

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №10
дисциплины «Основы программной инженерии»

Выполнила:
Панюкова Ксения Юрьевна
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Я изучила теоретический материал работы

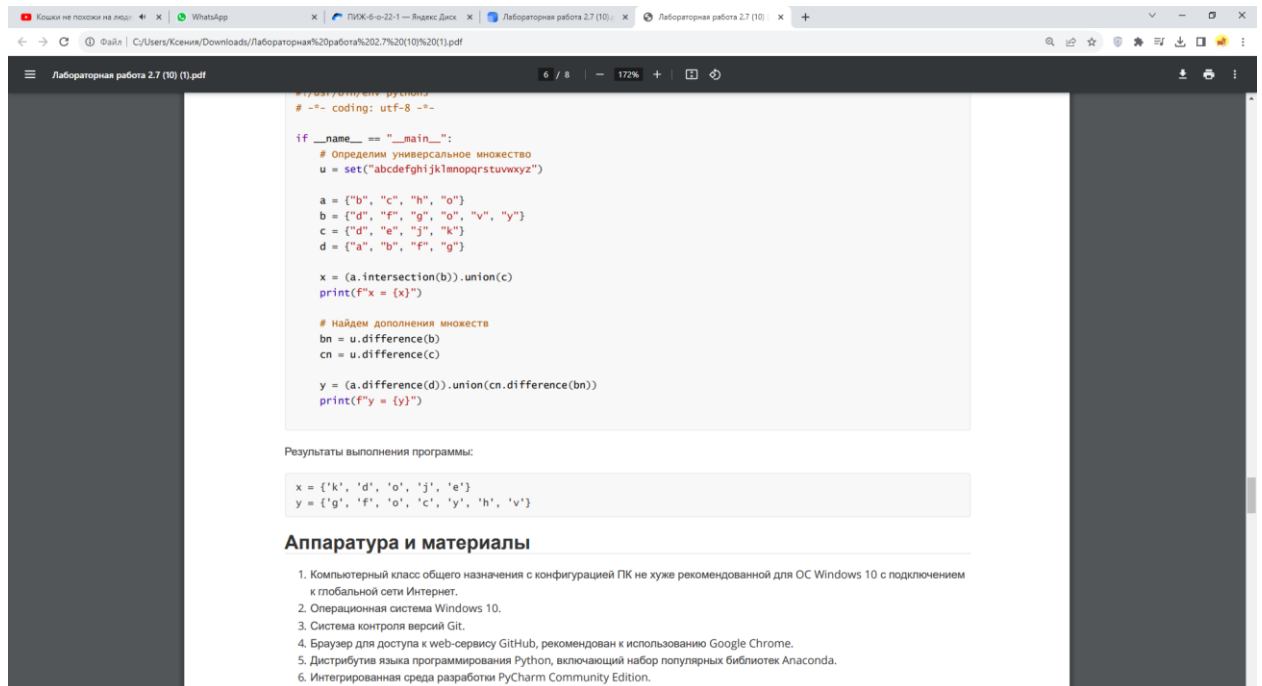



Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python


Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *
 MrFincheck

Repository name *

 PythonsAreNotPoisonous is available.

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

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:

