

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 №10
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

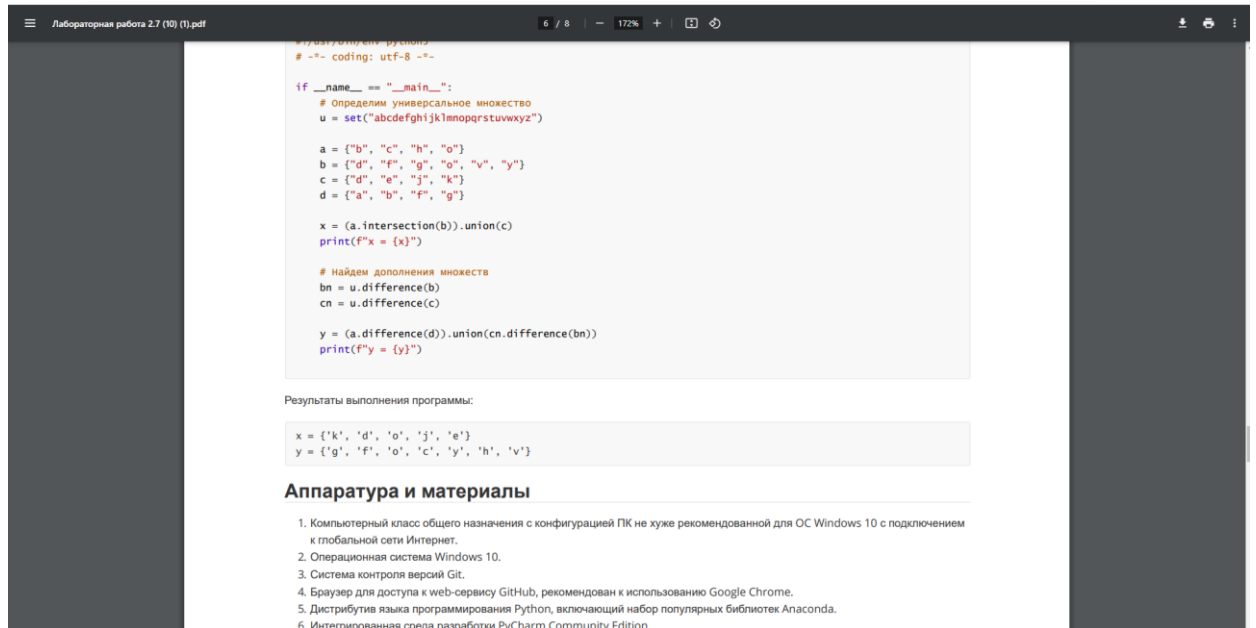


Рисунок 1.1 – Изучение материала для лабораторной работы


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 *

/ lab2(7)_10

✔ Your new repository will be created as lab2-7-_10.

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

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

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

.gitignore template: Python ▾

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

Choose a license

License: MIT 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](#).

