

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 №13**  
**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

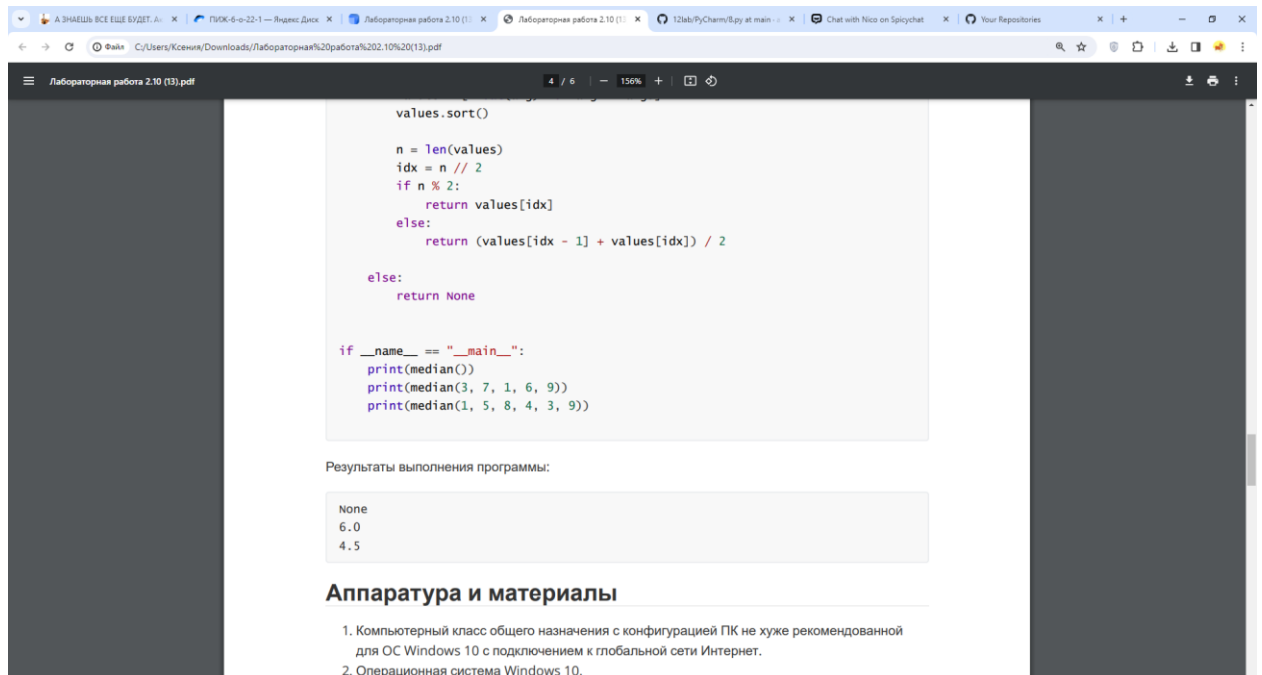


Рисунок 1.1 – Изучение материала для лабораторной работы

### 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 \*

☒ PythonsCanSwim is available.

Great repository names are short and memorable. Need inspiration? How about [bug-free-umbrella](#) ?

