

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.7

Тема: «Работа с множествами в языке Python»

Выполнил студент группы

ИВТ-б-о-21-1

Горшков В.И. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

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.

Ход работы:

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

VitaliyPitsa Update LICENSE		dafcf02e 1 hour ago	🕒 3 commits
📄 .gitignore	Initial commit		1 hour ago
📄 LICENSE	Update LICENSE		1 hour ago
📄 README.md	Update README.md		1 hour ago

Рисунок 1.1 – Созданный репозиторий

```
.gitignore – Блокнот
Файл  Правка  Формат  Вид  Справка
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
```

Рисунок 1.2 – Дополнил правила в .gitignore

```

c:\Users\Admin\Desktop\git\Python2>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

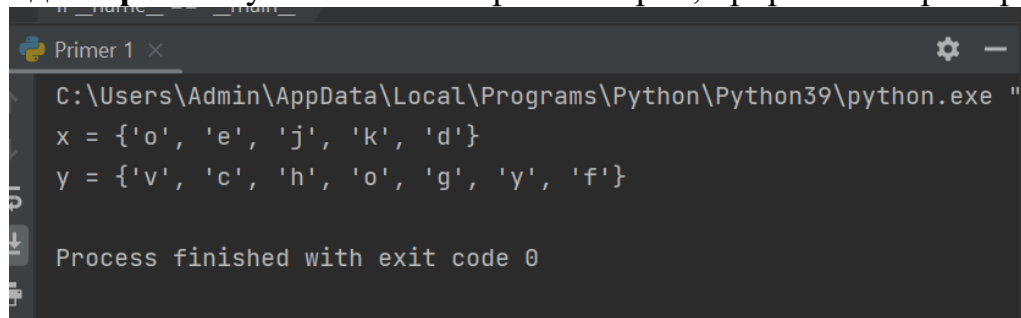
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/Python2/.git/hooks]

c:\Users\Admin\Desktop\git\Python2>

```

Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Создал проект Pycharm в папке репозитория, проработал примеры ЛР.



```

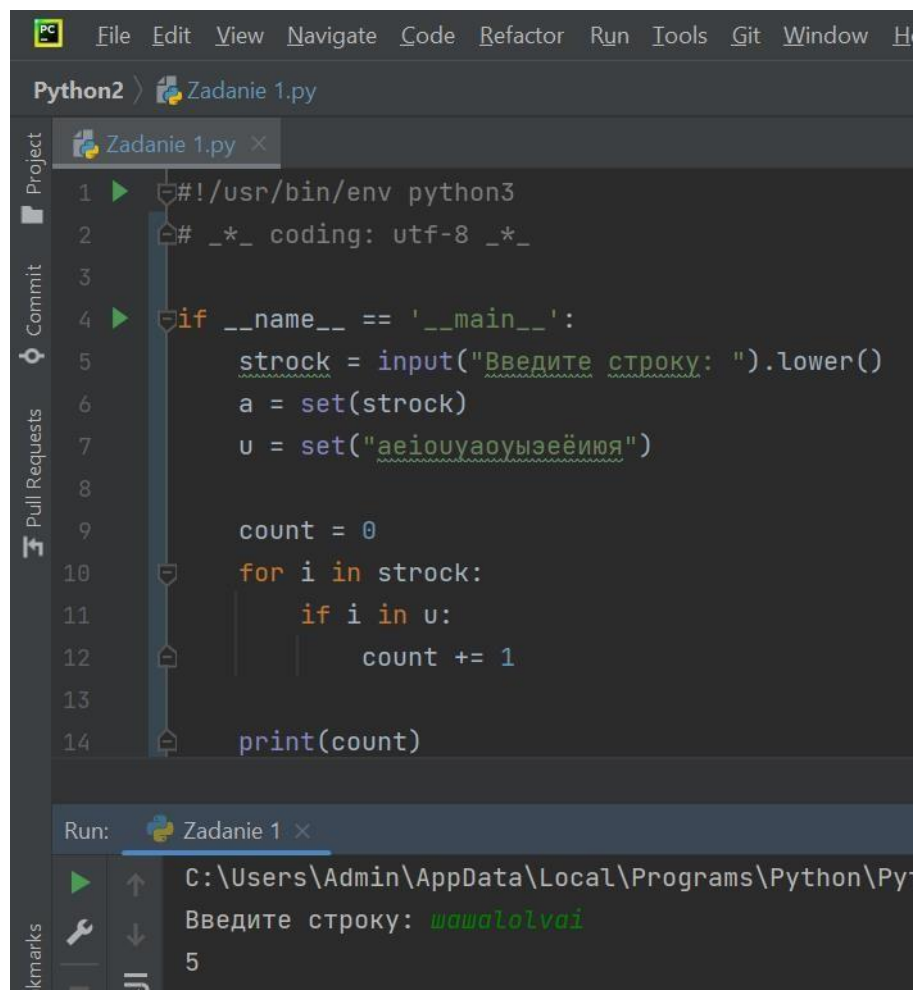
Primer 1 x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe "
x = {'o', 'e', 'j', 'k', 'd'}
y = {'v', 'c', 'h', 'o', 'g', 'y', 'f'}

Process finished with exit code 0

```

Рисунок 2.1 – Результат выполнения примера №1

3. Решить задачу №1: подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.



