

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 №15
дисциплины «Основы программной инженерии»

Выполнила:
Панюкова Ксения Юрьевна
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., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Я изучила теоретический материал работы

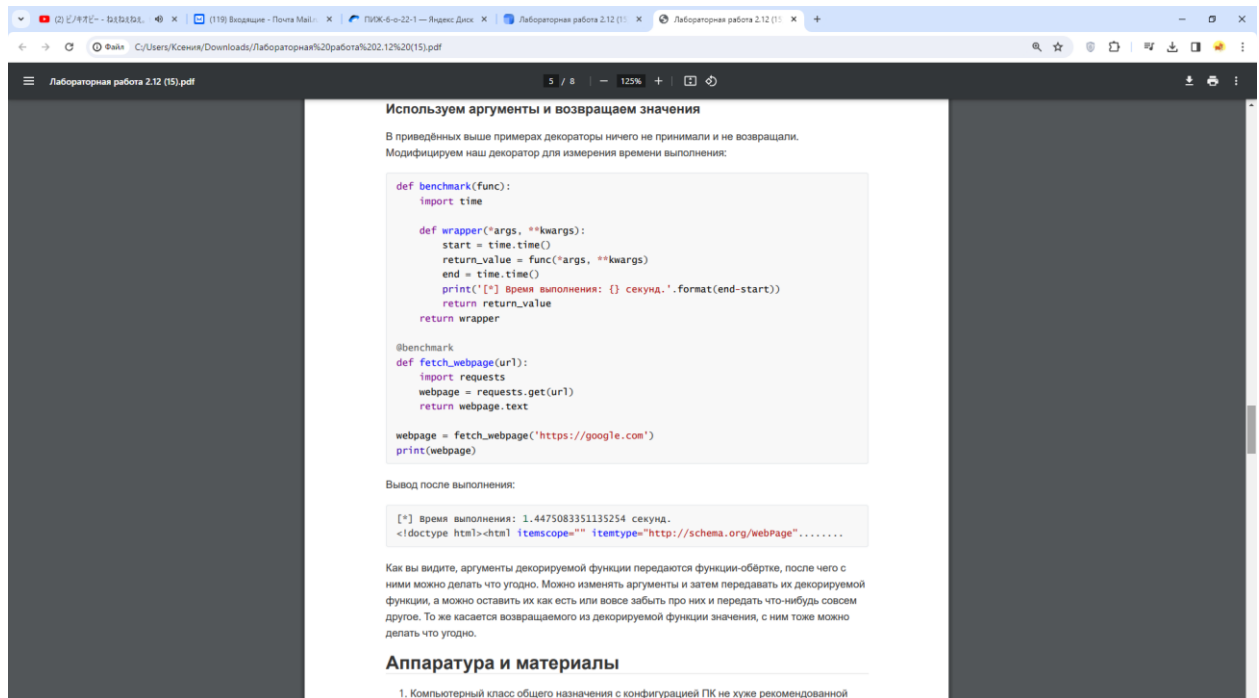


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создала 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 * Repository name *

MrFinicheck / RustAndPythonAreTheBes

✓ RustAndPythonAreTheBestProgrammingLanguages is available.

