

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с множествами в языке Python»

ОТЧЕТ
по лабораторной работе №10
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

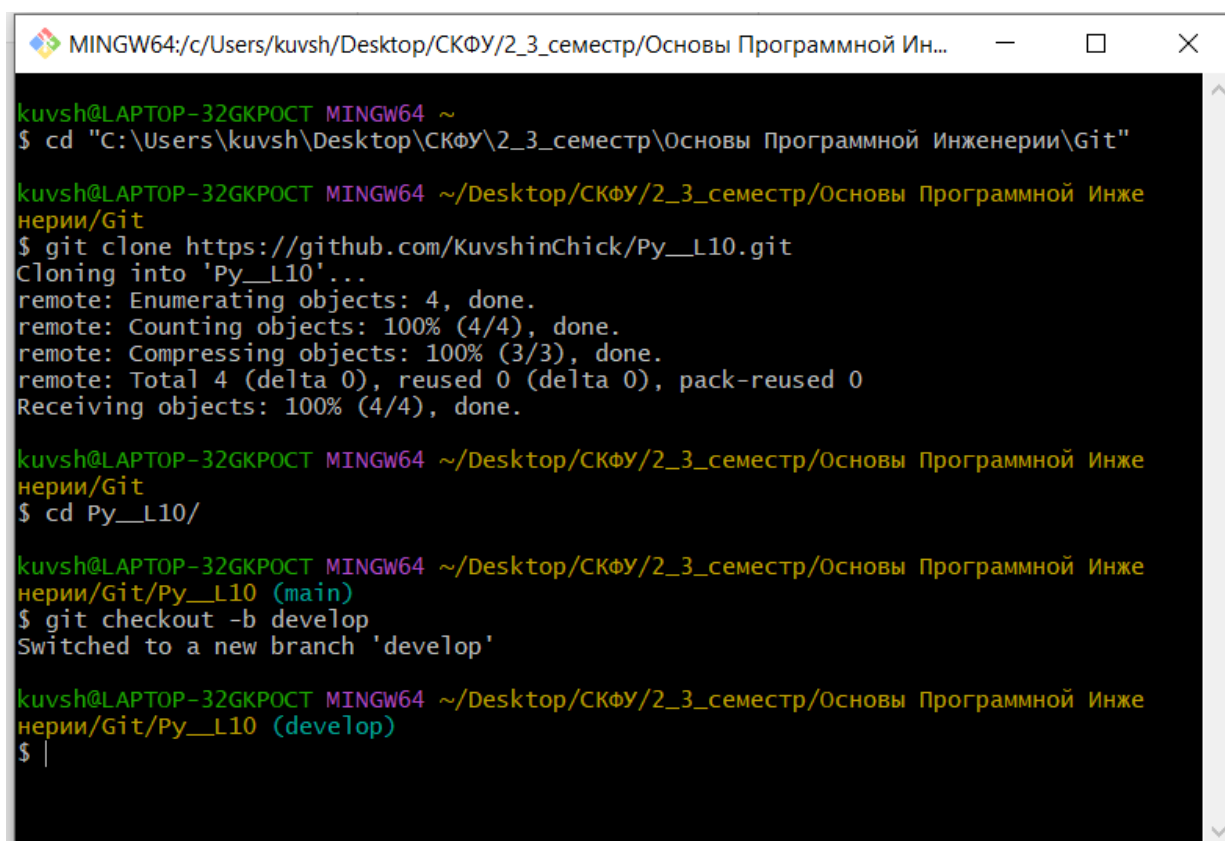
Ставрополь, 2022 г.

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

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L10.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
MINGW64:/c/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Ин...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git"