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

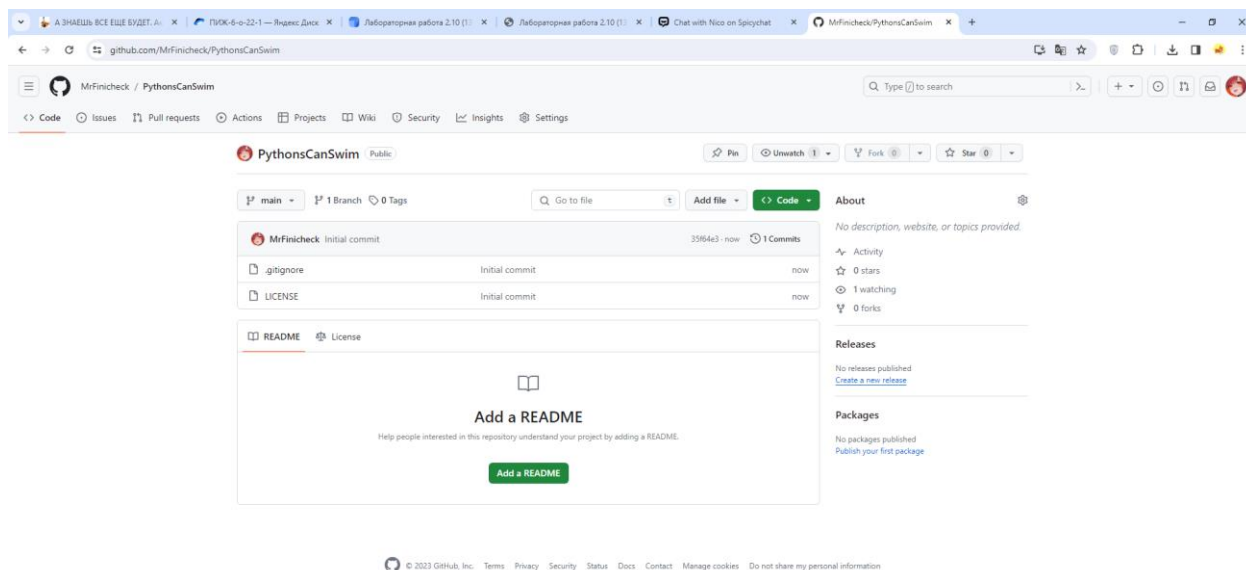


Рисунок 2.2 – Готовый репозиторий

### 3. Выполняю клонирование созданного репозитория

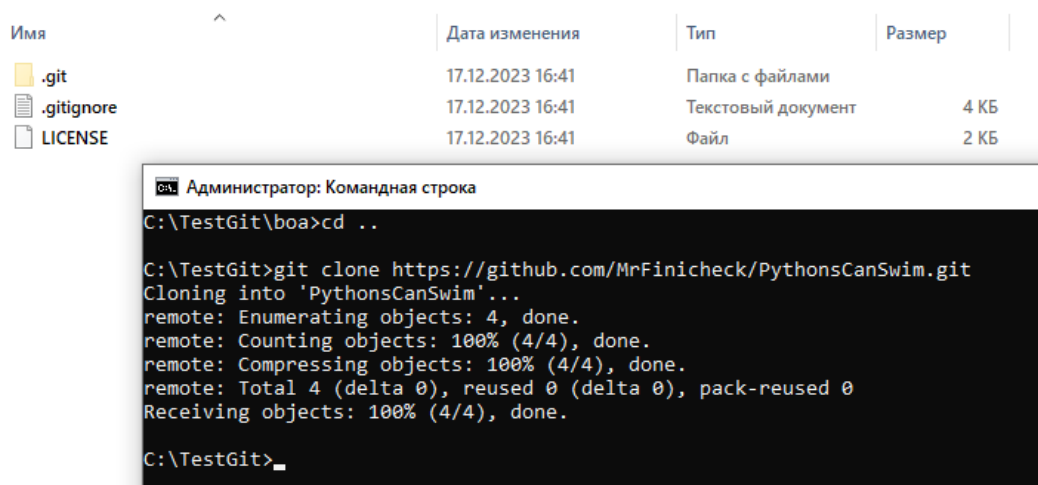
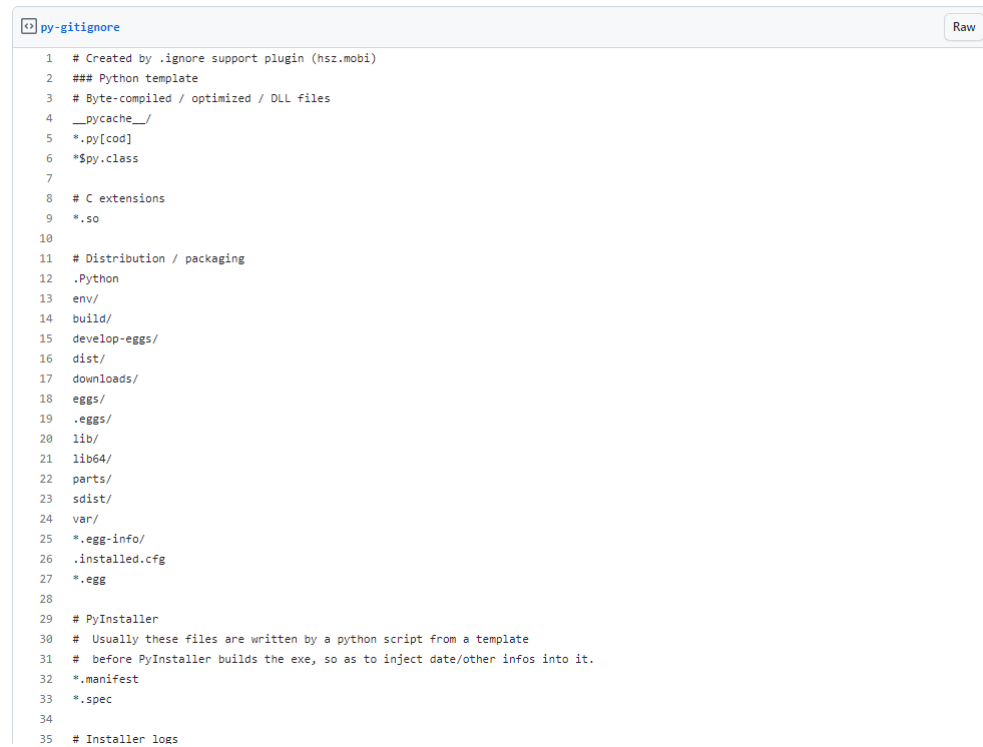


