

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 №2.7
дисциплины «Основы программной инженерии»

Выполнил:
Соколов Михаил Романович

(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: 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.

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и .gitignore файл для языка 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 ().*

Repository template
No template
Start your repository with a template repository's contents.

Owner * PoTtaTto / **Repository name *** fundamentals_of_software
✓ fundamentals_of_software_engineering_lab2_7 is available.

Great repository names are short and memorable. Need inspiration? How about ideal-octo-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 – Создание репозитория с заданными настройками

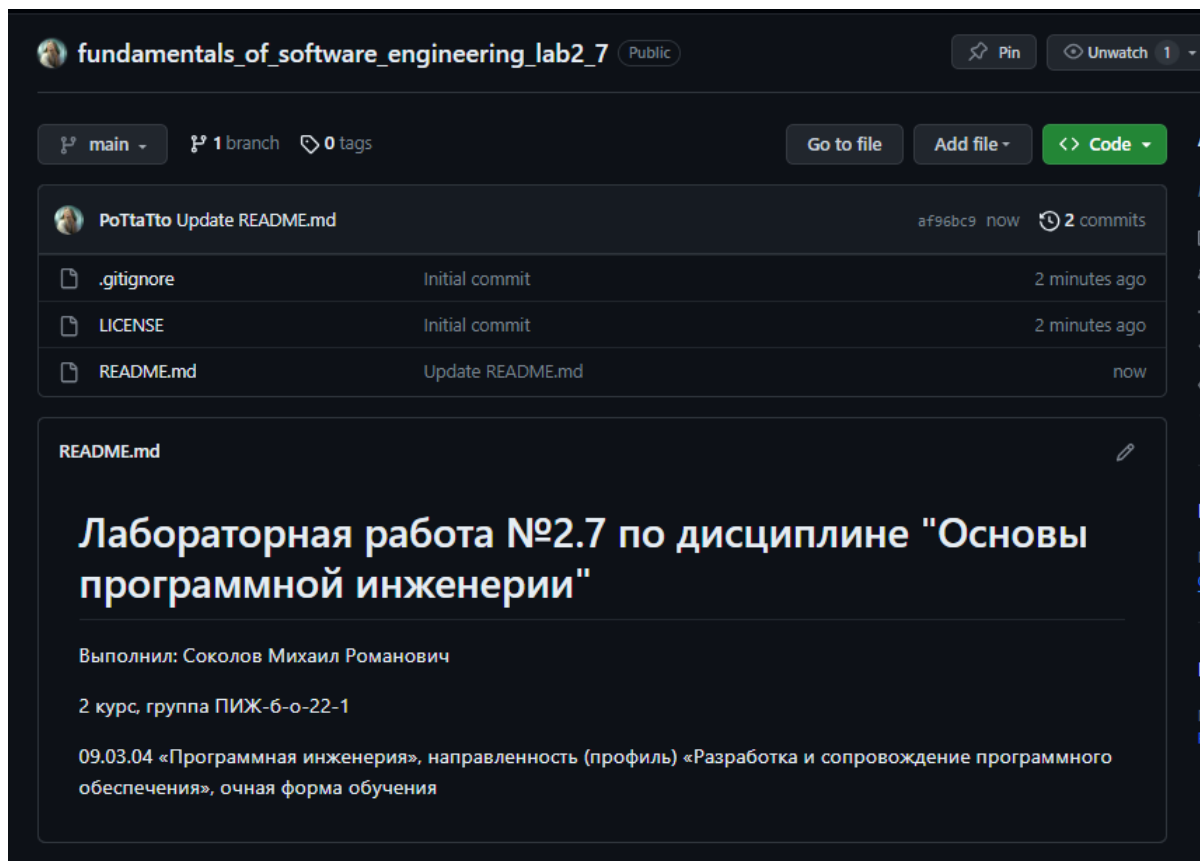


Рисунок 2 – Созданный репозиторий

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.7> git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab2_7
Cloning into 'fundamentals_of_software_engineering_lab2_7'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.7> |
```

Рисунок 3 – Клонирование репозитория

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.7> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.7> |
```

Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

```
.gitignore x
1 # Byte-compiled / optimized / DLL files
2 __pycache__/
3 *.py[cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
```

Рисунок 5 – Часть .gitignore файла, созданного GitHub

2. Проработаем пример лабораторной работы, фиксируя изменения.

