

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
Отчет по лабораторной работе № 2.3
«Работа со строками в языке Python»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » ноября 2022 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2022

Цель работы:

Приобретение навыков по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.3>

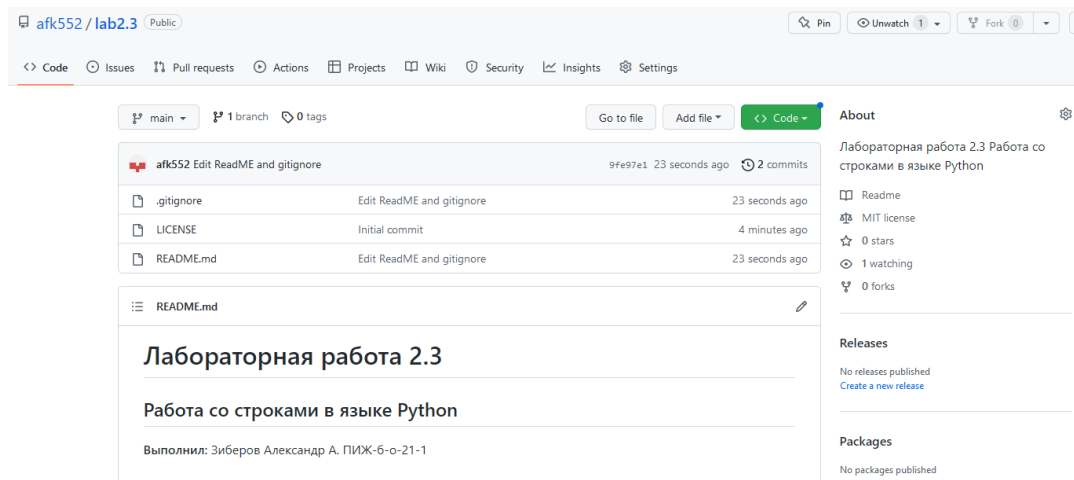


Рисунок 1 – Репозиторий GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

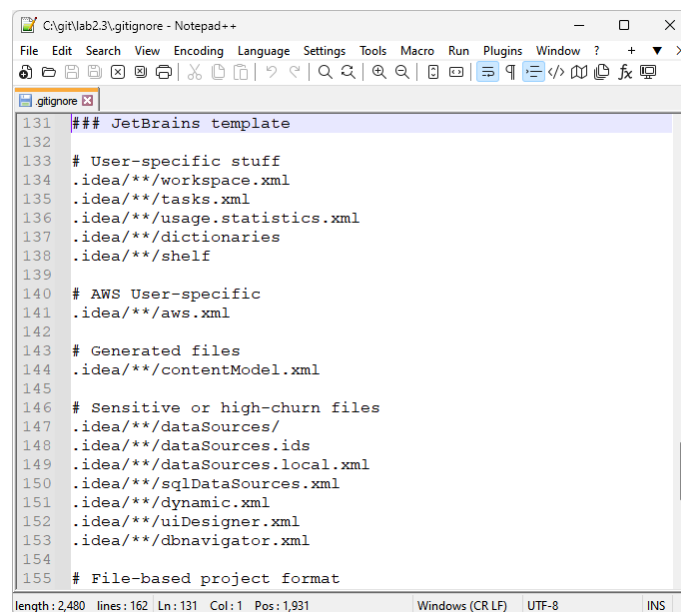
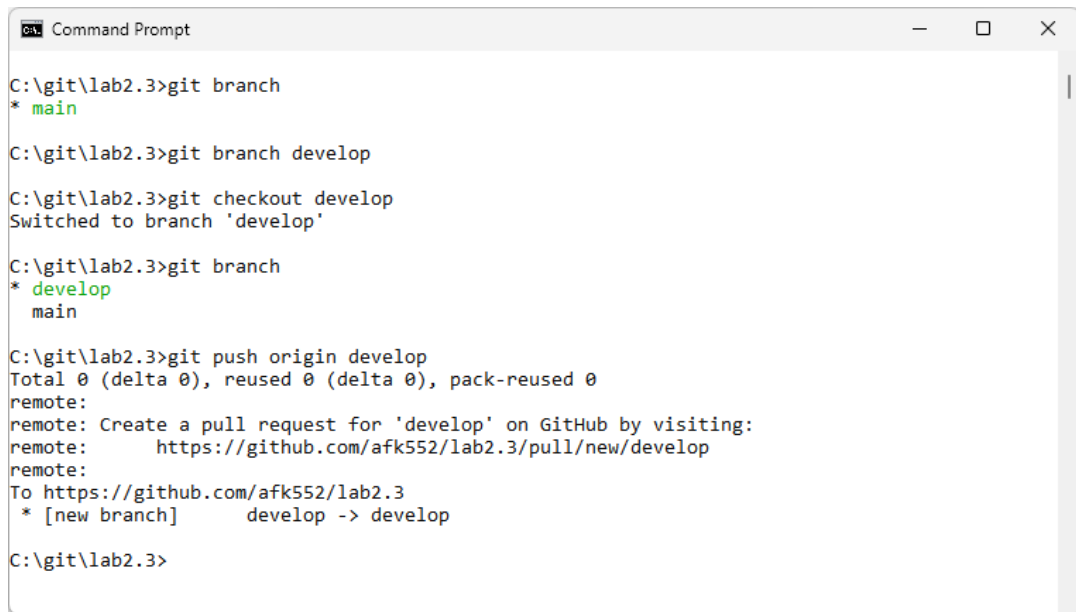


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\git\lab2.3>git branch
* main

C:\git\lab2.3>git branch develop

C:\git\lab2.3>git checkout develop
Switched to branch 'develop'

C:\git\lab2.3>git branch
* develop
  main

C:\git\lab2.3>git push 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/afk552/lab2.3/pull/new/develop
remote:
To https://github.com/afk552/lab2.3
 * [new branch]      develop -> develop

C:\git\lab2.3>
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

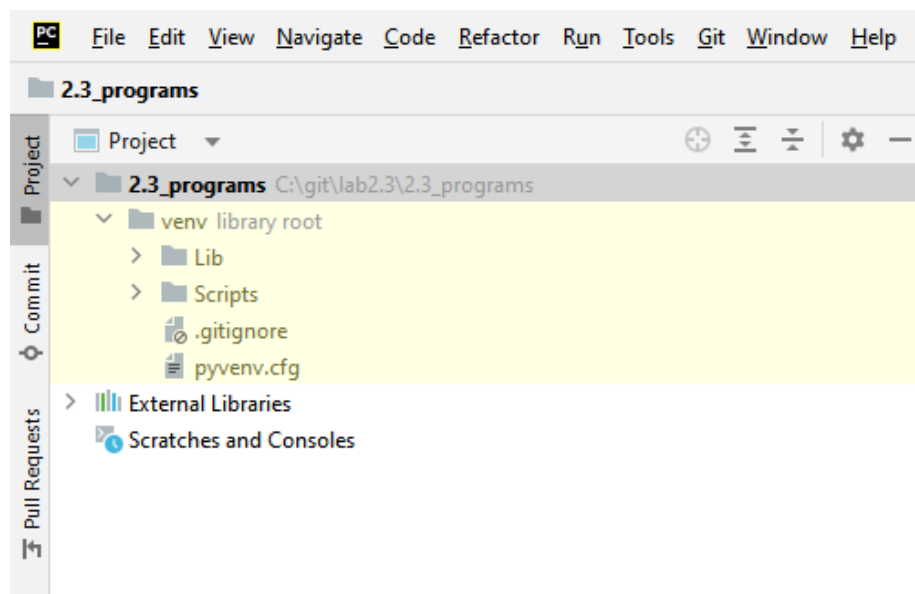


Рисунок 4 – Окно проекта в PyCharm

Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры. Рисунки 5-11.

```
example1.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     s = input("Введите предложение: ")
6     r = s.replace(' ', '_')
7     print(f"Предложение после замены: {r}")
8
```