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

Add .gitignore

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

Choose a license

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

 You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

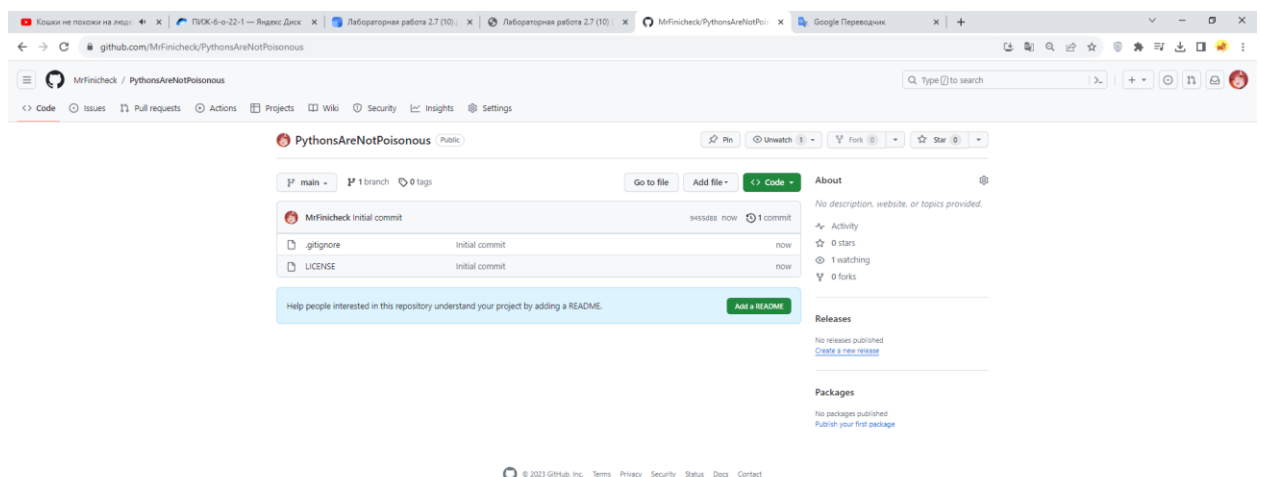


Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

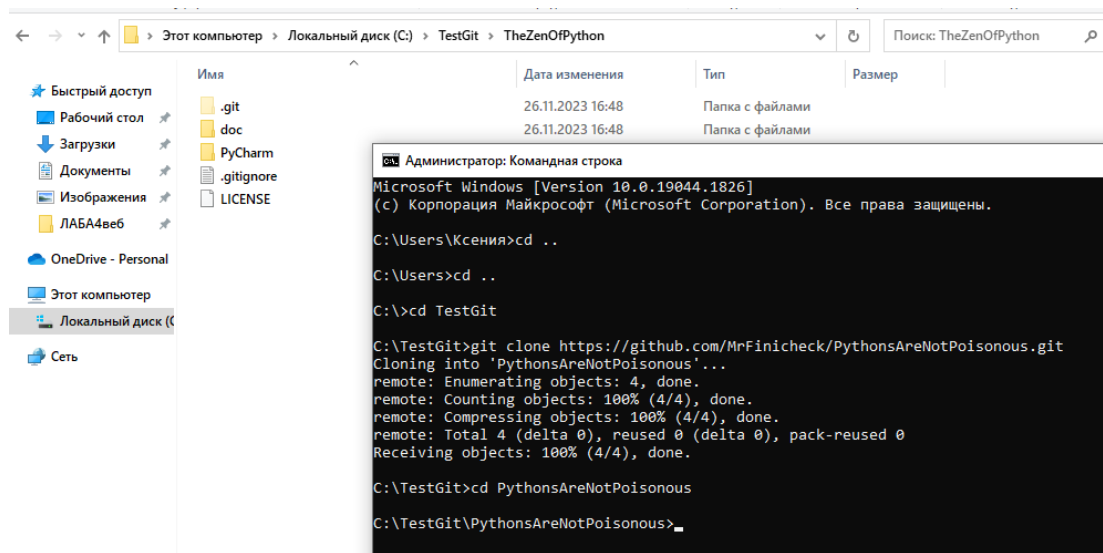


Рисунок 3.1 – Клонирование репозитория на локальный диск

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