Рисунок 2.1 – Настройка репозитория

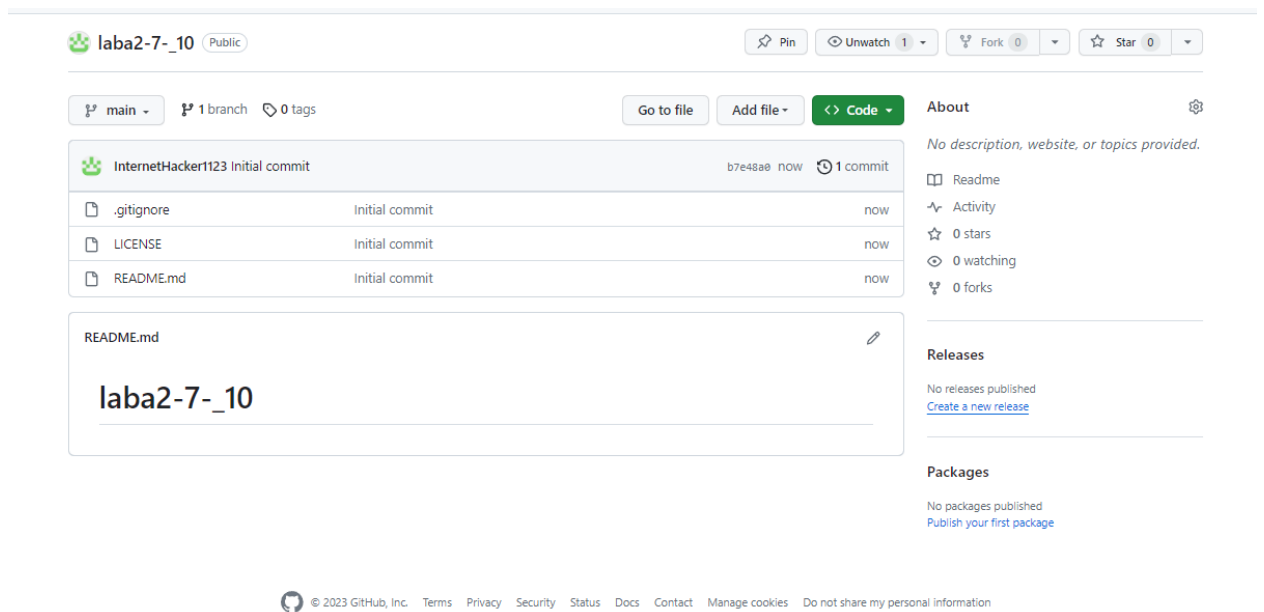


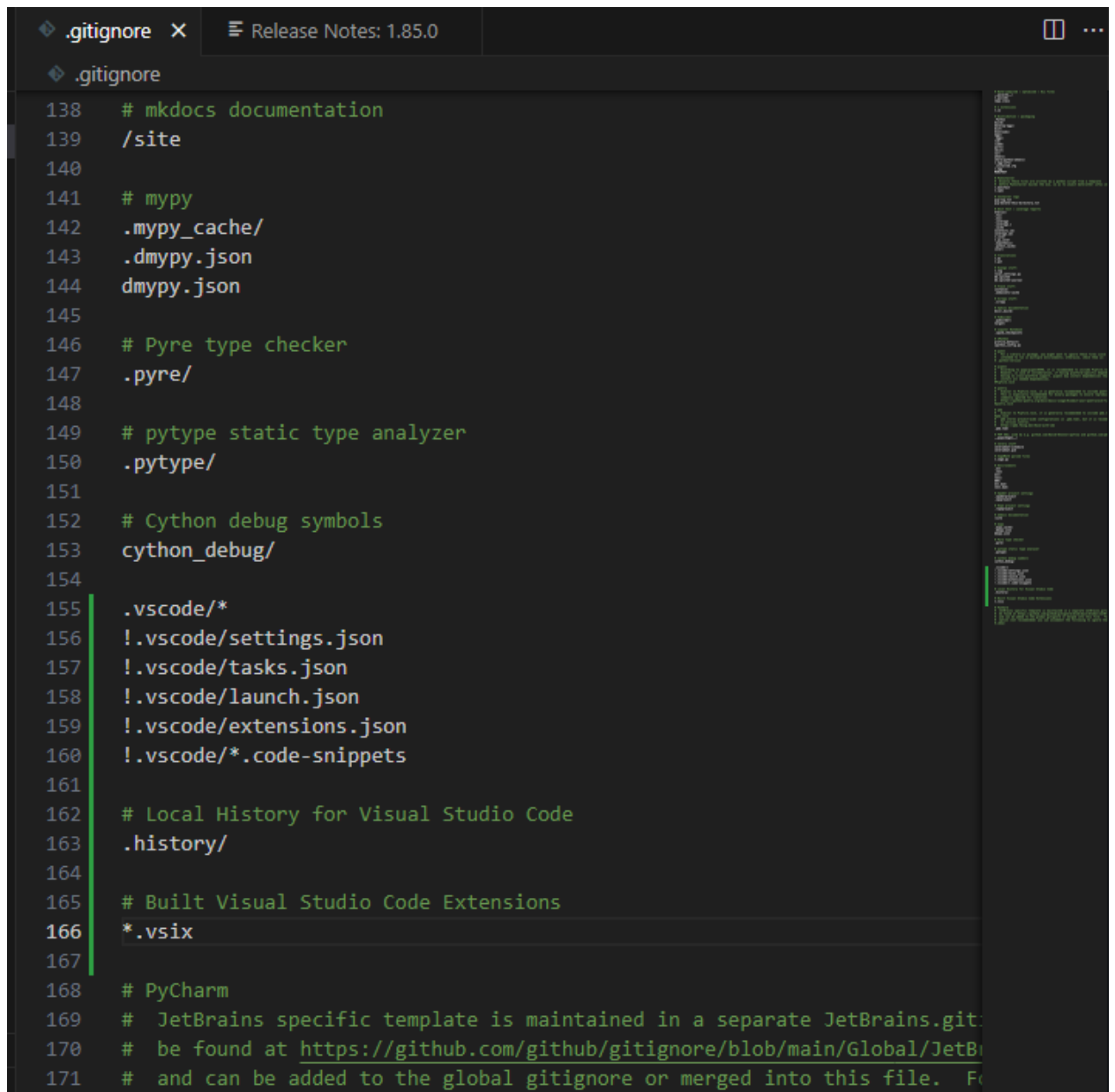
Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

```
C:\Users\tyt\Desktop\SE\laba10>git clone https://github.com/InternetHacker1123/laba2-7-_10.git
Cloning into 'laba2-7-_10'...
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.
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

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