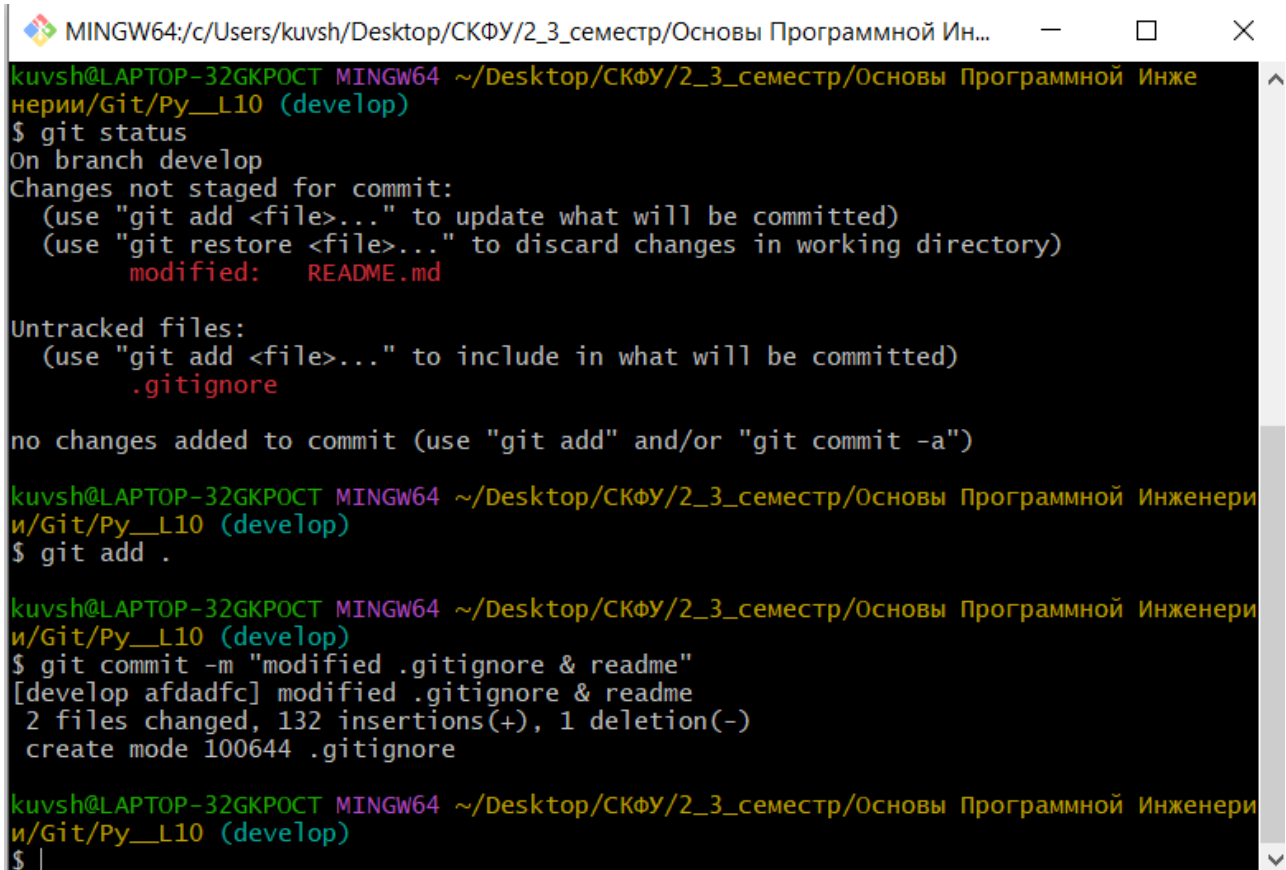
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L10.git
Cloning into 'Py__L10'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd Py__L10/

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L10 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L10 (develop)
$ |
```

Рисунок 9.1 – Клонирование репозитория и создание ветки develop

A screenshot of a Windows terminal window with a black background and white text. The window title is "MINGW64:/c/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Ин...". The user is at the prompt "kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L10 (develop)". They run "\$ git status", which shows that "README.md" is modified and ".gitignore" is untracked. Then they run "\$ git add .", followed by "\$ git commit -m 'modified .gitignore & readme'". The commit message is "[develop afdadfc] modified .gitignore & readme", and it shows "2 files changed, 132 insertions(+), 1 deletion(-)" and "create mode 100644 .gitignore". The prompt returns to "\$".

```
MINGW64:/c/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Ин...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L10 (develop)
$ git status
On branch develop
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:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженери
и/Git/Py__L10 (develop)
$ git add .

