

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ  
УНИВЕРСИТЕТ

Институт Цифрового Развития  
Кафедра инфокоммуникаций

ОТЧЁТ

по лабораторной работе №2

Дисциплина: «Языки программирования»

Выполнил: студент 2 курса  
группы ИТС-б-о-20-1

Харченко Екатерина Владимировна

Проверил:

К.т.н., доцент кафедры инфокоммуникаций

Воронкин Роман Александрович

Работа защищена с оценкой: \_\_\_\_\_

Ставрополь, 2021 г.



# 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 работы:


1. Создаем общедоступный репозиторий на GitHub, в котором будет использоваться лицензия MIT и язык программирования Python.


Владелец \*      Имя репозитория \*

 екатерина 533 / lab2.2 

Отличные имена репозитория короткие и запоминающиеся. Вам нужно вдохновение? Как насчет [подходящего-зонтика?](#)

Описание (необязательно)

☒  **Общедоступный**  
Любой пользователь в Интернете может увидеть это хранилище. Вы сами выбираете, кто может взять на себя обязательство.

☐  **Личное**  
Вы выбираете, кто может видеть и фиксироваться в этом хранилище.


Инициализируйте этот репозиторий с помощью:  
Пропустите этот шаг, если вы импортируете существующий репозиторий.

☐ **Добавьте файл README**  
в котором вы можете написать подробное описание своего проекта. [Узнайте больше.](#)

☐ **Add .gitignore**  
Выберите файлы, которые не следует отслеживать, из списка шаблонов. [Узнайте больше.](#)

☒ **Выберите лицензию**  
Лицензия указывает другим, что они могут и чего не могут делать с вашим кодом. [Узнайте больше.](#)

Лицензия: MIT License ▾

Это установит  main в качестве ветви по умолчанию. Измените имя по умолчанию в своих [настройках](#).

2. Выполняем клонирование созданного репозитория.

```
Microsoft Windows [Version 10.0.19042.1237]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Катя>git clone https://github.com/ekaterina533/lab2.2
Cloning into 'lab2.2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

C:\Users\Катя>cd lab2.2

C:\Users\Катя\lab2.2>
```

3. Организация репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\Катя\lab2.2>git flow init

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

Which branch should be used for integration of the "next release"?
- develop
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/Катя/lab2.2/.git/hooks]

C:\Users\Катя\lab2.2>git branch
  develop
* main

C:\Users\Катя\lab2.2>git checkout develop
Switched to branch 'develop'

C:\Users\Катя\lab2.2>git flow feature start feature_branch
Switched to a new branch 'feature/feature_branch'

Summary of actions:
- A new branch 'feature/feature_branch' was created, based on 'develop'
- You are now on branch 'feature/feature_branch'

Now, start committing on your feature. When done, use:

    git flow feature finish feature_branch
```

7. Проработка примера лабораторной работы. Создание для него отдельного модуль языка Python. Зафиксируем изменения в репозитории.

```
#!/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}")
```

Рис. 1.1 – Код программы примера.

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

Рис. 1.2 – Результат работы программы.

8. Решим задачу: подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == "__main__":
    s = input("Введите слово, или предложение:")
    c = 0
    a = {'y', 'e', 'ы', 'а', 'о', 'э', 'я', 'и', 'ю'}
    for n in s:
        if n in a:
            c = c + 1
    print("Количество гласных равно:", c)
```

Рис. 2.1 – Код программы.

```
Введите слово, или предложение:молоко
Количество гласных равно: 3
```

Рис. 2.2 – Результат работы программы.

9. Зафиксируем сделанные изменения в репозитории.

```
C:\Users\Катя\lab2.2>git add .
C:\Users\Катя\lab2.2>git commit -m "creating the 1задание.py"
[feature/feature_branch 07d37b0] creating the 1задание.py
1 file changed, 9 insertions(+)
create mode 100644 "1\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
```

10. Решим задачу: определите общие символы в двух строках, введенных с клавиатуры.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == "__main__":
    a=input("Введите слово:")
    b=input("Введите ещё одно слово:")
    c = set(a).intersection(set(b))
    print(c)
```

Рис. 3.1 – Код программы.

```
Введите слово:молоко
Введите ещё одно слово:солнце
{'о', 'л'}
```

Рис. 3.2 – Результат работы программы.

## 11. Зафиксируем сделанные изменения в репозитории.

```
C:\Users\Катя>cd lab2.2
C:\Users\Катя\lab2.2>git add .
C:\Users\Катя\lab2.2>git commit -m "update"
[feature/feature_branch 4bc7d4c] update
2 files changed, 10 insertions(+), 5 deletions(-)
create mode 100644 "2\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
```

## 13. Выполняем слияние ветки для разработки с веткой master/main.

Для того что бы выполнить слияние ветки develop и main, для начала нужно слить ветки feature и develop.