```
138 # mkdocs documentation
139 /site
140
141 # mypy
142 .mypy_cache/
143 .dmypy.json
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 *.*six
167
168 # PyCharm
169 # JetBrains specific template is maintained in a separate JetBrains.git
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
```

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\laba10\laba2-7-_10>git branch develop
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>git checkout develop
Switched to branch 'develop'

C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>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/laba2-7-_10/pull/new/develop
remote:
To https://github.com/InternetHacker1123/laba2-7-_10.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>
```

Рисунок 5.1 – Создание ветки develop от ветки main

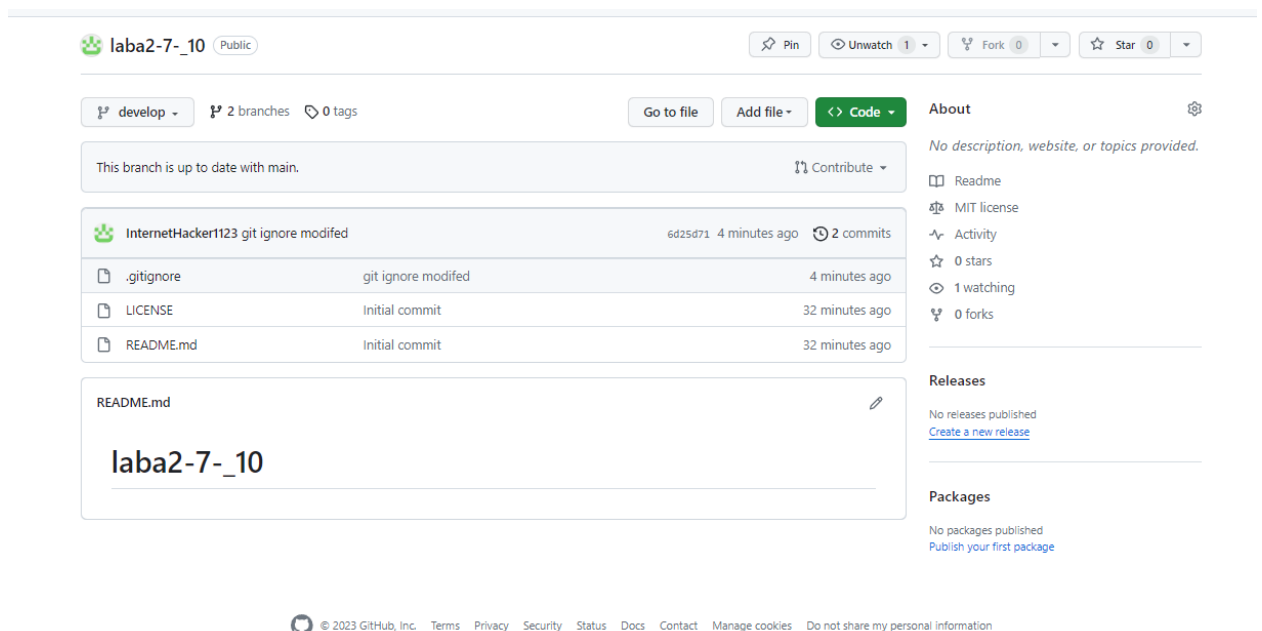


Рисунок 5.2 – Ветка develop на GitHub