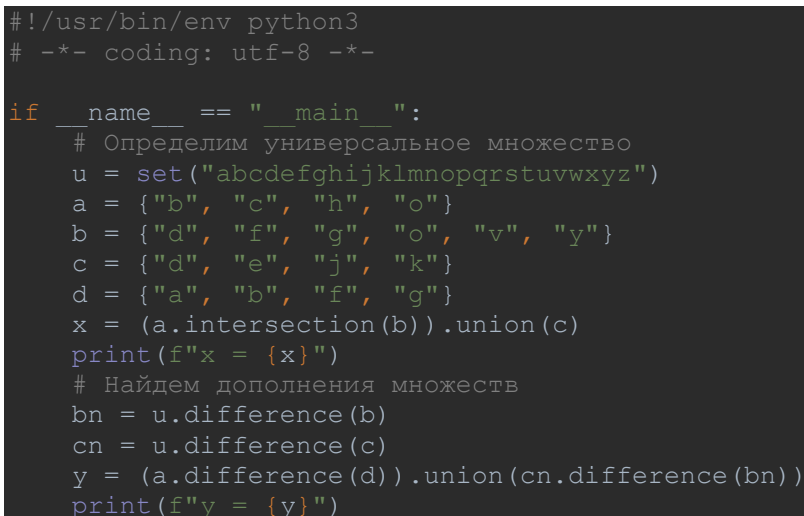
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженери
и/Git/Py__L10 (develop)
$ git commit -m "modified .gitignore & readme"
[develop afdadfc] modified .gitignore & readme
2 files changed, 132 insertions(+), 1 deletion(-)
create mode 100644 .gitignore

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженери
и/Git/Py__L10 (develop)
$
```

Рисунок 9.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.

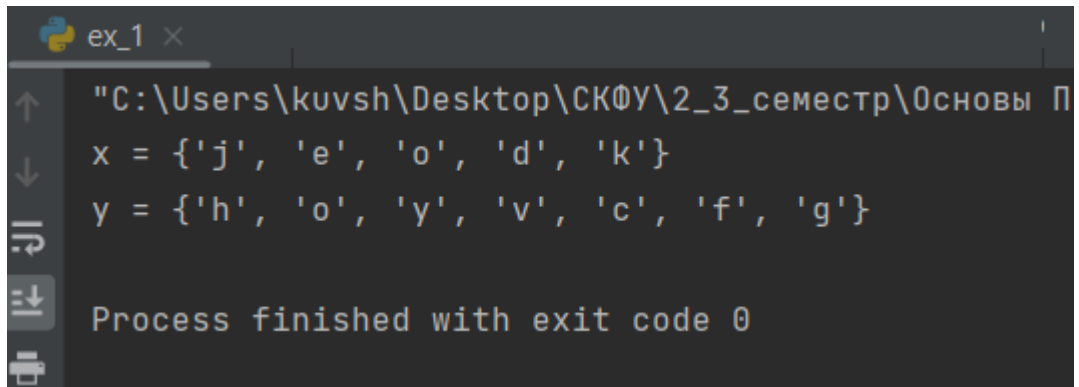
7. Проработайте примеры лабораторной работы. Создайте для них отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

A screenshot of a code editor showing Python code. The code defines a universal set 'u' and several subsets 'a', 'b', 'c', 'd'. It then performs set operations: 'x' is the union of the intersection of 'a' and 'b' with 'c'; 'bn' is the difference of 'u' and 'b'; 'cn' is the difference of 'u' and 'c'; 'y' is the union of the difference of 'a' and 'd' with the difference of 'cn' and 'bn'. The code prints the resulting sets 'x' and 'y'.

```
#!/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 9.3 – Код программы-примера



