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 №2.3
дисциплины «Основы программной инженерии»

Выполнил:
Юрьев Илья Евгеньевич
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:
Богданов С.С., ассистент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Работа со строками в языке Python.

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

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python:

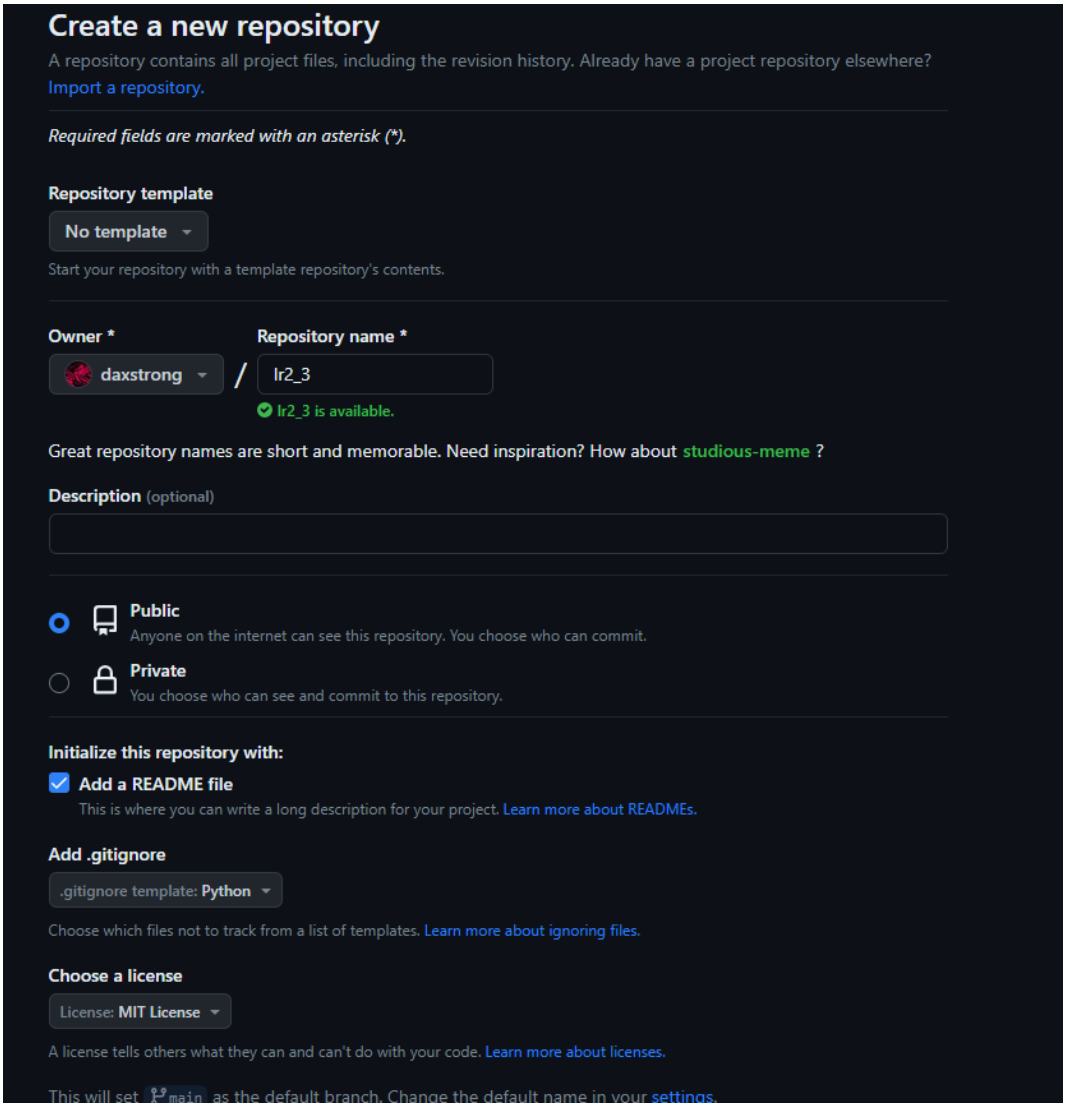

Рисунок 1 – Создание репозитория с заданными настройками

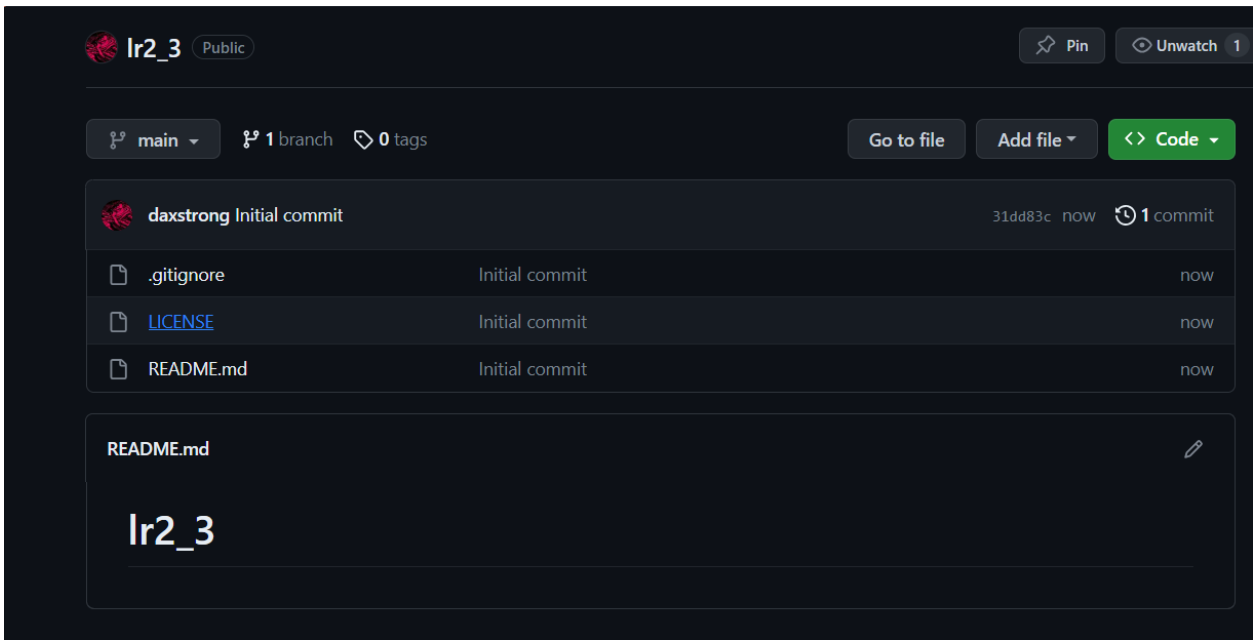

Рисунок 2 – Созданный репозиторий

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии
$ git clone https://github.com/daxstrong/lr2_3.git
Cloning into 'lr2_3'...
