

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»

Выполнил:
Звездин Алексей Сергеевич
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 изучил теоретический материал работы

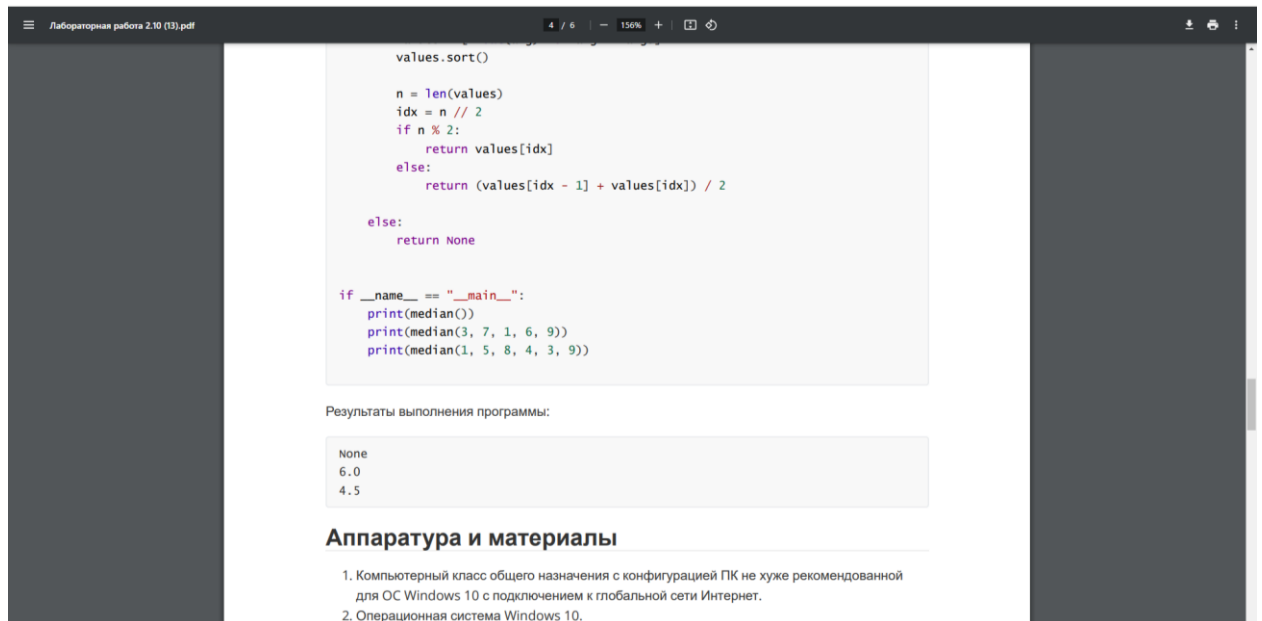


Рисунок 1.1 – Изучение материала для лабораторной работы


2. Создал 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 *

 InternetHacker1123 ▾

Repository name *

/

✔ Your new repository will be created as laba2.10-13-.

The repository name can only contain ASCII letters, digits, and the characters -, ., and _.

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

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.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