Рисунок 5 – Код программы из примера 1

```
Run: example1 x
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/example1.py
Введите предложение: Тестовое предложение с пробелами для проверки примера 1.
Предложение после замены: Тестовое_предложение_с_пробелами_для_проверки_примера_1.

Process finished with exit code 0
```

Рисунок 6 – Результат выполнения программы из примера 1

```
example2.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     word = input("Введите слово: ")
6     idx = len(word) // 2
7     if len(word) % 2 == 1:
8         # Длина слова нечетная.
9         r = word[:idx] + word[idx+1:]
10    else:
11        # Длина слова четная.
12        r = word[:idx-1] + word[idx+1:]
13    print(r)
14
```

Рисунок 7 – Код программы из примера 2

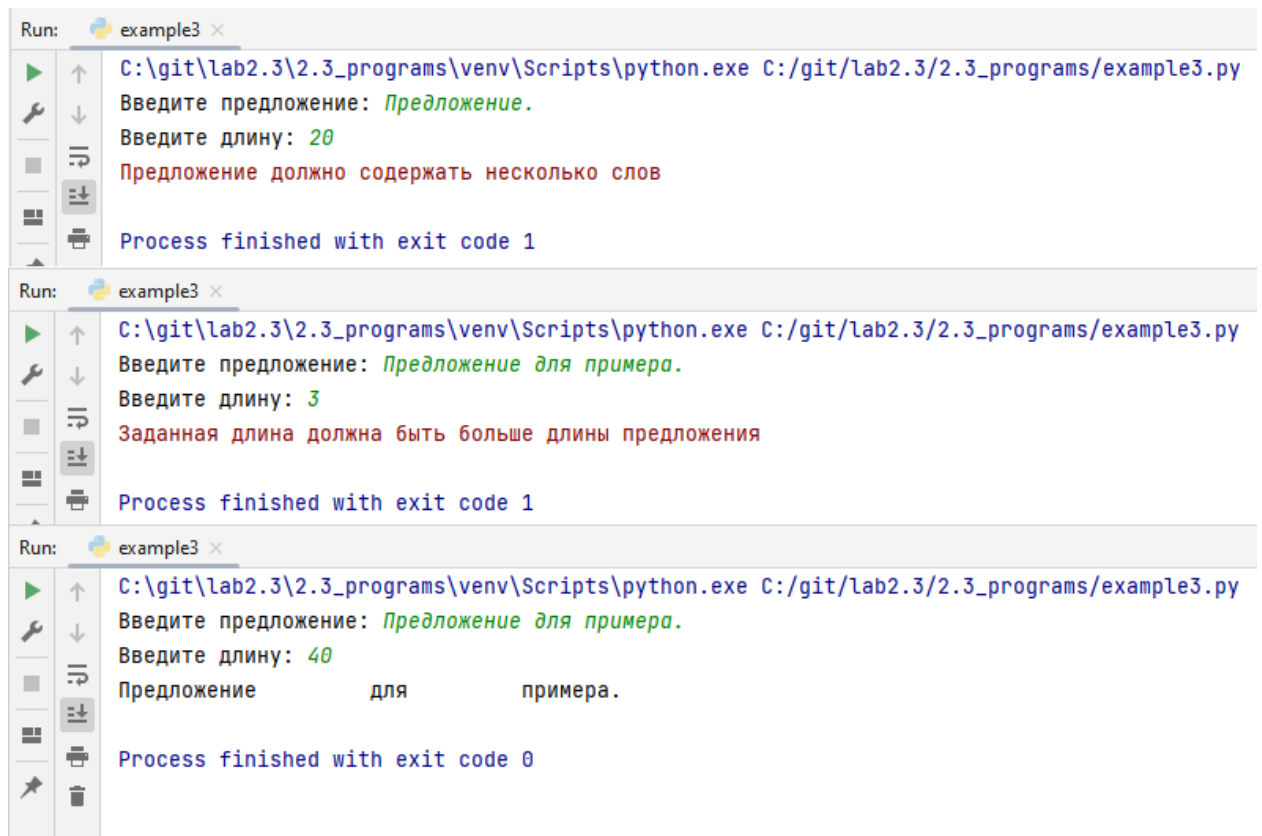
```
Run: example2 x
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/example2.py
Введите слово: Четное
Чеоо
Process finished with exit code 0

Run: example2 x
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/example2.py
Введите слово: Нечетное
Нечное
Process finished with exit code 0
```

Рисунок 8 – Результаты выполнения программы из примера 2

```
example3.py x
1 import sys
2
3 if __name__ == '__main__':
4     s = input("Введите предложение: ")
5     n = int(input("Введите длину: "))
6     # Проверить требуемую длину.
7     if len(s) >= n:
8         print(
9             "Заданная длина должна быть больше длины предложения",
10            file=sys.stderr
11        )
12        exit(1)
13    # Разделить предложение на слова.
14    words = s.split(' ')
15    # Проверить количество слов в предложении.
16    if len(words) < 2:
17        print(
18            "Предложение должно содержать несколько слов",
19            file=sys.stderr
20        )
21        exit(1)
22    # Количество пробелов для добавления.
23    delta = n
24    for word in words:
25        delta -= len(word)
26    # Количество пробелов на каждое слово.
27    w, r = delta // (len(words) - 1), delta % (len(words) - 1)
28    # Сформировать список для хранения слов и пробелов.
29    lst = []
30    # Пронумеровать все слова в списке и перебрать их.
31    for i, word in enumerate(words):
32        lst.append(word)
33        # Если слово не является последним, добавить пробелы.
34        if i < len(words) - 1:
35            # Определить количество пробелов.
36            width = w
37            if r > 0:
38                width += 1
39                r -= 1
40            # Добавить заданное количество пробелов в список.
41            if width > 0:
42                lst.append(' ' * width)
43
44    # Вывести новое предложение, объединив все элементы списка lst.
45    print(''.join(lst))
46
```