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.
```

Рисунок 3 – Клонирование репозитория

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_3 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 4 – Создание ветки develop

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8:

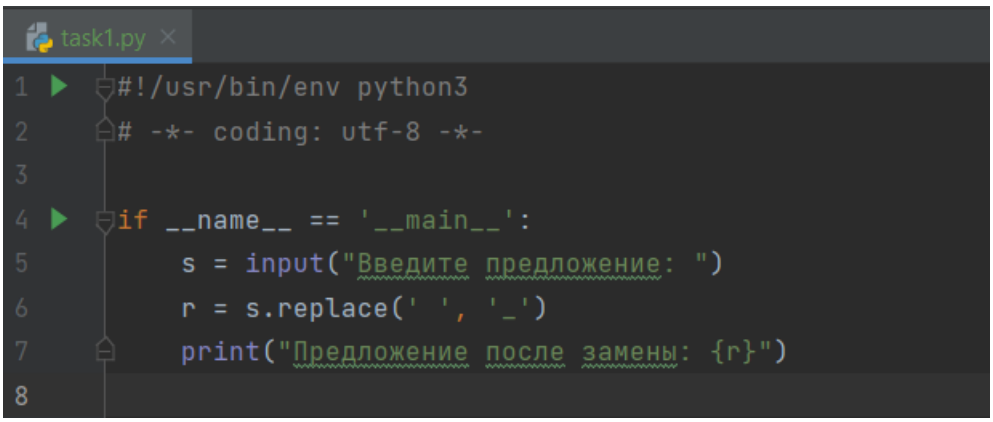

Рисунок 5 – Замена пробелов (задание №1)

