

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное  
учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Институт цифрового развития**

**Кафедра информационных систем и технологий**

Отчет по лабораторной работе №2,7.

Дисциплина: «Основы программной инженерии»

**Выполнил:**

Студент группы ПИЖ-б-о-22-  
1,

направление подготовки:  
09.03.04

«Программная инженерия»

ФИО: Рядская Мария

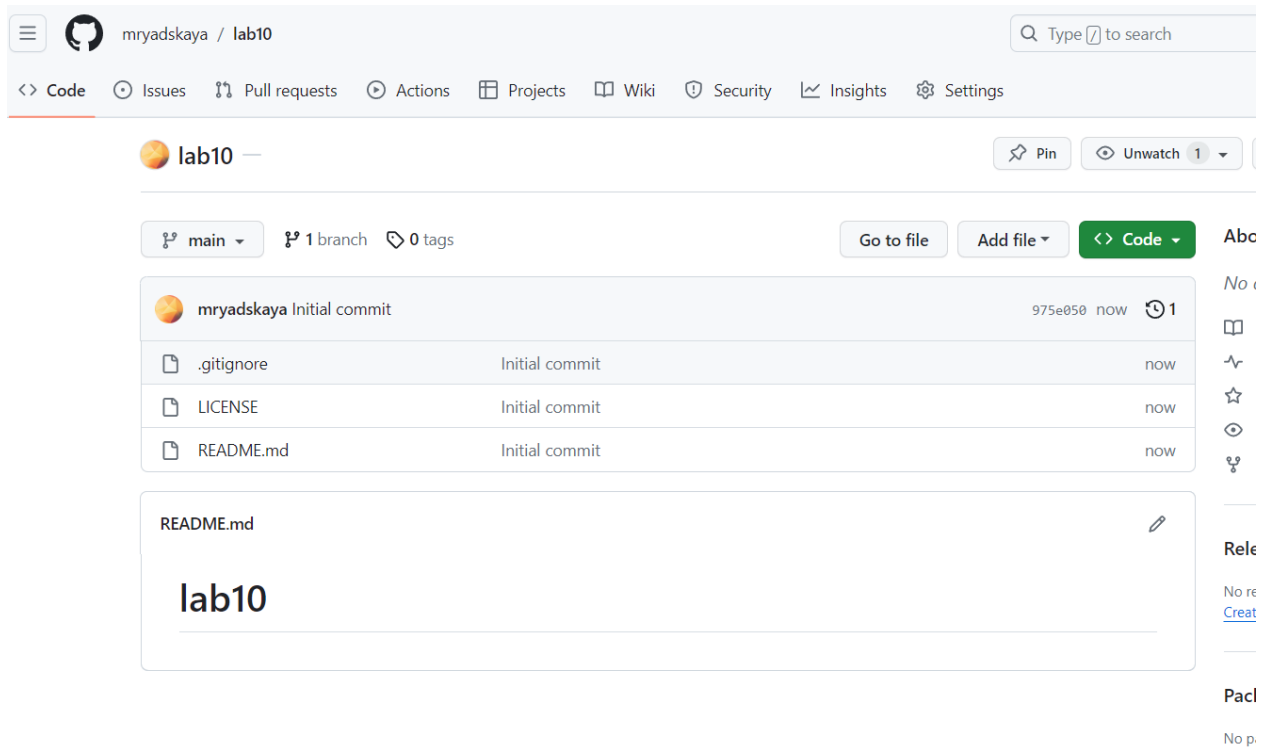
Александровна **Проверил:**

Воронкин Р. А.

Ставрополь 2023

- Изучила теоретический материал работы.

Создала репозиторий на git.hub.



- Организовать свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\git1\lab9>cd ..

C:\git1>git clone https://github.com/mryadskaya/lab10.git
Cloning into 'lab10'...
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:\git1>cd lab10

C:\git1\lab10>git branch develop

C:\git1\lab10>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/mryadskaya/lab10/pull/new/develop
remote:
To https://github.com/mryadskaya/lab10.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\git1\lab10>git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

C:\git1\lab10>
```

Проработать примеры лабораторной работы

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

if \_\_name\_\_ == "\_\_main\_\_"

Run   пример ×

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab4\PyCharm\пример.py

x = {'d', 'k', 'o', 'j', 'e'}

y = {'o', 'c', 'h', 'v', 'g', 'y', 'f'}

Process finished with exit code 0

## Задание 1

Подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      input_string = input("Введите строку: ")
6      input_string = input_string.lower()
7      vowels = set("aeiou")
8
9      # Найдем сумму
10     count = sum(1 for char in input_string if char in vowels)
11     print(f"Количество гласных в строке: {count}")
```

if \_\_name\_\_ == "\_\_main\_\_"

