

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
Отчет по лабораторной работе № 2.7
«Работа с множествами в языке Python»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » декабря 2022 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2022

Цель работы:

Приобретение навыков по работе с множествами при написании программ с помощью языка программирования Python версии 3.x.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.7>

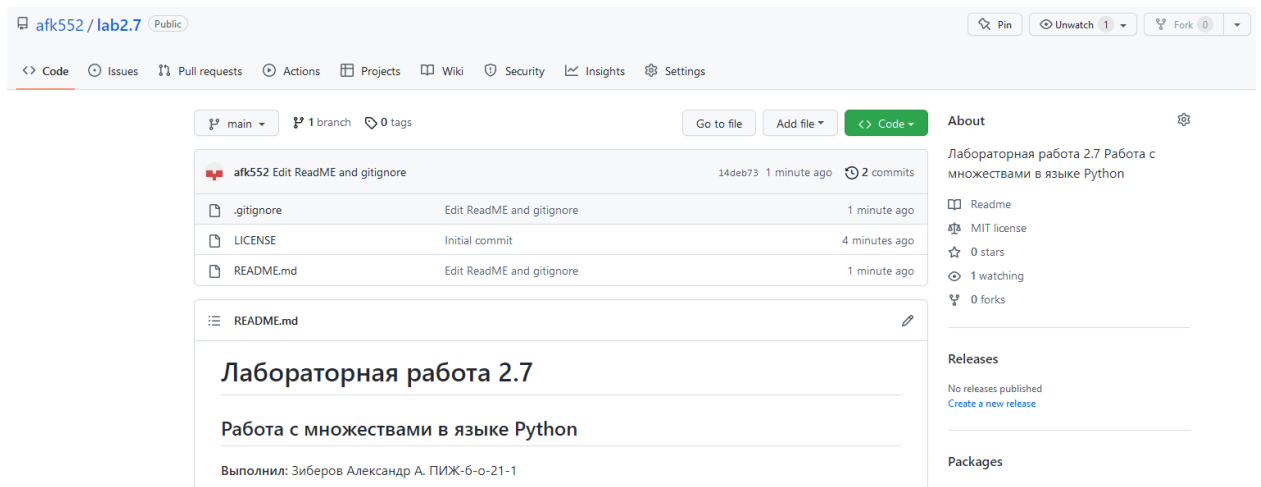


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

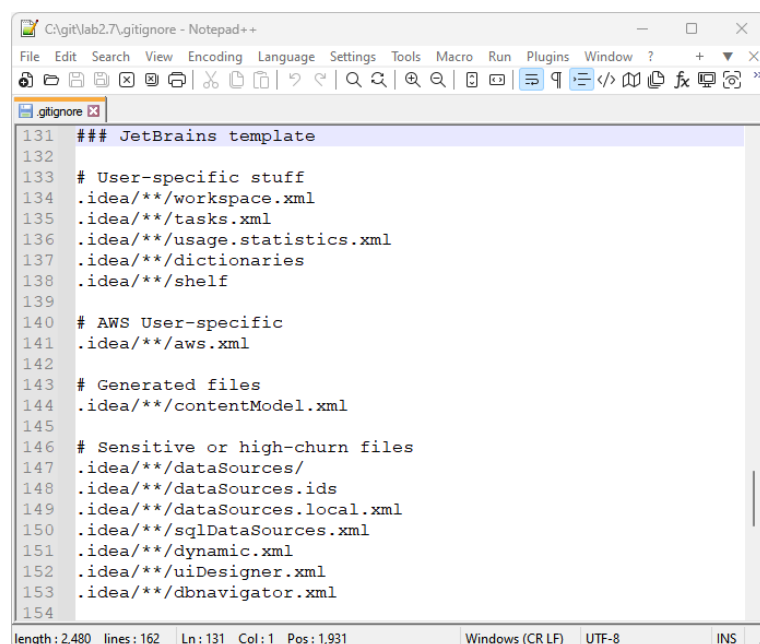


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\Windows\System32\cmd.exe

C:\git\lab2.7>git branch
* main

C:\git\lab2.7>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.7>git branch
* develop
  main

C:\git\lab2.7>git push 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/afk552/lab2.7/pull/new/develop
remote:
To https://github.com/afk552/lab2.7
 * [new branch]      develop -> develop