Рисунок 9 – Код программы из примера 3



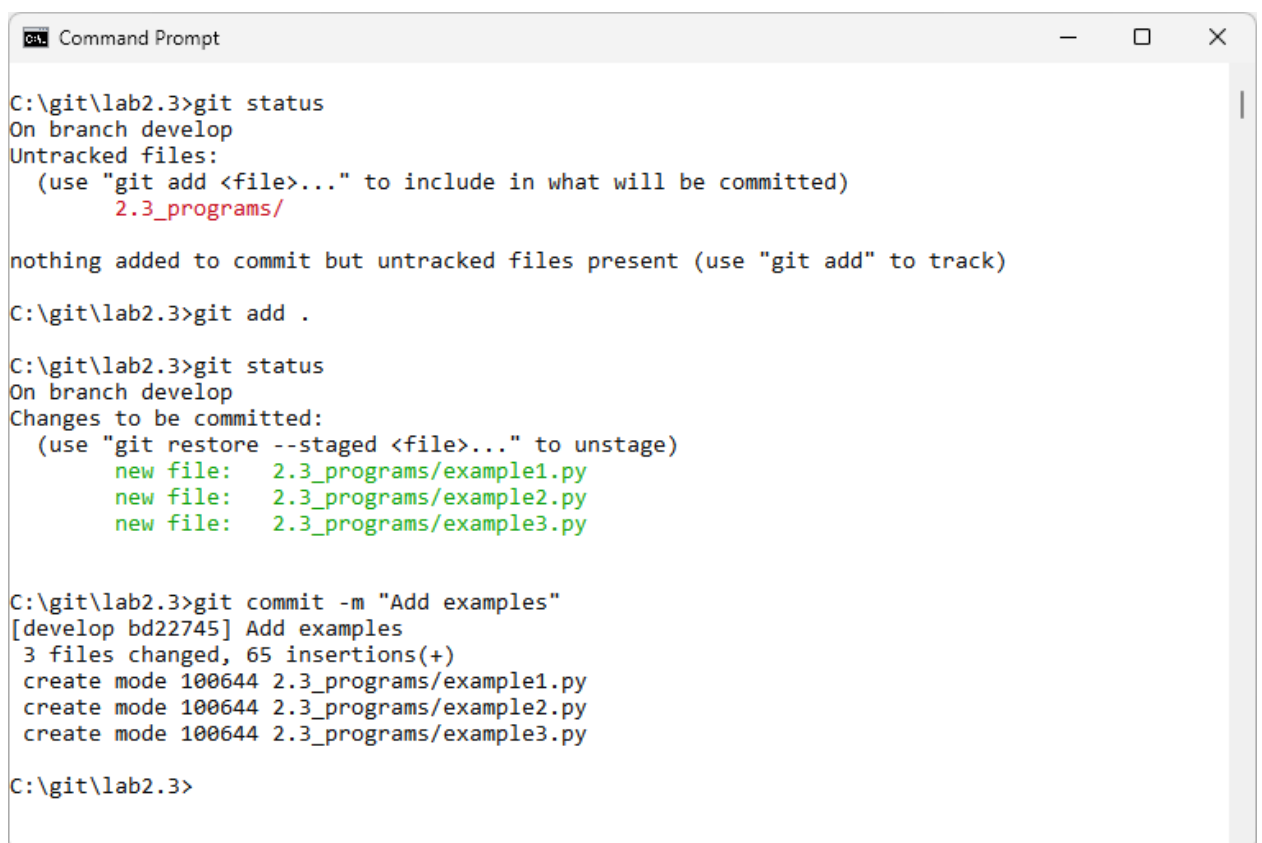
The image shows three sequential screenshots of a terminal window running a Python script named `example3.py`. The terminal path is `C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/example3.py`.

First screenshot: The script prompts for a sentence and a length. The user enters "Предложение." and "20". The script outputs "Предложение должно содержать несколько слов" (The sentence must contain several words) and "Process finished with exit code 1".

Second screenshot: The user enters "Предложение для примера." and "3". The script outputs "Заданная длина должна быть больше длины предложения" (The specified length must be greater than the length of the sentence) and "Process finished with exit code 1".

Third screenshot: The user enters "Предложение для примера." and "40". The script outputs "Предложение для примера." and "Process finished with exit code 0".

Рисунок 10 – Результаты выполнения программы из примера 3



The image shows a Windows Command Prompt window titled "Command Prompt" with the following text:

```
C:\git\lab2.3>git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    2.3_programs/

nothing added to commit but untracked files present (use "git add" to track)

C:\git\lab2.3>git add .

C:\git\lab2.3>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   2.3_programs/example1.py
    new file:   2.3_programs/example2.py
    new file:   2.3_programs/example3.py

C:\git\lab2.3>git commit -m "Add examples"
[develop bd22745] Add examples
 3 files changed, 65 insertions(+)
 create mode 100644 2.3_programs/example1.py
 create mode 100644 2.3_programs/example2.py
 create mode 100644 2.3_programs/example3.py

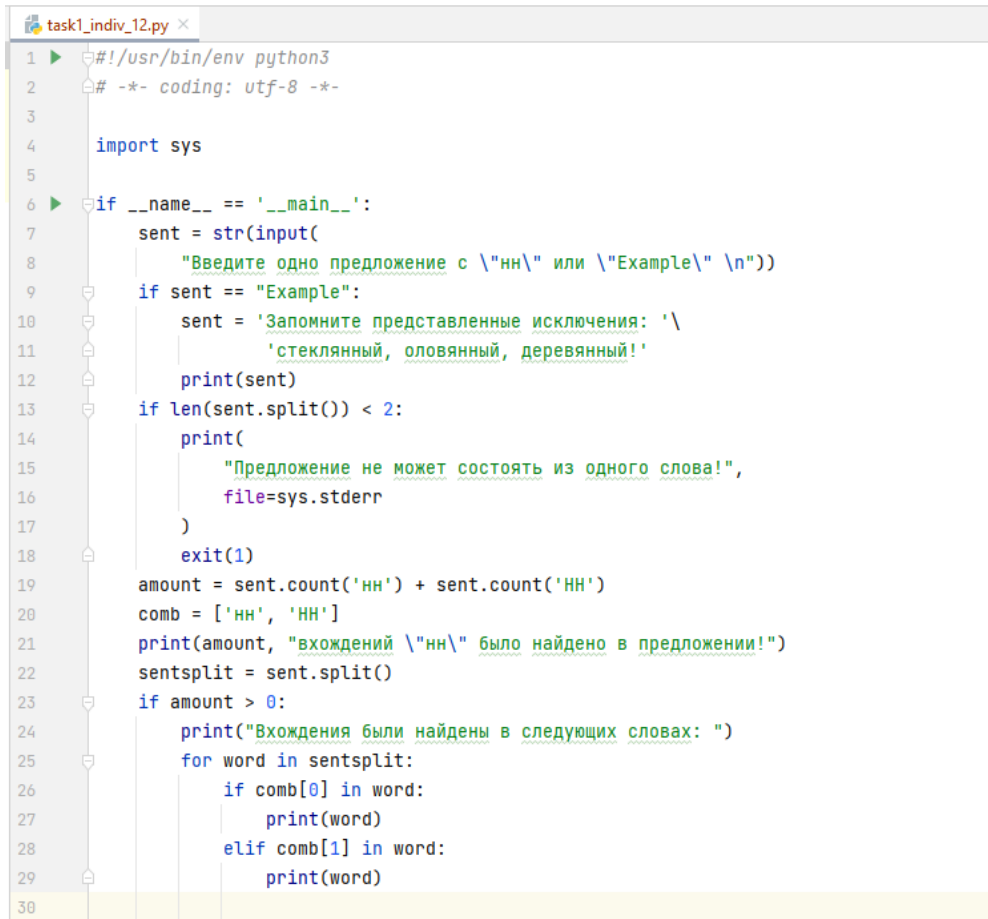
C:\git\lab2.3>
```

Рисунок 11 – Результаты выполнения программы из примера 3

Выполните индивидуальные задания, согласно своего варианта.