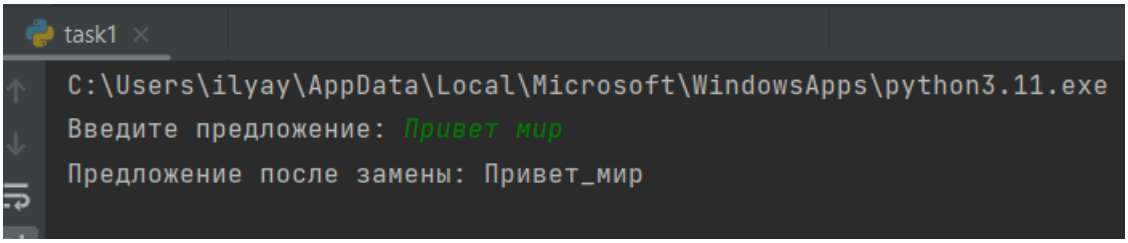

Рисунок 6 – Вывод программы (задание №1)

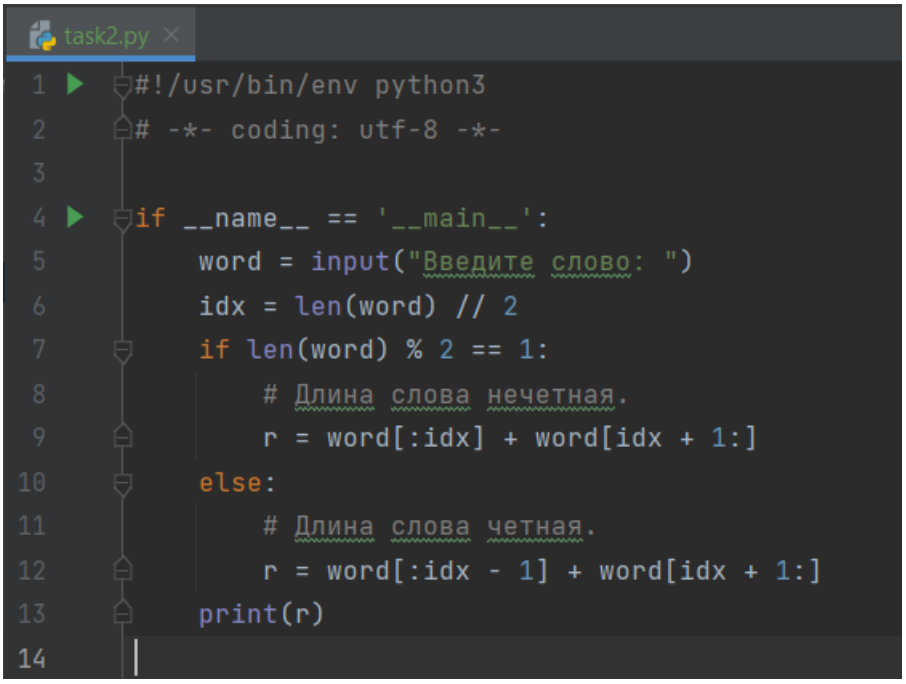

Рисунок 7 – Чётность и нечётность слова (задание №2)

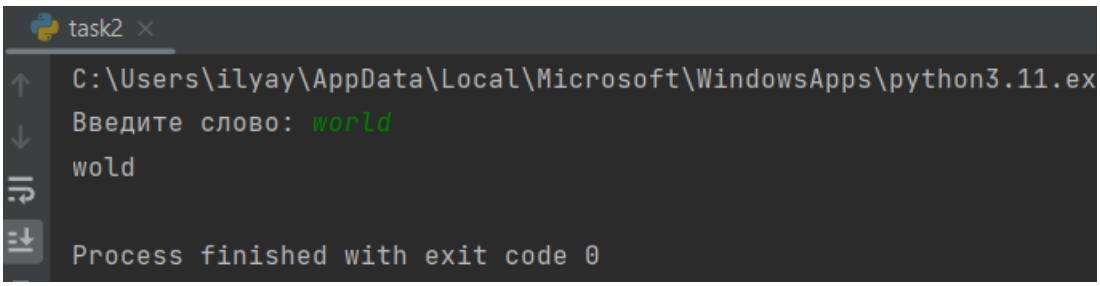

Рисунок 8 – Вывод программы (задание №2)

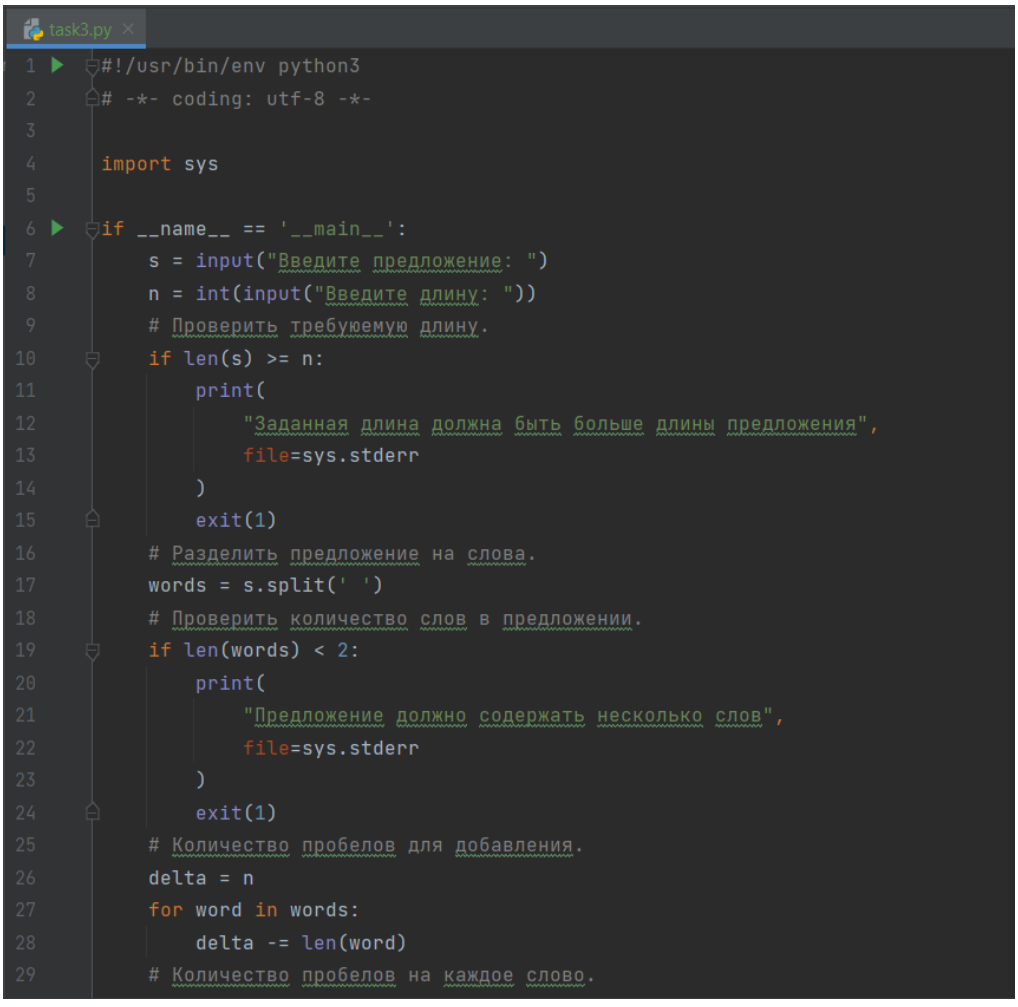

Рисунок 9 – Изменение длины строки (задание №3)