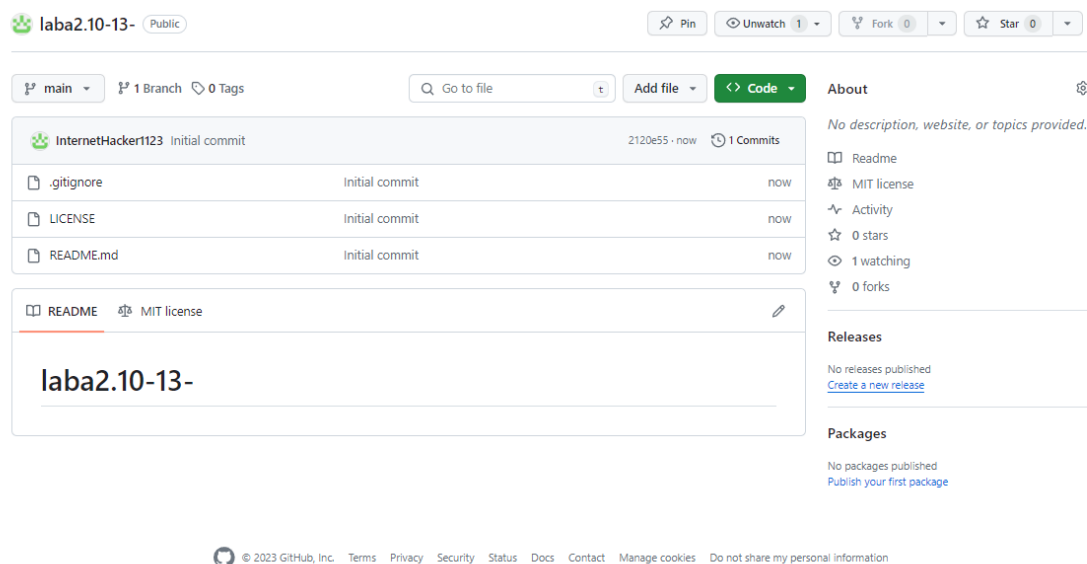


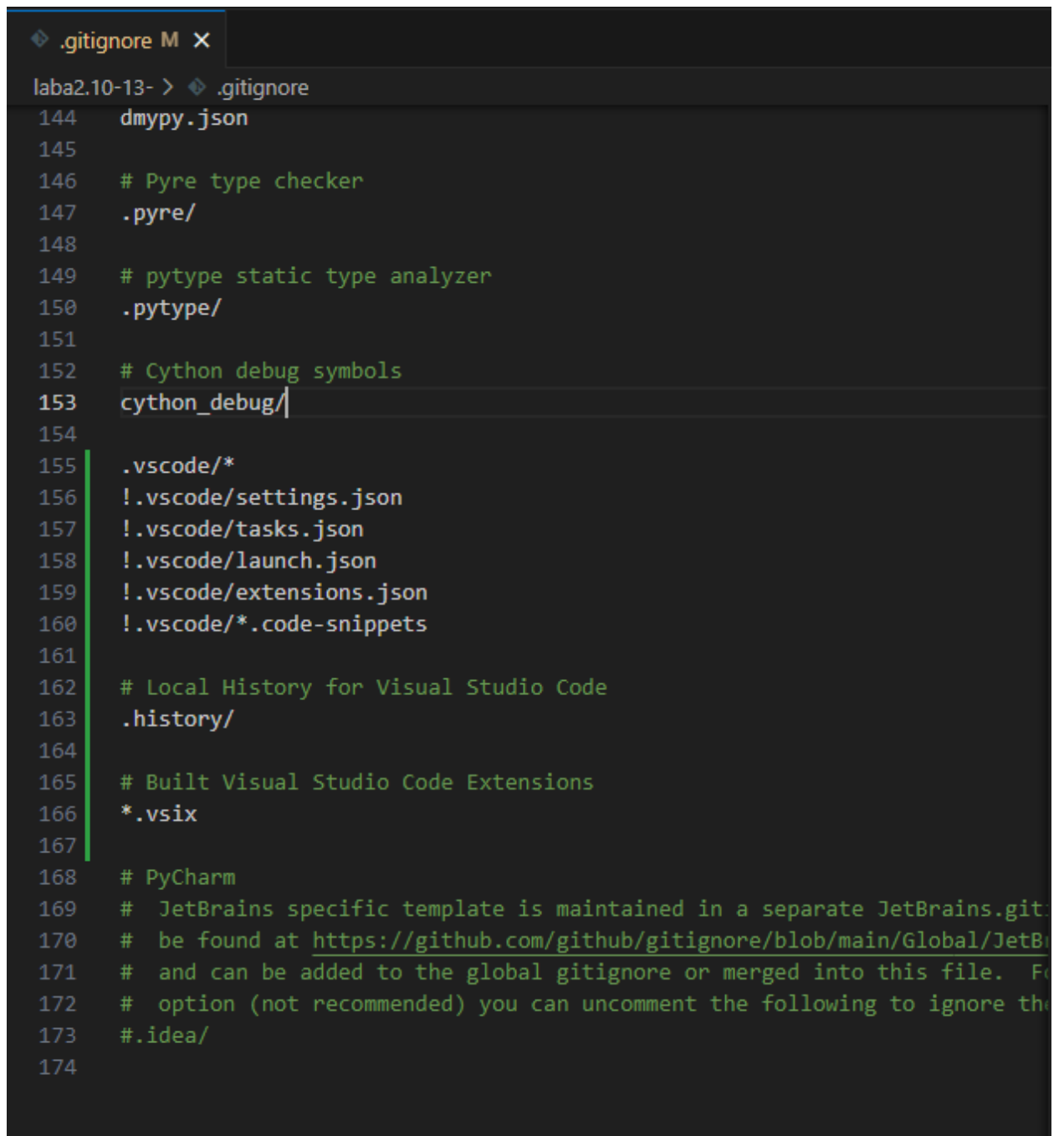
Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\laba13>git clone https://github.com/InternetHacker1123/laba2.10-13-.git
Cloning into 'laba2.10-13-'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\tyt\Desktop\SE\laba13>
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