6. Создал проект PyCharm в папке репозитория

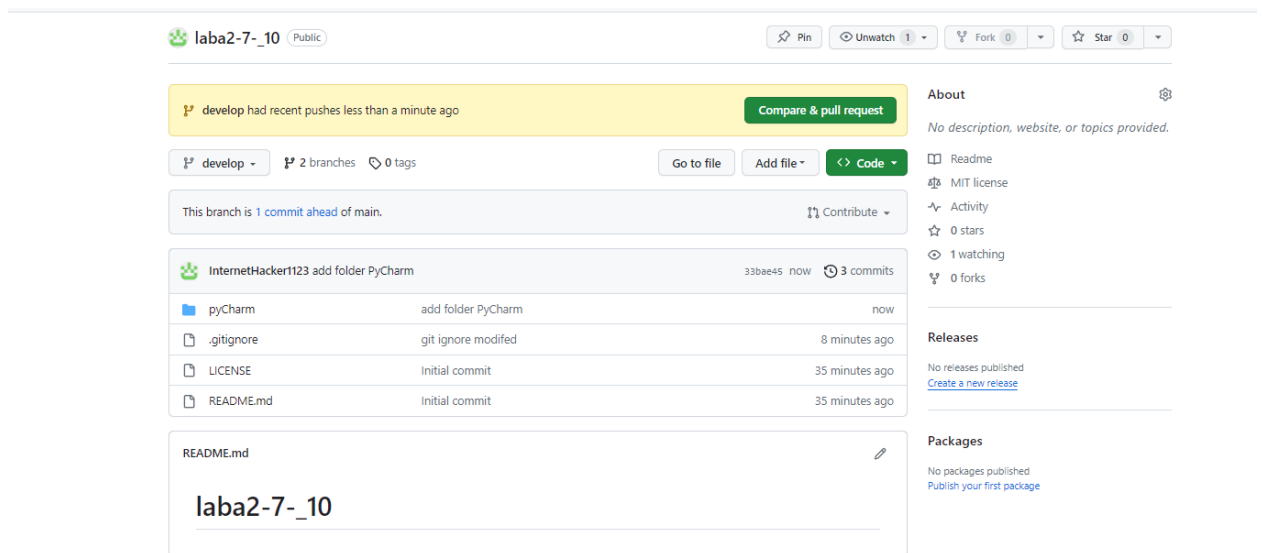
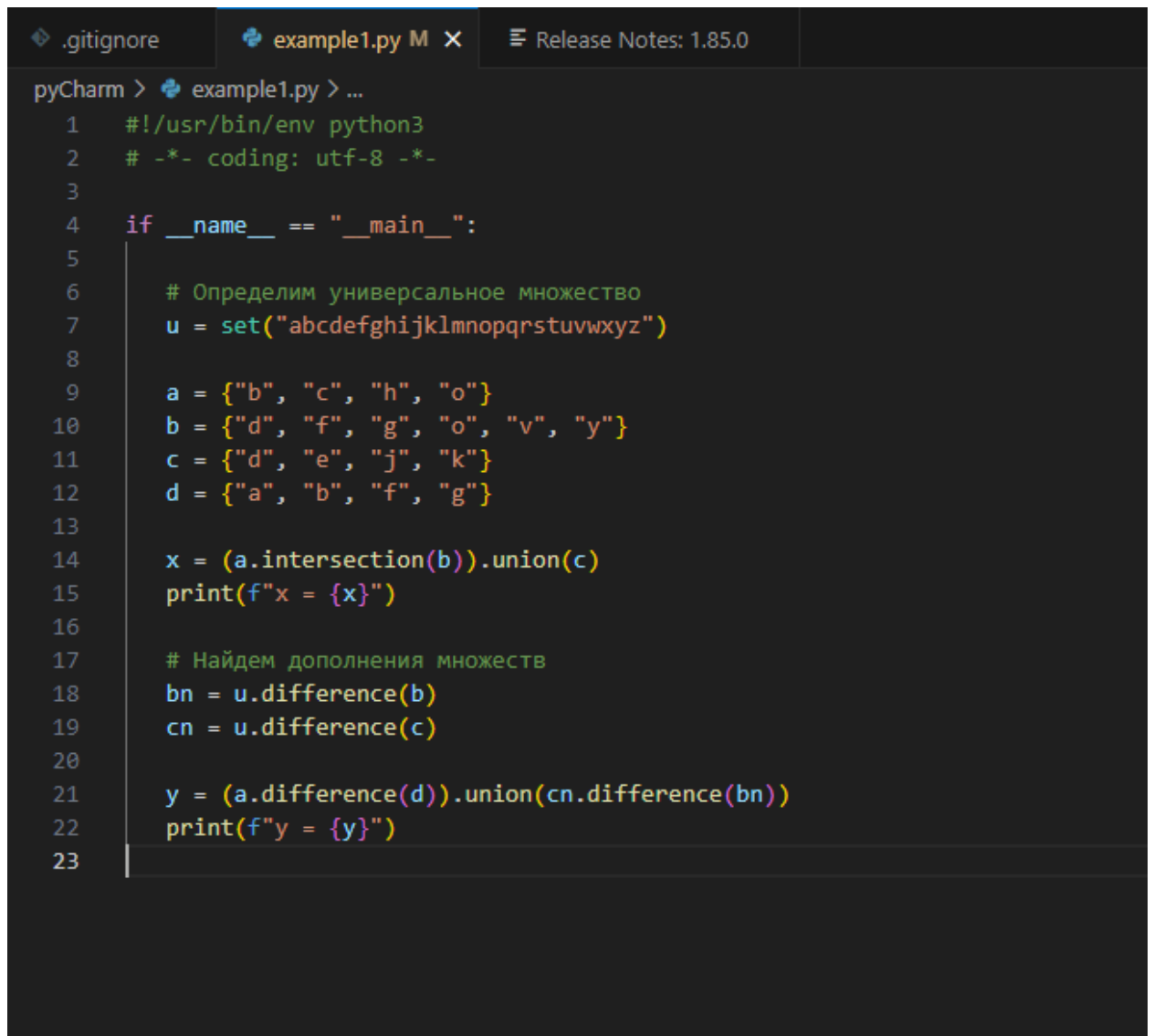


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработал примеры лабораторной работы. Создал для каждого примера отдельный модуль языка Python. Зафиксировал изменения в репозитории.



```
pyCharm > example1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5
6      # Определим универсальное множество
7      u = set("abcdefghijklmnopqrstuvwxyz")
8
9      a = {"b", "c", "h", "o"}
10     b = {"d", "f", "g", "o", "v", "y"}
11     c = {"d", "e", "j", "k"}
12     d = {"a", "b", "f", "g"}
13
14     x = (a.intersection(b)).union(c)
15     print(f"x = {x}")
16
17     # Найдем дополнения множеств
18     bn = u.difference(b)
19     cn = u.difference(c)
20
21     y = (a.difference(d)).union(cn.difference(bn))
22     print(f"y = {y}")
23
```

