

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

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

**Кафедра инфокоммуникаций
«Работа с кортежами в языке Python»**

**Отчет по лабораторной работе № 2.7
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

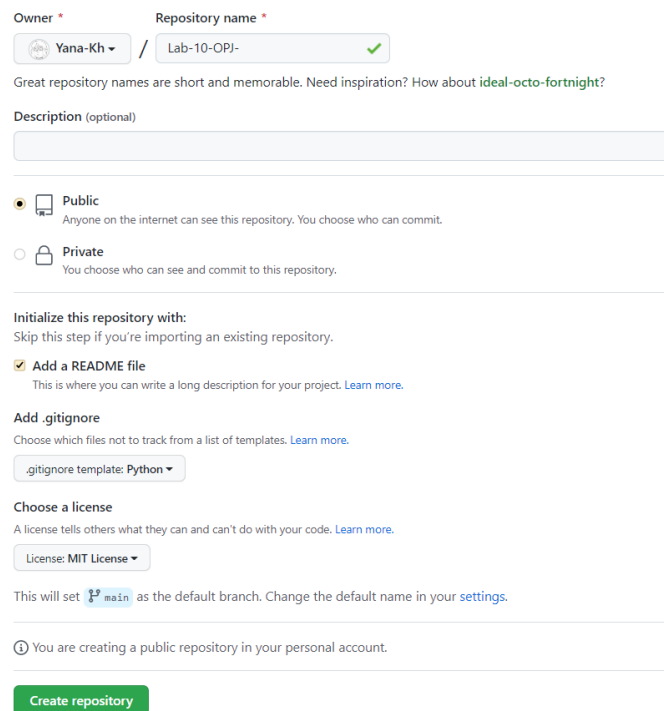
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

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

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



Owner * Repository name *

Yana-Kh / Lab-10-OPJ ✓

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

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

3. Выполните клонирование созданного репозитория.

```
C:\Users\антон>cd C:\Users\антон\Desktop\Git

C:\Users\антон\Desktop\Git>git clone https://github.com/Yana-Kh/Lab-10-OPJ.git
Cloning into 'Lab-10-OPJ'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\антон\Desktop\Git>cd C:\Users\антон\Desktop\Git\Lab-10-OPJ
C:\Users\антон\Desktop\Git\Lab-10-OPJ>
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
C:\Users\антон\Desktop\Git\Lab-10-OPJ>git add .

C:\Users\антон\Desktop\Git\Lab-10-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\антон\Desktop\Git\Lab-10-OPJ>_
```

Рисунок 3 – Дополнение файла .gitignore

5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\антон\Desktop\Git\Lab-10-OPJ>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/антон/Desktop/Git/Lab-10-OPJ/.git/hooks]
```

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

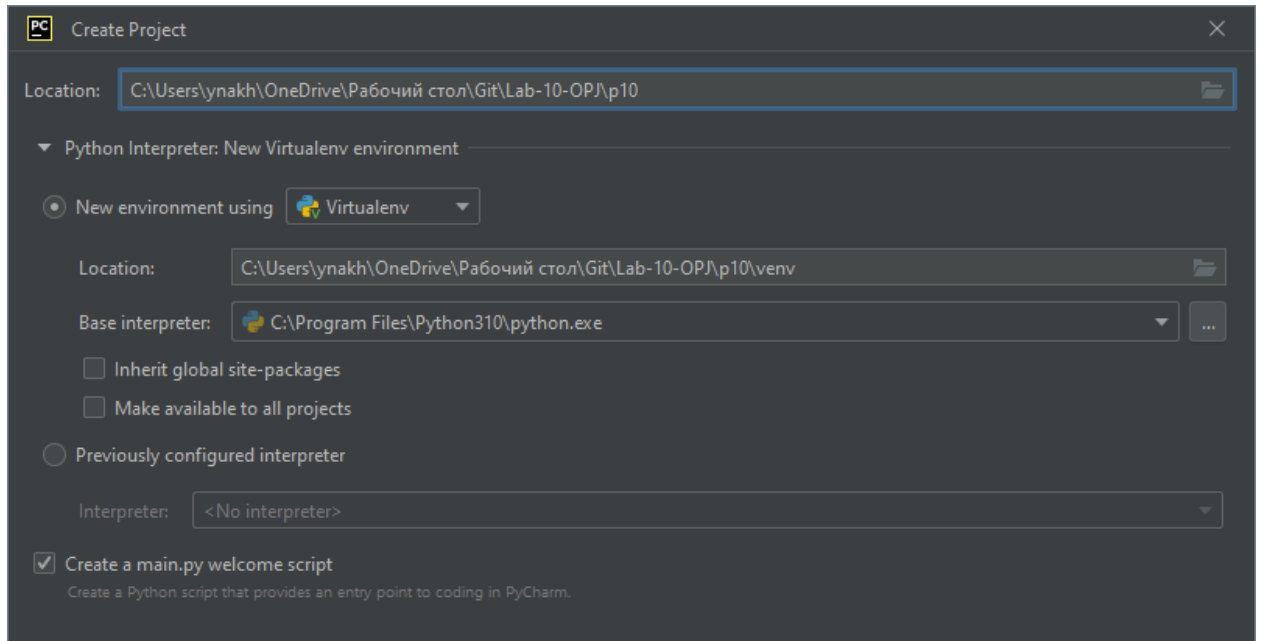


Рисунок 5 – Создание проекта PyCharm в папке репозитория

7. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

Пример 1. Определить результат выполнения операций над множествами. Считать элементы множества строками.

$$A = \{b, c, h, o\}; \quad B = \{d, f, g, o, v, y\}; \quad C = \{d, e, j, k\}; \quad D = \{a, b, f, g\};$$

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

Код:

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

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

Рисунок 6 – Результат работы программы

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

Код:

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

if __name__ == "__main__":
    s = input("Enter line: ")
    vowels = {'e', 'y', 'u', 'i', 'o', 'a'}
    n = 0
    for i in s:
        if i in vowels:
            n += 1
    print(f"Number of vowels per line: {n}")
```

```
Enter line: klpweihui
Number of vowels per line: 4
```