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

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

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

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

☐ You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

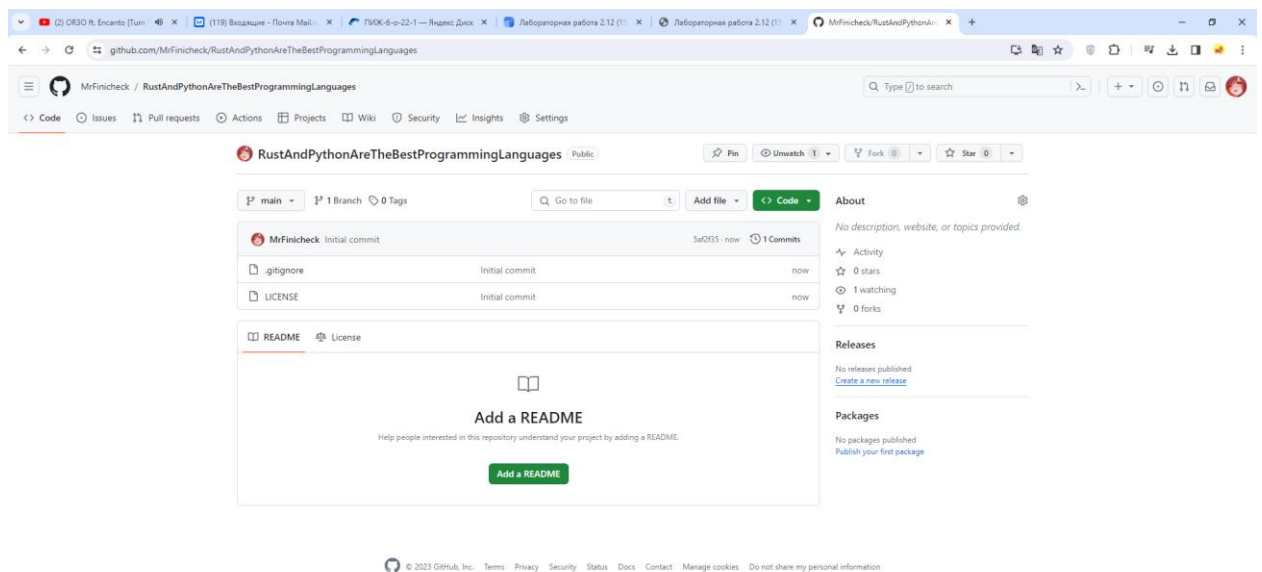


Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

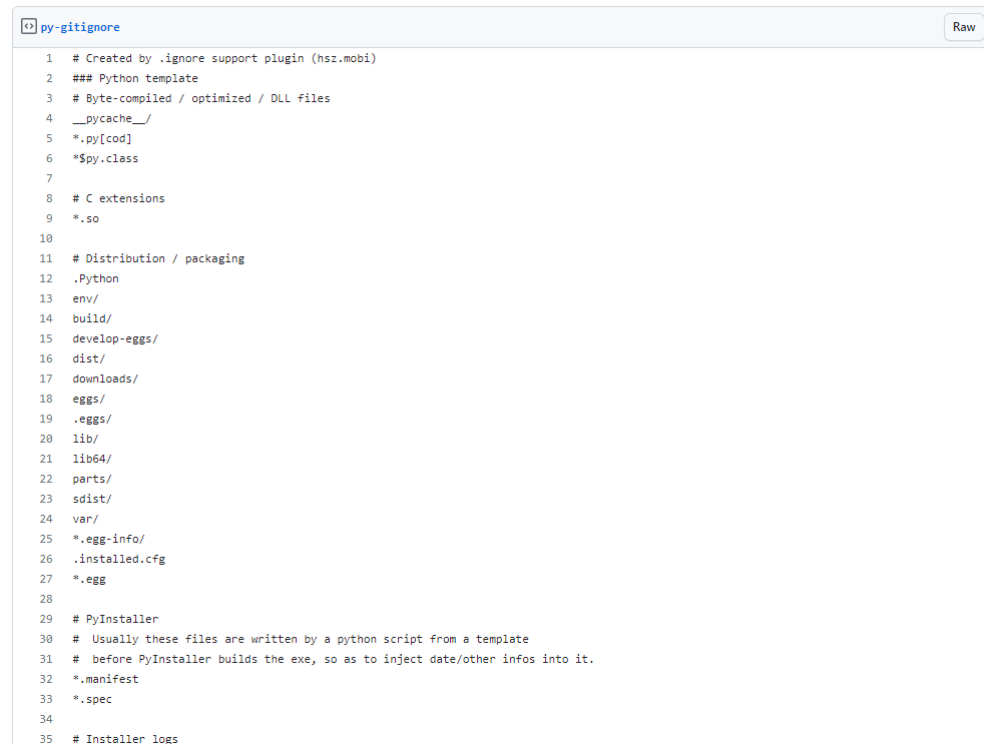
.git	23.12.2023 21:36	Папка с файлами	
.gitignore	23.12.2023 21:36	Текстовый документ	4 КБ
LICENSE	23.12.2023 21:36	Файл	2 КБ

```
Администратор: Командная строка
7688f97..91d7aa8 main -> main
C:\TestGit\PythonIsBetterThanJavascript>cd ..
C:\TestGit>git clone https://github.com/MrFinicheck/RustAndPythonAreTheBestProgrammingLanguages.git
Cloning into 'RustAndPythonAreTheBestProgrammingLanguages'...
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:\TestGit>cd RustAndPythonAreTheBestProgrammingLanguages
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

4. Дополнила файл .gitignore необходимыми правилами для работы с IDE PyCharm

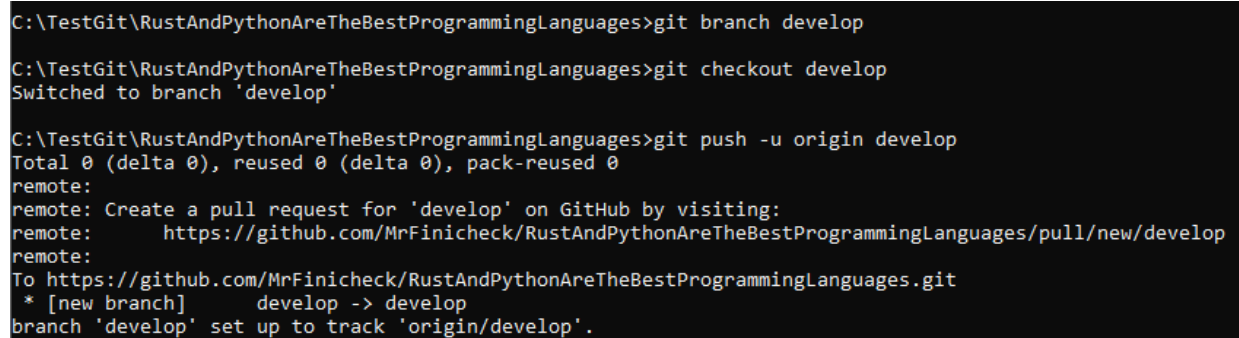
python pycharm gitignore



