

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:
Соколов Михаил Романович
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 .gitignore файл для языка 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 ().*

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * PoTtaTto ▾ / **Repository name *** fundamentals_of_software

✔ fundamentals_of_software_engineering_lab2_3 is available.

Great repository names are short and memorable. Need inspiration? How about [improved-fishstick](#) ?

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

.gitignore template: Python ▾

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

Choose a license

License: MIT License ▾

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

This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория с заданными настройками

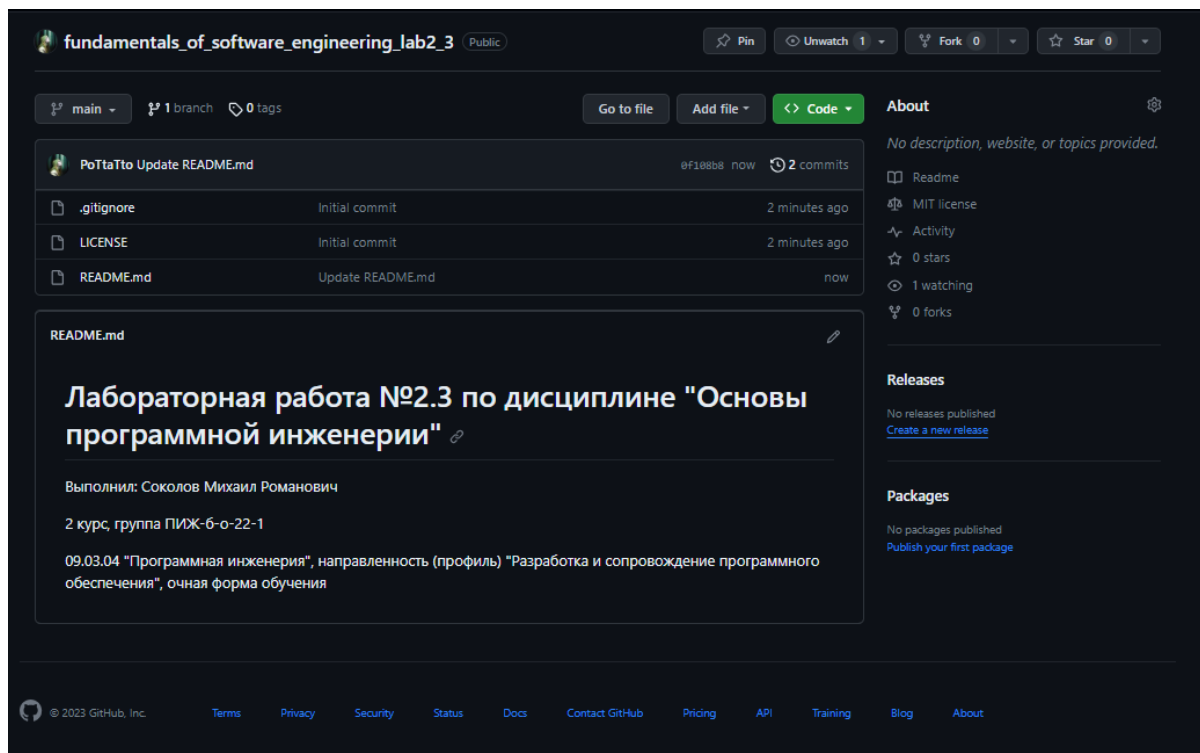


Рисунок 2 – Созданный репозиторий

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.3> git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab2_3
Cloning into 'fundamentals_of_software_engineering_lab2_3'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.3> |
```

Рисунок 3 – Клонирование репозитория

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.3> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.3>
```

Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

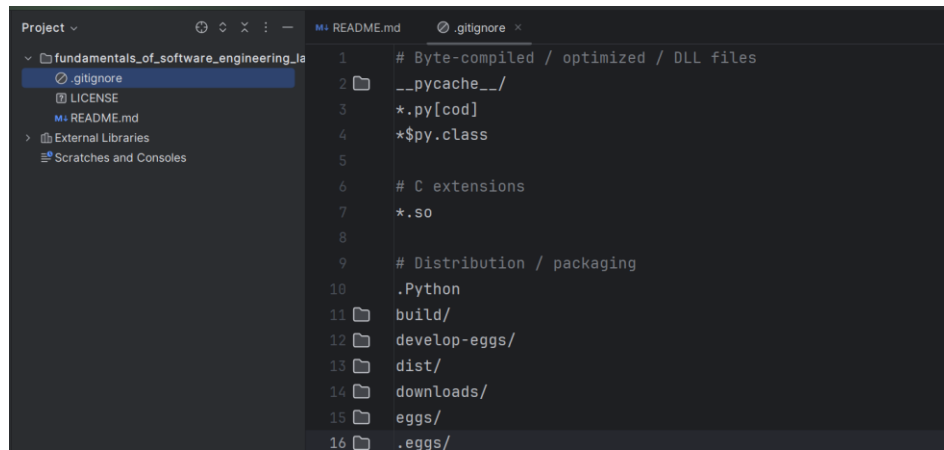


Рисунок 5 – Часть .gitignore, созданного GitHub

2. Проработаем примеры лабораторной работы, фиксируя изменения.
Создадим для каждого примера отдельный модуль языка Python:

