

**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
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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 с использованием лицензии 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 *

 MrPlatynum ▾

Repository name *

ProgrammEngineering10

✓ ProgrammEngineering10 is available.

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

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.](#)

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering10.git
Cloning into 'ProgrammEngineering10'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер

```
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
```

Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering10 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering10 (develop)
$
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8:

```
example1.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     # Определим универсальное множество
6     u = set("abcdefghijklmnopqrstuvwxyz")
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    # Найдем 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 – Пример №1

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering10/example1.py
x = {'d', 'j', 'e', 'o', 'k'}
y = {'f', 'c', 'g', 'o', 'v', 'y', 'h'}
```

Рисунок 6 – Вывод программы

3. Решите 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 ▶ if __name__ == "__main__":
5     vowels = {'a', 'e', 'i', 'o', 'u', 'y'}
6     user_input = input("Введите строку: ").lower() # Получаем строку от пользователя и приводим к нижнему регистру
7     vowel_count = sum(1 for char in user_input if char in vowels) # Подсчитываем количество гласных
8     result = vowel_count
9
10    print(f"Количество гласных букв в строке: {result}")
11
```

Рисунок 7 – Количество гласных в строке (Задание №1)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering10/task1.py
Введите строку: Hello world
Количество гласных букв в строке: 3
```

Рисунок 8 – Вывод программы

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 ▶ if __name__ == "__main__":
5     string_1 = input("Введите первую строку: ")
6     string_2 = input("Введите вторую строку: ")
7
8     set_1 = set(string_1)
9     set_2 = set(string_2)
10
11     common_characters = set_1.intersection(set_2)
12     common_characters = ', '.join(sorted(common_characters)) # Для красивого вывода, отсортируем и объединим символы
13
14     print(f"Общие символы в обеих строках: {common_characters}")
15
```

Рисунок 9 – Общие символы (Задание №2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering10/task2.py
Введите первую строку: Hello world
Введите вторую строку: world
Общие символы в обеих строках: d, l, o, r, w
```

Рисунок 10 – Вывод программы

5. Выполним индивидуальные задания:

6.
$$X = (A/B) \cup (C \cap D); \quad Y = (\bar{A} \cap \bar{B}) / (C \cup D).$$
$$A = \{b, f, g, m, o\}; \quad B = \{b, g, h, l, u\}; \quad C = \{e, f, m\}; \quad D = \{e, g, l, p, q, u, v\};$$

```
individual1.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    x = ((a.difference(b)).union(c.intersection(d)))
12    y = (all_.difference(a).intersection(all_.difference(b))).difference(c.union(d))
13
14    if x:
15        print(f'x = {x}')
16    else: print("x - пустое множество")
17    if y:
18        print(f'y = {y}')
19    else: print("y - пустое множество")
20
```

Рисунок 11 – Решение индивидуального задания

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering10/individual1.py
x = {'e', 'm', 'o', 'f'}
y - пустое множество
```

Рисунок 12 – Вывод программы

6. Зафиксируем сделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering10 (develop)
$ git log --oneline
64439fc (HEAD -> develop) add individual1.py
f976578 add task2.py
2b211c8 add task1.py
7f36d22 add example1.py
35336c2 (origin/main, origin/HEAD, main) Initial commit
```

Рисунок 13 – Коммиты ветки develop во время выполнения лабораторной работы

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering10 (develop)
$ git checkout main
Switched to branch 'main'
M       .gitignore
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering10 (main)
$ git merge develop
Updating 35336c2..64439fc
Fast-forward
 example1.py   | 21 ++++++
 individual1.py | 19 ++++++
 task1.py      | 10 ++++++
 task2.py      | 14 ++++++
 4 files changed, 64 insertions(+)
 create mode 100644 example1.py
 create mode 100644 individual1.py
 create mode 100644 task1.py
 create mode 100644 task2.py

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering10 (main)
$ |

```

Рисунок 14 – Слияние ветки develop в ветку main

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering10 (main)
$ git push origin main
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 2.19 KiB | 2.19 MiB/s, done.
Total 12 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/MrPlatynum/ProgrammEngineering10.git
 35336c2..64439fc main -> main

```

Рисунок 15 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

1. Что такое множества в языке Python?

Множество (set) в Python – это неупорядоченная коллекция уникальных элементов. Они используются для выполнения операций над уникальными элементами без дублирования.

2. Как осуществляется создание множеств в Python?

Множество можно создать, используя фигурные скобки {} и перечислив элементы множества через запятую. Например:

```
my_set = {1, 2, 3, 4}
```

3. Как проверить присутствие/отсутствие элемента в множестве?

Для проверки присутствия элемента в множестве можно использовать оператор in. Например:

```
my_set = {1, 2, 3, 4}
```

```
print(3 in my_set) # Выведет: True
```

```
print(5 not in my_set) # Выведет: True
```

4. Как выполнить перебор элементов множества?

Можно использовать цикл for для перебора элементов множества:

```
my_set = {1, 2, 3, 4}
```

```
for element in my_set:
```

```
    print(element)
```

5. Что такое set comprehension?

Set comprehension – это способ создания множества с использованием компактного синтаксиса, аналогичного списочным включениям. Например:

```
my_set = {x for x in range(10) if x % 2 == 0} # Создание множества четных чисел от 0 до 9
```

6. Как выполнить добавление элемента во множество?

Для добавления элемента в множество используется метод .add().
Например:

```
my_set = {1, 2, 3}
```

```
my_set.add(4)
```


7. Как выполнить удаление одного или всех элементов множества?

Методы `.remove()` и `.discard()` используются для удаления одного элемента, а метод `.clear()` - для удаления всех элементов множества. Например:

```
my_set = {1, 2, 3, 4}
```

```
my_set.remove(3)
```

```
my_set.discard(5) # Если элемент отсутствует, discard() не вызывает ошибку
```

```
my_set.clear() # Очистка множества
```

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

Объединение: `set1.union(set2)` или оператор `|`

Пересечение: `set1.intersection(set2)` или оператор `&`

Разность: `set1.difference(set2)` или оператор `-`

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

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

10. Каково назначение множеств `frozenset`?

`frozenset` – это неизменяемая версия множества. Однажды созданное `frozenset` не может быть изменено, но оно может быть использовано в качестве ключа в словарях или как элемент в другом множестве.

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

Для преобразования множества в список можно использовать `list(my_set)`.

Преобразование в строку: `str(my_set)`.

Множество не может быть преобразовано непосредственно в словарь, но можно создать словарь из множества ключей.