Рисунок 7.1 – Проработка примера 1

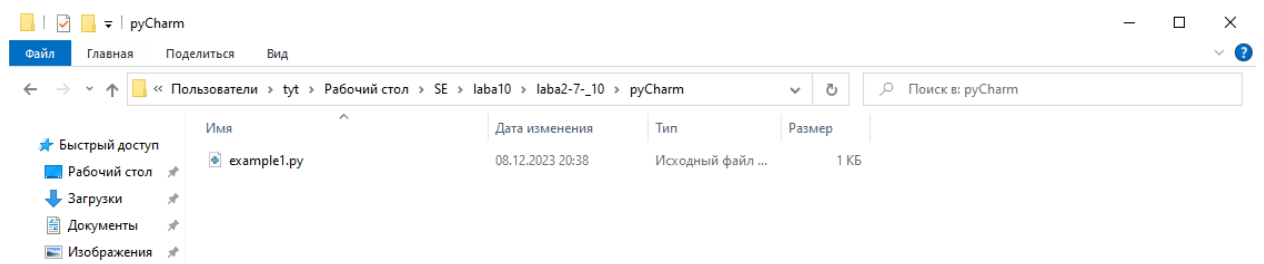
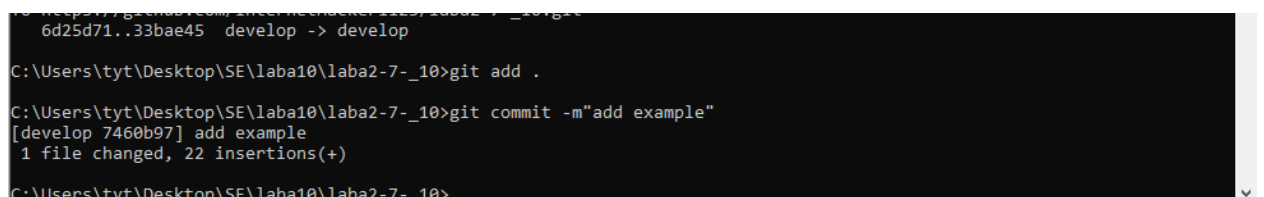


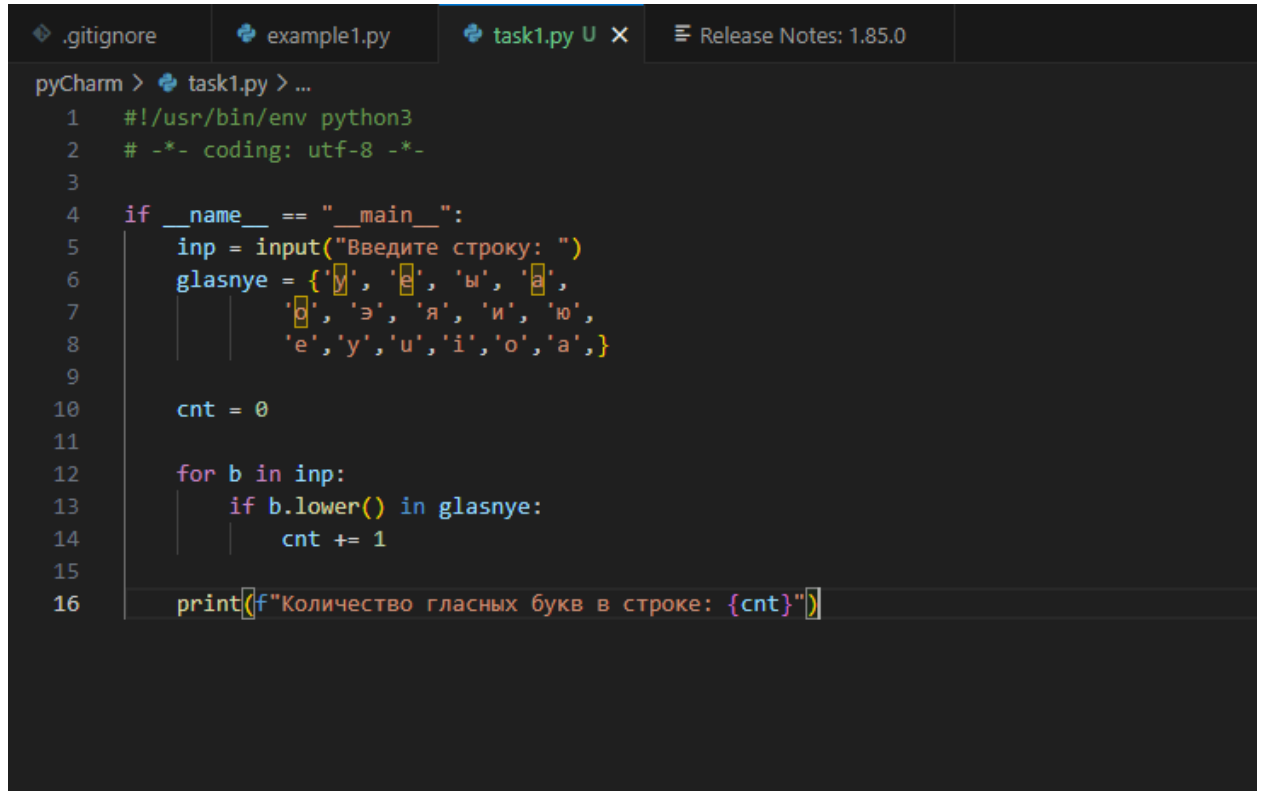
Рисунок 7.4 – Создание отдельного модуля для примера



```
6d25d71..33bae45 develop -> develop
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>git add .
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>git commit -m"add example"
[develop 7460b97] add example
1 file changed, 22 insertions(+)
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>
```