Рисунок 7 – Результат работы программы

9. Зафиксируйте сделанные изменения в репозитории.

```
Командная строка

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-0PJ>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   p10/ex1.py
        new file:   p10/ex8.py

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   p10/ex1.py
        modified:   p10/ex8.py

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-0PJ>git add .
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-0PJ>git commit -m"ex"
[develop 0f69ecf] ex
 2 files changed, 32 insertions(+)
 create mode 100644 p10/ex1.py
 create mode 100644 p10/ex8.py

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-0PJ>git push
fatal: The current branch develop has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin develop

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetUpRemote' in 'git help config'.
```

Рисунок 8 – Фиксация изменений в репозитории

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

Код:

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

if __name__ == "__main__":
    s1 = set(input("Enter line №1: "))
    s2 = set(input("Enter line №2: "))
    line = ', '.join(s1.intersection(s2))
    print(f"common characters in two lines: {line}")
```

```
Enter line №1: Hello
Enter line №2: House
common characters in two lines: {'e', 'H', 'o'}
```

Рисунок 9 – Результат работы программы

11. Зафиксируйте сделанные изменения в репозитории.

```

Командная строка
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:   p10/ex10.py

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   p10/ex10.py

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-OPJ>git add .
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-OPJ>git commit -m"ex"
[develop 55e6540] ex
 1 file changed, 7 insertions(+)
 create mode 100644 p10/ex10.py
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-OPJ>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 504 bytes | 504.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Yana-Kh/Lab-10-OPJ.git
   0f69ecf..55e6540  develop -> develop
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-10-OPJ>

```

Рисунок 10 – Фиксация изменений в репозитории

12. Выполните индивидуальные задания, согласно своему варианту.

Вариант 32 (3)

3.

$$\begin{aligned}
 A &= \{a, h, m, o, r\}; & B &= \{j, k, o, u, y\}; & C &= \{g, h, j\}; & D &= \{g, j, q\}; \\
 X &= (A \cap C) \cup (D \cap B); & Y &= (A \cap B) \cup (D/C).
 \end{aligned}
 \tag{4}$$

Код:

```

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

if __name__ == "__main__":
    a = {"a", "h", "m", "o", "r"}
    b = {"j", "k", "o", "u", "y"}
    c = {"g", "h", "j"}
    d = {"g", "j", "q"}

    x = (a.intersection(c)).union(d.intersection(b))
    y = (a.intersection(b).union(d.difference(c)))
    print(f"x = {x}, y = {y}")

```

```

x = {'j', 'h'}, y = {'o', 'q'}

Process finished with exit code 0

```

Рисунок 11 – Результат работы программы

Вопросы для защиты работы

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?

Присвоить переменной последовательность значений, выделив их фигурными скобками: `a = {1, 3, 5, 9}`

Вызвать set: `a = set('stroka')`

3. Как проверить присутствие/отсутствие элемента в множестве?

Для этого можно воспользоваться «in» или «not in»:

```
a = {1, 3, 5, 9}
```

```
print(2 in a) // false
```

4. Как выполнить перебор элементов множества?

С помощью цикла for:

```
for i in {1, 2, 3}
```

```
    print(i) //1 2 3
```

5. Что такое set comprehension?

Это генератор, позволяющий заполнять списки, а также другие наборы данных с учетом неких условий, например:

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

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

6. Как выполнить добавление элемента во множество?

Необходимо использовать метод add. Например:

```
a = {1, 3, 5, 9}
```

```
a.add(6)
```

7. Как выполнить удаление одного или всех элементов множества?

Для удаления элемента можно воспользоваться несколькими функциями:

`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.

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

$a = \{1, 3, 5, 7\}$

$b = \{1, 4, 6, 7, 8\}$

Объединение: `c = a.union(b)`

Пересечение: `c = a.intersection(b)`

Разность: `c = a.difference(b)`

Все команды возвращают множества

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

Для этого можно воспользоваться следующими командами (возвращают `true/false`):

`a.issubset(b)` – подмножество

`a.issuperset(b)` – надмножество

10. Каково назначение множеств `frozenset`?

Это множества, которые нельзя изменить (ни удалить, ни добавить новые)

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

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.

Чтобы получить из множества словарь, следует передать функции `dict` набор из нескольких пар значений, в каждом из которых будет находиться ключ (например, элементы множества – кортежи из двух элементов).

Для получения списка используется вызов `list`, получающий в качестве аргумента множество `a`.