4. Дополнил файл .gitignore необходимыми правилами для работы с VS Code

A screenshot of a code editor showing a .gitignore file. The editor has a dark theme. The file name ".gitignore" is in the top left corner of the editor window. The content of the file is as follows:

```
144 dmypy.json
145
146 # Pyre type checker
147 .pyre/
148
149 # pytype static type analyzer
150 .pytype/
151
152 # Cython debug symbols
153 cython_debug/
154
155 .vscode/*
156 !.vscode/settings.json
157 !.vscode/tasks.json
158 !.vscode/launch.json
159 !.vscode/extensions.json
160 !.vscode/*.code-snippets
161
162 # Local History for Visual Studio Code
163 .history/
164
165 # Built Visual Studio Code Extensions
166 *.vsix
167
168 # PyCharm
169 # JetBrains specific template is maintained in a separate JetBrains.giti
170 # be found at https://github.com/github/gitignore/blob/main/Global/JetBrains
171 # and can be added to the global gitignore or merged into this file. For
172 # option (not recommended) you can uncomment the following to ignore the
173 #.idea/
174
```

Рисунок 4.1 – .gitignore для VS Code

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

```
C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->git branch develop

C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->git checkout develop
Switched to branch 'develop'

C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->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/InternetHacker1123/laba2.10-13-/pull/new/develop
remote:
To https://github.com/InternetHacker1123/laba2.10-13-.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->
```