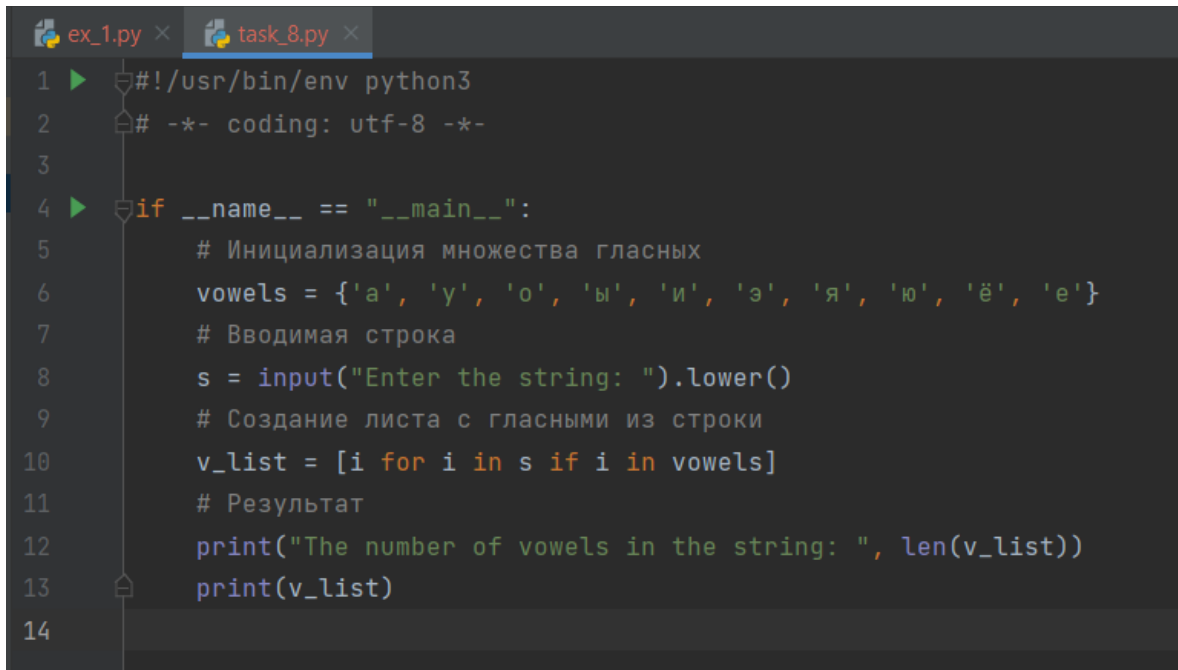
```
ex_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы П
x = {'j', 'e', 'o', 'd', 'k'}
y = {'h', 'o', 'y', 'v', 'c', 'f', 'g'}

Process finished with exit code 0
```

Рисунок 9.4 – Результат работы программы – примера

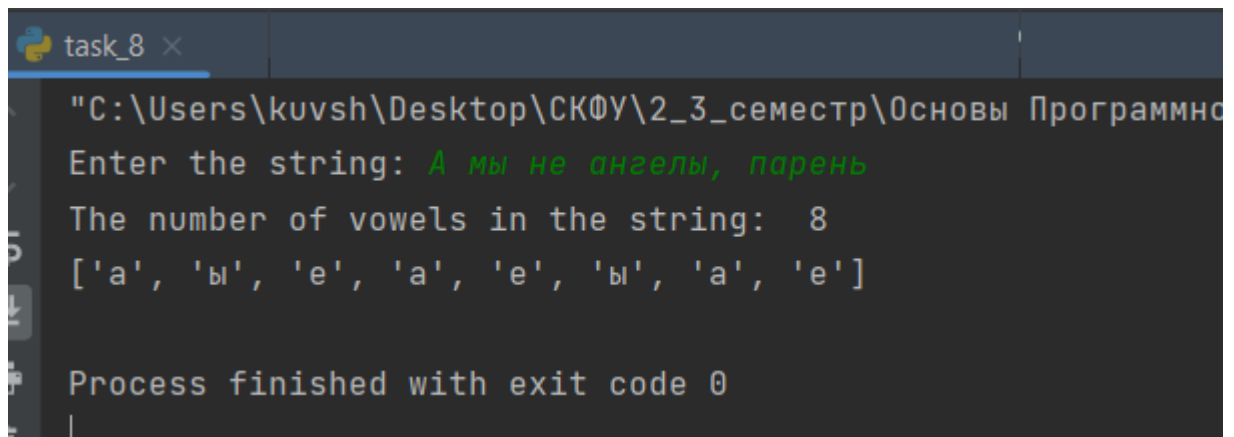
8. Решите задачу: подсчитайте количество гласных в строке, введенной с клавиатуры с использованием множеств.

9. Зафиксируйте сделанные изменения в репозитории.



```
ex_1.py x task_8.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == "__main__":
5     # Инициализация множества гласных
6     vowels = {'a', 'y', 'o', 'ы', 'и', 'э', 'я', 'ю', 'ё', 'е'}
7     # Вводимая строка
8     s = input("Enter the string: ").lower()
9     # Создание листа с гласными из строки
10    v_list = [i for i in s if i in vowels]
11    # Результат
12    print("The number of vowels in the string: ", len(v_list))
13    print(v_list)
14
```