Рисунок 7.5 – Фиксирование изменений в репозитории

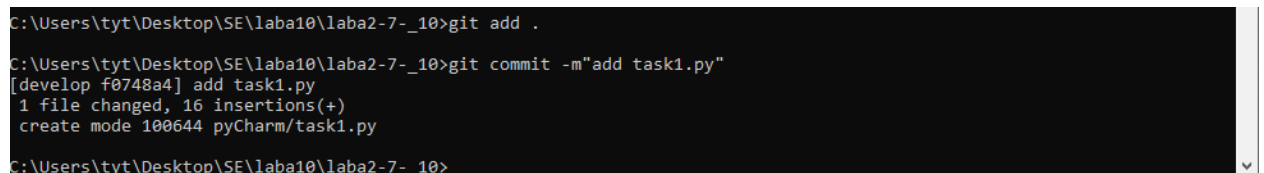
8. Решил задачу: подсчитал количество гласных в строке, введенной с клавиатуры с использованием множеств.



```
pyCharm > task1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      inp = input("Введите строку: ")
6      glasnye = {'а', 'е', 'ы', 'я',
7                 'о', 'э', 'я', 'и', 'ю',
8                 'е', 'у', 'и', 'о', 'а',}
9
10     cnt = 0
11
12     for b in inp:
13         if b.lower() in glasnye:
14             cnt += 1
15
16     print(f"Количество гласных букв в строке: {cnt}")
```

Рисунок 8.1 – Код программы task2.py

9. Зафиксировал сделанные изменения в репозитории



```
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>git add .
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>git commit -m"add task1.py"
[develop f0748a4] add task1.py
1 file changed, 16 insertions(+)
create mode 100644 pyCharm/task1.py
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>
```

Рисунок 9.1 – Коммит файлов в репозитории git

10. Решил задачу: определил общие символы в двух строках, введенных с клавиатуры.

```
.gitignore x example1.py task1.py task2.py U Release Notes: 1.85.0
pyCharm > task2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      stroka_1 = set(input("Введите первую строку: "))
6      stroka_2 = set(input("Введите вторую строку: "))
7      obshie_simvoly = stroka_1.intersection(stroka_2)
8      print(f"Общие символы: {obshie_simvoly}")
```