```
1 # Created by .ignore support plugin (hsz.mobi)
2 ### Python template
3 # Byte-compiled / optimized / DLL files
4 __pycache__/
5 *.py[cod]
6 *$py.class
7
8 # C extensions
9 *.so
10
11 # Distribution / packaging
12 .Python
13 env/
14 build/
15 develop-eggs/
16 dist/
17 downloads/
18 eggs/
19 .eggs/
20 lib/
21 lib64/
22 parts/
23 sdist/
24 var/
25 *.egg-info/
26 .installed.cfg
27 *.egg
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
34
35 # Installer logs
```

Рисунок 4.1 – .gitignore для IDE PyCharm

5. Организовала свой репозиторий в соответствии с моделью ветвления git-flow



```
C:\TestGit\RustAndPythonAreTheBestProgrammingLanguages>git branch develop
C:\TestGit\RustAndPythonAreTheBestProgrammingLanguages>git checkout develop
Switched to branch 'develop'
C:\TestGit\RustAndPythonAreTheBestProgrammingLanguages>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/MrFinicheck/RustAndPythonAreTheBestProgrammingLanguages/pull/new/develop
remote:
To https://github.com/MrFinicheck/RustAndPythonAreTheBestProgrammingLanguages.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
```

Рисунок 5.1 – Создание ветки develop от ветки main

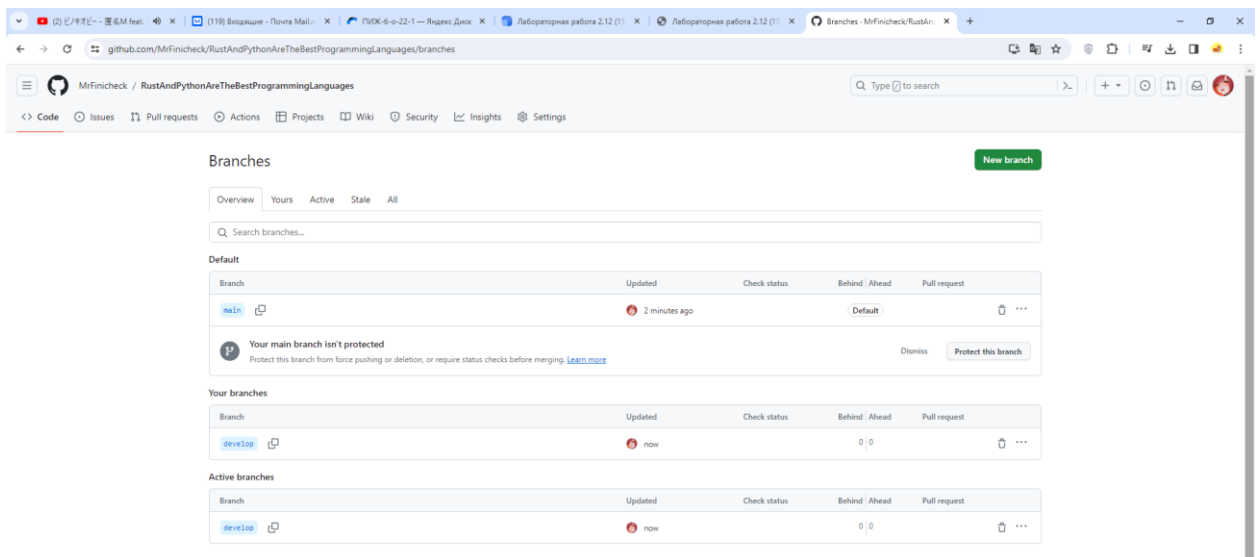


Рисунок 5.2 – Ветка develop на GitHub

6. Создала проект PyCharm в папке репозитория

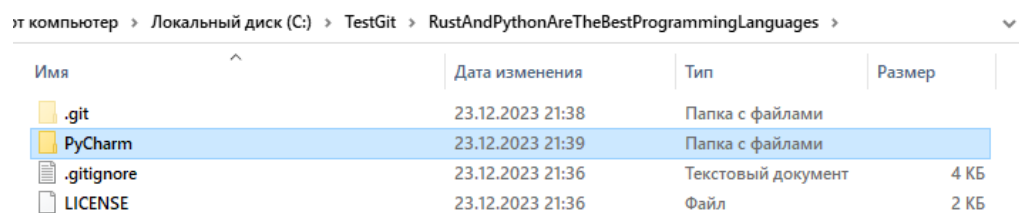


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработала примеры лабораторной работы.

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

def benchmark(func):
    import time

    def wrapper():
        start = time.time()
        func()
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end - start))

    return wrapper

@benchmark
def fetch_webpage():
    import requests
    webpage = requests.get('https://google.com')

if __name__ == '__main__':
    fetch_webpage()
```