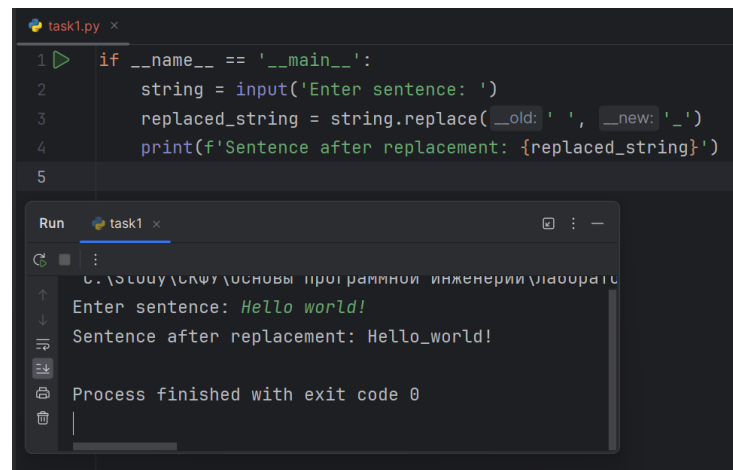


Рисунок 6 – Замена символов пробела на '-' (пример №1) (1)

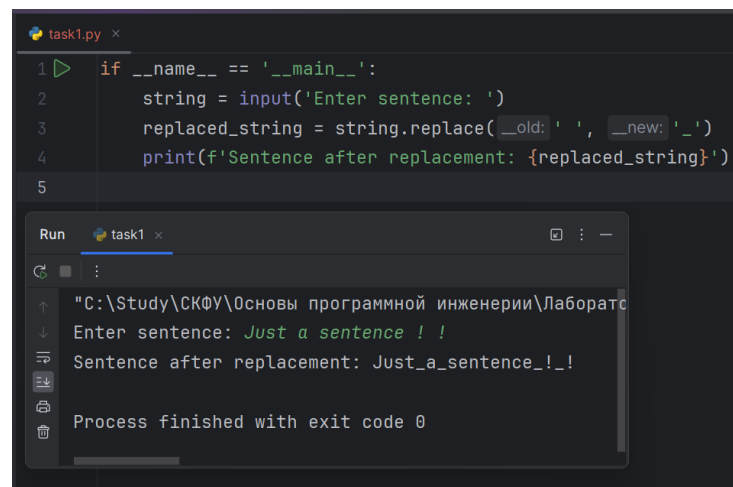
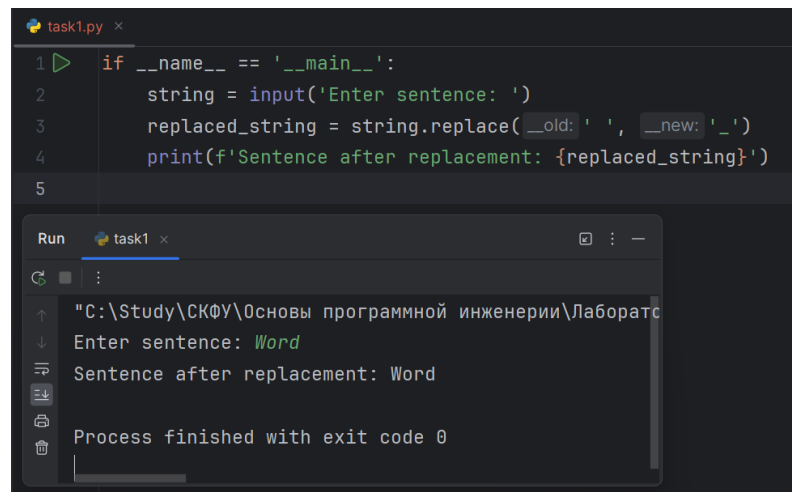


Рисунок 7 – Замена символов пробела на '-' (пример №1) (2)