```
    w, r = delta // (len(words) - 1), delta % (len(words) - 1)
    # Сформировать список для хранения слов и пробелов.
    lst = []
    # Пронумеровать все слова в списке и перебрать их.
    for i, word in enumerate(words):
        lst.append(word)
        # Если слово не является последним, добавить пробелы.
        if i < len(words) - 1:
            # Определить количество пробелов.
            width = w
            if r > 0:
                width += 1
                r -= 1
            # Добавить заданное количество пробелов в список.
            if width > 0:
                lst.append(' ' * width)
    # Вывести новое предложение, объединив все элементы списка lst.
    print(''.join(lst))

```

Рисунок 5 – Изменение длины строки (задание №3)

```
task3
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите предложение: Hello world
Введите длину: 20
Hello           world
```

Рисунок 6 – Вывод программы (задание №3)

3. Выполним индивидуальные задания:

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

if __name__ == '__main__':
    sentence = input("Введите предложение:")

    result = []

    for i in range(len(sentence) - 1):
        if sentence[i:i + 2] == "нн":
            result.append(sentence[i:i + 2])

    print(result)
```

Рисунок 7 – Нахождение всех буквосочетаний «нн» (задание №1)

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите предложение:Решённая учителем задача
['нн']
```

Рисунок 8 – Вывод программы (задание №1)

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

if __name__ == '__main__':
    sentence = input("Введите предложение: ")

    first_comma_index = sentence.find(',')

    second_comma_index = sentence.find(',', first_comma_index + 1)

    if second_comma_index == -1:
        print("Символы после первой запятой:", sentence[first_comma_index + 1:])
    else:
        print("Символы между первой и второй запятой:", sentence[first_comma_index + 1:second_comma_index])
```

Рисунок 9 – Нахождение символов, стоящих между первой и второй запятой (задание №2)

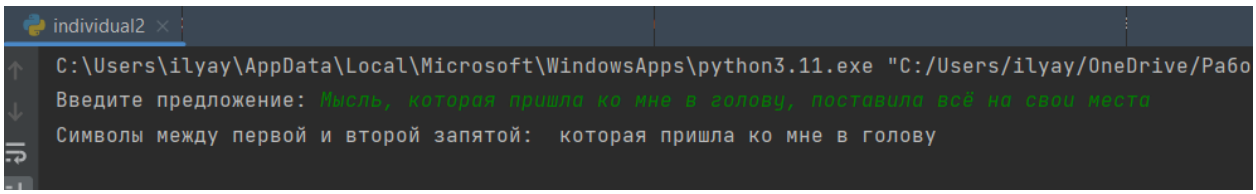

Рисунок 10 – Вывод программы (задание №2)

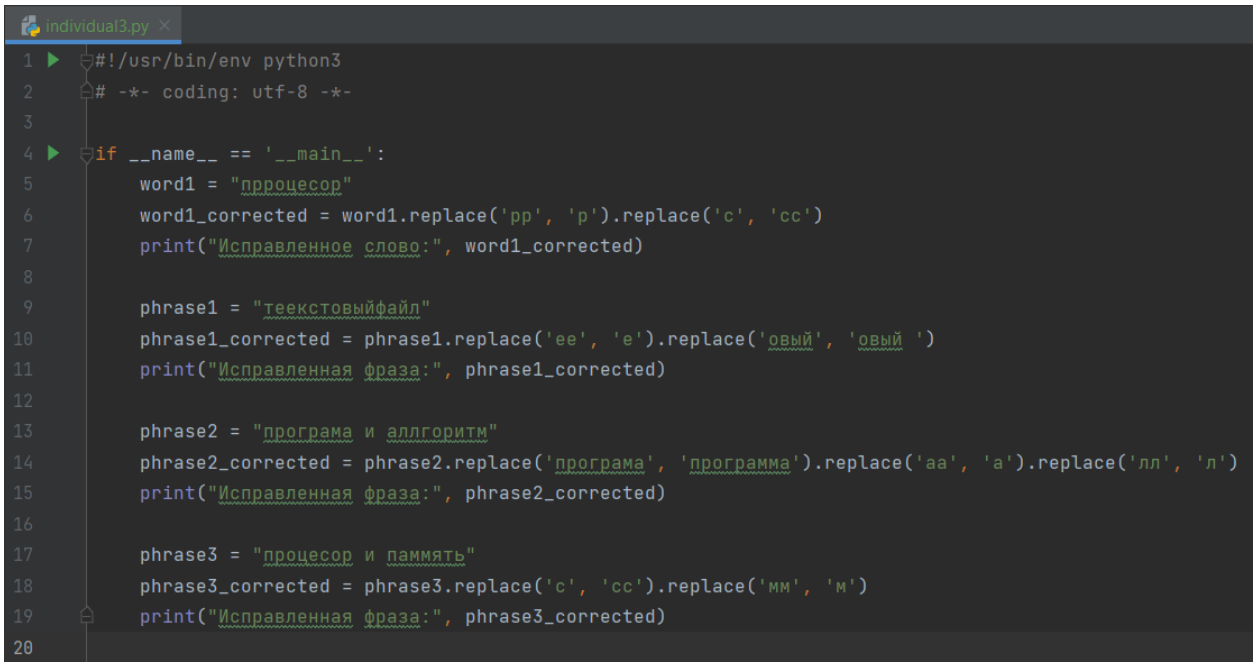

Рисунок 11 – Исправление ошибок в словах (задание №3)