Рисунок 7.1 – Код программы solution1.py в IDE PyCharm

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

def benchmark(func):
    import time

    def wrapper(*args, **kwargs):
        start = time.time()
        return_value = func(*args, **kwargs)
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end - start))
        return return_value

    return wrapper

@benchmark
def fetch_webpage(url):
    import requests
    webpage = requests.get(url)
    return webpage.text

if __name__ == '__main__':
    webpage = fetch_webpage('https://google.com')
    print(webpage)
```

Рисунок 7.2 – Код программы solution2.py в IDE PyCharm

8. Выполнила индивидуальное задание.

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

def tag_decorator(tag):
    def decorator(func):
        def wrapper(string):
            result = func(string)
            return f"<{tag}>{result}</{tag}>"

        return wrapper

    return decorator

@tag_decorator(tag="div")
def lowercase(string):
    return string.lower()

if __name__ == '__main__':
    result = lowercase(input("Введите строку: "))
    print(result)
```

Рисунок 8.1 – Код программы SOLUTION1.py в IDE PyCharm

9. Зафиксировала изменения в репозитории.

```
C:\TestGit\RustAndPythonAreTheBestProgrammingLanguages>git add PyCharm
C:\TestGit\RustAndPythonAreTheBestProgrammingLanguages>git commit -m"add files"
[develop a49ae65] add files
3 files changed, 78 insertions(+)
create mode 100644 PyCharm/excercise/solution1.py
create mode 100644 PyCharm/excercise/solution2.py
create mode 100644 PyCharm/individual/SOLUTION1.py
```

Рисунок 9.1 – Коммит файлов в репозитории git

Контрольные вопросы

1. Что такое декоратор?

Декоратор – это функция в Python, которая позволяет изменить поведение другой функции без изменения её кода. Он используется для добавления функциональности к существующей функции, обертывая её вокруг другой функции.

2. Почему функции являются объектами первого класса?

Функции в Python являются объектами первого класса, потому что они могут быть присвоены переменным, переданы как аргументы в функции, возвращены из другой функции и имеют те же свойства, что и другие типы данных в Python.

3. Каково назначение функций высших порядков?

Функции высших порядков – это функции, которые могут принимать другие функции в качестве аргументов или возвращать их как результат. Они позволяют абстрагировать действия и работать с функциями как с данными.

4. Как работают декораторы?

Декораторы работают путем обертывания одной функции внутри другой. Это позволяет изменять поведение декорируемой функции, не изменяя её код.

5. Какова структура декоратора функций?

Декоратор функции – это функция, которая принимает другую функцию в качестве аргумента, обычно использует внутреннюю функцию (wrapper), которая оборачивает оригинальную функцию и возвращает эту внутреннюю функцию.

6. Самостоятельно изучить как можно передать параметры декоратору, а не декорируемой функции?

В Python можно передать параметры декоратору, используя дополнительную обертку. Можно создать функцию-декоратор, которая принимает аргументы и возвращает другую функцию, которая уже будет декоратором.