The image shows a screenshot of a Python IDE. The main editor window displays a file named 'Zadanie 1.py' with the following code:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     strock = input("Введите строку: ").lower()
6     a = set(strock)
7     u = set("aeiouyaouyeёйюя")
8
9     count = 0
10    for i in strock:
11        if i in u:
12            count += 1
13
14    print(count)
```

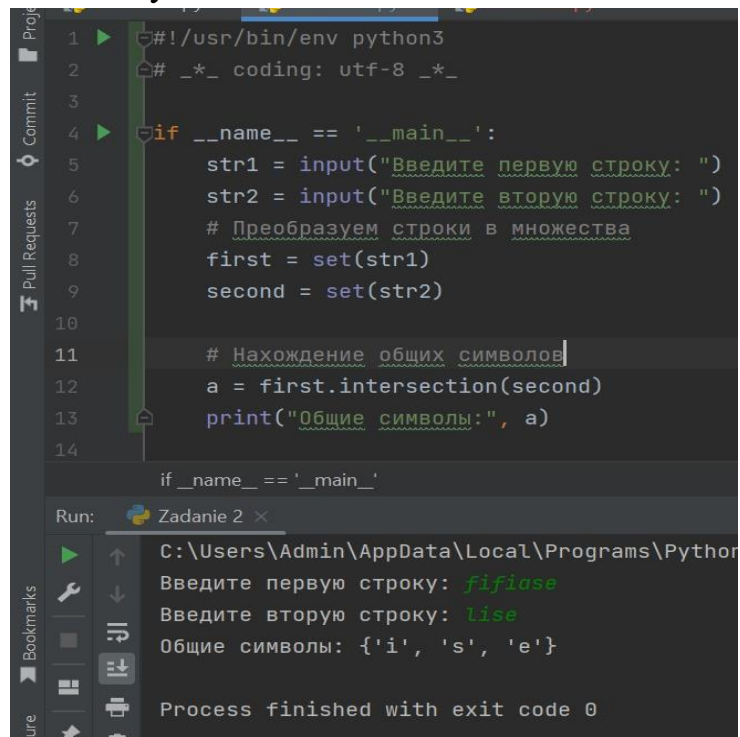
Below the editor, the 'Run' panel shows the execution path and the output:

```
Run: C:\Users\Admin\AppData\Local\Programs\Python\Py
Введите строку: wawalolvai
5
```

Рисунок 3.1 – Выполненное задание

4. Решите задачу №2: определите общие символы в двух строках, введенных с клавиатуры.

Рисунок 4.1 – Выполненное задание



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     str1 = input("Введите первую строку: ")
6     str2 = input("Введите вторую строку: ")
7     # Преобразуем строки в множества
8     first = set(str1)
9     second = set(str2)
10
11     # Нахождение общих символов
12     a = first.intersection(second)
13     print("Общие символы:", a)
14
```

Run: Zadanie 2

C:\Users\Admin\AppData\Local\Programs\Python\Python38-32\python.exe

Введите первую строку: *fifiase*

Введите вторую строку: *lise*

Общие символы: {'i', 's', 'e'}

Process finished with exit code 0

5. Индивидуальное задание В – 1. Определить результат выполнения операций над множествами.

1.
$$\begin{aligned} A &= \{b, e, f, k, t\}; & B &= \{f, i, j, p, y\}; & C &= \{j, k, l, y\}; & D &= \{i, j, s, t, u, y, z\}; \\ X &= (A \cap C) \cup (B \cap C); & Y &= (A \cap \bar{B}) \cup (D/C). \end{aligned} \quad (2)$$

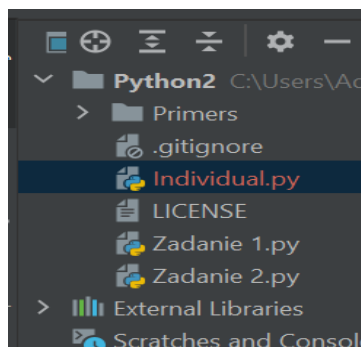


Рисунок 5.1 – Созданный проект

```
1  ▶  #!/usr/bin/ env python3
2  # -*- coding: utf-8 -*-
3
4
5  ▶  if __name__ == '__main__':
6      u = set("abcdefghijklmnopqrstuvwxyz")
7
8      a = {"a", "f", "i", "n", "o"}
9      b = {"f", "g", "h", "l", "u"}
10     c = {"i", "j", "u", "w"}
11     d = {"e", "g", "l", "p", "q", "u", "v"}
12
13     x = (a & c) | b
14     print(f"x = {x}")
15
16     bn = u.difference(b)
17     y = (a.intersection(bn)).difference(c.difference(d))
18     print(f"y = {y}")
```

Run: main x

```
"C:\Program Files\Python39\python.exe" C:/Users/Виталий/PycharmProjects/pythonProject4/main.py
x = {'l', 'i', 'f', 'h', 'g', 'u'}
y = {'o', 'a', 'n'}
```

Рисунок 5.2 – Индивидуальное задание

Вывод: в результате выполнения лабораторной работы были приобретены навыки для работы с множествами при написании программ с помощью языка программирования Python версии 3.x.