Создадим для примера отдельный модуль:

```
lab_task1.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 if __name__ == '__main__':
6     # Определим универсальное множество
7     u = set('abcdefghijklmnopqrstuvwxyz')
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    x = (a.intersection(b)).union(c)
13    print(f'x = {x}')
14    # Найдем дополнения множеств
15    bn = u.difference(b)
16    cn = u.difference(c)
17    y = (a.difference(d))
18    print(f'y = {y}')
```

Run lab_task1 x

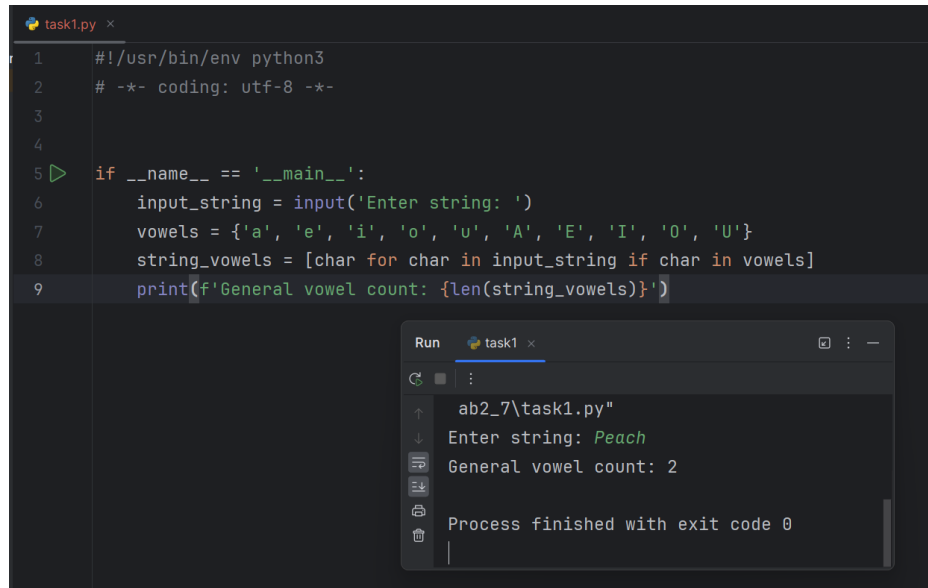
```
\lab_task1.py"
x = {'e', 'd', 'j', 'o', 'k'}
y = {'c', 'h', 'o'}

Process finished with exit code 0
```

Рисунок 6 – Определить результат выполнения операций над множествами.

Код и его выполнение

3. Решить задачу: подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств:



```
task1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      input_string = input('Enter string: ')
7      vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
8      string_vowels = [char for char in input_string if char in vowels]
9      print(f'General vowel count: {len(string_vowels)}')
```

Run task1 x

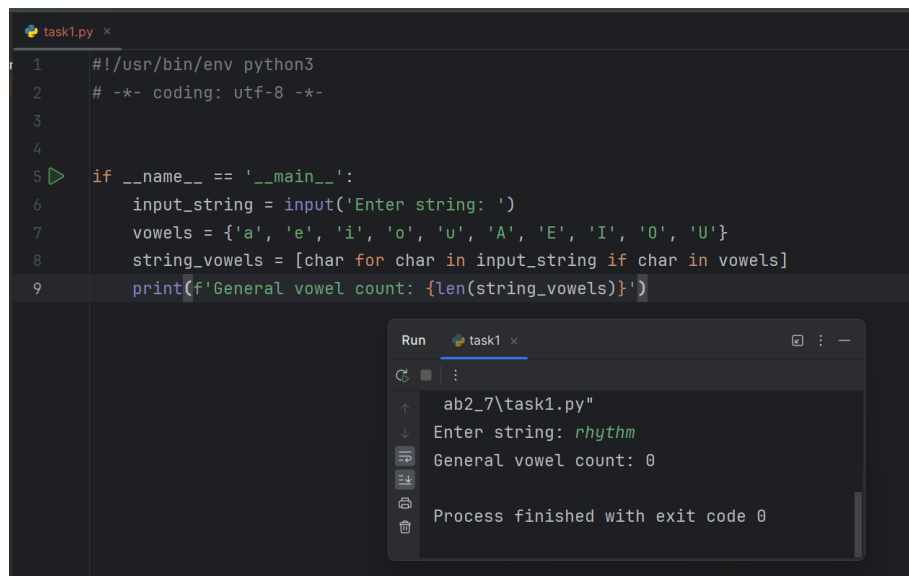
ab2_7\task1.py"