Run   задание 1 ×

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\git1\lab4\PyCharm\задание 1.py"

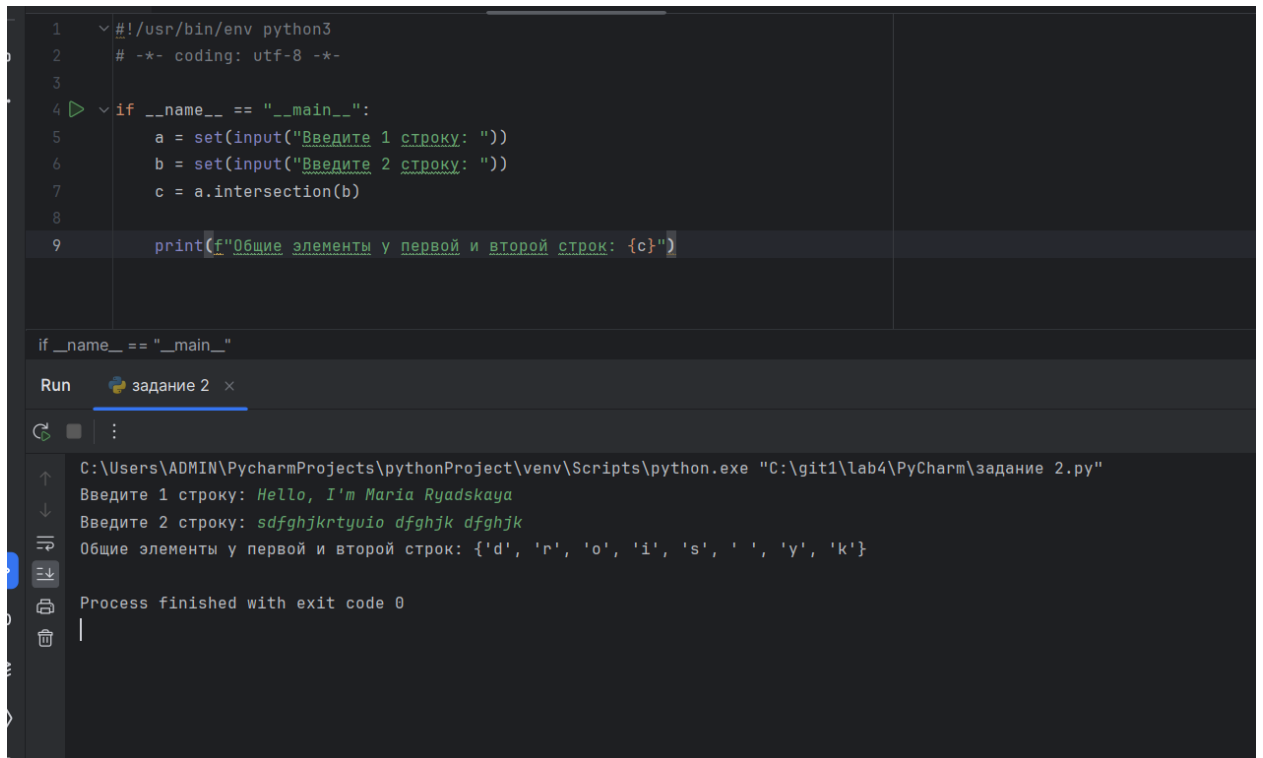
Введите строку: Hello, I'm Maria Ryadskaya