Рисунок 3.1 – Клонирование репозитория на локальный диск

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

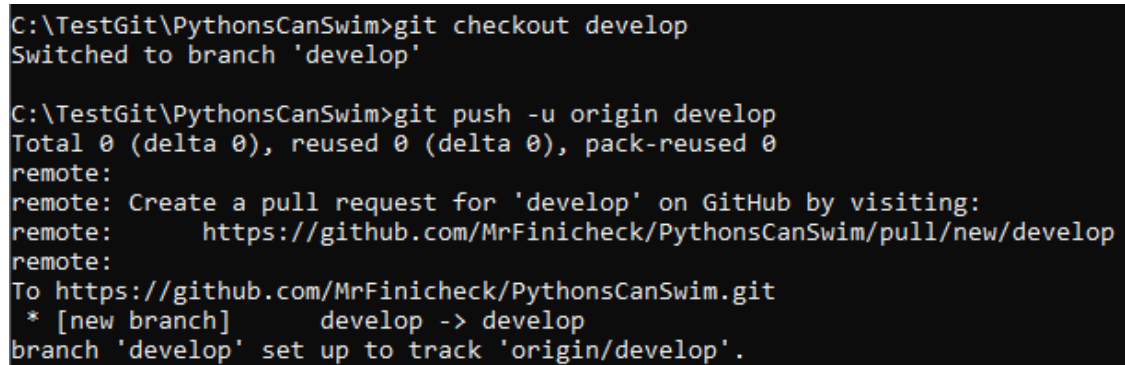
python pycharm gitignore

A screenshot of a PyCharm IDE window showing a .gitignore file. The window title is 'py-gitignore'. The file content is a standard Python .gitignore template, listing various files and directories to be ignored, such as \_\_pycache\_\_, \*.py[cod], \*.py.class, \*.so, .Python, env/, build/, develop-eggs/, dist/, downloads/, eggs/, .eggs/, lib/, lib64/, parts/, sdist/, var/, \*.egg-info/, .installed.cfg, \*.egg, \*.manifest, \*.spec, and \*.spec. The file is numbered from 1 to 35.

```
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

A screenshot of a terminal window showing the execution of git commands. The first command is 'git checkout develop', which switches to the 'develop' branch. The second command is 'git push -u origin develop', which pushes the 'develop' branch to the 'origin' remote. The output shows that the push was successful and provides a link to create a pull request on GitHub.