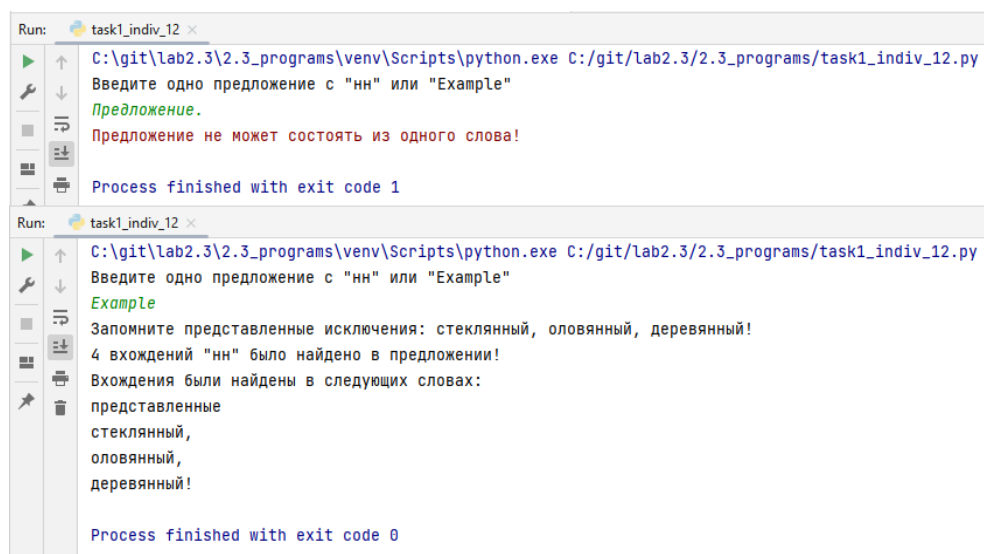
Вариант 12

Задание 1. Дано предложение. Вывести все имеющиеся в нем буквосочетания **нн**. Рисунки 12, 13.



```
task1_indiv_12.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      sent = str(input(
8          "Введите одно предложение с \"нн\" или \"Example\" \n"))
9      if sent == "Example":
10         sent = 'Запомните представленные исключения: '\
11             'стеклянный, оловянный, деревянный!'
12         print(sent)
13         if len(sent.split()) < 2:
14             print(
15                 "Предложение не может состоять из одного слова!",
16                 file=sys.stderr
17             )
18             exit(1)
19         amount = sent.count('нн') + sent.count('НН')
20         comb = ['нн', 'НН']
21         print(amount, "вхождений \"нн\" было найдено в предложении!")
22         sentsplit = sent.split()
23         if amount > 0:
24             print("Вхождения были найдены в следующих словах: ")
25             for word in sentsplit:
26                 if comb[0] in word:
27                     print(word)
28                 elif comb[1] in word:
29                     print(word)
30
```

Рисунок 12 – Код программы индивидуального задания 1

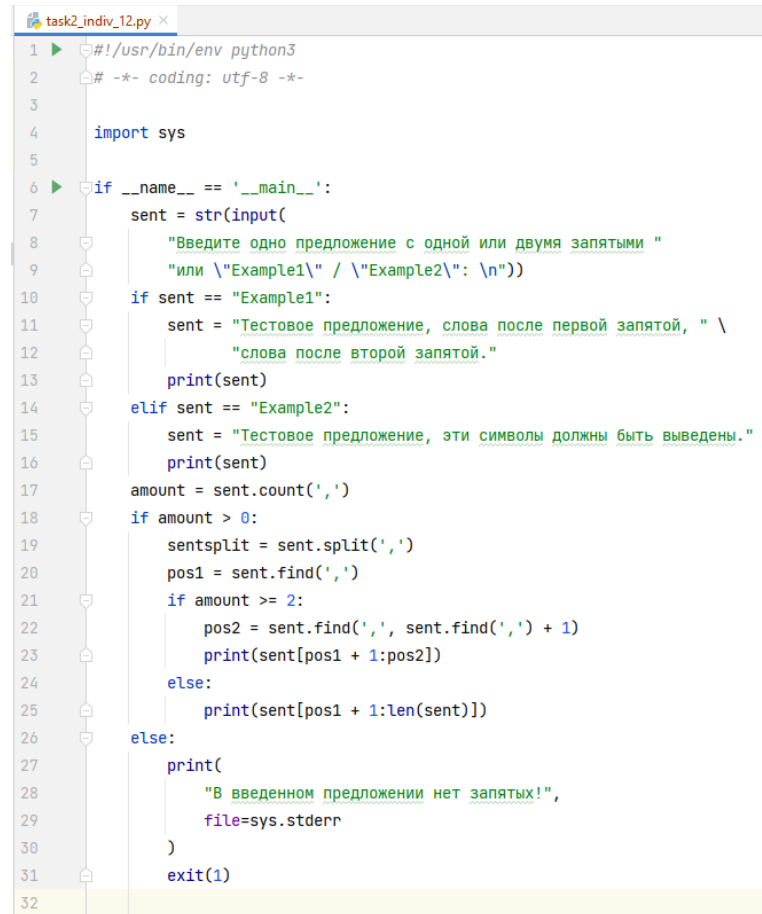


```
Run: task1_indiv_12
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/task1_indiv_12.py
Введите одно предложение с "нн" или "Example"
Предложение.
Предложение не может состоять из одного слова!
Process finished with exit code 1

Run: task1_indiv_12
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/task1_indiv_12.py
Введите одно предложение с "нн" или "Example"
Example
Запомните представленные исключения: стеклянный, оловянный, деревянный!
4 вхождений "нн" было найдено в предложении!
Вхождения были найдены в следующих словах:
представленные
стеклянный,
оловянный,
деревянный!
Process finished with exit code 0
```