Enter string: *Peach*

General vowel count: 2

Process finished with exit code 0

Рисунок 7 – Код задачи и результат выполнения (1)



```
task1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      input_string = input('Enter string: ')
7      vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
8      string_vowels = [char for char in input_string if char in vowels]
9      print(f'General vowel count: {len(string_vowels)}')
```

Run task1 x

ab2_7\task1.py"

Enter string: *rhythm*

General vowel count: 0

Process finished with exit code 0

Рисунок 8 – Код задачи и результат выполнения (2)

4. Решите задачу: определите общие символы в двух строках, введенных с клавиатуры:

```
task2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      input_str1 = input('Enter first string: ')
7      input_str2 = input('Enter second string: ')
8      general_symbols = set(input_str1).intersection(set(input_str2))
9
10     print('General symbols: ', ', '.join(general_symbols))
```

Run task2 x

Enter first string: *Star*
Enter second string: *Stairs*
General symbols: S, a, r, t
Process finished with exit code 0

Рисунок 9 – Код задачи и результат выполнения (1)

```
task2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      input_str1 = input('Enter first string: ')
7      input_str2 = input('Enter second string: ')
8      general_symbols = set(input_str1).intersection(set(input_str2))
9
10     print('General symbols: ', ', '.join(general_symbols))
```

Run task2 x

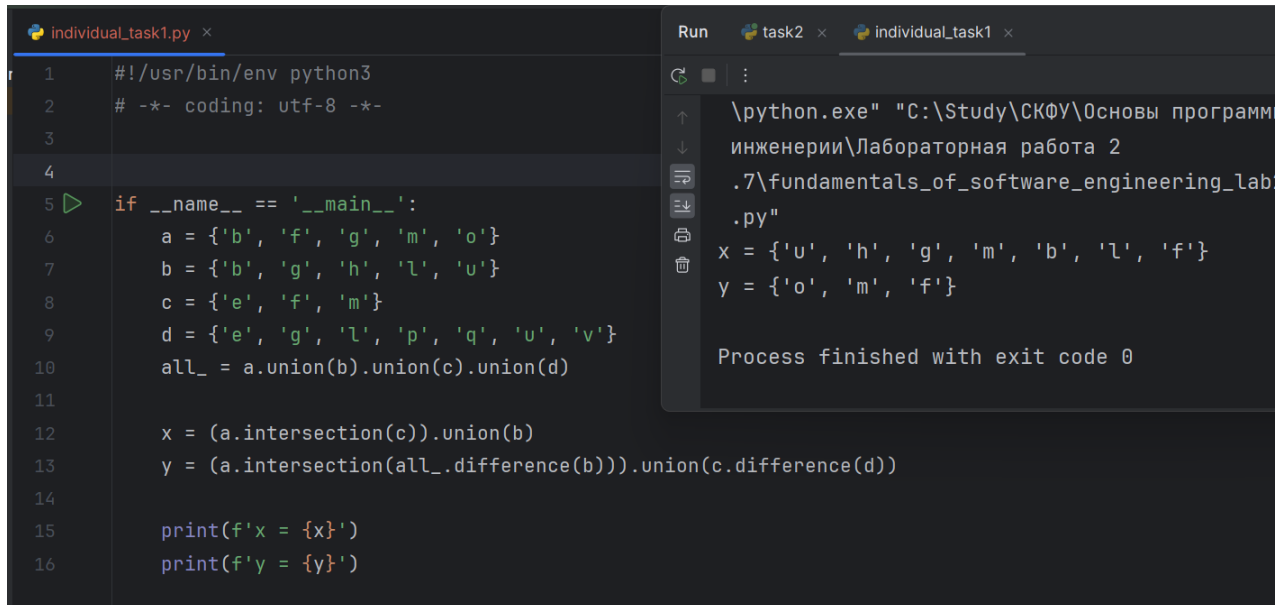
Enter first string: *python*
Enter second string: *galleries*
General symbols:
Process finished with exit code 0

Рисунок 10 – Код задачи и результат выполнения (2)

5. Решим индивидуальное задание (вариант №7):

$$A = \{b, f, g, m, o\}; \quad B = \{b, g, h, l, u\}; \quad C = \{e, f, m\}; \quad D = \{e, g, l, p, q, u, v\};$$
$$X = (A \cap C) \cup B; \quad Y = (A \cap \bar{B}) \cup (C/D).$$