Рисунок 9.5 – Код программы



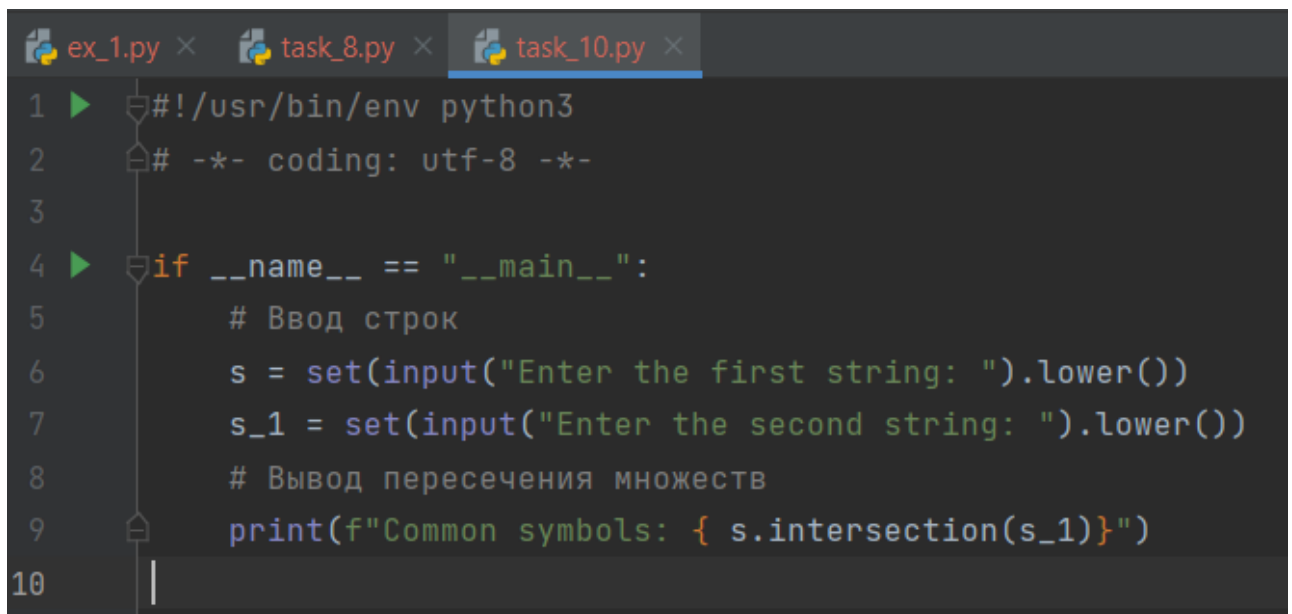
```
task_8 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программно
Enter the string: А мы не ангелы, парень
The number of vowels in the string: 8
['a', 'ы', 'е', 'а', 'е', 'ы', 'а', 'е']

Process finished with exit code 0
```

Рисунок 9.6 – Результат работы программы

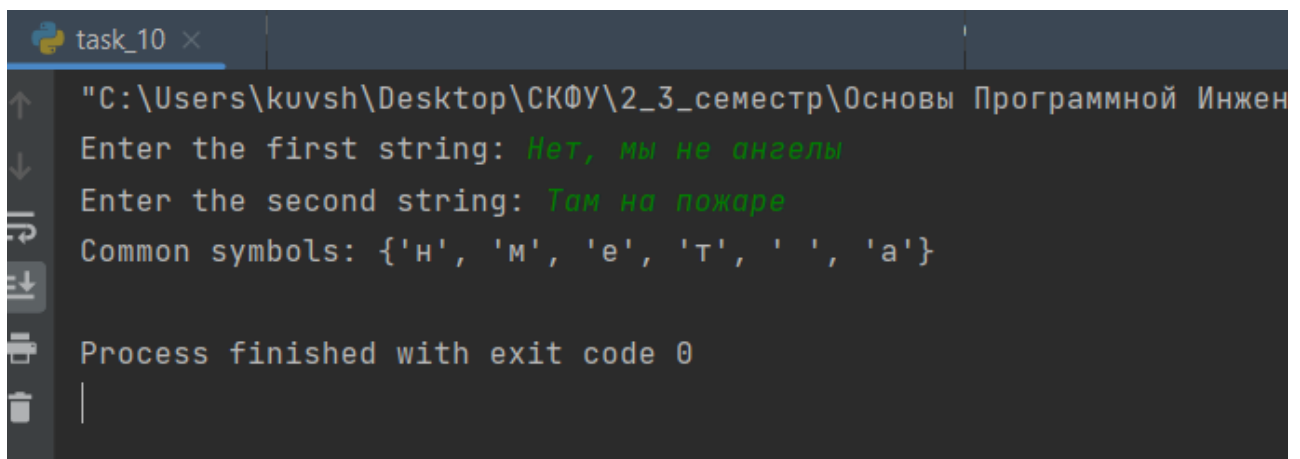
10. Решите задачу: определите общие символы в двух строках, введенных с клавиатуры.