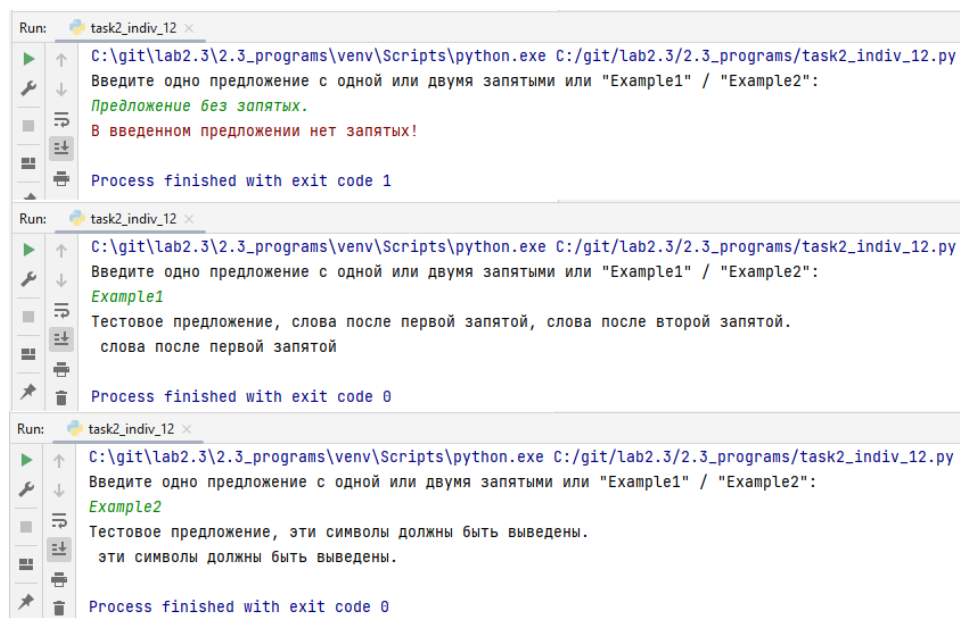
Рисунок 13 – Результаты выполнения программы индивидуального задания 1

Задание 2. Дано предложение. Напечатать все символы, расположенные между первой и второй запятыми. Если второй запятой нет, то должны быть напечатаны все символы, расположенные после единственной имеющейся запятой. Рисунки 14, 15.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      sent = str(input(
8          "Введите одно предложение с одной или двумя запятыми "
9          "или \"Example1\" / \"Example2\": \n"))
10     if sent == "Example1":
11         sent = "Тестовое предложение, слова после первой запятой, " \
12             "слова после второй запятой."
13         print(sent)
14     elif sent == "Example2":
15         sent = "Тестовое предложение, эти символы должны быть выведены."
16         print(sent)
17     amount = sent.count(',')
18     if amount > 0:
19         sentsplit = sent.split(',')
20         pos1 = sent.find(',')
21         if amount >= 2:
22             pos2 = sent.find(',', sent.find(',') + 1)
23             print(sent[pos1 + 1:pos2])
24         else:
25             print(sent[pos1 + 1:len(sent)])
26     else:
27         print(
28             "В введенном предложении нет запятых!",
29             file=sys.stderr
30         )
31     exit(1)
32
```

Рисунок 14 – Код программы индивидуального задания 2



```
Run: task2_indiv_12
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/task2_indiv_12.py
Введите одно предложение с одной или двумя запятыми или "Example1" / "Example2":
Предложение без запятых.
В введенном предложении нет запятых!
Process finished with exit code 1

Run: task2_indiv_12
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/task2_indiv_12.py
Введите одно предложение с одной или двумя запятыми или "Example1" / "Example2":
Example1
Тестовое предложение, слова после первой запятой, слова после второй запятой.
слова после первой запятой
Process finished with exit code 0

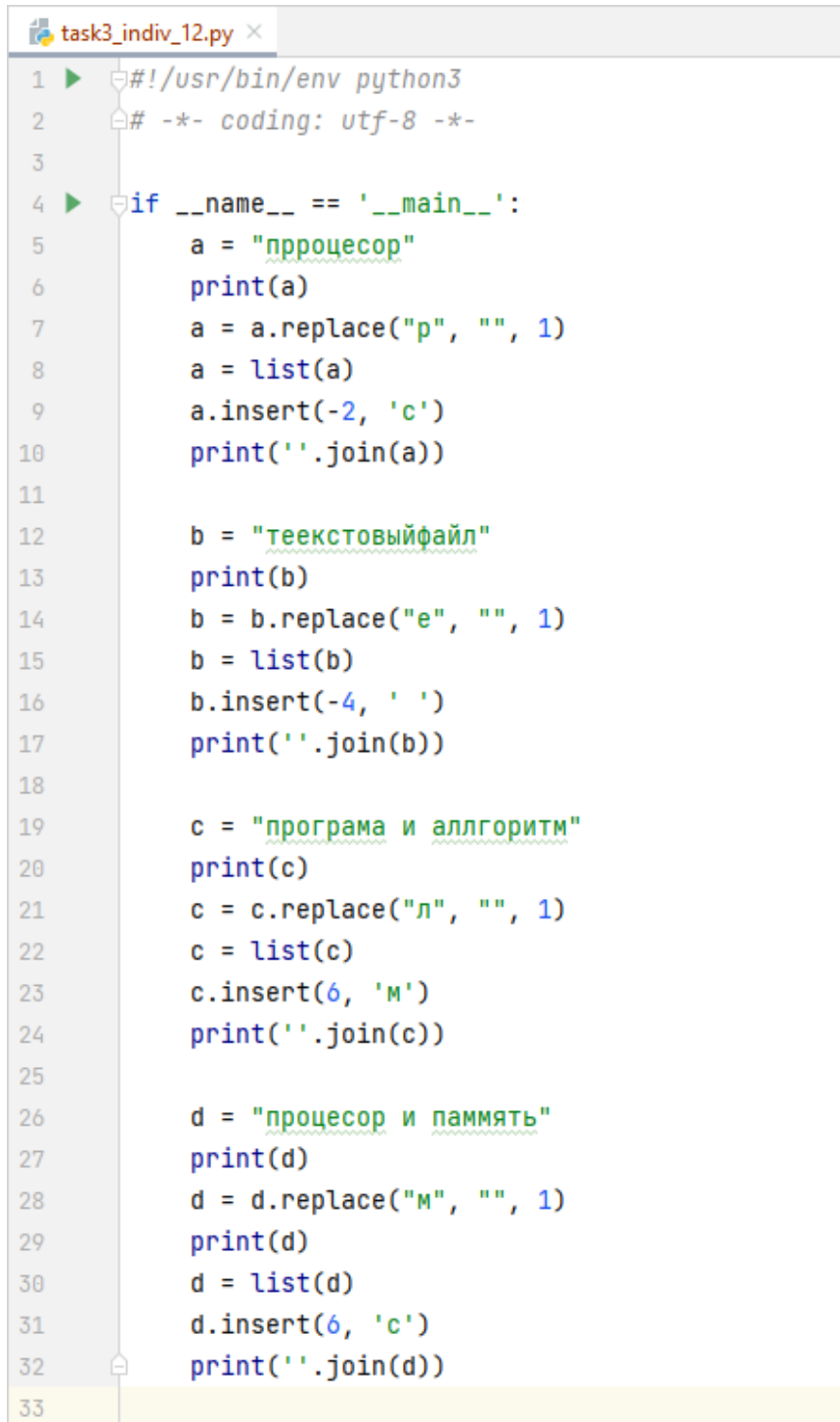
Run: task2_indiv_12
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/task2_indiv_12.py
Введите одно предложение с одной или двумя запятыми или "Example1" / "Example2":
Example2
Тестовое предложение, эти символы должны быть выведены.
эти символы должны быть выведены.
Process finished with exit code 0
```

Рисунок 15 – Результаты выполнения программы индивидуального задания 2

Задание 3. Путем вставок и удаления символов исправить ошибки:

- в слове прроцесор;
- во фразе теекстовыйфайл;
- во фразе програма и аллгоритм;
- во фразе процессор и паммать.

Рисунки 16, 17.