Рисунок 11 – Условия задачи



```
individual_task1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      a = {'b', 'f', 'g', 'm', 'o'}
7      b = {'b', 'g', 'h', 'l', 'u'}
8      c = {'e', 'f', 'm'}
9      d = {'e', 'g', 'l', 'p', 'q', 'u', 'v'}
10     all_ = a.union(b).union(c).union(d)
11
12     x = (a.intersection(c)).union(b)
13     y = (a.intersection(all_.difference(b))).union(c.difference(d))
14
15     print(f'x = {x}')
16     print(f'y = {y}')
```

```
Run task2 x individual_task1 x
\python.exe" "C:\Study\СКФУ\Основы программ
инженерии\Лабораторная работа 2
.7\fundamentals_of_software_engineering_lab
.py"
x = {'u', 'h', 'g', 'm', 'b', 'l', 'f'}
y = {'o', 'm', 'f'}

Process finished with exit code 0
```

Рисунок 12 – Код задачи и результат выполнения

6. Сделаем merge веток develop/master и отправим изменения на удаленный репозиторий:

```
4c7f4d6 (HEAD -> develop) individual_task1.py is added
fe7aae5 task2.py is added
7095533 task1.py is added
bac761a lab_task1.py is added
af96bc9 (origin/main, origin/HEAD, main) Update README.md
b3811be Initial commit
```

Рисунок 13 – Коммиты проекта

```
4c7f4d6 (HEAD -> main, develop) individual_task1.py is added
fe7aae5 task2.py is added
7095533 task1.py is added
bac761a lab_task1.py is added
af96bc9 (origin/main, origin/HEAD) Update README.md
b3811be Initial commit
```

Рисунок 14 – Изменения после merge

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.7\fundamentals_of_software_engineering_lab2_7> git push origin
main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 1.90 KiB | 1.90 MiB/s, done.
Total 13 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/PoTtaTto/fundamentals\_of\_software\_engineering\_lab2\_7
  af96bc9..4c7f4d6  main -> main
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.7\fundamentals_of_software_engineering_lab2_7>
```

Рисунок 15 – Отправление изменений на удаленный репозиторий

Ответы на контрольные вопросы:

1. Что такое множества в языке Python?

В Python множество представляет собой коллекцию уникальных и неупорядоченных элементов. Оно поддерживает операции множественной математики, такие как объединение, пересечение, разность, и позволяет быстро проверять принадлежность элементов.

2. Как осуществляется создание множеств в Python?

Множества в Python можно создать с помощью фигурных скобок `{ }` или функции `set()`, указав элементы через запятую внутри фигурных скобок или передавая итерируемый объект в функцию `set()`.

3. Как проверить присутствие/отсутствие элемента в множестве?

Для проверки присутствия элемента в множестве можно использовать оператор `in`. Например, `element in set_name` вернет `True`, если `element` присутствует в `set_name`, иначе `False`.

4. Как выполнить перебор элементов множества?

Перебор элементов множества можно выполнить с помощью цикла `for`. Например: `for element in set_name`.

5. Что такое set comprehension?

Set comprehension - это способ создания множества на основе выражения итерации. Например: `{x for x in range(10)}` создаст множество с элементами от 0 до 9.

6. Как выполнить добавление элемента во множество?

Элемент можно добавить во множество с помощью метода `add()`. Например, `set_name.add(element)` добавит `element` в `set_name`.

7. Как выполнить удаление одного или всех элементов множества?

Для удаления одного элемента можно использовать метод `remove()` или `discard()`. Для удаления всех элементов используется метод `clear()`.

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

Операции над множествами выполняются с помощью операторов `|` для объединения, `&` для пересечения и `-` для разности.

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

Методы `issuperset()` и `issubset()` используются для проверки того, является ли одно множество надмножеством или подмножеством другого соответственно.

10. Каково назначение множеств `frozenset`?

`frozenset` – это неизменяемая версия множества, то есть после создания нельзя изменить его содержимое. Это полезно, когда требуется использовать множество в качестве ключа словаря или как элемент другого множества.

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

Множество можно преобразовать в список с помощью `list(set_name)`, в строку с помощью `str(set_name)`, а в словарь нельзя преобразовать напрямую, но его элементы можно использовать при создании словаря.