Рисунок 5.1 – Создание ветки develop от ветки main

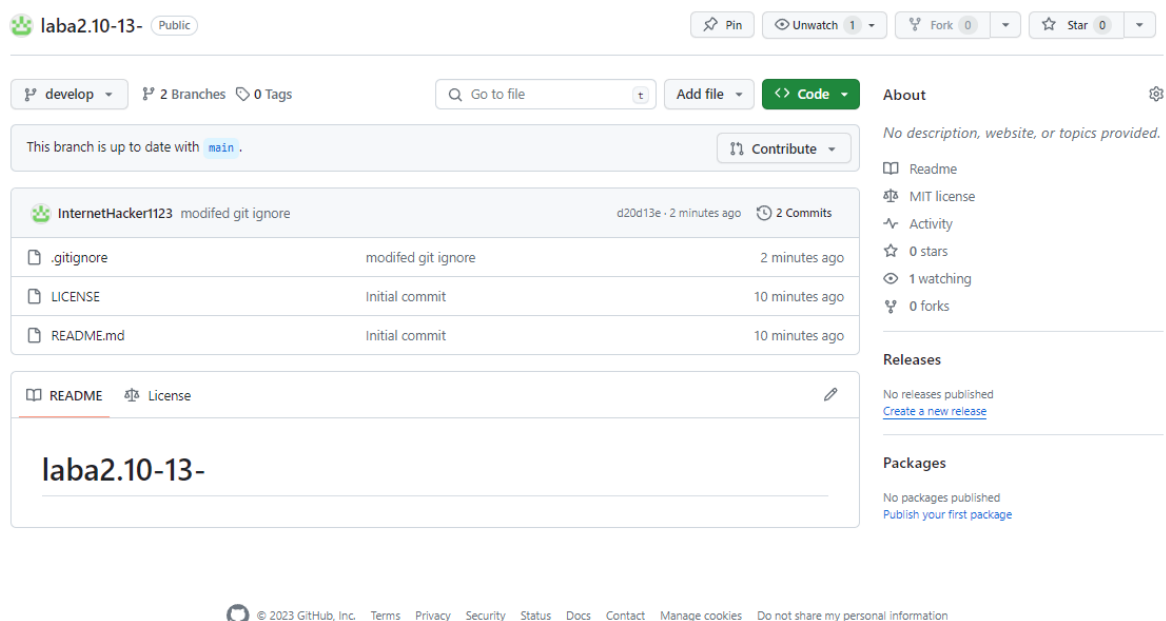


Рисунок 5.2 – Ветка develop на GitHub

6. Создал проект PyCharm в папке репозитория

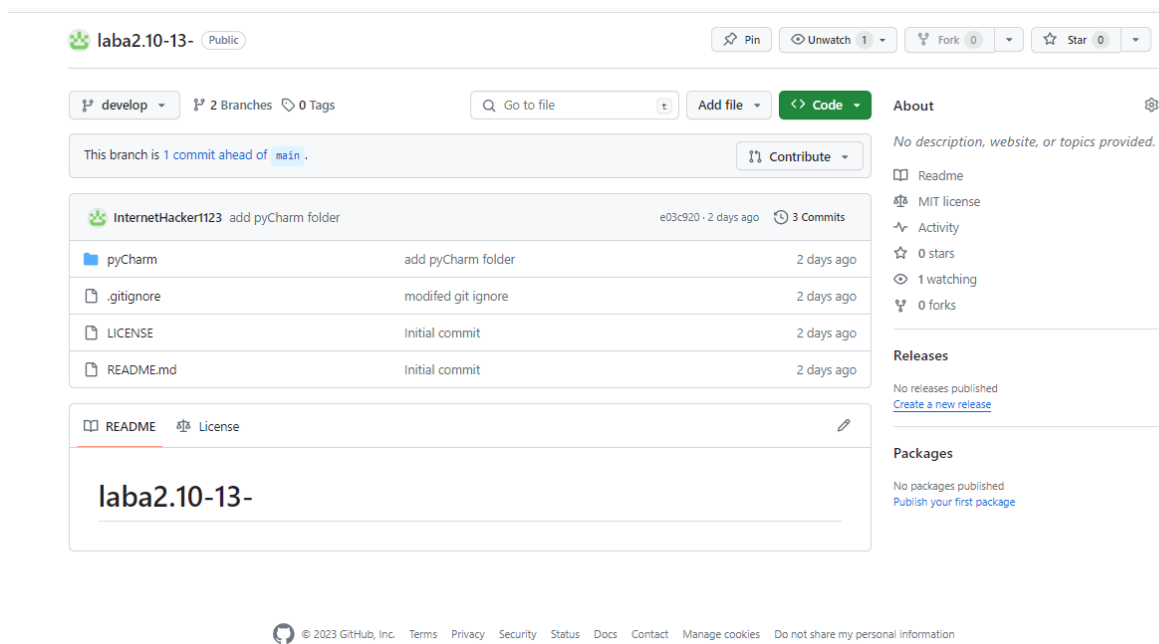


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработал примеры лабораторной работы.