C:\git\lab2.7>
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

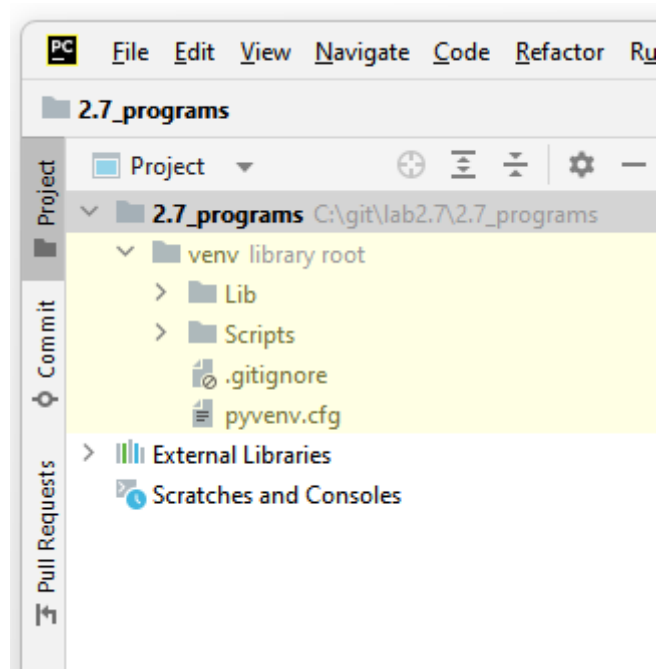
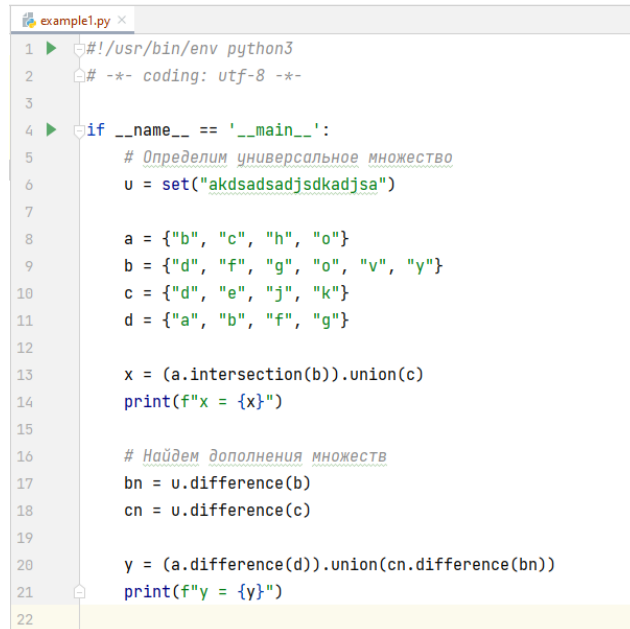


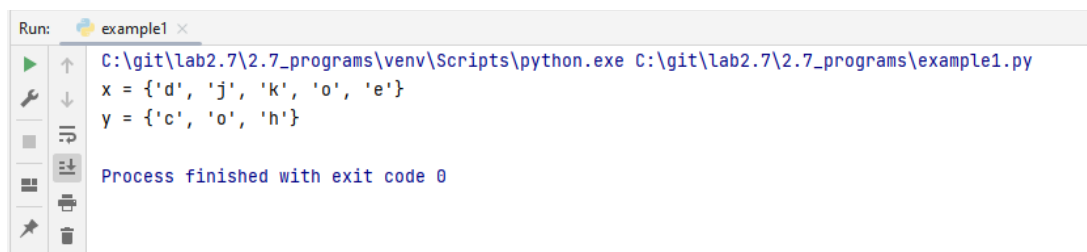
Рисунок 4 – Окно проекта в PyCharm

Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры. Рисунки 5-6.



