

Лабораторная работа №2

Выполнил Эсеналиев Арсен

ИВТ-б-о-21-1

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


1. Создам общедоступный репозиторий на GitHub с MIT

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

Repository name *


 Arsen445 ▾

/


LB2.7 ✓

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

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:
Skip this step if you're importing an existing repository.


☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)


Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

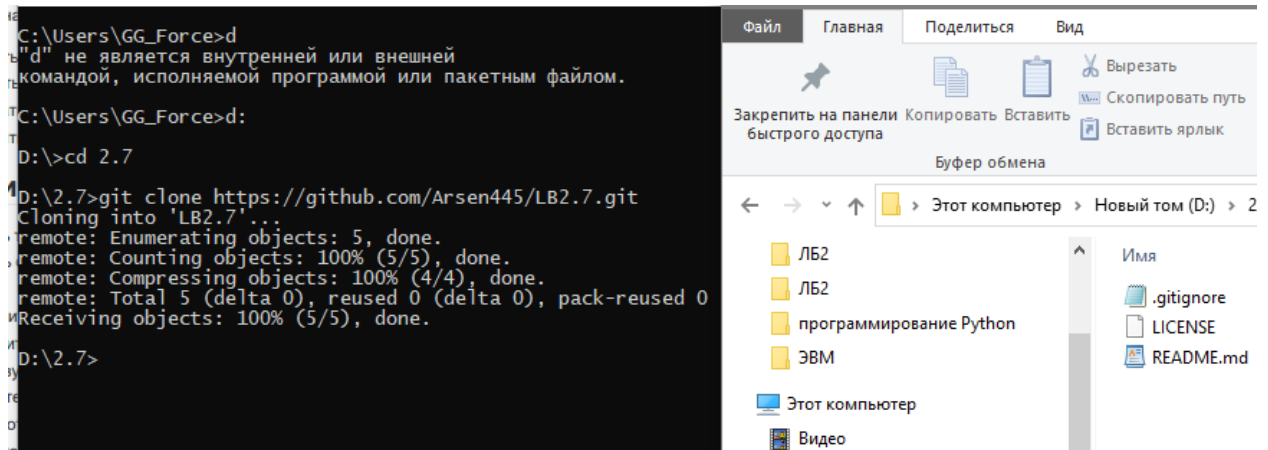
License: MIT License ▾

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

2. Выполнил клонирование созданного репозитория.



3. Организовал свой репозиторий в соответствии с моделью ветвления git-flow. (Перешел с главной main на develop)

```
D:\2.7\LB2.7>git flow init

which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Fatal: Local branch '' does not exist.

D:\2.7\LB2.7>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? [D:/2.7/LB2.7/.git/hooks]
```

4. Создал проект PyCharm в папке репозитория.



5. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```

# Created by https://www.toptal.com/developers/gitignore/api/windows,pycharm+all,python
# Edit at https://www.toptal.com/developers/gitignore?templates=windows,pycharm+all,python

### PyCharm+all ###
# 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
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

# Gradle
.idea/**/gradle.xml
.idea/**/libraries

# Gradle and Maven with auto-import
# When using Gradle or Maven with auto-import, you should exclude module files,
# since they will be recreated, and may cause churn. Uncomment if using
# auto-import.
# .idea/artifacts
# .idea/compiler.xml
# .idea/jarRepositories.xml
# .idea/modules.xml
# .idea/*.iml
# .idea/modules
# * .iml

```

6. Проработал пример лабораторной работы.

```

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

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

    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}")

