

**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»

Выполнила:

студентка 2 курса

группы Пиж-б-о-21-1

Джолдошова Мээрим

Бекболотовна

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.

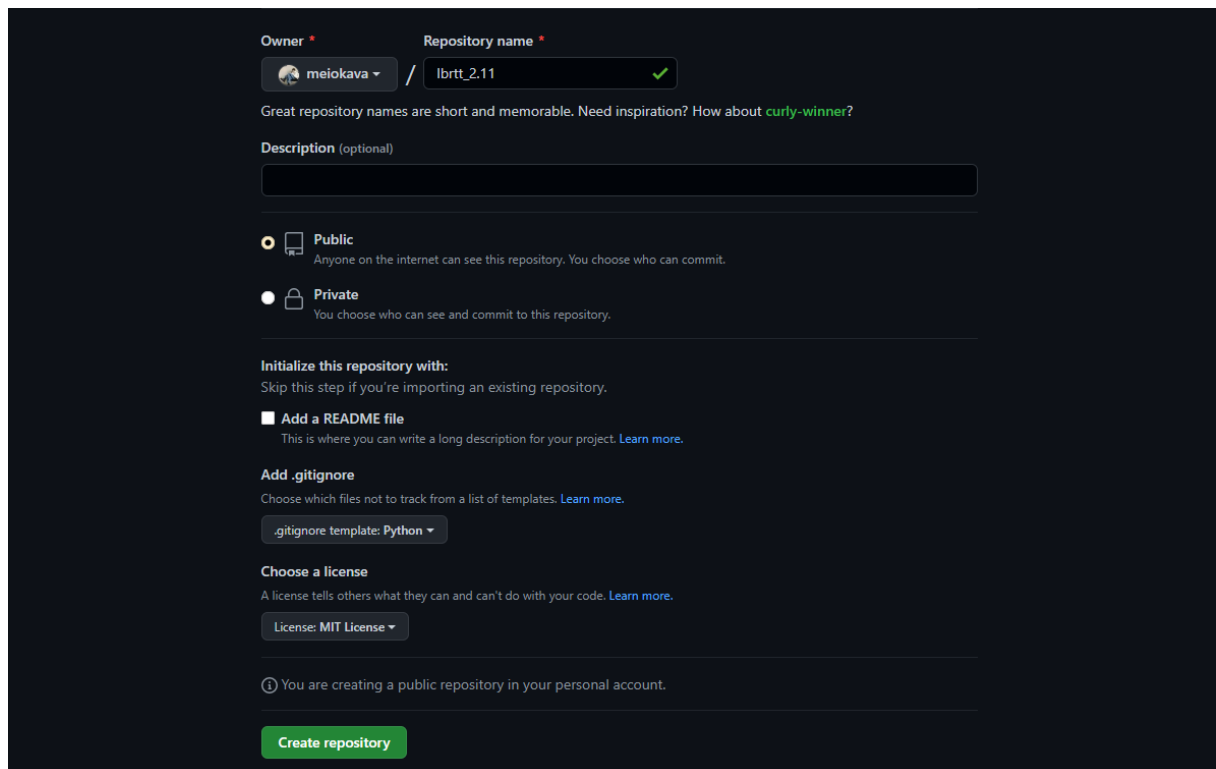
The image shows the GitHub 'Create new repository' page. At the top, the 'Owner' is set to 'meiokava' and the 'Repository name' is 'lbrtt_2.11', which has a green checkmark. Below this, there's a note about repository names being short and memorable, with a link to 'curly-winner?'. The 'Description' field is empty. Under 'Visibility', the 'Public' option is selected with a radio button, and a note says 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is unselected. The 'Initialize this repository with:' section has a note to skip if importing an existing repository. The 'Add a README file' option is selected with a checkbox, with a note 'This is where you can write a long description for your project. Learn more.' The 'Add .gitignore' section has a note to choose from a list of templates, with a link to 'Learn more.' and a dropdown menu showing '.gitignore template: Python'. The 'Choose a license' section has a note 'A license tells others what they can and can't do with your code. Learn more.' and a dropdown menu showing 'License: MIT License'. At the bottom, there's a note 'You are creating a public repository in your personal account.' and a green 'Create repository' button.