Количество гласных в строке: 9

Process finished with exit code 0

## Задание 2

Определите общие символы в двух строках, введенных с клавиатуры.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      a = set(input("Введите 1 строку: "))
6      b = set(input("Введите 2 строку: "))
7      c = a.intersection(b)
8
9      print(f"Общие элементы у первой и второй строк: {c}")
```

if \_\_name\_\_ == "\_\_main\_\_"

Run задание 2 x

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\git1\lab4\PyCharm\задание 2.py"

Введите 1 строку: Hello, I'm Maria Ryadskaya

Введите 2 строку: sdfghjkrtyuio dfghjk dfghjk

Общие элементы у первой и второй строк: {'d', 'r', 'o', 'i', 's', ' ', 'y', 'k'}

Process finished with exit code 0

Индивидуальное задание. Определить результат выполнения операций над множествами. Считать элементы множества строками. Проверить результаты вручную.

$$A = \{c, m, n, o, q\}; \quad B = \{c, d, m, w\}; \quad C = \{m, n, q\}; \quad D = \{c, m, p\};$$

$$X = (A \cup B) \cap C; \quad Y = (A \cap B) \cup (C/D).$$

The screenshot shows the PyCharm IDE with a project named 'пример.py'. The main editor displays a Python script for set operations. The script defines a universal set 'u' and four sets 'a', 'b', 'c', and 'd'. It then calculates the intersection of (a union b) with c, and the union of (a intersection b) with (c minus d). The results are printed as sets.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      # Определим универсальное множество
6      u = set("abcdefghijklmnopqrstuvwxyz")
7      a = {"c", "m", "n", "o", "q"}
8      b = {"c", "g", "p", "q"}
9      c = {"m", "n", "q"}
10     d = {"c", "m", "p"}
11
12     x = (a.union(b)).intersection(c)
13     print(f"x = {x}")
14
15     db = u.difference(b)
16     y = (a.intersection(db)).union(c.difference(d))
17     print(f"y = {y}")

```

The Run window shows the execution output:

```

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab4\PyCharm\индивидуальное.py
x = {'q', 'n', 'm'}
y = {'o', 'n', 'q', 'm'}
Process finished with exit code 0

```

- Зафиксировал все изменения в github в ветке develop.
- Слила ветки.

```
C:\git1\lab10>git add .

C:\git1\lab10>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 1.
py"
    new file:   "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 2.
py"
    new file:   "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\32
1\203\320\260\320\273\321\214\320\275\320\276\320\265.py"
    new file:   "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200.py"

C:\git1\lab10>git commit -m"сохранение"
[develop 7ca3cf0] сохранение
4 files changed, 56 insertions(+)
 create mode 100644 "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 1.
py"
 create mode 100644 "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 2.
py"
 create mode 100644 "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\32
1\203\320\260\320\273\321\214\320\275\320\276\320\265.py"
 create mode 100644 "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200.py"

C:\git1\lab10>git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.60 KiB | 410.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mryadskaya/lab10.git
 975e050..7ca3cf0  develop -> develop

C:\git1\lab10>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git1\lab10>git merg develop
git: 'merg' is not a git command. See 'git --help'.

The most similar command is
    merge
```

Вывод: приобрел навыки по работе с множествами при написании программ с помощью языка программирования Python версии 3.x.

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. Существует и другой способ создания множеств, который подразумевает использование вызова `set`.

Аргументом этой функции может быть набор неких данных или даже строка с

текстом.

## 3. Как проверить присутствие/отсутствие элемента в множестве? Для этого используется `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?



Чтобы внести новые значения, потребуется вызывать метод `add`.

Аргументом в данном случае будет добавляемый элемент последовательности.

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. Таким образом можно перенести уникальные данные из одного набора чисел в другой.

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`. Функция позволяет найти элементы, уникальные для второго набора данных, которых в нем нет.

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.

Чтобы узнать, является ли множество `a` надмножеством `b`, необходимо вызвать метод `issuperset` и вывести результат его работы на экран.

10. Каково назначение множеств `frozenset`? `frozenset` в Python - это неизменяемая (`immutable`) версия типа данных "множество" (`set`). Основное назначение `frozenset` заключается в том, что оно может использоваться в ситуациях, где требуется неизменяемое множество, то есть множество, элементы

которого нельзя изменить после его создания. Вот некоторые случаи, когда `frozenset` может быть полезным:

- Ключи в словаре: Поскольку словари Python могут использовать только неизменяемые объекты в качестве ключей, `frozenset` может быть использован в качестве ключа для словаря.

- Элементы множества в другом множестве: Вы можете создать множество, содержащее `frozenset`, чтобы использовать его в качестве элемента

другого множества, так как `frozenset` является неизменяемым и поэтому может

быть элементом множества.

- Защита от изменений: Если вам нужно гарантировать, что набор элементов останется неизменным и не будет изменен случайно или намеренно, вы можете использовать `frozenset` вместо `set`.

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

Для преобразования множества в строку используется конкатенация текстовых значений, которую обеспечивает функция `join`. В этом случае ее аргументом является набор данных в виде нескольких строк. Запятая в кавычках

выступает в качестве символа, разделяющего значения.

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`.