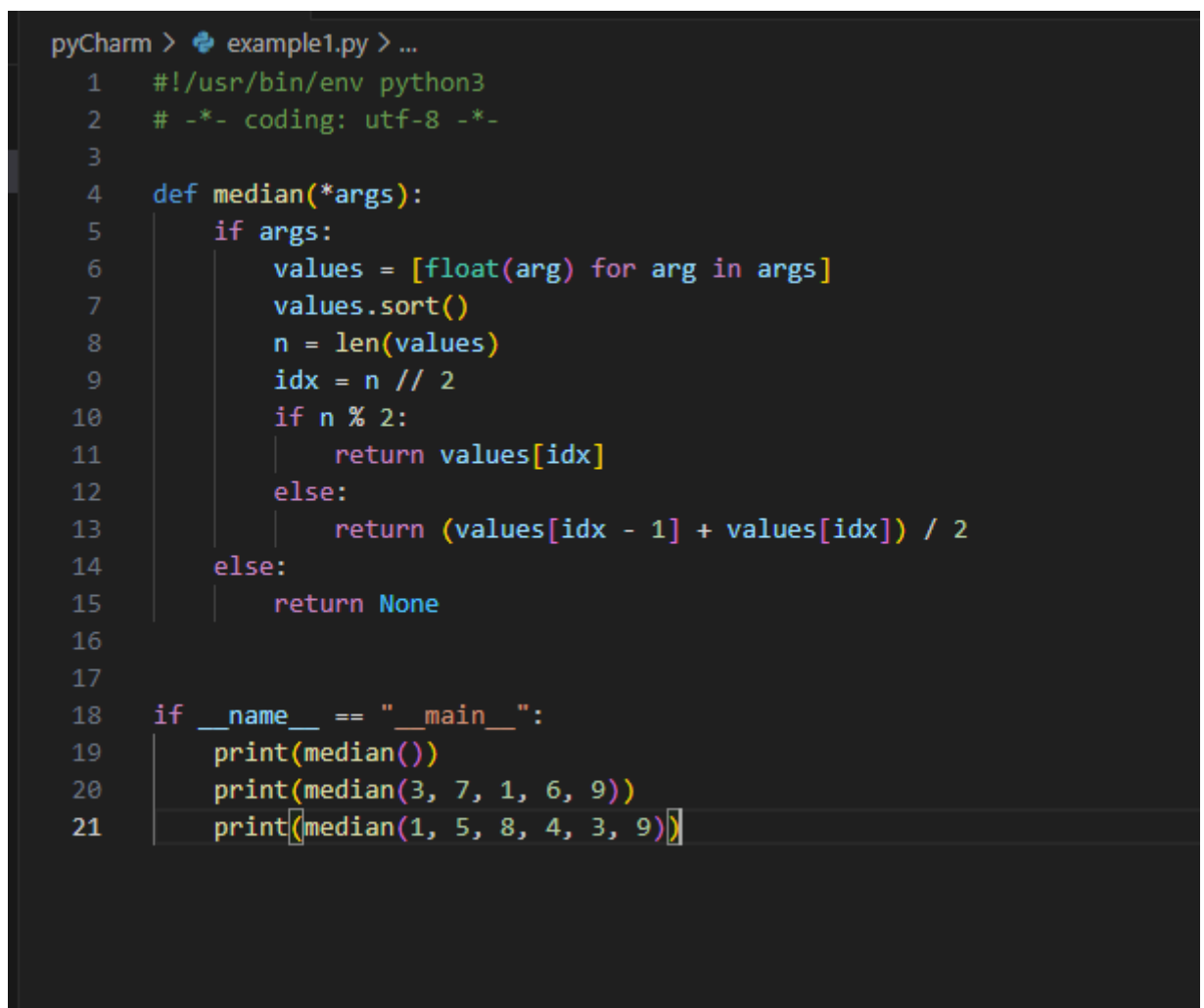
