаете работу по замыканиям функций.'.format(s, n)
9         return a
10    return name
11
12
13 ▶ if __name__ == '__main__':
14     p = surname('Яковлева')
15     print(p('Елизавета'))
16
17
18 ind x
19 C:\nana\2.11\Scripts\python.exe C:\Users\Elizaveta\Desktop\git\2.11\prog
20 Уважаемая, Яковлева Елизавета! Вы делаете работу по замыканиям функций.
21
22 Process finished with exit code 0
```

Рисунок 7. Результат выполнения индивидуального задания

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

Контр. вопросы и ответы на них:

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.