

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего
образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

**Кафедра
инфокоммуникаций
Институт цифрового
развития**

ОТЧЁТ

по лабораторной работе №2.7

Дисциплина: «Основы программной инженерии»

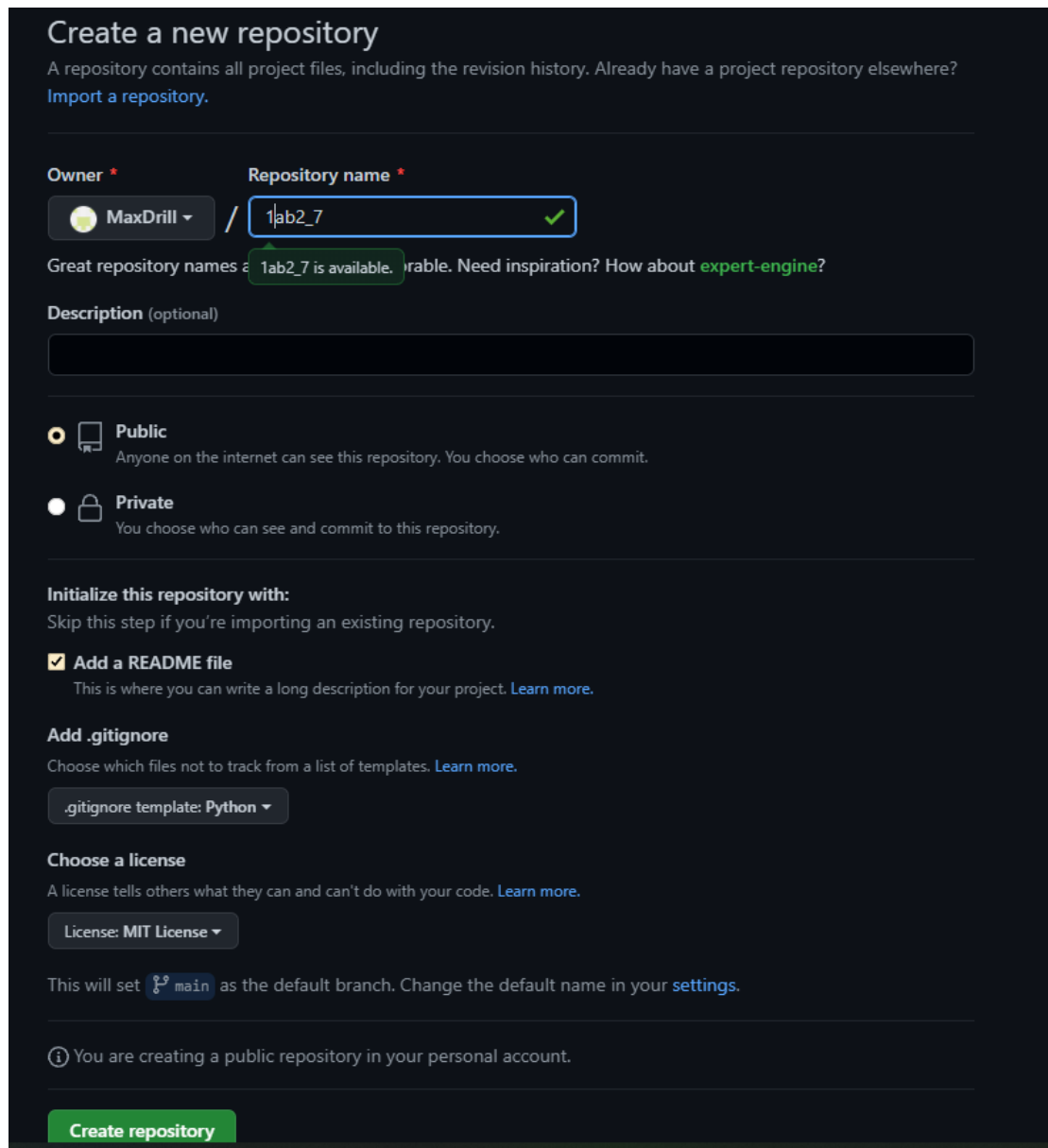
Тема: «Работа с множествами в языке Python»

Выполнил:
студент 2 курса
группы Пиж-б-о-21-1
Коныжев Максим
Викторович

Ставрополь 2022

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

1. Был создан репозиторий в Github в который были добавлены правила gitignore для работы IDE PyCharm, была выбрана лицензия MIT, сам репозиторий был клонирован на локальный сервер и был организован в соответствии с моделью ветвления git-flow.