Рисунок 1 – Создание репозитория

```
c:\Users\мвидео>cd/d c:\lbrtt_2.11
c:\lbrtt_2.11>git clone https://github.com/meiokava/lbrtt_2.11.git
Cloning into 'lbrtt_2.11'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
c:\lbrtt_2.11>
```

Рисунок 2 – Клонирование репозитория

```

C:\Users\mvideo>cd/d c:\lbrtt_2.11

C:\lbrtt_2.11>git clone https://github.com/meiokava/lbrtt_2.11.git
Cloning into 'lbrtt_2.11'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\lbrtt_2.11>cd/d c:\lbrtt_2.11\lbrtt_2.11

C:\lbrtt_2.11\lbrtt_2.11>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:/lbrtt_2.11/lbrtt_2.11/.git/hooks]

C:\lbrtt_2.11\lbrtt_2.11>

```

Рисунок 3 – Организация репозитория с моделью ветвления git-flow

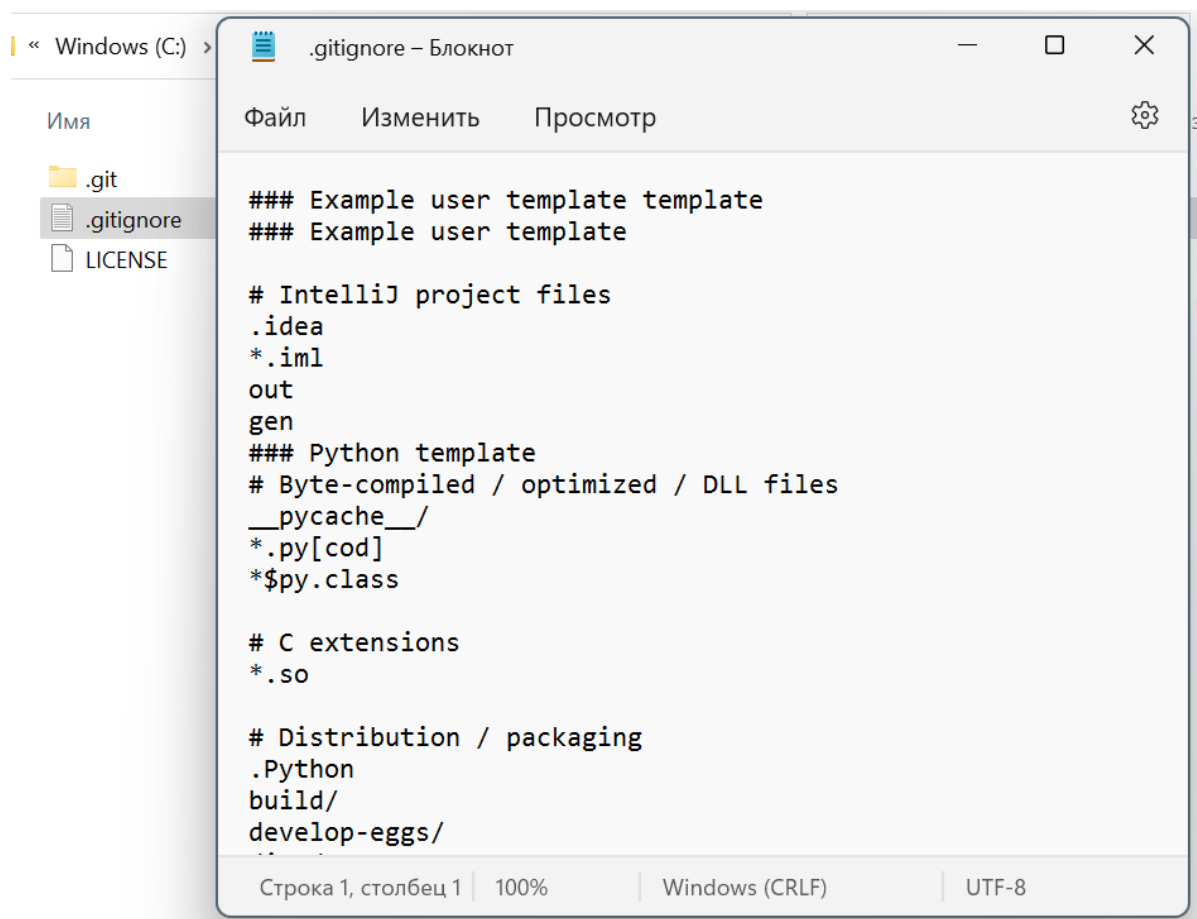


Рисунок 4 – Дополнение файла gitignore

2. Была создана папка PyCharm в которой хранятся примеры из

лабораторной работы.