```
task3_indiv_12.py x
1  > #!/usr/bin/env python3
2  > # -*- coding: utf-8 -*-
3
4  > if __name__ == '__main__':
5      a = "прроцесор"
6      print(a)
7      a = a.replace("p", "", 1)
8      a = list(a)
9      a.insert(-2, 'c')
10     print(''.join(a))
11
12     b = "теекстовыйфайл"
13     print(b)
14     b = b.replace("e", "", 1)
15     b = list(b)
16     b.insert(-4, ' ')
17     print(''.join(b))
18
19     c = "програма и аллгоритм"
20     print(c)
21     c = c.replace("л", "", 1)
22     c = list(c)
23     c.insert(6, 'м')
24     print(''.join(c))
25
26     d = "процесор и паммать"
27     print(d)
28     d = d.replace("м", "", 1)
29     print(d)
30     d = list(d)
31     d.insert(6, 'c')
32     print(''.join(d))
33
```

Рисунок 16 – Код программы индивидуального задания 3



Рисунок 17 – Результат выполнения программы индивидуального задания 3

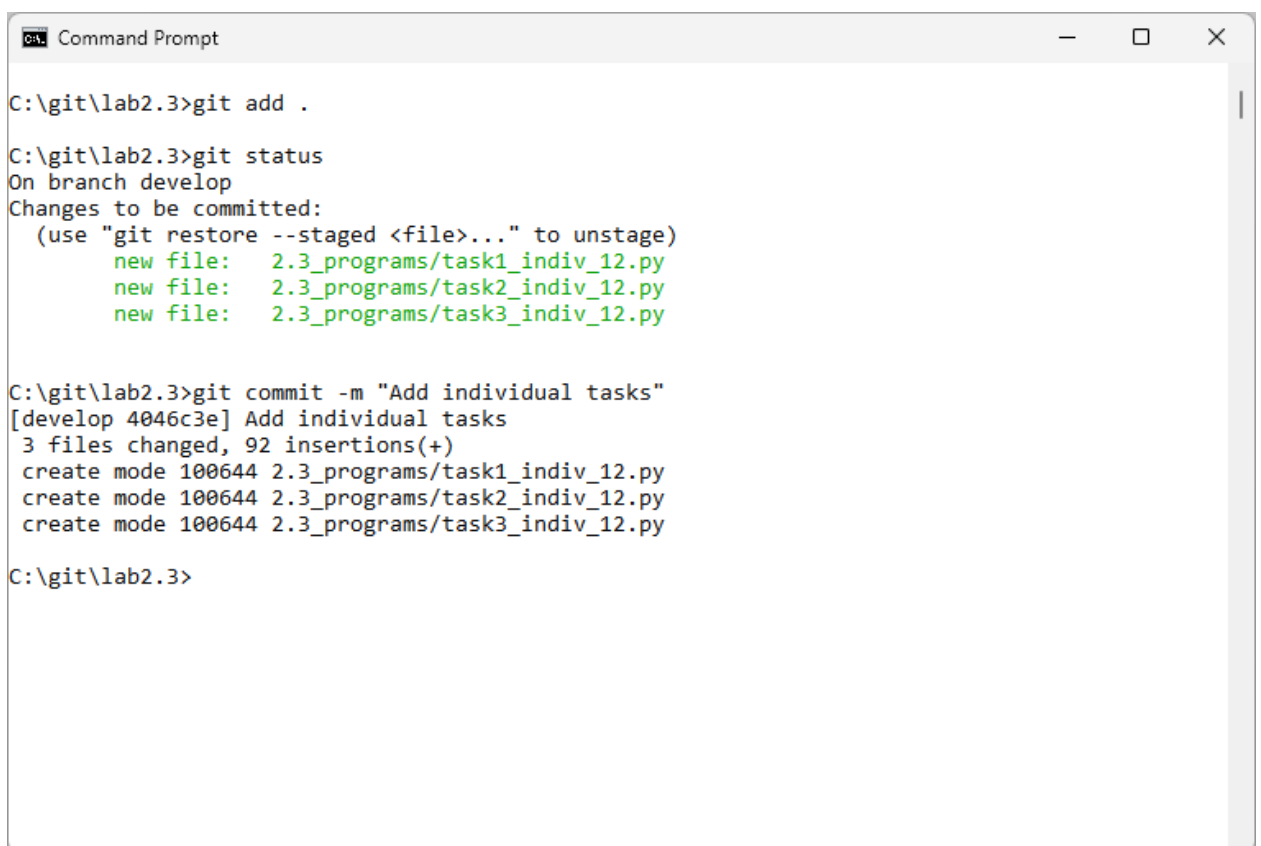
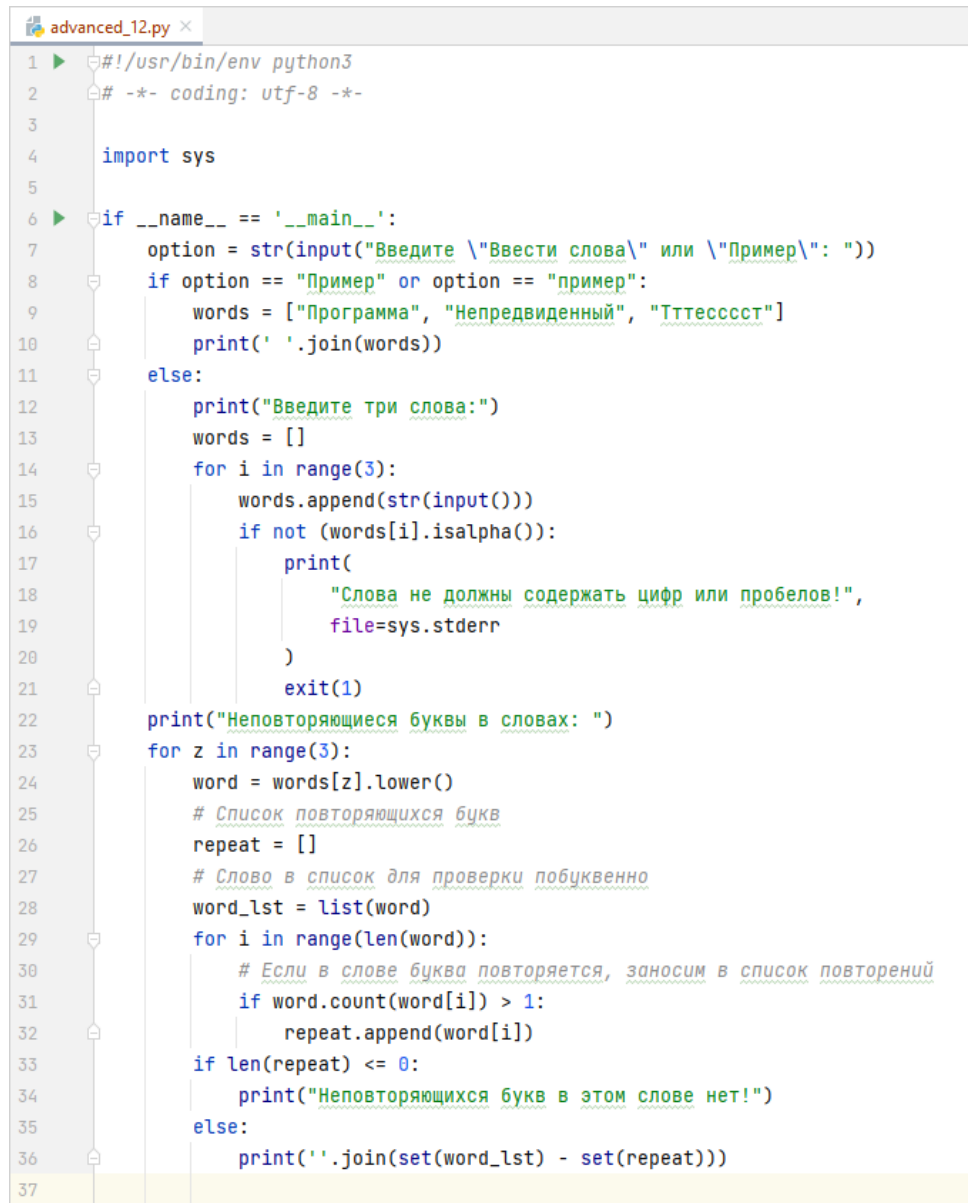


Рисунок 18 – Окно командной строки

Задание повышенной сложности.

Даны три слова. Напечатать неповторяющиеся в них буквы. Рисунки 19, 20.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     option = str(input("Введите \"Ввести слова\" или \"Пример\": "))
8     if option == "Пример" or option == "пример":
9         words = ["Программа", "Непредвиденный", "Тттесссст"]
10        print(' '.join(words))
11    else:
12        print("Введите три слова:")
13        words = []
14        for i in range(3):
15            words.append(str(input()))
16            if not (words[i].isalpha()):
17                print(
18                    "Слова не должны содержать цифр или пробелов!",
19                    file=sys.stderr
20                )
21            exit(1)
22        print("Неповторяющиеся буквы в словах: ")
23        for z in range(3):
24            word = words[z].lower()
25            # Список повторяющихся букв
26            repeat = []
27            # Слово в список для проверки побуквенно
28            word_lst = list(word)
29            for i in range(len(word)):
30                # Если в слове буква повторяется, заносим в список повторений
31                if word.count(word[i]) > 1:
32                    repeat.append(word[i])
33            if len(repeat) <= 0:
34                print("Неповторяющихся букв в этом слове нет!")
35            else:
36                print(' '.join(set(word_lst) - set(repeat)))
37
```