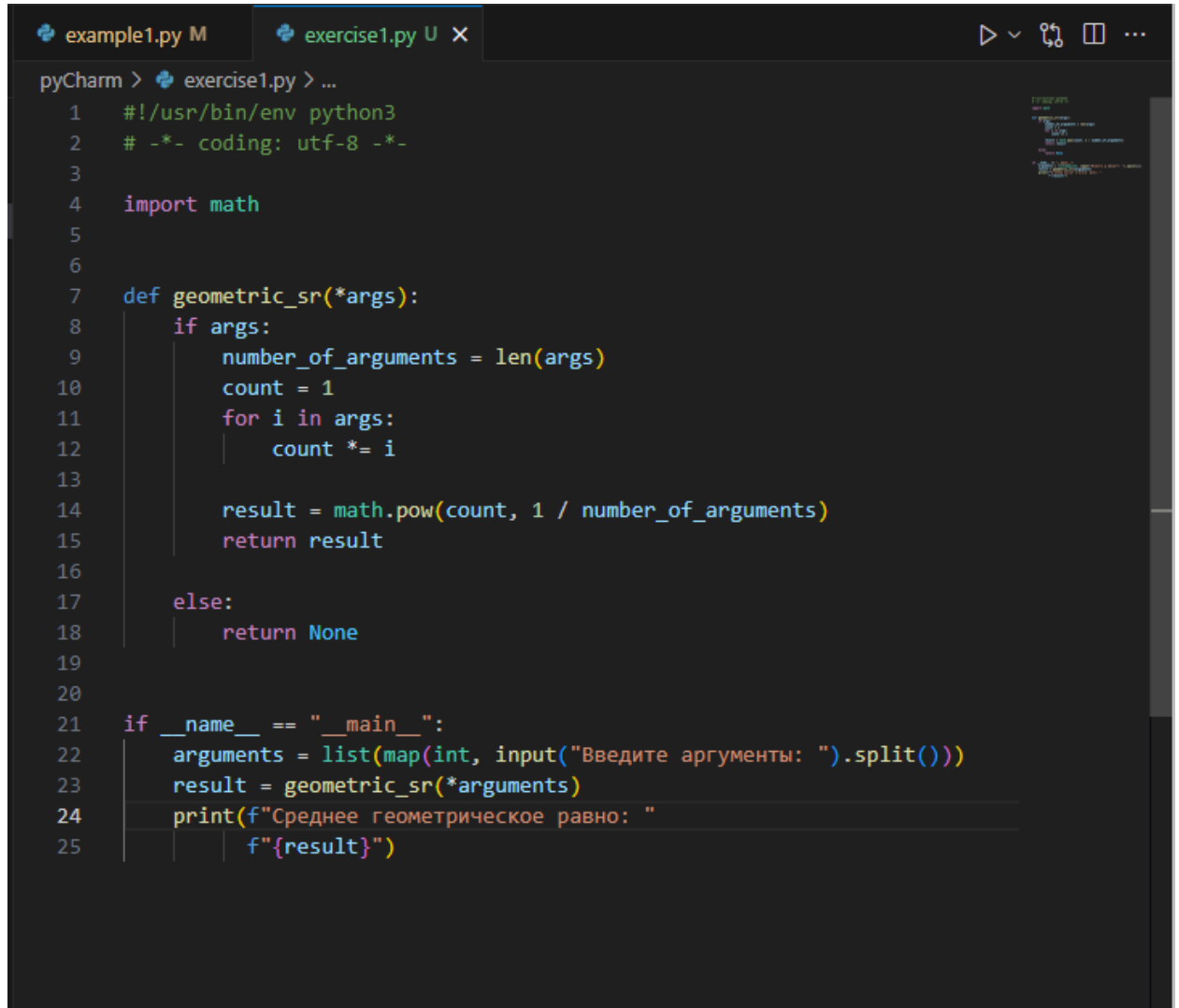