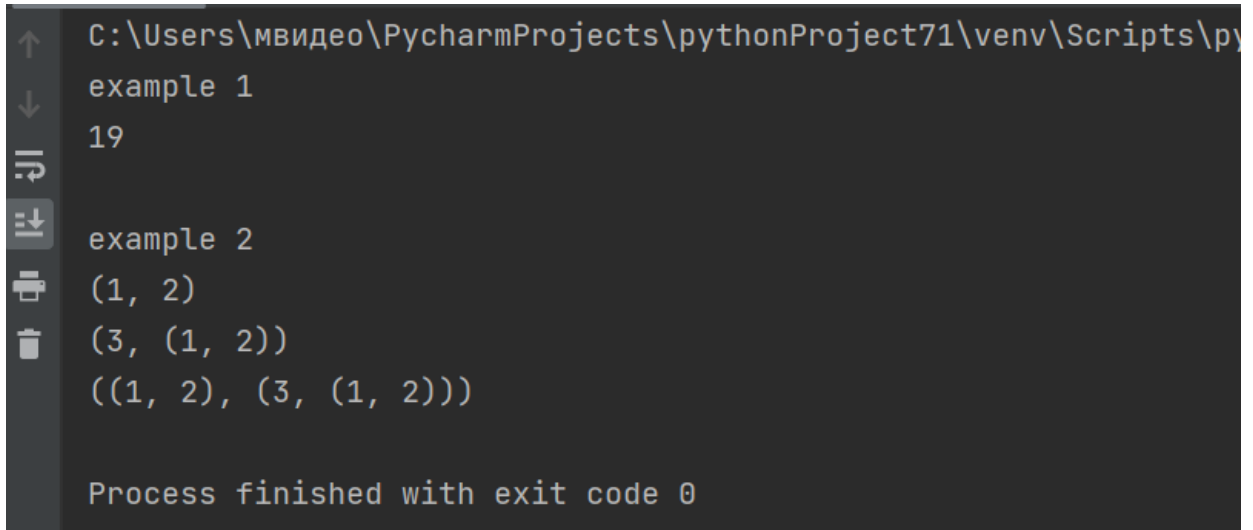
| | | | |
|---|------------|------------------|------------------|
|  | .git | 22.12.2022 11:07 | Папка с файлами |
|  | PyCharm | 22.12.2022 11:11 | Папка с файлами |
|  | .gitignore | 21.12.2022 14:09 | Файл "GITIGNORE" |
|  | LICENSE | 22.12.2022 11:06 | Файл |

Рисунок 5 – Папка PyCharm



```
C:\Users\мвидео\PycharmProjects\pythonProject71\venv\Scripts\python.exe
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

Вариант 5

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.

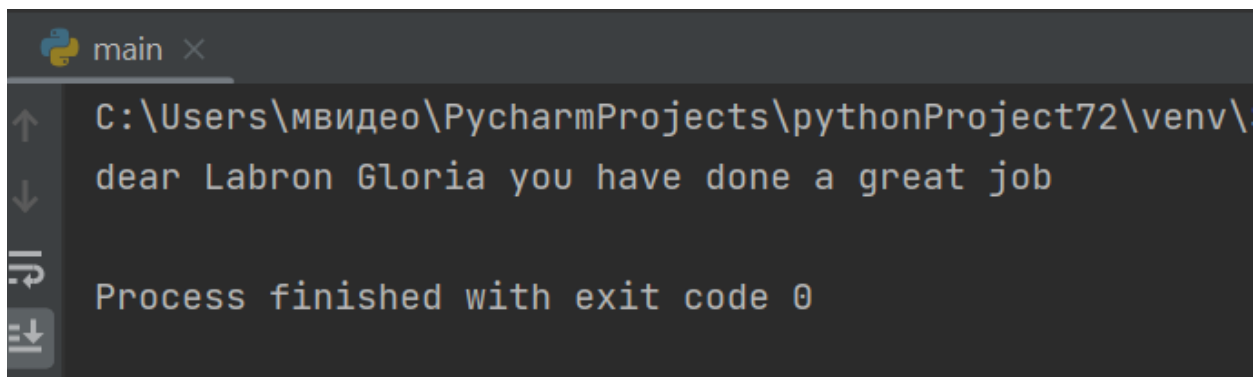
Код программы:

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

def print_msg(ms):

    def printer(nm, fm):
        data = ms.format(n=nm, f=fm)
        return data
    return printer

if __name__ == '__main__':
    d = "dear {f} {n} you have done a great job"
    another = print_msg(d)
    print(another("Gloria", "Labron"))
```



```
main x
C:\Users\мвидео\PycharmProjects\pythonProject72\venv\
dear Labron Gloria you have done a great job
Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

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 уровня интерпретатора. Для их использования не нужно импортировать модули.