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 ***

MaxDrill / 1ab2_7 ✓

Great repository names are short, simple, and easy to remember. Need inspiration? How about [expert-engine?](#)

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.

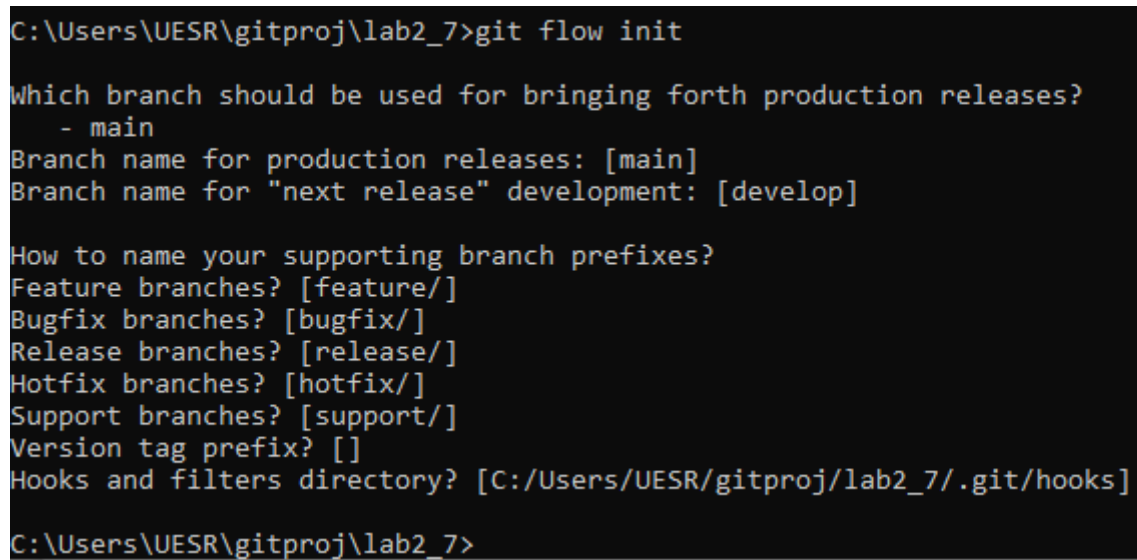
Create repository

Рисунок 1 – Создание репозитория

A screenshot of a code editor showing a .gitignore file. The file contains 149 lines of text, with 121 lines of code and 2.12 KB of size. The content lists various files and directories to be ignored, including IntelliJ project files, Python templates, C extensions, and distribution/packaging files. The editor has a dark theme and a sidebar on the right with buttons for 'Raw', 'Blame', and a search icon.

```
149 lines (121 sloc) | 2.12 KB
1  ### Example user template template
2  ### Example user template
3
4  # IntelliJ project files
5  .idea
6  *.iml
7  out
8  gen
9  ### Python template
10 # Byte-compiled / optimized / DLL files
11 __pycache__/
12 *.py[co]
13 *.py.class
14
15 # C extensions
16 *.so
17
18 # Distribution / packaging
19 .Python
20 build/
21 develop-eggs/
22 dist/
23 download/
```

Рисунок 2 – Изменение gitignore

A screenshot of a terminal window showing the output of the 'git flow init' command. The terminal displays a series of prompts and user input for configuring the git-flow workflow. The prompts include questions about production releases, supporting branch prefixes, and the hooks directory. The user has provided answers for each prompt, such as 'main' for production releases and 'feature/' for feature branches. The terminal shows the command being executed and the resulting configuration.

```
C:\Users\UESR\gitproj\lab2_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? [C:/Users/UESR/gitproj/lab2_7/.git/hooks]

C:\Users\UESR\gitproj\lab2_7>
```