Рисунок 7.1 – Код программы example1.py

8. Решил поставленную задачу: написал функцию, вычисляющую среднее геометрическое своих аргументов.



```
pyCharm > exercise1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6
7  def geometric_sr(*args):
8      if args:
9          number_of_arguments = len(args)
10         count = 1
11         for i in args:
12             count *= i
13
14         result = math.pow(count, 1 / number_of_arguments)
15         return result
16
17     else:
18         return None
19
20
21 if __name__ == "__main__":
22     arguments = list(map(int, input("Введите аргументы: ").split()))
23     result = geometric_sr(*arguments)
24     print(f"Среднее геометрическое равно: "
25           f"{result}")
```

Рисунок 8.1 – Код программы exercise8.py

9. Решил поставленную задачу: написал функцию, вычисляющую среднее гармоническое своих аргументов.


```
pyCharm > exercise2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def harmonic_sr(*args):
5      if args:
6          number_of_arguments = len(args)
7          count = 0
8          for i in args:
9              count += 1/i
10         result = number_of_arguments/count
11         return result
12
13     else:
14         return None
15
16
17 if __name__ == "__main__":
18     arguments = list(map(int, input("Введите аргументы: ").split()))
19     result = harmonic_sr(*arguments)
20     print(f"Среднее гармоническое равно: {result}")
```

Рисунок 9.1 – Код программы exercise2.py в IDE PyCharm

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

```
on311/python.exe c:/Users/tyt/Desktop/SE/laba13/laba2.10-13-/pyChar
Введите аргументы: 1 2 3 4 5 6
Среднее геометрическое равно: 2.993795165523909
PS C:\Users\tyt\Desktop\SE\laba13\laba2.10-13-> |
```

Рисунок 10.1 – Результат выполнения примера exercise1.py

11. Зафиксировал сделанные изменения в репозитории.

```
C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->git commit -m "add exercises"
[develop a156797] add exercises
3 files changed, 66 insertions(+)
create mode 100644 pyCharm/exercise1.py
create mode 100644 pyCharm/exercise2.py

C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->
```

Рисунок 11.1 – Коммит файлов в репозитории git

12. Решил индивидуальное задание согласно своему варианту.

```
pyCharm > individual.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def geometric_sr(*args):
5      if args:
6          cnt = 0
7          result = 0
8          for i in args:
9              if i < 0:
10                 cnt += 1
11                 continue
12                 if cnt == 1:
13                     result += i
14                     continue
15                     if cnt == 2:
16                         break
17             return result
18         else:
19             return None
20
21
22 if __name__ == "__main__":
23     arguments = list(map(int, input("Введите аргументы: ").split()))
24     result = geometric_sr(*arguments)
25     print(f"Результат: {result}")
```

Рисунок 12.1 – Код программы individual.py

13. Самостоятельно подобрал или придумал задачу с переменным числом именованных аргументов. Привел 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.

```
pyCharm > individual2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def func(*args):
5      if args:
6          return args
7      else:
8          return None
9
10
11 if __name__ == "__main__":
12     arguments = list(map(str, input("Введите дела: ").split()))
13     result = func(*arguments)
14     for i, item in enumerate(result):
15         print(f"Дело {i}: {item}")
```

Рисунок 13.1 – Код программы individual.py

14. Зафиксировал изменения в репозитории.

```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->add ;
"add" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.
C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->add .
"add" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.
C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->git add .
C:\Users\tyt\Desktop\SE\laba13\laba2.10-13->git commit -m"add all tasks"
[develop 0f86bb2] add all tasks
2 files changed, 40 insertions(+)
create mode 100644 pyCharm/individual.py
create mode 100644 pyCharm/individual2.py
```

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