Ответы на 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?

Создать можно, просто присвоив переменной последовательность значений, выделив их фигурными скобками. Следующий пример показывает код, в котором создается множество целых чисел под названием `a`, после функция `print` выводит на экран его содержимое.

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

```
print(a)
```

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

Существует и другой способ создания множеств, который подразумевает использование вызова `set`. Аргументом этой функции может быть набор неких данных или даже строка с текстом, как это показано в следующем примере.

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

```
print(a)
```

```
{'d', 'a', 't'}
```

3. Как проверить присутствие/отсутствие элемента в множестве?

Проверка, есть ли данное значение в множестве. Для этого используется `in`.

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

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

```
True
```

Наоборот, проверка отсутствия. Используется `not in`.

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

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

False

4. Как выполнить перебор элементов множества?

Перебор всех элементов.

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

```
    print(a)
```

0

1

2

5. Что такое set comprehension?

Set Comprehensions – для создания множества можно в Python воспользоваться генератором, позволяющих заполнять списки, а также другие наборы данных с учетом неких условий. Следующий код демонстрирует генерацию множества `a` с циклом `for` для нескольких чисел:

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

```
print(a)
```

{0, 1, 2, 3}

6. Как выполнить добавление элемента во множество?

Чтобы внести новые значения, потребуется вызывать метод `add`. Аргументом в данном случае будет добавляемый элемент последовательно. В примере кода на Python добавим в множество элемент со значением 4.

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

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

```
print(a)
```


`{0, 1, 2, 3, 4}`

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.

Избавиться от лишних значений в наборе данных с помощью `remove`. В качестве входного параметра здесь выступает элемент, который нужно удалить (в примере удалим число со значением 3).

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

Объединение

Чтобы объединить все элементы двух разных множеств, стоит воспользоваться методом `union` на одном из объектов. Следующий пример демонстрирует работу данной функции, где создается последовательность чисел под именем `c`.

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

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

```
c = a.union(b)
```

```
print(c)
```

```
{0, 1, 2, 3, 4}
```

Пересечение

Чтобы найти общие элементы для двух разных множеств, следует применить функцию `intersection`, принимающую в качестве аргумента один из наборов данных. Код, приведенный ниже, создает новую последовательность чисел из пересечения двух множеств в Python 3.

Разность

Чтобы вычислить разность для двух разных множеств, необходимо воспользоваться методом `difference`. Функция позволяет найти элементы, уникальные для второго набора данных, которых в нем нет. Следующий код демонстрирует эту операцию.

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

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

```
c = a.difference(b)
```

```
print(c)
```

```
{0}
```

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. Так как не все элементы набора чисел `a` присутствуют в `b`, функция вернет `False`.

```
a = {0, 1, 2, 3, 4}
```

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

```
print(a.issubset(b))
```

```
False
```

Определение надмножества

Чтобы узнать, является ли множество *a* надмножеством *b*, необходимо вызвать метод `issuperset` и вывести результат его работы на экран. Поскольку все элементы набора чисел *b* присутствуют в *a*, функция возвращает `True`.

```
a = {0, 1, 2, 3, 4}
```

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

```
print(a.issuperset(b))
```

```
True
```

10. Каково назначение множеств `frozenset`?

Тип `frozenset` – множество, содержимое которого не поддается изменению имеет тип `frozenset`. Значения из этого набора нельзя удалить, как и добавить новые. В следующем примере демонстрируется создание при помощи стандартной функции.

```
a = frozenset({"hello", "world"})
```

```
print(a)
```

```
frozenset({'hello', 'world'})
```

Поскольку содержимое `frozenset` должно всегда оставаться статичным, перечень функций, с которыми такое множество может взаимодействовать, имеет ограничения.

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

Строка

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. Метод `type` возвращает тип данных объекта в конце приведенного кода.

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

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

```
print(b)
```

```
print(type(b))
```

```
set,dict,list,str
```

```
<class 'str'>
```

Словарь

Чтобы получить из множества словарь, следует передать функции `dict` набор из нескольких пар значений, в каждом из которых будет находиться ключ. Функция `print` демонстрирует на экране содержимое полученного объекта, а `type` отображает его тип.

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

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

```
print(b)
```

```
print(type(b))
```

```
{'b': 4, 'a': 2}
```

```
<class 'dict'>
```

Следует отметить, что каждый элемент для такого преобразования — кортеж состоящий из двух значений:

1. ключ будущего словаря;
2. значение, соответствующее ключу.

Список

По аналогии с предыдущими преобразованиями можно получить список неких объектов. На этот раз используется вызов `list`, получающий в качестве аргумента множество `a`. На выходе функции `print` отображаются уникальные значения для изначального набора чисел.

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

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

```
print(b)
```

```
print(type(b))
```

```
[0, 1, 2, 3]
```

```
<class 'list'>
```