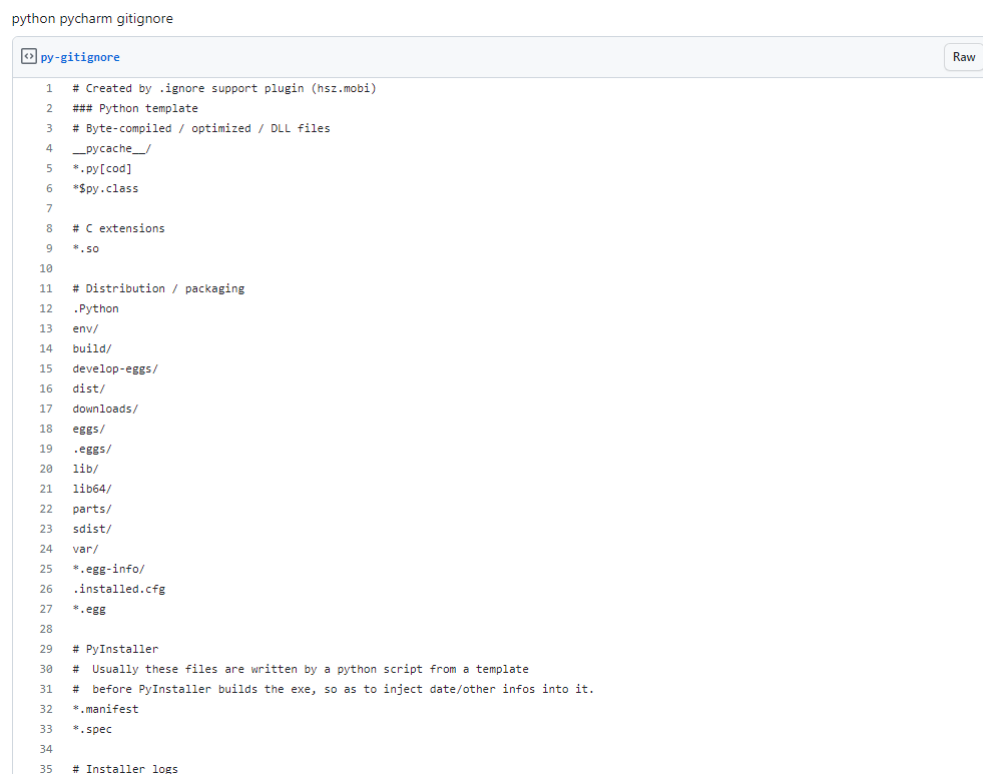


Рисунок 4.1 – .gitignore для IDE PyCharm

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

```

C:\TestGit\PythonsAreNotPoisonous>git branch develop

C:\TestGit\PythonsAreNotPoisonous>git checkout develop
Switched to branch 'develop'

C:\TestGit\PythonsAreNotPoisonous>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MrFinichcheck/PythonsAreNotPoisonous/pull/new/develop
remote:
To https://github.com/MrFinichcheck/PythonsAreNotPoisonous.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