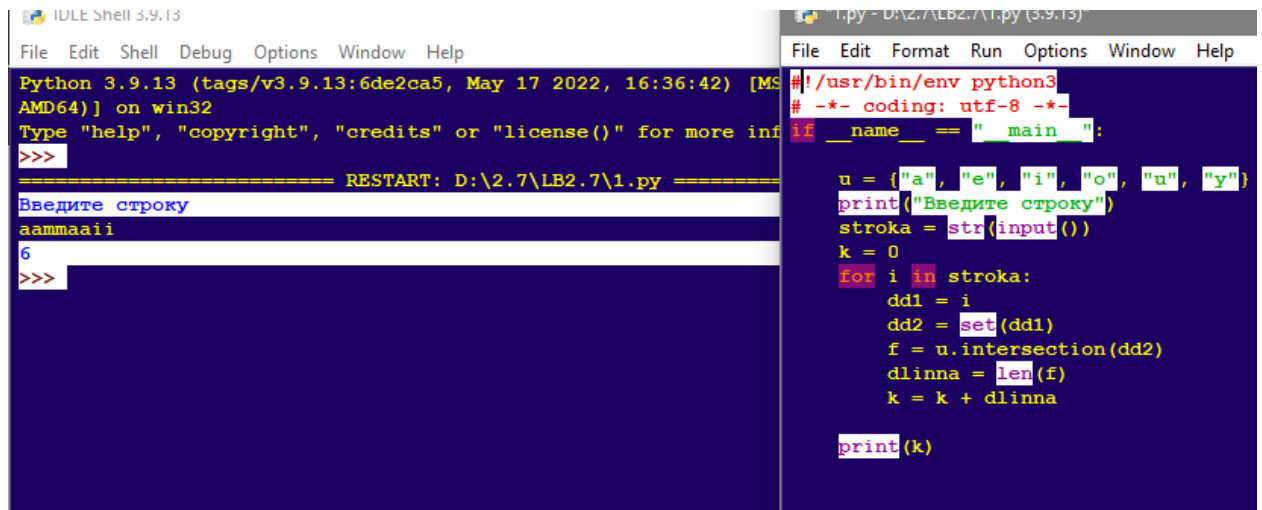
```

```

Traceback (most recent call last):
  File "C:/Python39/ew.py", line 2, in <
    for i in {a}:
TypeError: unhashable type: 'set'
>>>
===== RESTART: C:/E
Traceback (most recent call last):
  File "C:/Python39/ew.py", line 2, in <
    for i in {a}:
TypeError: unhashable type: 'set'
>>>
===== RESTART: C:/E
0
1
2
3
>>>
===== RESTART: C:/E
x = {'j', 'd', 'k', 'o', 'e'}
y = {'o', 'h', 'f', 'c', 'v', 'y', 'g'}
>>>

```

7. Решите задачу: подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.



```
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSI64] on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
Введите строку
aammaaii
6
>>>

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == "__main__":
    u = {"a", "e", "i", "o", "u", "y"}
    print("Введите строку")
    stroka = str(input())
    k = 0
    for i in stroka:
        dd1 = i
        dd2 = set(dd1)
        f = u.intersection(dd2)
        dlinna = len(f)
        k = k + dlinna

    print(k)
```

8. Зафиксируйте сделанные изменения в репозитории.

```
D:\REP2_6\LB2.6>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   1.png
    new file:   1zad.py
    new file:   2zad.py

D:\REP2_6\LB2.6>git commit -m "new"
[develop 981ed95] new
 3 files changed, 10 insertions(+)
 create mode 100644 1.png
 create mode 100644 1zad.py
 create mode 100644 2zad.py

D:\REP2_6\LB2.6>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 14.62 KiB | 14.62 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Arsen445/LB2.6.git
   d6888d4..981ed95  develop -> develop

D:\REP2_6\LB2.6>
```


Контрольные вопросы:

1. Что такое множества в языке Python?

Множеством в языке программирования Python называется неупорядоченная совокупность уникальных значений.

2. Как осуществляется создание множеств в Python?

присвоив переменной последовательность значений, выделив их фигурными скобками.

```
a = {1, 2, 0, 1, 3, 2}
print(a)
```

3. Как проверить присутствие/отсутствие элемента в множестве?

Операцией in

- 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?

```
for a in {0, 1, 2}:
    print(a)

0
1
2
```

5. Что такое set comprehension?

Генератор последовательностей из заданных данных

```
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.

```
a = {0, 1, 2, 3}  
a.add(4)  
print(a)
```

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.

```
a = {0, 1, 2, 3}  
a.remove(3)  
print(a)
```

Чтобы не удалять каждый элемент отдельно, используется метод clear

```
a = {0, 1, 2, 3}  
a.clear()  
print(a)
```

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

```
a = {0, 1, 2, 3}  
b = {4, 3, 2, 1}  
c = a.union(b)  
print(c)
```


Добавление

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, как это показано в следующем примере.

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
a.update(b)
print(a)

{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.

```
a = {0, 1, 2, 3}
b = {4, 3, 2, 1}
c = a.intersection(b)
print(c)

{1, 2, 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?

Определение подмножества

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 ?

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

Преобразование множеств

Иногда возникает необходимость представления уже готовой последовательности значений в качестве совсем другого типа данных. Возможности языка позволяют конвертировать любое множество в строку, словарь или список при помощи стандартных функций.

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'>
```