

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»

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

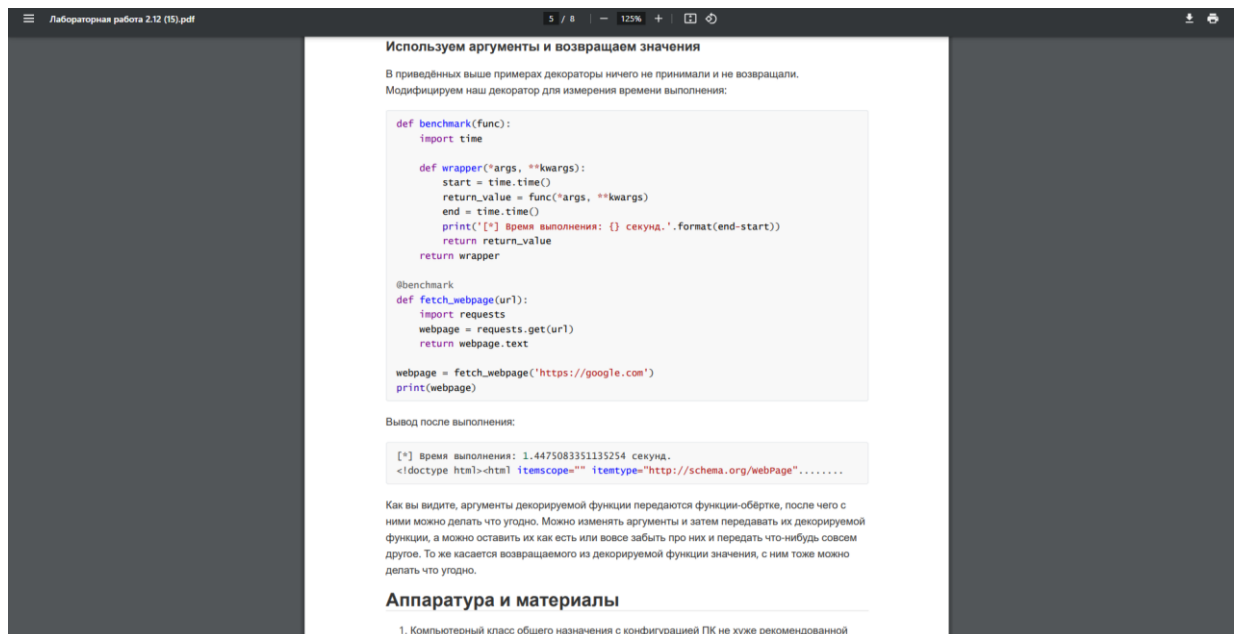


Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python

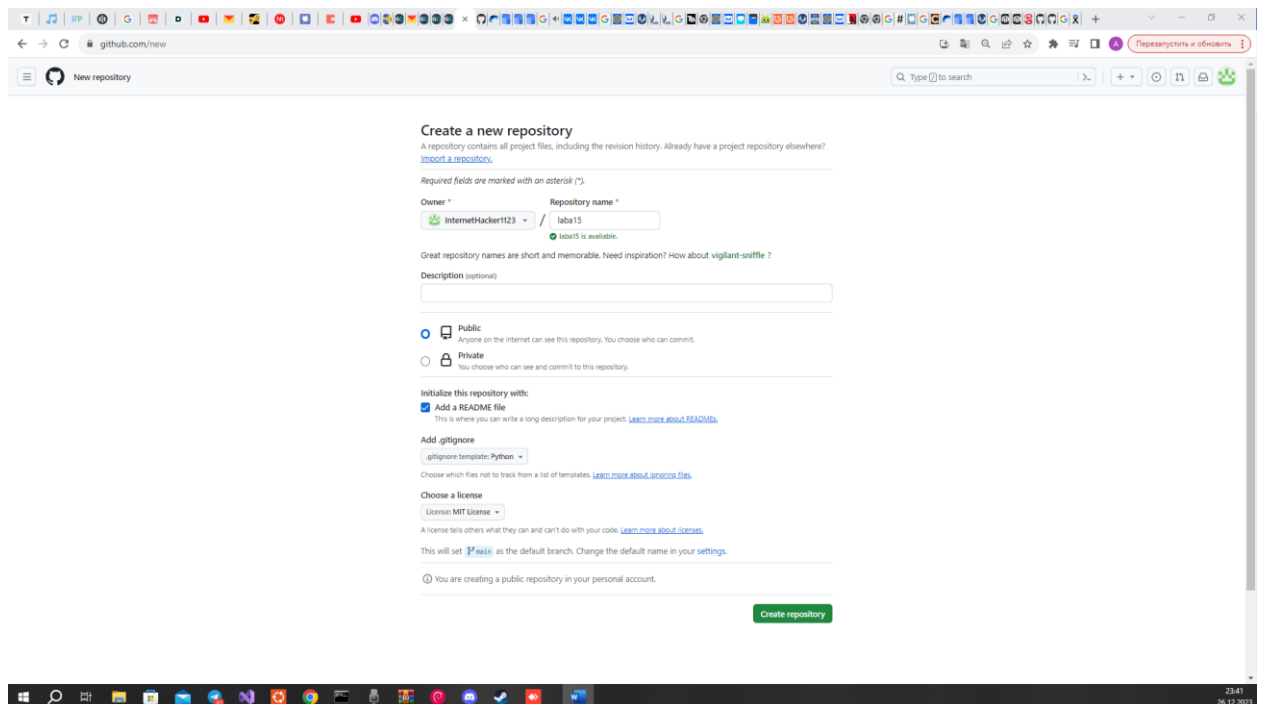


Рисунок 2.1 – Настройка репозитория

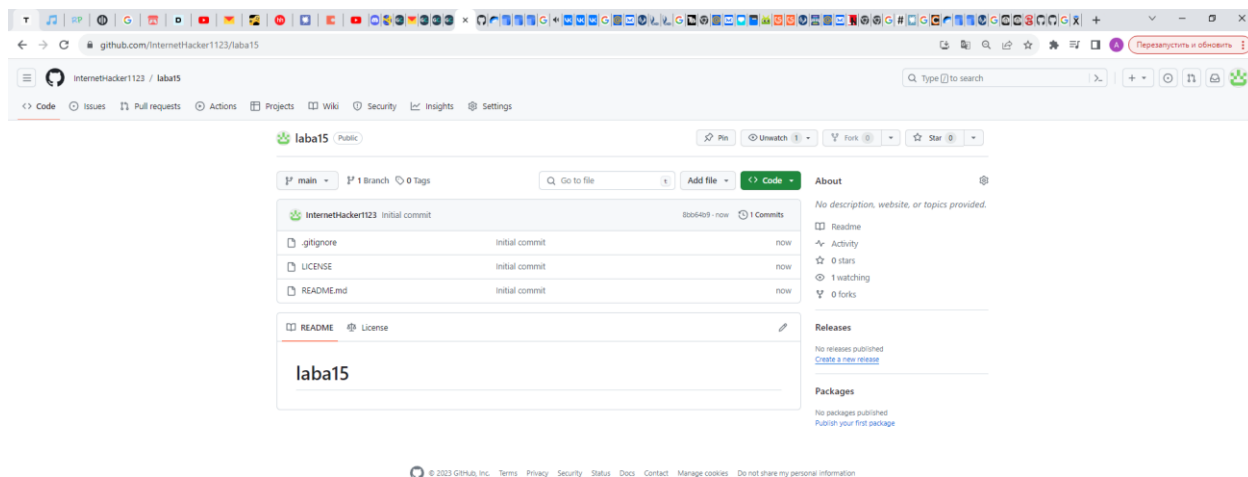


Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

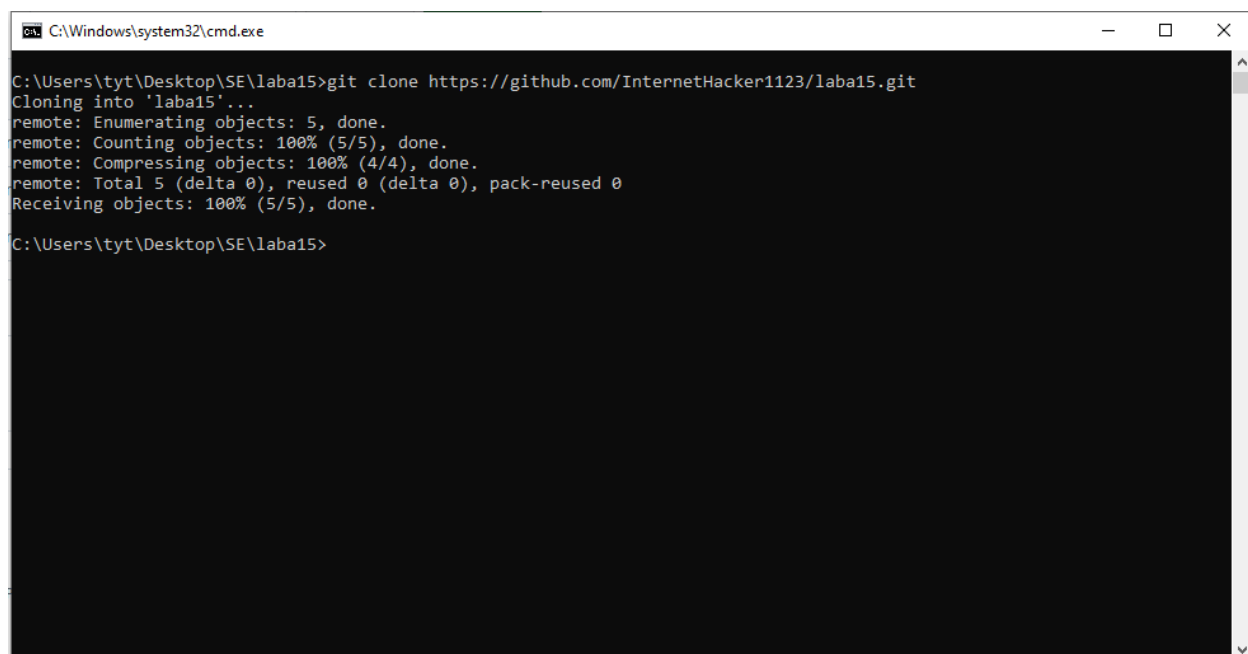


Рисунок 3.1 – Клонирование репозитория на локальный диск

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

```
.gitignore M x
.gitignore
135 # Rope project settings
136 .ropeproject
137
138 # mkdocs documentation
139 /site
140
141 # mypy
142 .mypy_cache/
143 .dmy.py.json
144 dmy.py.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.gitignore that can
170 # be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
171 # and can be added to the global gitignore or merged into this file. For a more nuclear
172 # option (not recommended) you can uncomment the following to ignore the entire idea folder.
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\laba15\laba15>git branch develop
C:\Users\tyt\Desktop\SE\laba15\laba15>git checkout develop
Switched to branch 'develop'
M       .gitignore

C:\Users\tyt\Desktop\SE\laba15\laba15>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/laba15/pull/new/develop
remote:
To https://github.com/InternetHacker1123/laba15.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\Users\tyt\Desktop\SE\laba15\laba15>
```