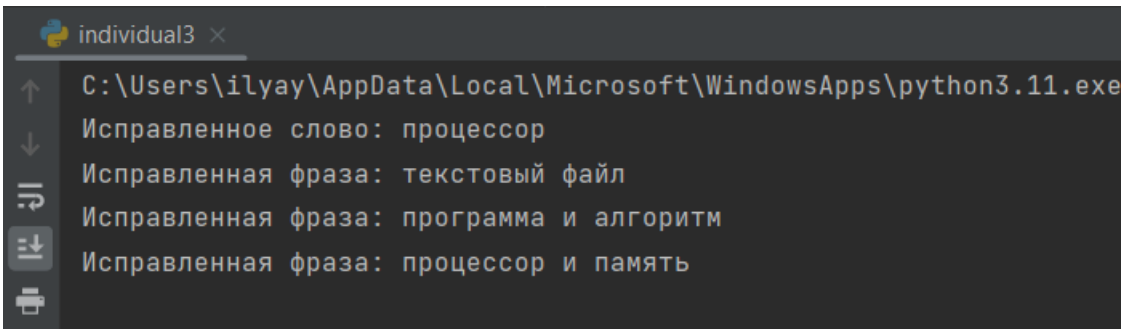

Рисунок 12 – Вывод программы (задание №3)

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

if __name__ == '__main__':
    word1 = input("Введите первое слово: ")
    word2 = input("Введите второе слово: ")
    word3 = input("Введите третье слово: ")

    unique_letters = []

    for letter in word1:
        if letter not in unique_letters and letter not in word2 and letter not in word3:
            unique_letters.append(letter)

    for letter in word2:
        if letter not in unique_letters and letter not in word1 and letter not in word3:
            unique_letters.append(letter)

    for letter in word3:
        if letter not in unique_letters and letter not in word1 and letter not in word2:
            unique_letters.append(letter)

    print("Неповторяющиеся буквы:", ''.join(unique_letters))
```

Рисунок 13 – Нахождение неповторяющихся букв в словах (задание №3)

```
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите первое слово: привет
Введите второе слово: ветер
Введите третье слово: парк
Неповторяющиеся буквы: иак
```

Рисунок 14 – Вывод программы (задание №4)

4. Зафиксируем проделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_3 (develop)
$ git log
commit 4a652372811dc68002ca3acf063f457de2c1e4b4 (HEAD -> develop)
Author: dexstrong <ilya.yurev.04@inbox.ru>
Date:   Mon Nov 27 17:56:48 2023 +0300

    Final changes

commit 31dd83c29cd17c802cd12f758365209103942e7f (origin/main, origin/HEAD, main)
Author: Ilya Yurev <112946692+daxstrong@users.noreply.github.com>
Date:   Mon Nov 27 16:42:14 2023 +0300

    Initial commit
```

Рисунок 15 – Коммиты ветки develop

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_3 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_3 (main)
$ git merge develop
Updating 31dd83c..4a65237
Fast-forward
 .idea/.gitignore                                |  8 +++++
 .idea/inspectionProfiles/profiles_settings.xml |  6 ++++
 .idea/lr2_3.iml                                 |  8 +++++
 .idea/misc.xml                                  |  4 +++
 .idea/modules.xml                               |  8 +++++
 .idea/vcs.xml                                   |  6 ++++
 individual1.py                                  | 13 +++++++
 individual2.py                                  | 14 ++++++++
 individual3.py                                  | 19 ++++++++++
 individual4.py                                  | 23 ++++++++++++
 task1.py                                        |  7 ++++
 task2.py                                        | 13 +++++++
 task3.py                                        | 47 +++++++++++++++++++++++++
 13 files changed, 176 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
 create mode 100644 .idea/lr2_3.iml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 individual1.py
 create mode 100644 individual2.py
 create mode 100644 individual3.py
 create mode 100644 individual4.py
 create mode 100644 task1.py
 create mode 100644 task2.py
 create mode 100644 task3.py
```

Рисунок 16 – Слияние веток main и develop

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженер
ии/lr2_3 (main)
$ git push origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 12 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (17/17), 3.93 KiB | 3.93 MiB/s, done.
Total 17 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/daxstrong/lr2_3.git
   31dd83c..4a65237  main -> main
```