Рисунок 10.1 – Код программы task2.py

11. Зафиксировал сделанные изменения в репозитории.

```
C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\tyt\Desktop\SE\laba10\laba2-7-_10>git merge develop
Updating 6d25d71..cdf74bc
Fast-forward
 pyCharm/example1.py | 22 +++++
 pyCharm/individual.py | 19 +++++
 pyCharm/task1.py | 16 +++++
 pyCharm/task2.py | 8 +++++
4 files changed, 65 insertions(+)
create mode 100644 pyCharm/example1.py
create mode 100644 pyCharm/individual.py
create mode 100644 pyCharm/task1.py
create mode 100644 pyCharm/task2.py
```

Рисунок 11.1 – Коммит файлов в репозитории git

Контрольные вопросы

1. Что такое множества в языке Python?

Множеством в языке программирования Python называется неупорядоченная совокупность уникальных значений. В качестве элементов этого набора данных могут выступать любые неизменяемые объекты, такие как числа, символы, строки. В отличие от массивов и списков, порядок следования значений не учитывается при обработке его содержимого. Над одним, а также несколькими множествами можно выполнять ряд операций,

благодаря функциям стандартной библиотеки языка программирования Python.

2. Как осуществляется создание множеств в Python?

Сделать это можно, просто присвоив переменной последовательность значений, выделив их фигурными скобками, либо используя функцию `set()`.

3. Как проверить присутствие/отсутствие элемента в множестве?

Проверка, есть ли данное значение в множестве. Для этого используется `in`. Наоборот, проверка отсутствия. Используется `not in`.

4. Как выполнить перебор элементов множества?

```
for a in {0, 1, 2}:  
    print(a)
```

5. Что такое `set comprehension`?

Для создания множества можно в Python воспользоваться генератором, позволяющих заполнять списки, а также другие наборы данных с учетом неких условий.

6. Как выполнить добавление элемента во множество?

```
a = {0, 1, 2, 3}  
a.add(4)  
print(a)
```

```
{0, 1, 2, 3, 4}
```

7. Как выполнить удаление одного или всех элементов множества?

Для удаления элементов из множества используются следующие функции в Python (кроме очистки, которая будет рассмотрена ниже):

- `remove` — удаление элемента с генерацией исключения в случае, если такого элемента нет;
- `discard` — удаление элемента без генерации исключения, если элемент отсутствует;
- `pop` — удаление первого элемента, генерируется исключение при попытке удаления из пустого множества.

Иногда необходимо полностью убрать все элементы. Чтобы не удалять каждый элемент отдельно, используется метод `clear`, не принимающий аргументов.

8. Как выполняются основные операции над множествами: объединение, пересечение, разность?

Чтобы объединить все элементы двух разных множеств, стоит воспользоваться методом `union` на одном из объектов. Чтобы добавить все элементы из одного множества к другому, необходимо вызывать метод `update` на первом объекте. Чтобы найти общие элементы для двух разных множеств, следует применить функцию `intersection`, принимающую в качестве аргумента один из наборов данных. Чтобы вычислить разность для двух разных множеств, необходимо воспользоваться методом `difference`.

9. Как определить, что некоторое множество является надмножеством или подмножеством другого множества?

Чтобы выяснить, является ли множество `a` подмножеством `b`, стоит попробовать вывести на экран результат выполнения метода `issubset`, как в следующем примере.

10. Каково назначение множеств `frozenset`?

Множество, содержимое которого не поддается изменению, имеет тип `frozenset`. Значения из этого набора нельзя удалить, как и добавить новые.

11. Как осуществляется преобразование множеств в строку, список, словарь?

Для преобразования множества в строку используется конкатенация текстовых значений, которую обеспечивает функция `join`. Чтобы получить из множества словарь, следует передать функции `dict` набор из нескольких пар значений, в каждом из которых будет находиться ключ. По аналогии с предыдущими преобразованиями можно получить список неких объектов. На этот раз используется вызов `list`, получающий в качестве аргумента множество `a`.