```

Рисунок 5.1 – Создание ветки develop от ветки main

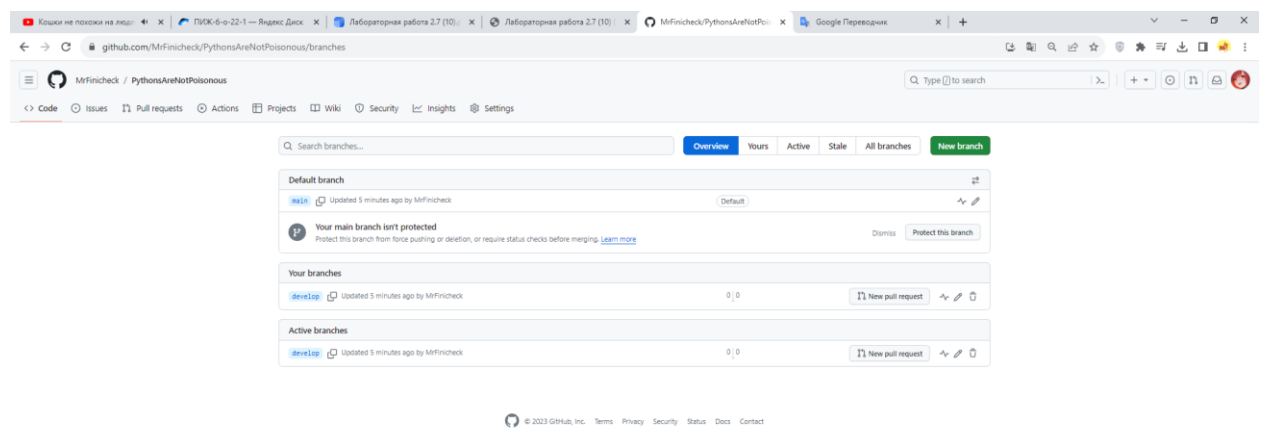


Рисунок 5.2 – Ветка develop на GitHub

6. Создала проект PyCharm в папке репозитория

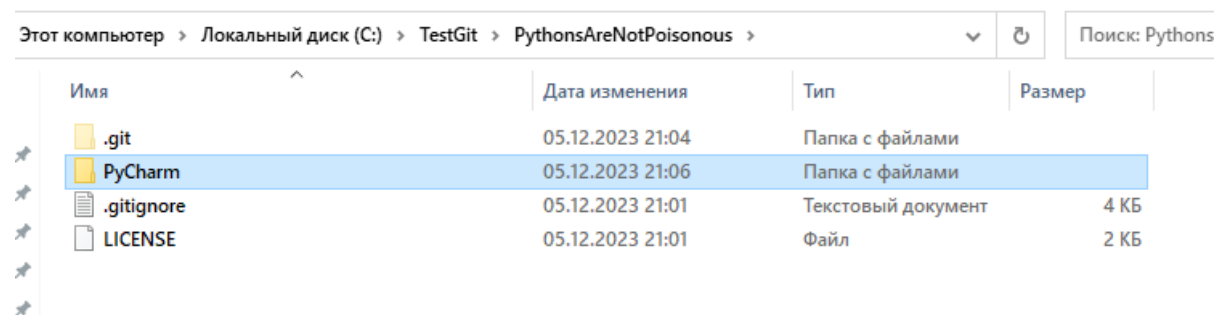


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработала примеры лабораторной работы. Создала для каждого примера отдельный модуль языка Python. Зафиксировала 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}")
```

Рисунок 7.1 – Проработка примера 1

Имя	Дата изменения	Тип	Размер
EXAMPLE.py	05.12.2023 21:15	Исходный файл Pyt...	1 КБ

Рисунок 7.4 – Создание отдельного модуля для примера

```
C:\TestGit\PythonsAreNotPoisonous>git add PyCharm
C:\TestGit\PythonsAreNotPoisonous>git commit -m"adding example"
[develop 2d13c0a] adding example
1 file changed, 21 insertions(+)
create mode 100644 PyCharm/Examples/EXAMPLE.py
```

Рисунок 7.5 – Фиксирование изменений в репозитории

8. Решила задачу: подсчитала количество гласных в строке, введенной с клавиатуры с использованием множеств.

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

if __name__ == "__main__":
    # Ввести строку.
    user_input = input("Введите предложение: ")

    # Указать гласные.
    vowels = {'a', 'e', 'ё', 'и', 'o', 'y',
              'ы', 'э', 'ю', 'я', 'а', 'е',
              'i', 'o', 'u'}

    # Инициализировать счетчик.
    count = 0
    for char in user_input:
        # Посчитать количество гласных.
        if char.lower() in vowels:
            count += 1

    # Вывести количество гласных.
    print(f"Количество гласных: {count}")
```

Рисунок 8.1 – Код программы Task8.py в IDE PyCharm

9. Зафиксировала сделанные изменения в репозитории

```
C:\TestGit\PythonsAreNotPoisonous>git add PyCharm

C:\TestGit\PythonsAreNotPoisonous>git commit -m"add 8 task"
[develop f693097] add 8 task
3 files changed, 30 insertions(+), 1 deletion(-)
create mode 100644 PyCharm/Tasks/Task10.py
create mode 100644 PyCharm/Tasks/Task8.py
```

Рисунок 9.1 – Коммит файлов в репозитории git

10. Решила задачу: определила общие символы в двух строках, введенных с клавиатуры.

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

if __name__ == "__main__":
    # Ввести две строки.
    sentence1 = set(input("Введите первую строку: "))
    sentence2 = set(input("Введите вторую строку: "))

    # Найти общие элементы.
    inter_sentence = sentence1.intersection(sentence2)

    # Вывести общие символы у двух строк.
    print(f"Общие символы у двух строк: {inter_sentence}")
```

Рисунок 10.1 – Код программы Task10.py в IDE PyCharm

11. Зафиксировала сделанные изменения в репозитории.

```
C:\TestGit\PythonsAreNotPoisonous>git add PyCharm
C:\TestGit\PythonsAreNotPoisonous>git commit -m"add task 10"
[develop f1e2e0f] add task 10
2 files changed, 18 insertions(+), 1 deletion(-)
```

Рисунок 11.1 – Коммит файлов в репозитории git

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?

Сделать это можно, просто присвоив переменной последовательность значений, выделив их фигурными скобками, либо используя функцию `set()`.

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?

```
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.

6. Как выполнить добавление элемента во множество?

```
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.

Иногда необходимо полностью убрать все элементы. 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 добавить все элементы из одного множества к другому, необходимо вызывать метод `update` на первом объекте. 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`, как в следующем примере.

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`.