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

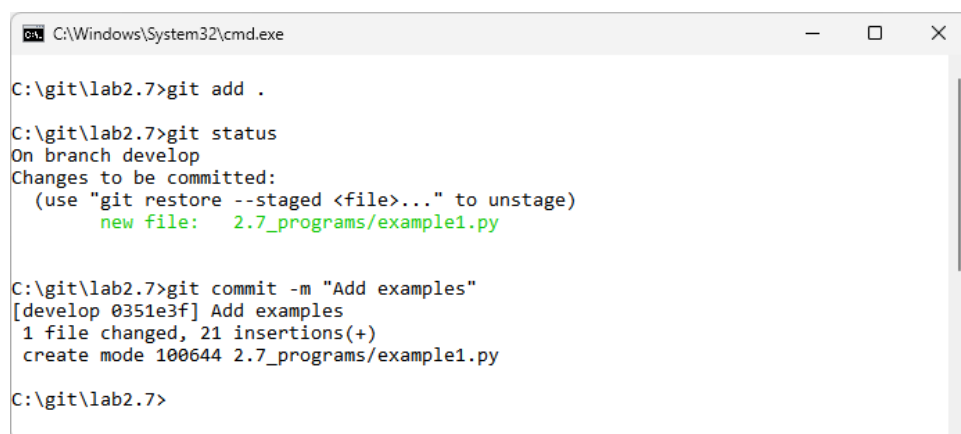
Рисунок 5 – Код программы из примера 1



```
Run: example1 x
C:\git\lab2.7\2.7_programs\venv\Scripts\python.exe C:\git\lab2.7\2.7_programs\example1.py
x = {'d', 'j', 'k', 'o', 'e'}
y = {'c', 'o', 'h'}

Process finished with exit code 0
```

Рисунок 6 – Результат выполнения программы из примера 1



```
C:\Windows\System32\cmd.exe

C:\git\lab2.7>git add .

C:\git\lab2.7>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.7_programs/example1.py

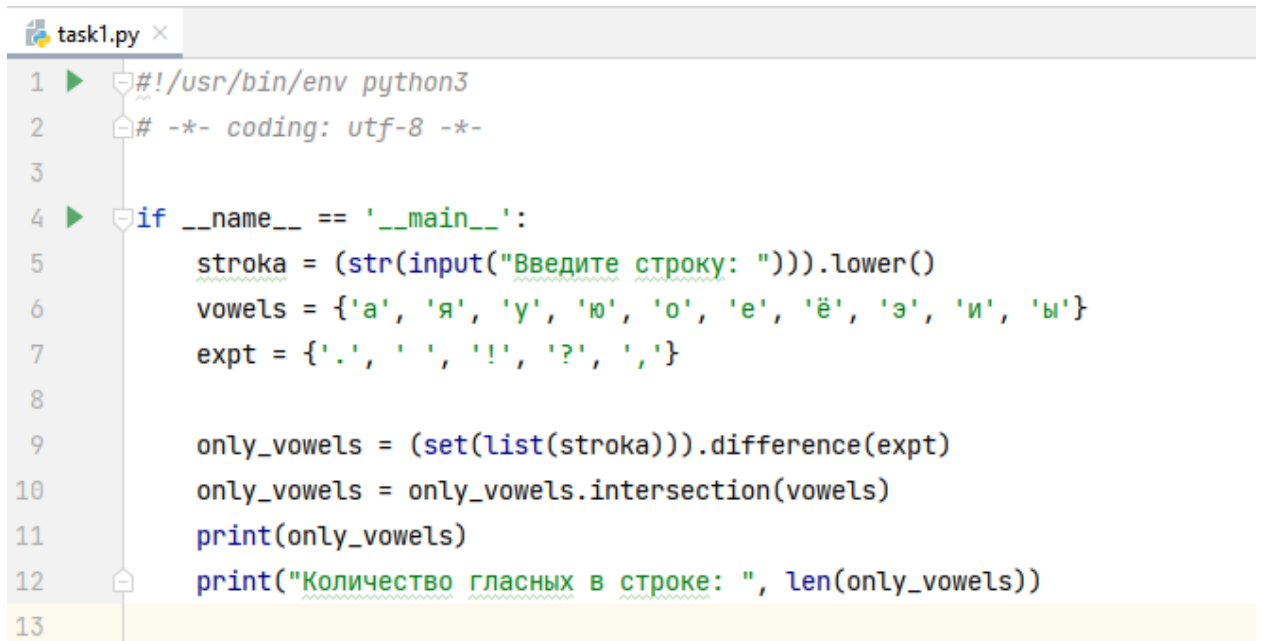
C:\git\lab2.7>git commit -m "Add examples"
[develop 0351e3f] Add examples
 1 file changed, 21 insertions(+)
 create mode 100644 2.7_programs/example1.py

C:\git\lab2.7>
```

Рисунок 7 – Окно командной строки

Задание 1 (9). Подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.