11. Зафиксируйте сделанные изменения в репозитории.



```
ex_1.py x task_8.py x task_10.py x
1  ▶  #!/usr/bin/env python3
2  #  -*- coding: utf-8 -*-
3
4  ▶  if __name__ == "__main__":
5      # Ввод строк
6      s = set(input("Enter the first string: ").lower())
7      s_1 = set(input("Enter the second string: ").lower())
8      # Вывод пересечения множеств
9      print(f"Common symbols: { s.intersection(s_1)}")
10
```

Рисунок 9.7 – Код программы



```
task_10 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инжен
Enter the first string: Нет, мы не ангелы
Enter the second string: Там на пожаре
Common symbols: {'н', 'м', 'е', 'т', ' ', 'а'}

Process finished with exit code 0
```

Рисунок 9.8 – Результат работы программы

12. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

13. Выполните слияние ветки для разработки с веткой master/main.

14. Отправьте сделанные изменения на сервер GitHub.

15. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Индивидуальное задание

15.
$$A = \{c, m, n, o, q\}; \quad B = \{c, d, m, w\}; \quad C = \{m, n, q\}; \quad D = \{c, m, p\};$$
$$X = (A \cup B) \cap C; \quad Y = (A \cap \bar{B}) \cup (C/D).$$
 (16)

```
ex_1.py x task_8.py x task_10.py x ind_1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      # Определим универсальное множество
6      u = set("abcdefghijklmnopqrstuvwxyz")
7      # Инициализация множеств
8      A = {'c', 'm', 'n', 'o', 'q'}
9      B = {'c', 'd', 'm', 'w'}
10     C = {'m', 'n', 'q'}
11     D = {'c', 'm', 'p'}
12     # Найдем дополнения множества
13     Bn = u.difference(B)
14     # Проверка и вывод
15     print(f"Step_1, A∪B: {A.union(B)}")
16     print(f"Step_2, (A∪B)∩C: {(A.union(B)).intersection(C)}")
17     print(f"X = {(A.union(B)).intersection(C)}")
18     print("-----")
19     print(f"Step_1, A∩¬B: {A.intersection(Bn)}")
20     print(f"Step_2, C/D: {(C.difference(D))}")
21     print(f"Step_3, (A∩¬B)∪(C/D): {(A.intersection(Bn)).union(C.difference(D))}")
22     print(f"Y = {(A.intersection(Bn)).union(C.difference(D))}")
23
```

Рисунок 9.9 – Код программы

```
ind_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программ
Step_1, A∪B: {'d', 'w', 'm', 'n', 'c', 'o', 'q'}
Step_2, (A∪B)∩C: {'n', 'm', 'q'}
X = {'n', 'm', 'q'}
-----
Step_1, A∩¬B: {'n', 'o', 'q'}
Step_2, C/D: {'n', 'q'}
Step_3, (A∩¬B)∪(C/D): {'n', 'o', 'q'}
Y = {'n', 'o', 'q'}

Process finished with exit code 0
```

Рисунок 9.10 – Результат работы программы

Вопросы для защиты работы:

1. Что такое множества в языке Python?

Множеством в языке программирования Python называется неупорядоченная совокупность уникальных значений.

2. Как осуществляется создание множеств в Python?

Перед тем как начать работу с множеством, необходимо для начала его создать. Сделать это можно, просто присвоив переменной последовательность значений, выделив их фигурными скобками.

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

{0, 1, 2, 3}
```

Существует и другой способ создания множеств, который подразумевает использование вызова `set`. Аргументом этой функции может быть набор неких данных или даже строка с текстом, как это показано в следующем примере.

```
a = set('data')
print(a)

{'d', 'a', 't'}
```

3. Как проверить присутствие/отсутствие элемента в множестве?

Проверка, есть ли данное значение в множестве. Для этого используется `in`.

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

True
```

4. Как выполнить перебор элементов множества?

Перебор всех элементов.

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

```
0  
1  
2
```

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. Следующий код демонстрирует генерацию множества `a` с циклом `for` для нескольких чисел.

```
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`. Аргументом в данном случае будет добавляемый элемент последовательности. В примере кода на Python добавим в множество элемент со значением 4.

```
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 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`. 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'>
```

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'>
```

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'>
```