Рисунок 19 – Код программы задания повышенной сложности 12



```
Run: advanced_12 x
C:\git\lab2.3\2.3_programs\venv\Scripts\python.exe C:/git/lab2.3/2.3_programs/advanced_12.py
Введите "Ввести слова" или "Пример": Пример
Программа Непредвиденный Тттесссст
Неповторяющиеся буквы в словах:
пог
првийы
е
Process finished with exit code 0
```

Рисунок 20 – Результат выполнения программы задания повышенной сложности 12

```
Command Prompt

C:\git\lab2.3>git add .

C:\git\lab2.3>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.3_programs/advanced_12.py

C:\git\lab2.3>git commit -m "Add advanced task"
[develop cbac3d7] Add advanced task
 1 file changed, 36 insertions(+)
 create mode 100644 2.3_programs/advanced_12.py

C:\git\lab2.3>
```

Рисунок 21 – Окно командной строки

Выполните слияние ветки для разработки с веткой main / master.
Отправьте сделанные изменения на сервер GitHub.

```
Command Prompt

C:\git\lab2.3>git branch
* develop
  main

C:\git\lab2.3>git push origin develop
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (16/16), 4.08 KiB | 2.04 MiB/s, done.
Total 16 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/afk552/lab2.3
   9fe97e1..cbac3d7  develop -> develop

C:\git\lab2.3>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git\lab2.3>git merge develop
Updating 9fe97e1..cbac3d7
Fast-forward
 2.3_programs/advanced_12.py | 36 ++++++
 2.3_programs/example1.py    |  7 ++++++
 2.3_programs/example2.py    | 13 ++++++
 2.3_programs/example3.py    | 45 ++++++
 2.3_programs/task1_indiv_12.py | 29 ++++++
 2.3_programs/task2_indiv_12.py | 31 ++++++
 2.3_programs/task3_indiv_12.py | 32 ++++++
 7 files changed, 193 insertions(+)
 create mode 100644 2.3_programs/advanced_12.py
 create mode 100644 2.3_programs/example1.py
 create mode 100644 2.3_programs/example2.py
 create mode 100644 2.3_programs/example3.py
 create mode 100644 2.3_programs/task1_indiv_12.py
 create mode 100644 2.3_programs/task2_indiv_12.py
 create mode 100644 2.3_programs/task3_indiv_12.py

C:\git\lab2.3>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/afk552/lab2.3
   9fe97e1..cbac3d7  main -> main

C:\git\lab2.3>
```