Рисунок 5.1 – Создание ветки develop от ветки main

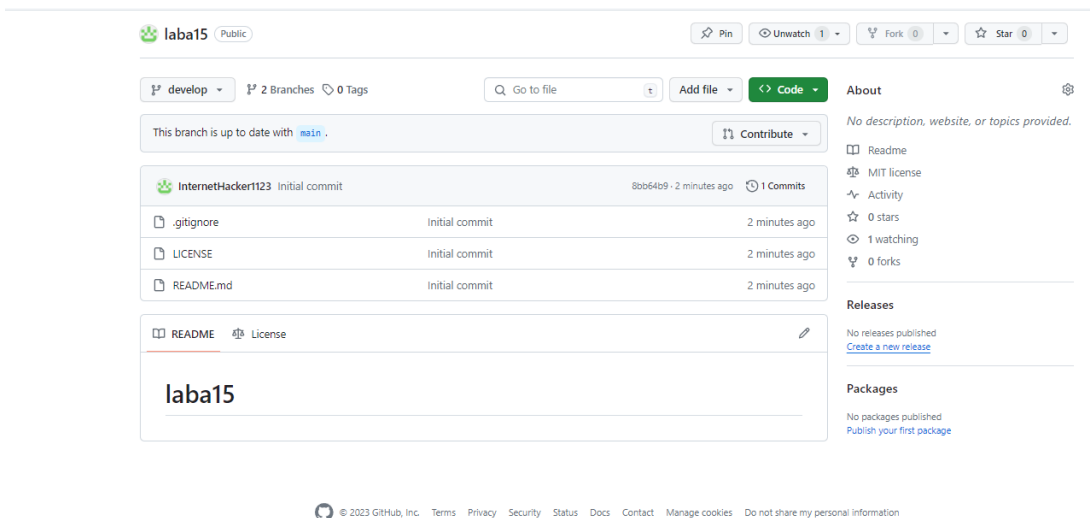


Рисунок 5.2 – Ветка develop на GitHub