```
C:\TestGit\PythonsCanSwim>git checkout develop
Switched to branch 'develop'

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

Рисунок 5.1 – Создание ветки develop от ветки main

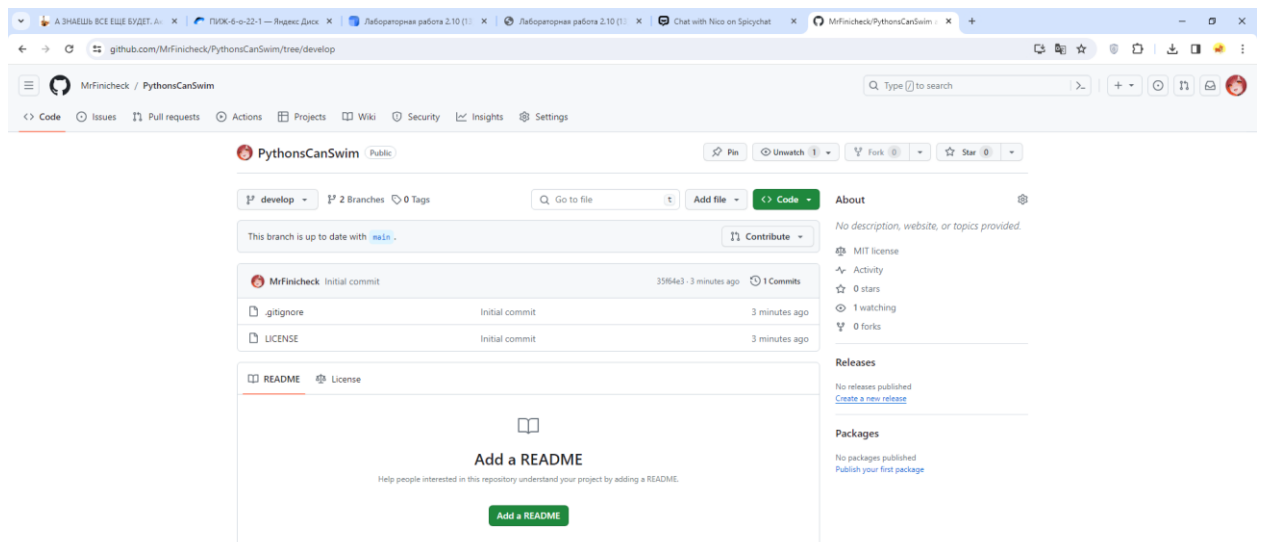


Рисунок 5.2 – Ветка develop на GitHub

## 6. Создала проект PyCharm в папке репозитория

Имя	Дата изменения	Тип	Размер
.git	17.12.2023 16:43	Папка с файлами	
PyCharm	17.12.2023 16:45	Папка с файлами	
.gitignore	17.12.2023 16:41	Текстовый документ	4 КБ
LICENSE	17.12.2023 16:41	Файл	2 КБ

Рисунок 6.1 – Репозиторий с проектом PyCharm

## 7. Проработала примеры лабораторной работы.

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

def median(*args):
    if args:
        values = [float(arg) for arg in args]
        values.sort()
        n = len(values)
        idx = n // 2
        if n % 2:
            return values[idx]
        else:
            return (values[idx - 1] + values[idx]) / 2
    else:
        return None

if __name__ == "__main__":
    print(median())
    print(median(*args: 3, 7, 1, 6, 9))
    print(median(*args: 1, 5, 8, 4, 3, 9))
```

Рисунок 7.1 – Код программы exercise1.py в IDE PyCharm

8. Решила поставленную задачу: написала функцию, вычисляющую среднее геометрическое своих аргументов.

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

import math

def geometric_mean(*args):
    if args:
        number_of_arguments = len(args)
        count = 1
        for i in args:
            count *= i

        solution = math.pow(count, 1 / number_of_arguments)
        return solution

    else:
        return None

if __name__ == "__main__":
    arguments = list(map(int, input("Введите аргументы: ").split()))
    result = geometric_mean(*arguments)
    print(f"Среднее геометрическое введенных вами чисел равно: "
          f"{result}")
```

Рисунок 8.1 – Код программы EXERCISE8.py в IDE PyCharm

9. Решила поставленную задачу: написала функцию, вычисляющую среднее геометрическое своих аргументов.