Рисунок 22 – Окно командной строки

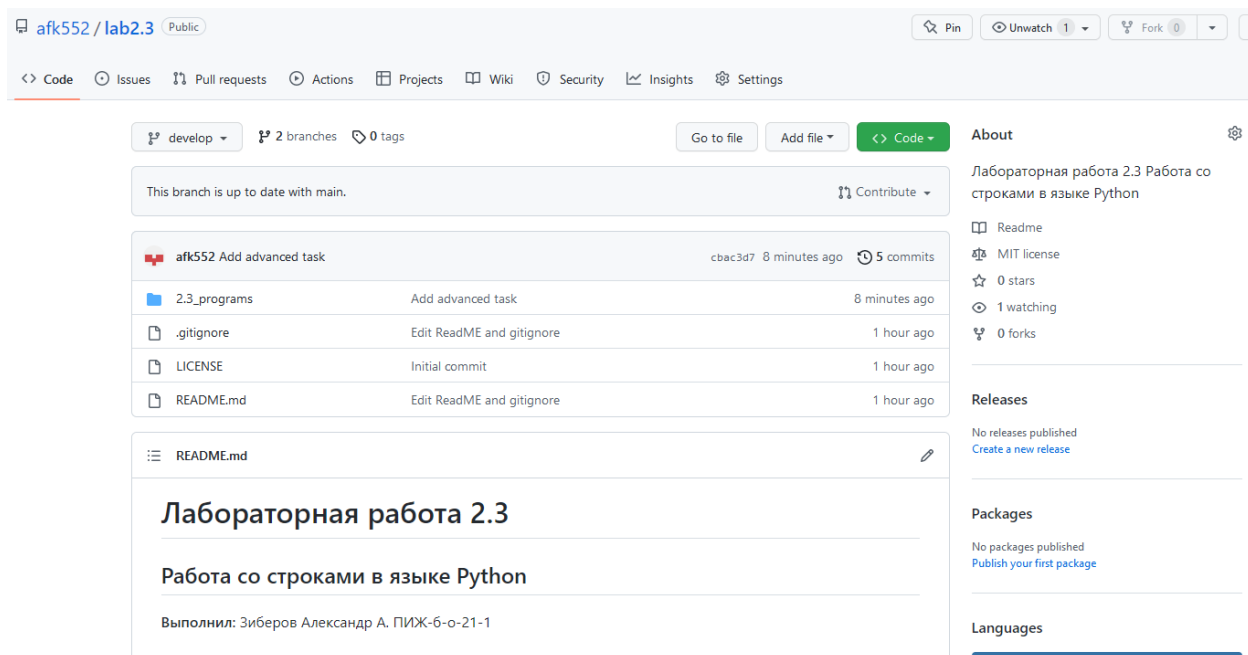


Рисунок 23 – Удаленный репозиторий на GitHub

Вывод: В результате выполнения работы были изучены строки в языке Python версии 3 и получены навыки работы с ними в программировании.

Контрольные вопросы:

1. Что такое строки в языке Python?

Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

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

Строки в апострофах и кавычках, экранированные последовательности – служебные символы, “Сырые” строки, строки в тройных апострофах или кавычках.

3. Какие операции и функции существуют для строк?

Сложение (+), умножение (*), принадлежность подстроки (in), встроенные функции:

len() – длина строки

`str()` – изменяет тип объекта на `string`

`ord()` – Преобразует символ в целое число

`chr()` – Преобразует целое число в символ

4. Как осуществляется индексирование строк?

<название переменной> [число от 0 до длины строки - 1]

5. Как осуществляется работа со срезами для строк?

Если `s` это строка, выражение формы `s[m:n]` возвращает часть `s`, начинающуюся с позиции `m`, и до позиции `n`, но не включая позицию.

6. Почему строки Python относятся к неизменяемому типу данных?

В Python нельзя изменить некоторый одиночный символ в строке, например, через `s[2] = 'z'`, не говоря уже о том, чтобы вставить символ внутрь строки. Но можно сделать это срезами (`s = s[:2] + 'z' = s[3:]`). Только это будет совершенно другая строка, размещённая по совершенно другому адресу в памяти, а `s` — переустановленная ссылка на эту новую строку. Но изменить строку или её длину (её структурность) по текущей ссылке — невозможно. В этом и состоит неизменяемость объекта — это не константность, так как его значение можно изменить, но это будет уже ссылка на другой объект с этим новым значением.

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

<название переменной строки>.`istitle()`

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

`if stroka1 in stroka2`

9. Как найти индекс первого вхождения подстроки в строку?

`stroka1.find(stroka2)`

10. Как подсчитать количество символов в строке?

`len(stroka)`

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

`stroka.count(<символ>)`

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

"F-строки" обеспечивают краткий, читаемый способ включения значения выражений Python внутри строк.

```
name = "Александр"
age = 100
print(f"Меня зовут {name} Мне {age} лет.")
>>> Меня зовут Александр. Мне 100 лет.
```

13. Как найти подстроку в заданной части строки?

```
stroka.find(<подстрока>[, <start>[,<end>]])
```

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

```
name = "Александр"
age = 100
print("Меня зовут {}. Мне {} лет.".format(name, age))
>>> Меня зовут Александр. Мне 100 лет.
```

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

```
stroka.isdigit()
```

16. Как разделить строку по заданному символу?

```
'foo.bar.baz.qux'.rsplit(sep='.')
```

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

```
stroka.islower()
```

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

```
stroka[0].islower()
```

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

Нет, но можно прибавить число как строку, предварительно его преобразовав к типу string при помощи str(<число>).

20. Как «перевернуть» строку?

Общая парадигма для разворота (reverse) строки: stroka = stroka[::-1]

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

`s.join(<iterable>)` возвращает строку, которая является результатом конкатенации объекта `<iterable>` с разделителем `s`.

`'-'.join[<iterable>]`

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

К верхнему регистру – `stroka.upper()`, к нижнему – `stroka.lower()`

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

Первый: `stroka[0].upper()`, второй: `stroka[len(s)-1].upper()`

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

`stroka.isupper()`

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

`s.splitlines()` делит `s` на строки и возвращает их в списке. Любой из следующих символов или последовательностей символов считается границей строки.

Метод `splitlines()` обычно используется программистами для разделения строки на разрывы строк. Он возвращает список всех строк в указанной строке. Эта функция используется для разрыва заданной строки на разрывах строк, таких как `\n` (символы новой строки) или `\r` (возврат каретки) и т. д.

Воспользовался бы, если было бы необходимо получить содержимое строки до переноса на новую строку.

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

`stroka.replace(<что заменить>, <на что заменить>)`

Не написав третий параметр (количество раз), `replace` заменит все вхождения.

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

`string.endswith(<suffix>[, <start>[, <end>]])` – заканчивается,
`string.startswith(<suffix>[, <start>[, <end>]])` – начинается.

`'python'.endswith('yt', 0, 4)`

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

`stroka.isspace()`

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

Строка увеличится в три раза (появятся три копии строки в одной).

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

`stroka.title()`

Метод `str. title()` возвращает копию строки `str`, в которой у каждого слова в строке, первый символ имеет верхний регистр, а остальные символы слова переводятся в нижний регистр.

31. Как пользоваться методом `partition()`?

`string.partition(<sep>)` делит строку на основе разделителя.
`s.partition(<sep>)` отделяет от `s` подстроку длиной от начала до первого вхождения `<sep>`. Возвращаемое значение представляет собой кортеж из трех частей:

Часть `s` до `<sep>`

Разделитель `<sep>`

Часть `s` после `<sep>`

32. В каких ситуациях пользуются методом `rfind()`?

Метод `rfind()` похож на метод `find()`, но он, в отличие от `find()`, просматривает строку не слева направо, а справа налево, возвращая индекс первого найденного вхождения искомой подстроки.

Следовательно, его удобно использовать для нахождения индекса последнего вхождения подстроки в строку.