Рисунки 8, 9.



```
1  > #!/usr/bin/env python3
2  > # -*- coding: utf-8 -*-
3
4  > if __name__ == '__main__':
5      строка = (str(input("Введите строку: "))).lower()
6      vowels = {'a', 'я', 'y', 'ю', 'o', 'e', 'ё', 'э', 'и', 'ы'}
7      expt = {'.', ' ', '!', '?', ',', ''}
8
9      only_vowels = (set(list(строка))).difference(expt)
10     only_vowels = only_vowels.intersection(vowels)
11     print(only_vowels)
12     print("Количество гласных в строке: ", len(only_vowels))
13
```

Рисунок 8 – Код программы задания 1



Run: task1 x

```
C:\git\lab2.7\2.7_programs\venv\Scripts\python.exe C:\git\lab2.7\2.7_programs\task1.py
Введите строку: Тестовая строка для проверки.
а,о,и,я,е
Количество гласных в строке: 5
Process finished with exit code 0
```

Run: task1 x

```
C:\git\lab2.7\2.7_programs\venv\Scripts\python.exe C:\git\lab2.7\2.7_programs\task1.py
Введите строку: Стрк
В строке нет гласных букв!
Process finished with exit code 0
```

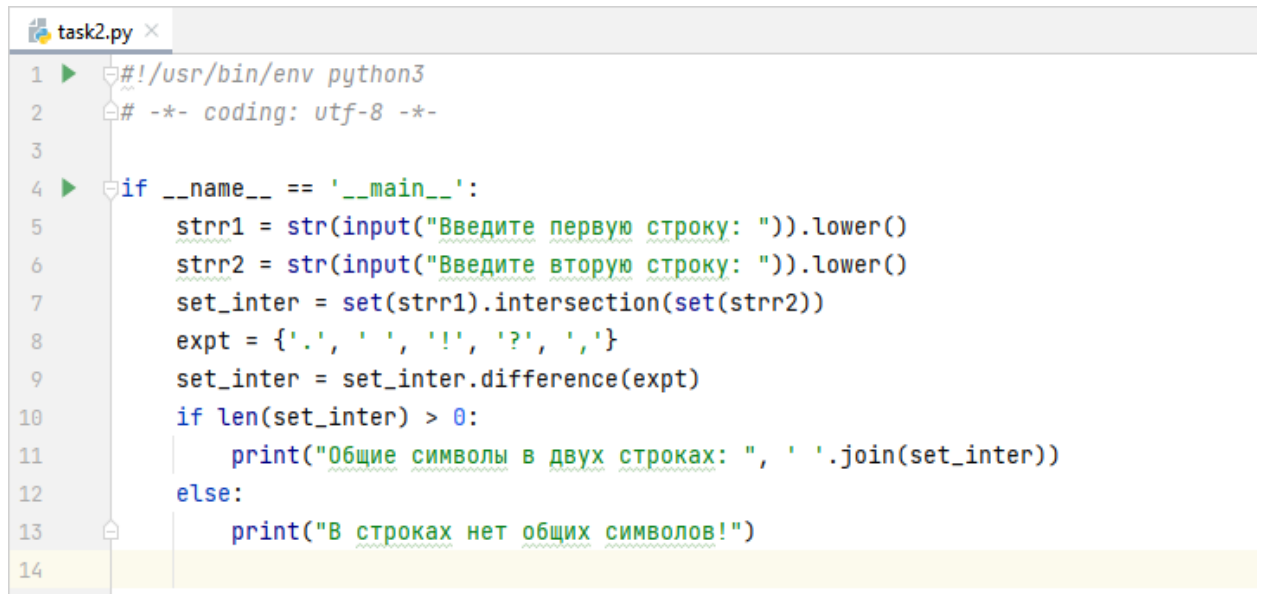
Run: task1 x

```
C:\git\lab2.7\2.7_programs\venv\Scripts\python.exe C:\git\lab2.7\2.7_programs\task1.py
Введите строку:
Введена пустая строка!
Process finished with exit code 1
```