6. Создал проект PyCharm в папке репозитория

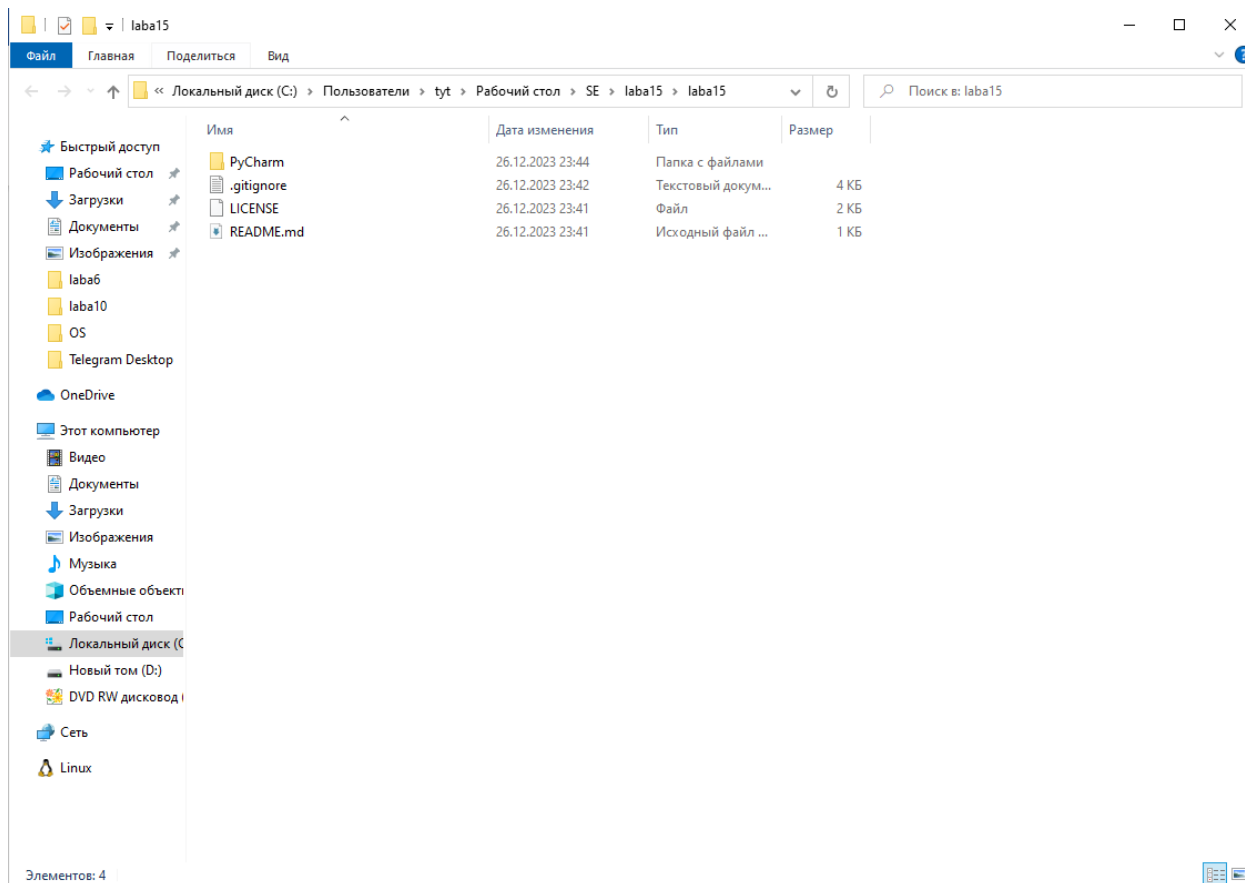
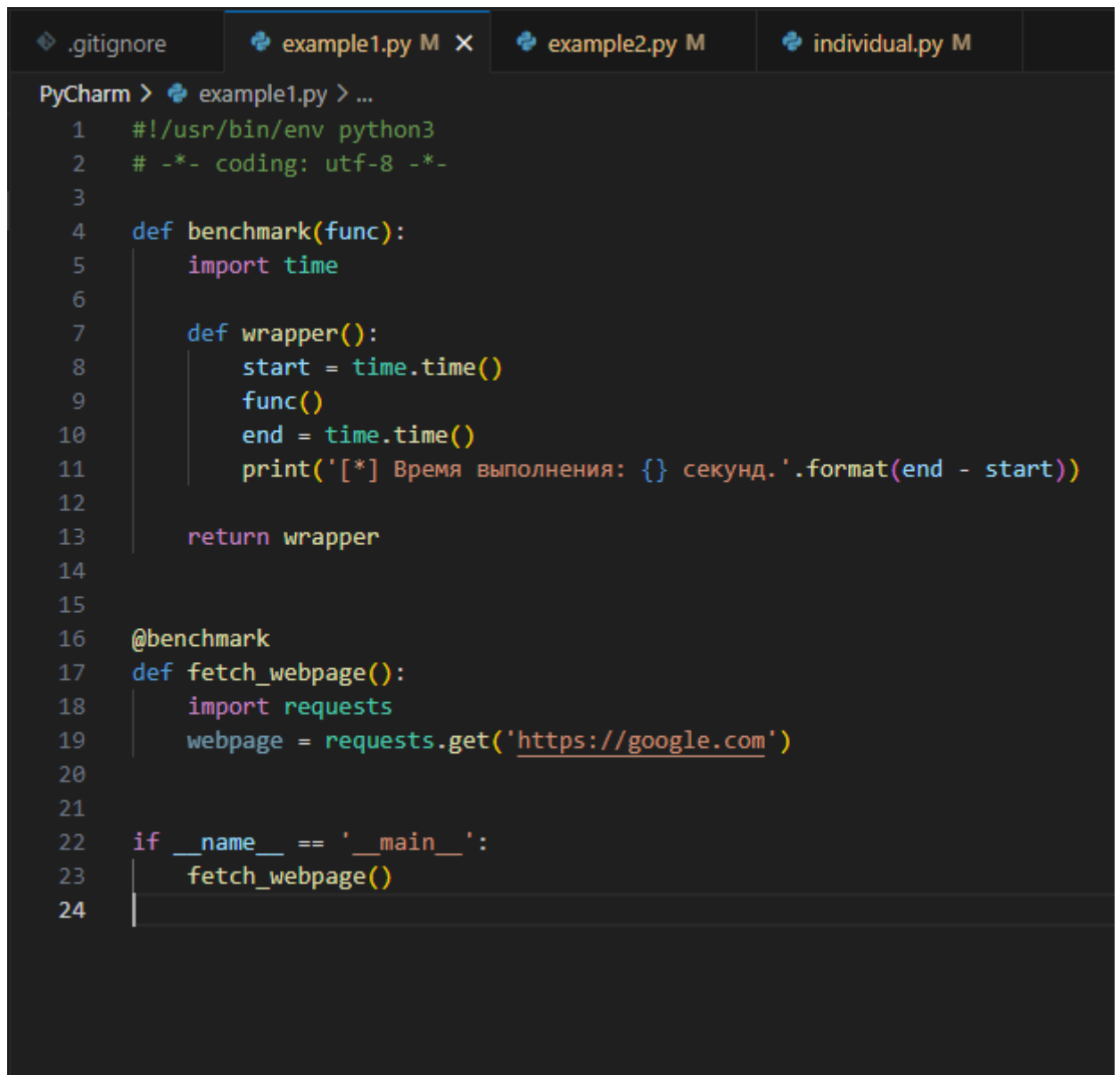