Рисунок 3 – Организация репозитория в соответствии с моделью ветвления
git-flow

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == "__main__":
4      # Определим универсальное множество
5      u = set("abcdefghijklmnopqrstuvwxyz")
6      a = {"b", "c", "h", "o"}
7      b = {"d", "f", "g", "o", "v", "y"}
8      c = {"d", "e", "j", "k"}
9      d = {"a", "b", "f", "g"}
10
11     x = (a.intersection(b)).union(c)
12     print(f"x = {x}")
13
14     # Найдем дополнения множеств
15     bn = u.difference(b)
16     cn = u.difference(c)
17
18     y = (a.difference(d)).union(cn.difference(bn))
19     print(f"y = {y}")
20
21     if __name__ == "__main__":
22         lab10_idz x
23         C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\python.exe C:/Users/UESR/Desktop/
24         X = {'l', 'f', 'h', 'b', 'm', 'u', 'g'}
25         Y = {'f', 'e', 'o', 'm'}
26
27     Process finished with exit code 0
```

Рисунок 5 – Результат работы программы

3. Выполнил задания.

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

Код программы:

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

if __name__ == "__main__":
    # Все согл
    a = set("bcd fghilkmnpqrstvwxyz")
    mark = set(".?,:;!;'`\" ")

    # Ввод строки
    x = set(input("Введите строку: ").lower())

    # Находим все гласные в строке
    gl = x.difference(a)
    gl = gl.difference(mark)

    count = len(gl)

    print(f"Все гласные буквы из введенной строки: = {gl}")
    print(f"Кол-во гласных букв: = {count}")
```

```

C:\Users\DESK\Desktop\2.1\proj\venv\Scripts\python.exe C:/Users/DESK/Desktop/2.1/proj/venv/Scripts/python.exe
Введите строку: qwertyasd
Все гласные буквы из введенной строки: = {'e', 'y', 'a'}
Кол-во гласных букв: = 3

Process finished with exit code 0

```

Рисунок 6 – Результат работы программы

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

Код программы:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == "__main__":
    # Ввод строки
    x1 = set(input("Введите первую строку: ").lower())
    x2 = set(input("Введите вторую строку: ").lower())

    mark = set(".,!;'\`\" ")

    gl = x1.intersection(x2)
    gl = gl.difference(mark)

    count = len(gl)

    print(f"Общие символы: = {gl}")
    print(f"Кол-во общих символов: = {count}")

```

```

C:\Users\DESK\Desktop\2.1\proj\venv\Scripts\python.exe
Введите первую строку: hello! Nice day
Введите вторую строку: Hello. Good
Общие символы: = {'l', 'o', 'd', 'h', 'e'}
Кол-во общих символов: = 5

Process finished with exit code 0

```

Рисунок 7 – Результат работы программы

Индивидуальное задание

Вариант 7

$$A = \{b, f, g, m, o\}; \quad B = \{b, g, h, l, u\}; \quad C = \{e, f, m\}; \quad D = \{e, g, l, p, q, u, v\};$$

$$7. \quad X = (A \cap C) \cup B; \quad Y = (A \cap \bar{B}) \cup (C/D).$$

$$A \cap C = \{f, m\}$$

$$X = (A \cap C) \cup B = \{f, m, b, g, h, l, u\}$$

$$A = \{n, y, w, p, i, x, t, z, l, a, d, v, r, h, c, j, u, q, s, k, e\}$$

$$B = \{r, n, o, p, x, z, i, a, t, m, d, f, y, k, e, j, w, s, c, v, q\}$$

$$C = \{r, n, o, p, h, x, b, z, i, a, t, d, y, k, u, j, w, l, s, c, v, q, g\}$$

$$A \cap B = \{f, o, m\}$$

$$C/D = \{f, m, e\}$$

$$Y = (A \cap B) \cup (C/D) = \{o, m, f, e\}$$

```
# -*- coding: utf-8 -*-
if __name__ == "__main__":

    # Universe
    u = set("abcdefghijklmnopqrstuvwxyz")

    # Set Data
    a = {"b", "f", "g", "m", "o"}
    b = {"b", "g", "h", "l", "u"}
    c = {"e", "f", "m"}
    d = {"e", "g", "l", "p", "q", "u", "v"}

    # Definition X
    x = b.union(a.intersection(c))
    print(f'X = {x}')

    # Inverses for a b and c
    ne_a = u.difference(a)
    ne_b = u.difference(b)
    ne_c = u.difference(c)

    # Definition Y
    y = (a.intersection(ne_b)).union(c.difference(b))
    print(f'Y = {y}')
```

```
C:\Users\UESR\Desktop\2.1\proj\venv\Scripts\python.exe
X = {'g', 'f', 'h', 'm', 'b', 'l', 'u'}
Y = {'e', 'm', 'o', 'f'}

Process finished with exit code 0
```

Рисунок 8 – Результат работы программы

```

C:\Users\UESR\gitproj\lab2_6>git add .