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

import math

def harmonic_mean(*args):
    if args:
        number_of_arguments = len(args)
        count = 0
        for i in args:
            count += 1/i

        return number_of_arguments/count

    else:
        return None

if __name__ == "__main__":
    arguments = list(map(int, input("Введите аргументы: ").split()))
    result = harmonic_mean(*arguments)

    print(f"Среднее гармоническое введенных вами аргументов равно: "
          f"{result}")
```

Рисунок 9.1 – Код программы EXERCISE9.py в IDE PyCharm

10. Привела в отчете скриншоты результатов выполнения примера при различных исходных данных, вводимых с клавиатуры.

```
C:\Users\Ксения\PycharmProjects\pyt
None
6.0
4.5
```

Рисунок 10.1 – Результат выполнения примера exercise1.py

11. Зафиксировала сделанные изменения в репозитории.

```
C:\TestGit\PythonsCanSwim>git add PyCharm

C:\TestGit\PythonsCanSwim>git commit -m"add tasks"
[develop ba8ce23] add tasks
5 files changed, 116 insertions(+)
create mode 100644 PyCharm/Examples/exercise1.py
create mode 100644 PyCharm/Individual/Single.py
create mode 100644 PyCharm/Tasks/EXERCISE13.py
create mode 100644 PyCharm/Tasks/EXERCISE8.py
create mode 100644 PyCharm/Tasks/EXERCISE9.py

C:\TestGit\PythonsCanSwim>
```

Рисунок 11.1 – Коммит файлов в репозитории git

12. Решила индивидуальное задание согласно своему варианту.

```
def sum_after_last_zero(*args):
    last_zero_index = -1
    for i in range(len(args) - 1, -1, -1):
        if args[i] == 0:
            last_zero_index = i
            break

    # Если нулей нет.
    if last_zero_index == -1:
        return None

    return sum(args[last_zero_index + 1:])

if __name__ == '__main__':
    arguments = list(map(int, input("Введите аргументы: ").split()))
    result = sum_after_last_zero(*arguments)

    print(f"Сумму аргументов, расположенных после последнего аргумента,  
f"равного нулю"  
f"{result}")
```

Рисунок 12.1 – Код программы single.py в IDE PyCharm

13. Самостоятельно подобрала или придумала задачу с переменным числом именованных аргументов. Привела решение этой задачи.

Задача: Напишите функцию, которая принимает переменное количество именованных аргументов (например, предметы и их цены) и выводит их на экран в виде списка.

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

def print_items(**kwargs):
    for item, price in kwargs.items():
        print(f"{item}: {price}")

if __name__ == '__main__':
    print_items(apple=2, banana=1, orange=1.5, pear=2.5)
```

Рисунок 13.1 – Код программы single.py в IDE PyCharm

14. Зафиксировала изменения в репозитории.



```
C:\TestGit\PythonsCanSwim>git commit -m"add other tasks"
[develop 16b3375] add other tasks
2 files changed, 7 insertions(+), 2 deletions(-)
```

Рисунок 14.1 – Коммит файлов в репозитории git

### Контрольные вопросы

1. Какие аргументы называются позиционными в Python?

Позиционные аргументы — это аргументы, которые передаются в функцию в определенном порядке, и их значения связываются с параметрами функции в том же порядке.

2. Какие аргументы называются именованными в Python?

Именованные аргументы — это аргументы, которые передаются в функцию с указанием имени параметра.

3. Для чего используется оператор \*?

Этот оператор позволяет «распаковывать» объекты, внутри которых хранятся некие элементы.

4. Каково назначение конструкций \*args и \*\*kwargs?

Каждая из этих конструкций используется для распаковки аргументов соответствующего типа, позволяя вызывать функции со списком аргументов переменной длины