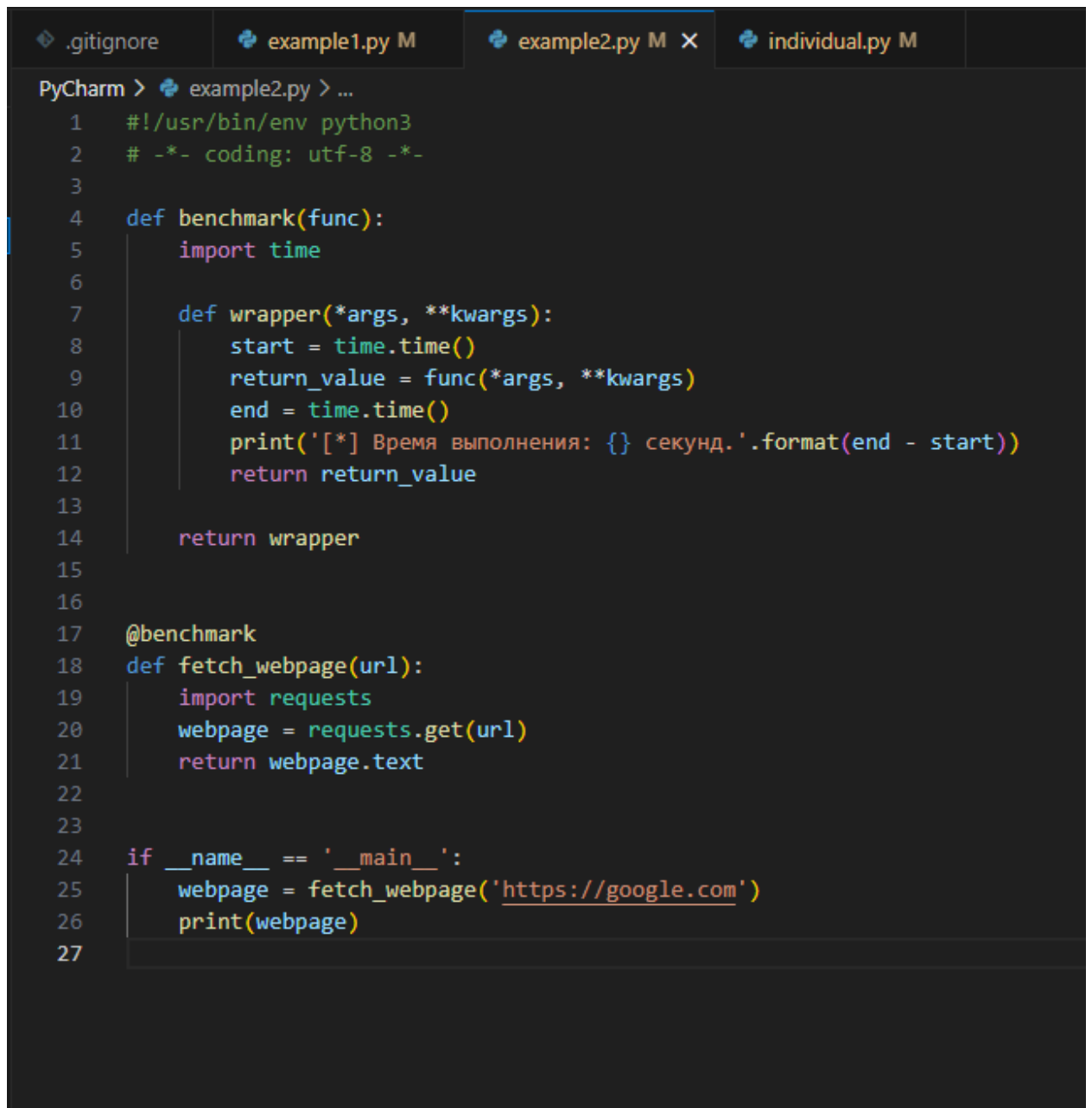
Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработал примеры лабораторной работы.



```
.gitignore example1.py M x example2.py M individual.py M
PyCharm > example1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def benchmark(func):
5      import time
6
7      def wrapper():
8          start = time.time()
9          func()
10         end = time.time()
11         print('[*] Время выполнения: {} секунд.'.format(end - start))
12
13     return wrapper
14
15
16 @benchmark
17 def fetch_webpage():
18     import requests
19     webpage = requests.get('https://google.com')
20
21
22 if __name__ == '__main__':
23     fetch_webpage()
24
```