Рисунок 22 – Отправка изменений на удаленный репозиторий

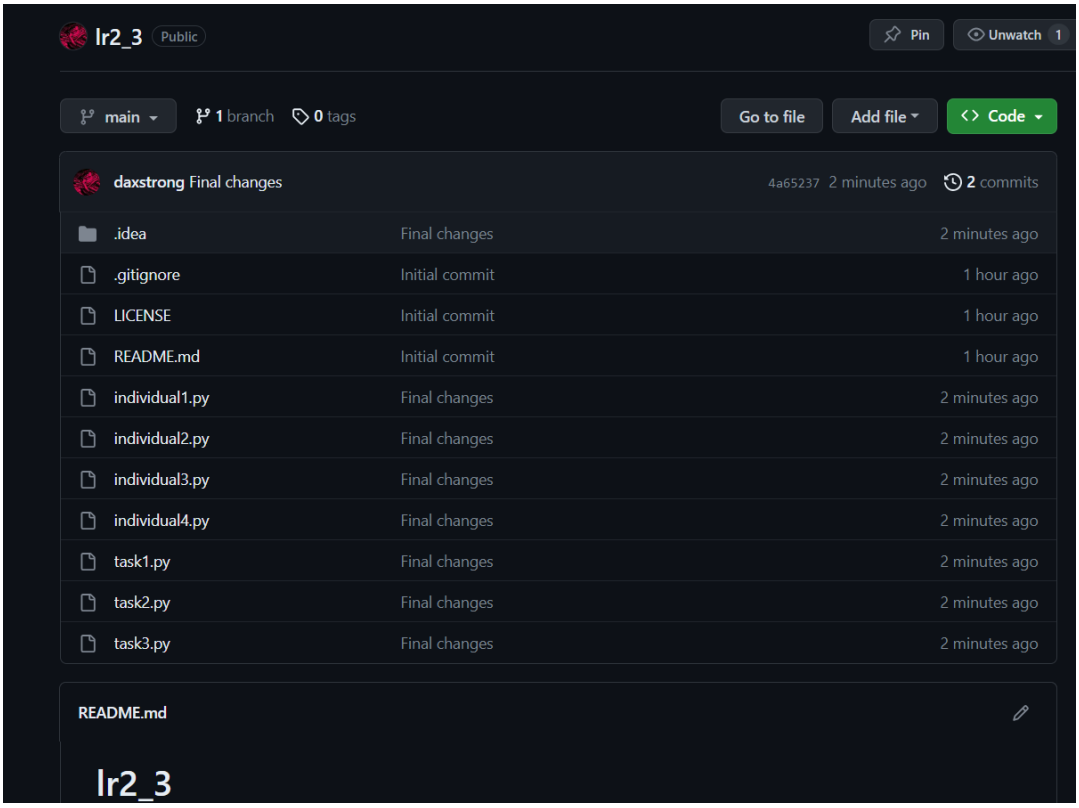

Рисунок 23 – Изменения удаленного репозитория

Ответы на контрольные вопросы:

1. Что такое строки в языке Python?

Строки в Python - это последовательности символов, заключенные в кавычки (одинарные, двойные или тройные). Они используются для хранения текстовой информации.

2. Какие существуют способы задания строковых литералов в языке Python?

Строки можно задать с помощью одинарных ('), двойных (") или тройных (''' или """) кавычек.

3. Какие операции и функции существуют для строк?

Python предоставляет множество операций и методов для работы со строками: конкатенация (+), умножение (*), доступ по индексу ([]), методы для поиска, замены, разделения строк и многое другое.

4. Как осуществляется индексирование строк?

Строки в Python индексируются, начиная с 0. Можно получить доступ к символам строки по их индексу, используя квадратные скобки: my_string[0].

5. Как осуществляется работа со срезами для строк?