Рисунок 9 – Результат выполнения программы задания 1

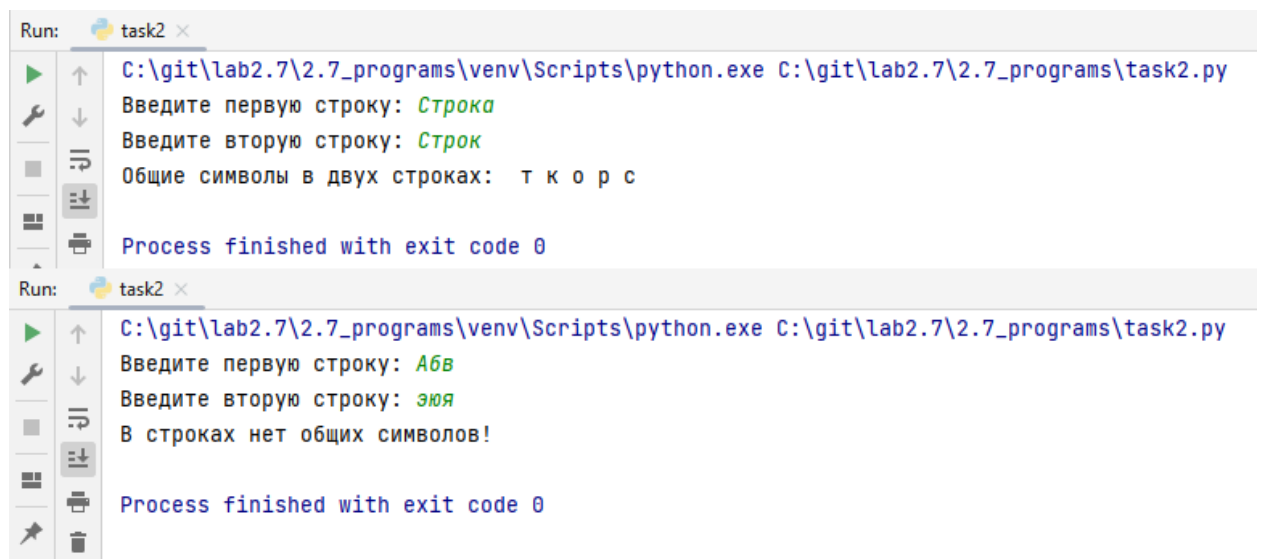
Задание 2 (10). Определите общие символы в двух строках, введенных с клавиатуры.

Рисунки 10, 11.

A screenshot of a code editor showing a Python script named task2.py. The script uses set operations to find common characters between two input strings, excluding punctuation. The code is as follows:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     strr1 = str(input("Введите первую строку: ")).lower()
6     strr2 = str(input("Введите вторую строку: ")).lower()
7     set_inter = set(strr1).intersection(set(strr2))
8     expt = {'.', ' ', '!', '?', ',', ''}
9     set_inter = set_inter.difference(expt)
10    if len(set_inter) > 0:
11        print("Общие символы в двух строках: ", ' '.join(set_inter))
12    else:
13        print("В строках нет общих символов!")
14
```

Рисунок 10 – Код программы задания 2

Two screenshots of a terminal window showing the execution of the program. The first screenshot shows the input strings "Строка" and "Строк", resulting in the output "Общие символы в двух строках: т к о р с". The second screenshot shows the input strings "Абв" and "эюя", resulting in the output "В строках нет общих символов!". Both screenshots show the command prompt path and the exit code 0.

```
Run: task2
C:\git\lab2.7\2.7_programs\venv\Scripts\python.exe C:\git\lab2.7\2.7_programs\task2.py
Введите первую строку: Строка
Введите вторую строку: Строк
Общие символы в двух строках: т к о р с
Process finished with exit code 0

Run: task2
C:\git\lab2.7\2.7_programs\venv\Scripts\python.exe C:\git\lab2.7\2.7_programs\task2.py
Введите первую строку: Абв
Введите вторую строку: эюя
В строках нет общих символов!
Process finished with exit code 0
```

Рисунок 11 – Результат выполнения программы задания 1

Выполните индивидуальные задания, согласно своего варианта.

Вариант 12

Задание 1. Определить результат выполнения операций над множествами. Считать элементы множества строками. Проверить результаты вручную. Номер варианта задания необходимо получить у преподавателя.

Рисунки 12-14.