Рисунок 7.1 – Код программы example1.py



```
PyCharm > example2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def benchmark(func):
5      import time
6
7      def wrapper(*args, **kwargs):
8          start = time.time()
9          return_value = func(*args, **kwargs)
10         end = time.time()
11         print('[*] Время выполнения: {} секунд.'.format(end - start))
12         return return_value
13
14     return wrapper
15
16
17 @benchmark
18 def fetch_webpage(url):
19     import requests
20     webpage = requests.get(url)
21     return webpage.text
22
23
24 if __name__ == '__main__':
25     webpage = fetch_webpage('https://google.com')
26     print(webpage)
27
```

Рисунок 7.2 – Код программы example2.py

8. Выполнил индивидуальное задание.


```
.gitignore x example1.py M example2.py M individual.py x
PyCharm > individual.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6
7  # Объявляем функцию для вычисления площади круга
8  def calculate_circle_area(radius):
9      return math.pi * radius**2
10
11
12 # Объявляем декоратор для вывода сообщения
13 def print_circle_area_message(func):
14     def wrapper(radius):
15         result = func(radius)
16         print(f"Площадь круга равна = {result}")
17     return wrapper
18
19
20 # Применяем декоратор к функции
21 decorated_calculate_circle_area = print_circle_area_message(
22     calculate_circle_area)
23
24 if __name__ == '__main__':
25     decorated_calculate_circle_area(5)
26
```

Рисунок 8.1 – Код программы individual.py

9. Зафиксировал изменения в репозитории.

```
C:\Users\tyt\Desktop\SE\laba15\laba15>git add .
C:\Users\tyt\Desktop\SE\laba15\laba15>git commit -m"add tasks"
[develop 08c5571] add tasks
3 files changed, 74 insertions(+)
C:\Users\tyt\Desktop\SE\laba15\laba15>git push
Enumerating objects: 11, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 6 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.11 KiB | 1.11 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/InternetHacker1123/laba15.git
c78f405..08c5571 develop -> develop
C:\Users\tyt\Desktop\SE\laba15\laba15>
```

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.