```
C:\Users\Катя\lab2.2>git flow feature finish feature_branch
Switched to branch 'develop'
Merge made by the 'recursive' strategy.
...320\260\320\264\320\260\320\275\320\270\320\265.py" | 10 ++++++++
...320\260\320\264\320\260\320\275\320\270\320\265.py" | 7 ++++++
...320\260\320\264\320\260\320\275\320\270\320\265.py" | 15 ++++++++
"320\277\321\200\320\270\320\274\320\265\321\200.py" | 18 ++++++++
4 files changed, 50 insertions(+)
create mode 100644 "1\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
create mode 100644 "2\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
create mode 100644 "\320\270\320\275\320\264.\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
create mode 100644 "\320\277\321\200\320\270\320\274\320\265\321\200.py"
Deleted branch feature/feature_branch (was ca32fc3).

Summary of actions:
- The feature branch 'feature/feature_branch' was merged into 'develop'
- Feature branch 'feature/feature_branch' has been locally deleted
- You are now on branch 'develop'
```

Рис. 4.1 – Сливание веток feature и develop.

```
C:\Users\Катя\lab2.2>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\Катя\lab2.2>git merge develop
Updating 5bd1530..bf19803
Fast-forward
...320\260\320\264\320\260\320\275\320\270\320\265.py" | 10 ++++++++
...320\260\320\264\320\260\320\275\320\270\320\265.py" | 7 ++++++
...320\260\320\264\320\260\320\275\320\270\320\265.py" | 15 ++++++++
"320\277\321\200\320\270\320\274\320\265\321\200.py" | 18 ++++++++
4 files changed, 50 insertions(+)
create mode 100644 "1\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
create mode 100644 "2\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
create mode 100644 "\320\270\320\275\320\264.\320\267\320\260\320\264\320\260\320\275\320\270\320\265.py"
create mode 100644 "\320\277\321\200\320\270\320\274\320\265\321\200.py"
```

Рис. 4.2 – Сливание веток develop и main.



14. Отправляем сделанные изменения на сервер GitHub.

```
C:\Users\Катя\lab2.2>git push
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 8 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (17/17), 2.60 KiB | 887.00 KiB/s, done.
Total 17 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/ekaterina533/lab2.2
5bd1530..bf19803 main -> main
```

15. Решаем индивидуальное задание вариант 18:

18 вариант

$$A = \{b, c, g, I, w\}; \quad B = \{e, g, h, q, w\}; \quad C = \{c, d, k, l, y\}; \quad D = \{a, g, h, u, v, z\};$$
$$X = (A \cap C) \cup B; \quad Y = (\bar{A} \cap D) \cup (C/B).$$

```
# индивидуальное задание вариант 18
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == "__main__":
    u = set("abcdefghijklmnopqrstuvwxyz")
    a = {"b", "c", "g", "I", "w"}
    b = {"e", "g", "h", "q", "w"}
    c = {"c", "d", "k", "l", "y"}
    d = {"a", "g", "h", "u", "v", "z"}
    x = (a.intersection(c)).union(b)
    print(f"x = {x}")
    bn = u.difference(a)
    cn = bn.intersection(d)
    y = cn.union(c.difference(b))
    print(f"y = {y}")
```

Рис. 5.1 – Код программы.

```
x = {'c', 'w', 'h', 'q', 'e', 'g'}
y = {'c', 'k', 'd', 'z', 'h', 'l', 'v', 'u', 'a', 'y'}
```

Рис. 5.2 – Результат работы 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

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?

Проверка, есть ли данное значение в множестве. Для этого используется `in`. Наоборот, проверка отсутствия. Используется `not in`.

4. Как выполнить перебор элементов множества?

С помощью команды `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. `a = {i for i in [1, 2, 0, 1, 3, 2]} print(a)`.

6. Как выполнить добавление элемента во множество?

Чтобы внести новые значения, потребуется вызывать метод `add`. Аргументом в данном случае будет добавляемый элемент последовательности `a.add(4)`.

7. Как выполнить удаление одного или всех элементов множества?

`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.

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 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.

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.

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

10. Каково назначение множеств frozenset?

Множество, содержимое которого не поддается изменению имеет тип frozenset. Значения из этого набора нельзя удалить, как и добавить новые.

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

Для преобразования множества в строку используется конкатенация текстовых значений, которую обеспечивает функция join.

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, получающий в качестве аргумента множество a.



Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с множествами на языке программирования Python.