

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»

Выполнила:
Ламская Ксения Вячеславовна
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Лабораторная работа 2.7 Работа с множествами в языке Python.

Цель работы: приобретение навыков по работе с множествами при написании программ с помощью языка программирования Python версии 3.x.


Порядок выполнения работы

1. Создание репозитория GitHub.

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 *	Repository name *
<div> ksenia-lamskaya ▾</div>	<div>10laba</div>
	✓ 10laba is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-succotash](#) ?

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](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создания репозитория

2. Проработала примеры из лабораторной работы.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == "__main__":
    # Определим универсальное множество
    u = set("abcdefghijklmnopqrstuvwxyz")

    a = {"b", "c", "h", "o"}
    b = {"d", "f", "g", "o", "v", "y"}
    c = {"d", "e", "j", "k"}
    d = {"a", "b", "f", "g"}

    x = (a.intersection(b)).union(c)
    print(f"x = {x}")

    # Найдем дополнения множеств
    bn = u.difference(b)
    cn = u.difference(c)

    y = (a.difference(d)).union(cn.difference(bn))
    print(f"y = {y}")
```

Рисунок 2.1 – Код из примера 1

```
x = {'d', 'k', 'j', 'e', 'o'}
y = {'g', 'y', 'h', 'o', 'v', 'f', 'c'}
```

Рисунок 2.2 – Вывод программы из примера 1

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

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == '__main__':
    input_string = input('Введите строку: ').lower()
    vowels = set('aeiouy')

    count = sum(1 for char in input_string if char in vowels)
    print(f'Количество гласных в строке: {count}')
```

Рисунок 3.1 – Код программы

```
Введите строку: qwertyuio asdfghjkl zxcvbnm
Количество гласных в строке: 6
```

Рисунок 3.2 – Вывод программы

4. Определите общие символы в двух строках, введенных с клавиатуры.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    a = set(input('Введите 1 строку: '))
    b = set(input('Введите 2 строку: '))
    c = a.intersection(b)

    print(f'Общие элементы: {c}')
```

Рисунок 4.1 – Код программы

```
Введите 1 строку: qwertypoi
Введите 2 строку: qwlkjfh
Общие элементы: {'q', 'w'}
```

Рисунок 4.2 – Вывод программы

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

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

$$A = \{a, b, h, k, o, r\}; \quad B = \{b, g, h, l, s\}; \quad C = \{k, l, z\}; \quad D = \{g, j, p, q, u, v\};$$

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == "__main__":
    # Определим универсальное множество
    u = set("abcdefghijklmnopqrstuvwxyz")

    a = {"a", "b", "h", "k", "o", "r"}
    b = {"b", "g", "h", "l", "s"}
    c = {"k", "l", "z"}
    d = {"g", "j", "p", "q", "u", "v"}

    x = (a.union(b)).intersection(d)
    print(f"x = {x}")

    # Найдем дополнения множеств
    an = u.difference(a)
    bn = u.difference(b)

    y = (an.intersection(bn)).difference(c.union(d))
    print(f"y = {y}")
```

Рисунок 5.1 – Код программы

```
x = {'g'}
y = {'y', 'm', 'i', 'n', 't', 'f', 'x', 'c', 'd', 'e', 'w'}
```

Рисунок 5.2 – Вывод программы

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`.