```
indiv_12.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      # Универсальное множество
6      univ = set("abcdef")
7      A = {'b', 'k', 'n', 'o', 'q'}
8      B = {'a', 'b', 'k', 'u'}
9      C = {'o', 'p'}
10     D = {'a', 'm', 'n', 'y', 'z'}
11
12     X = (A.union(B)).intersection(D)
13     # Взяли дополнение для множества A
14     an = univ.difference(A)
15     an_int_d = an.intersection(D)
16     c_diff_b = C.difference(B)
17
18     Y = an_int_d.union(c_diff_b)
19     print("A", A, "\nB", B, "\nC", C, "\nD", D, "\nUniv.", univ, "\n")
20     print("X = (A U B) ∩ D => ", X)
21     print("Y = (¬A ∩ D) ∪ (C / B) => ", Y)
22
```

Рисунок 12 – Код программы индивидуального задания 1

```
Run: indiv_12 x
C:\git\lab2.7\2.7_programs\venv\Scripts\python.exe C:\git\lab2.7\2.7_programs\indiv_12.py
A {'b', 'k', 'o', 'q', 'n'}
B {'b', 'a', 'k', 'u'}
C {'o', 'p'}
D {'n', 'y', 'a', 'z', 'm'}
Univ. {'c', 'a', 'e', 'b', 'd', 'f'}

X = (A U B) ∩ D => {'a', 'n'}
Y = (¬A ∩ D) ∪ (C / B) => {'o', 'p', 'a'}

Process finished with exit code 0
```

Рисунок 13 – Результаты выполнения программы индивидуального задания 1

$A = \{ 'b', 'k', 'o', 'q', 'n' \}$
 $B = \{ 'b', 'a', 'k', 'u' \}$
 $C = \{ 'o', 'p' \}$
 $D = \{ 'n', 'y', 'a', 'z', 'm' \}$
 $Univ. = \{ 'c', 'a', 'e', 'b', 'd', 'f' \}$

$X = (A \cup B) \cap D \Rightarrow \{ 'a', 'n' \}$
 $Y = (\neg A \cap D) \cup (C / B) \Rightarrow \{ 'o', 'p', 'a' \}$

$I \quad X = (A \overset{1}{\cup} B) \overset{2}{\cap} D$
 $A = \{ 'o', 'b', 'n', 'q', 'k' \} \quad B = \{ 'b', 'u', 'a', 'k' \}$
 $A \cup B = 'o' 'b' 'n' 'q' 'k' 'u' \quad (1) \cap D = 'n' 'a' \quad \checkmark$

$II \quad Y = (\neg A \overset{1}{\cap} D) \overset{2}{\cup} (C / B)$
 $\neg A \cap D = 'f' 'a' 'd' 'c' 'e' \quad (1) \quad \neg A \cap D = 'a' \quad (2)$
 $C / B = 'o' 'p'$
 $(1) \cup (2) = 'a' 'o' 'b' \quad \checkmark$

Рисунок 14 – Проверка результатов программы индивидуального задания 1

```

C:\Windows\System32\cmd.exe

C:\git\lab2.7>git add .

C:\git\lab2.7>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   2.7_programs/indiv_12.py

C:\git\lab2.7>git commit -m "Add individual task"
[develop fa7f9e4] Add individual task
 1 file changed, 21 insertions(+)
 create mode 100644 2.7_programs/indiv_12.py

C:\git\lab2.7>

```

Рисунок 15 – Окно командной строки

Выполните слияние ветки для разработки с веткой main / master. Отправьте сделанные изменения на сервер GitHub. Рисунки 16, 17.

```

C:\git\lab2.7>git branch
* develop
  main

C:\git\lab2.7>git push origin develop
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 1.25 KiB | 640.00 KiB/s, done.
Total 13 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), completed with 3 local objects.
To https://github.com/afk552/lab2.7
  446e55a..3d5f1a9 develop -> develop

C:\git\lab2.7>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git\lab2.7>git merge develop
Updating 14deb73..3d5f1a9
Fast-forward
 2.7_programs/example1.py | 21 ++++++
 2.7_programs/indiv_12.py | 21 ++++++
 2.7_programs/task1.py    | 17 ++++++
 2.7_programs/task2.py    | 13 ++++++
 4 files changed, 72 insertions(+)
 create mode 100644 2.7_programs/example1.py
 create mode 100644 2.7_programs/indiv_12.py
 create mode 100644 2.7_programs/task1.py
 create mode 100644 2.7_programs/task2.py