C:\Users\UESR\gitproj\lab2_6>git commit -m "Add proj"
[develop aa04d41] Add proj
4 files changed, 201 insertions(+)
create mode 100644 proj/idz9.py
create mode 100644 proj/lab9_obsh.py
create mode 100644 proj/num1_lr9.py
create mode 100644 proj/num2_lr9.py

C:\Users\UESR\gitproj\lab2_6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\UESR\gitproj\lab2_6>

```

Рисунок 9 – Коммит изменений

```

C:\Users\UESR\gitproj\lab2_7>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\UESR\gitproj\lab2_7>git merge develop
Updating c0c1831..7ca0101
Fast-forward
 proj/lab10_idz.py | 26 ++++++
 proj/lab10_num1.py | 19 ++++++
 proj/lab10_num2.py | 18 ++++++
 proj/lab10_num3.py | 16 ++++++
4 files changed, 79 insertions(+)
create mode 100644 proj/lab10_idz.py
create mode 100644 proj/lab10_num1.py
create mode 100644 proj/lab10_num2.py
create mode 100644 proj/lab10_num3.py

C:\Users\UESR\gitproj\lab2_7>

```

Рисунок 10 – Слияние веток main и develop

Контрольные вопросы:

1. Что такое словари в языке Python?

Словари в Python – это изменяемые отображения ссылок на объекты, доступные по ключу.

2. Может ли функция len() быть использована при работе со словарями?

Функция `len()` возвращает длину (количество элементов) в объекте. Аргумент может быть последовательностью, такой как строка, байты, кортеж, список или диапазон или коллекцией (такой как словарь, множество или неизменяемое множество).

3. Какие методы обхода словарей Вам известны?

Самый очевидный вариант обхода словаря — это попытаться напрямую запустить цикл `for` по объекту словаря, так же как мы делаем это со списками, кортежами, строками и любыми другими итерируемыми объектами. `for something in currencies: print(something)`

4. Какими способами можно получить значения из словаря по ключу?

С помощью метода `.get()`

5. Какими способами можно установить значение в словаре по ключу?

С помощью функции `dict.update()`

6. Что такое словарь включений?

Словарь включений аналогичен списковым включениям, за исключением того, что он создаёт объект словаря вместо списка.

7. Самостоятельно изучите возможности функции `zip()` приведите примеры ее использования.

Функция `zip()` в Python создает итератор, который объединяет элементы из нескольких источников данных. Эта функция работает со списками, кортежами, множествами и словарями для создания списков или кортежей, включающих все эти данные. Предположим, что есть список имен и номером сотрудников, и их нужно объединить в массив кортежей. Для этого можно использовать функцию `zip()`. Вот пример программы, которая делает именно это:

```
employee_numbers = [2, 9, 18, 28]
employee_names = ["Дима", "Марина", "Андрей", "Никита"]
zipped_values = zip(employee_names, employee_numbers)
zipped_list = list(zipped_values)
print(zipped_list)
```


Функция `zip` возвращает следующее:

```
[('Дима', 2), ('Марина', 9), ('Андрей', 18), ('Никита', 28)]
```

8. Самостоятельно изучите возможности модуля `datetime`. Каким функционалом по работе с датой и временем обладает этот модуль? `Datetime` — важный элемент любой программы, написанной на Python. Этот модуль позволяет управлять датами и временем, представляя их в таком виде, в котором пользователи смогут их понимать.

`datetime` включает различные компоненты. Так, он состоит из объектов следующих типов:

- 🕒 `date` — хранит дату
- 🕒 `time` — хранит время
- 🕒 `datetime` — хранит дату и время

Как получить текущие дату и время?

```
import datetime  
  
dt_now = datetime.datetime.now()  
print(dt_now)
```

Результат:

```
2022-09-11 15:43:32.249588
```

Получить текущую дату:

```
from datetime import date  
  
current_date = date.today()  
print(current_date)
```

Результат:

```
2022-09-11
```

Получить текущее время:

```
import datetime
```

```
current_date_time = datetime.datetime.now()  
current_time = current_date_time.time()  
print(current_time)
```

Результат:

15:51:05.627643