Срезы позволяют получать подстроки из строки. Используйте my_string[start:end], чтобы получить подстроку с индексами от start до end-1.

6. Почему строки Python относятся к неизменяемому типу данных?

Строки в Python являются неизменяемыми, что означает, что после создания строки ее содержимое нельзя изменить, можно лишь создать новую строку.

7. Как проверить то, что каждое слово в строке начинается с заглавной буквы?

Можно использовать метод istitle(), который возвращает True, если каждое слово начинается с заглавной буквы.

8. Как проверить строку на вхождение в неё другой строки?

Для этого используется оператор in, например: substring in my_string.

9. Как найти индекс первого вхождения подстроки в строку?

Можно воспользоваться методом find() или index(), которые возвращают индекс первого вхождения подстроки.

10. Как подсчитать количество символов в строке?

Используйте функцию len(my_string), чтобы узнать длину строки.

11. Как подсчитать то, сколько раз определённый символ встречается в строке?

Метод count() позволяет подсчитать количество определенного символа в строке.

12. Что такое f-строки и как ими пользоваться?

f-строки позволяют встраивать значения переменных или выражений в строку. Используйте f"текст {переменная}" для подстановки значений.

13. Как найти подстроку в заданной части строки?

Можно использовать метод find() или index() с указанием диапазона индексов.

14. Как вставить содержимое переменной в строку, воспользовавшись методом format()?

С помощью метода format() можно вставить содержимое переменной в строку, указав место для подстановки {}.

15. Как узнать о том, что в строке содержатся только цифры?

Методы isdigit(), isnumeric() или isdecimal() позволяют проверить, содержатся ли в строке только цифры.

16. Как разделить строку по заданному символу?

Используйте метод split() с указанием символа или подстроки, по которой нужно разделить строку.

17. Как проверить строку на то, что она составлена только из строчных букв?

Метод islower() вернет True, если все символы строки являются строчными буквами.

18. Как проверить то, что строка начинается со строчной буквы?

Можно использовать метод islower() для проверки первого символа строки.

19. Можно ли в Python прибавить целое число к строке?

Нет, операция сложения числа и строки в Python вызовет ошибку. Однако, можно преобразовать число в строку и затем сконкатенировать строки.

20. Как «перевернуть» строку?

Используйте срезы для "переворота" строки: my_string[::-1].

21. Как объединить список строк в одну строку, элементы которой разделены дефисами?

Метод join() позволяет объединить строки списка с помощью определенного разделителя, например: ' - '.join(list_of_strings).

22. Как привести всю строку к верхнему или нижнему регистру?

Методы upper() и lower() соответственно приводят всю строку к верхнему или нижнему регистру.

23. Как преобразовать первый и последний символы строки к верхнему регистру?

Можно использовать методы capitalize() для первого символа и срезы для последнего: my_string[:-1] + my_string[-1].upper().

24. Как проверить строку на то, что она составлена только из прописных букв?

Метод isupper() вернет True, если все символы строки являются прописными буквами.

25. В какой ситуации вы воспользовались бы методом splitlines()?

splitlines() используется для разделения строки на отдельные строки по символу переноса строки '\n'.

26. Как в заданной строке заменить на что-либо все вхождения некоей подстроки?

Метод replace() заменяет все вхождения подстроки на другую строку.

27. Как проверить то, что строка начинается с заданной последовательности символов, или заканчивается заданной последовательностью символов?

Методы startswith() и endswith() проверяют, начинается или заканчивается ли строка соответственно с указанной последовательностию символов.

28. Как узнать о том, что строка включает в себя только пробелы?

Метод isspace() возвращает True, если строка состоит только из пробельных символов.

29. Что случится, если умножить некую строку на 3?

При умножении строки на число, она повторится нужное количество раз: 'abc' * 3 вернет 'abcabcabc'.

30. Как привести к верхнему регистру первый символ каждого слова в строке?

Можно использовать метод title(), который преобразует первый символ каждого слова к верхнему регистру.

31. Как пользоваться методом partition()?

partition() разбивает строку на три части по заданному разделителю, возвращая кортеж из трех элементов: текст до разделителя, сам разделитель и текст после разделителя.

32. В каких ситуациях пользуются методом rfind()?

rfind() используется для поиска последнего вхождения подстроки в строку.