C:\git\lab2.7>

```

Рисунок 16 – Окно командной строки

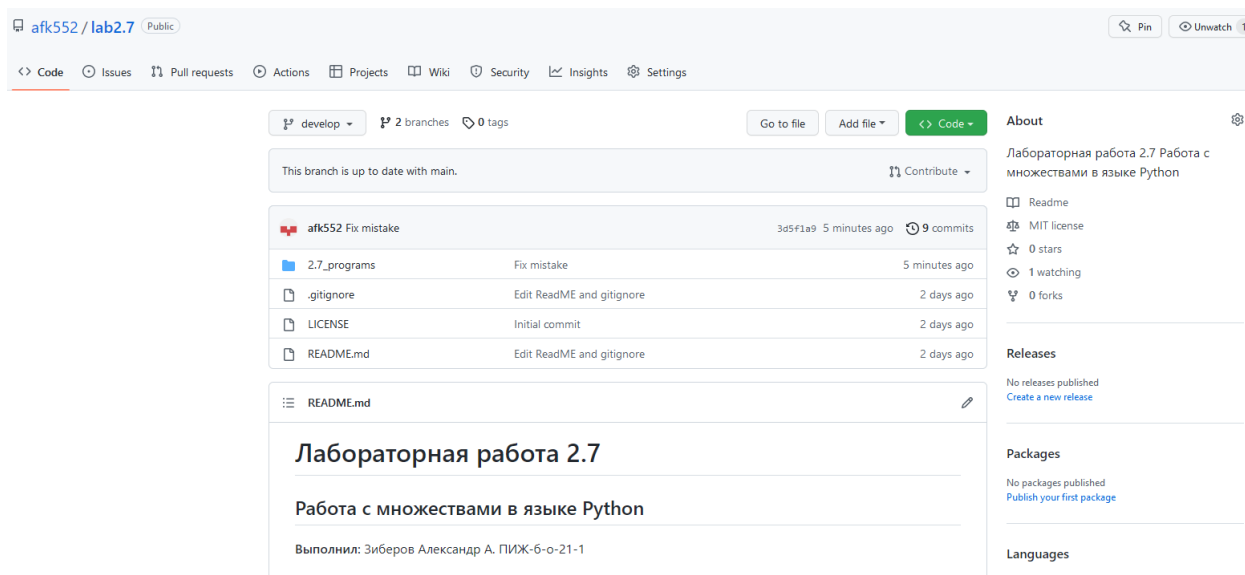


Рисунок 17 – Удаленный репозиторий на GitHub

Вывод: В результате выполнения работы были изучены словари в языке программирования Python 3, работа и методы взаимодействия с ними и написание программ с их использованием.

Контрольные вопросы:

1. Что такое множества в языке Python?

Множеством в языке программирования Python называется неупорядоченная совокупность уникальных значений. В качестве элементов 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?

```
a = {1, 2, 0, 1, 3, 2}
```

```
a = set('data')
```

3. Как проверить присутствие/отсутствие элемента в множестве?

```
a = {0, 1, 2, 3}
```

```
print(2 in a)
```

```
True
```

```
a = {0, 1, 2, 3}
```

```
print(2 not in a)
```

```
False
```

4. Как выполнить перебор элементов множества?

```
for a in {0, 1, 2}:
```

```
    print(a)
```

5. Что такое set comprehension?

```
a = {i for i in [1, 2, 0, 1, 3, 2]}
```

6. Как выполнить добавление элемента во множество?

```
a = {0, 1, 2, 3}
```

```
a.add(4)
```

7. Как выполнить удаление одного или всех элементов множества?

Удаление одного элемента:

```
a = {0, 1, 2, 3}
```

```
a.remove(3)
```

Удаление всех элементов множества:

```
a.clear()
```

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

Объединение множеств: - `a.union(b)` или `a | b`

Пересечение множеств: `a.intersection(b)` или `a & b`

Разность множеств: `a.difference(b)` или `a - b`

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

Подмножество множества: `a.issubset(b)`

Надмножество множества: `a.issuperset(b)`

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?**

В строку:

```
a = {'set', 'str', 'dict', 'list'}
```

```
b = ','.join(a)
```

В словарь:

```
a = {('a', 2), ('b', 4)}
```

```
b = dict(a)
```

В список:

```
a = {1, 2, 0, 1, 3, 2}
```

```
b = list(a)
```