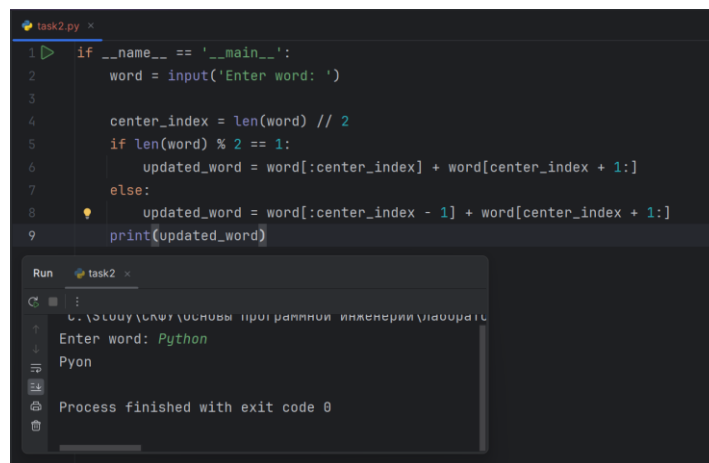


```
task1.py
1 if __name__ == '__main__':
2     string = input('Enter sentence: ')
3     replaced_string = string.replace(__old: ' ', __new: '_')
4     print(f'Sentence after replacement: {replaced_string}')
5

Run task1
"C:\Study\СКФУ\Основы программной инженерии\Лаборатория\
Enter sentence: Word
Sentence after replacement: Word

Process finished with exit code 0
```

Рисунок 8 – Замена символов пробела на '-' (пример №1) (3)

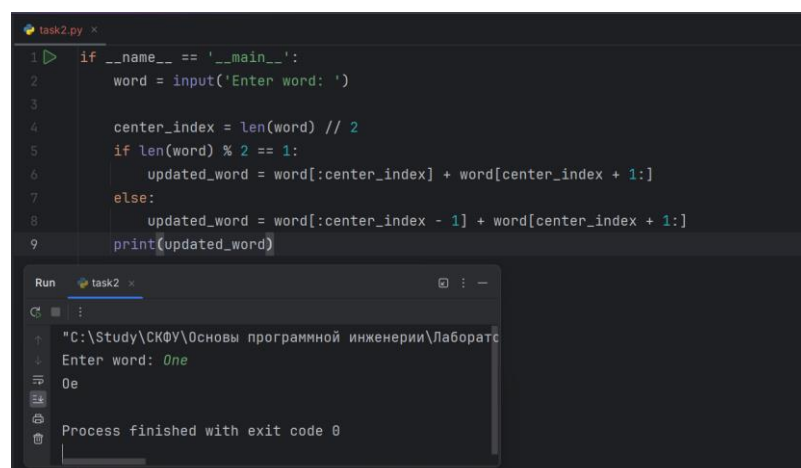


```
task2.py
1 if __name__ == '__main__':
2     word = input('Enter word: ')
3
4     center_index = len(word) // 2
5     if len(word) % 2 == 1:
6         updated_word = word[:center_index] + word[center_index + 1:]
7     else:
8         updated_word = word[:center_index - 1] + word[center_index + 1:]
9     print(updated_word)

Run task2
"C:\Study\СКФУ\Основы программной инженерии\Лаборатория\
Enter word: Python
Pyon

Process finished with exit code 0
```

Рисунок 9 – Удаление среднего символа, если длина слова четная, иначе удалить два средних символа слова (пример №2) (1)



```
task2.py
1 if __name__ == '__main__':
2     word = input('Enter word: ')
3
4     center_index = len(word) // 2
5     if len(word) % 2 == 1:
6         updated_word = word[:center_index] + word[center_index + 1:]
7     else:
8         updated_word = word[:center_index - 1] + word[center_index + 1:]
9     print(updated_word)

Run task2
"C:\Study\СКФУ\Основы программной инженерии\Лаборатория\
Enter word: One
Oe

Process finished with exit code 0
```

Рисунок 10 – Удаление среднего символа, если длина слова четная, иначе удалить два средних символа слова (пример №2) (2)

```
1 import sys
2
3
4 if __name__ == '__main__':
5     string = input('Enter sentence: ')
6     length = int(input('Enter length: '))
7
8     if len(string) >= length:
9         print('Input length must be greater than length of sentence!', file=sys.stderr)
10        exit(1)
11
12    words = string.split()
13    if len(words) < 2:
14        print('Sentence must contain at least several words!', file=sys.stderr)
15        exit(1)
16
17    delta = length - sum([len(word) for word in words])
18    w, r = delta // (len(words) - 1), delta % (len(words) - 1)
19
20    lst = []
21    for i, word in enumerate(words):
22        lst.append(word)
23        if i < len(words) - 1:
24            width = w
25            if r > 0:
26                width += 1
27                r -= 1
28            if width > 0:
29                lst.append(' ' * width)
30    print(''.join(lst))
```

Run task3 x

Enter sentence: hello python world
Enter length: 40
hello python world
Process finished with exit code 0

Рисунок 11 – Вставка пробелов между словами, пока не будет достигнута нужная длина (пример №3) (1)

```
1 import sys
2
3
4 if __name__ == '__main__':
5     string = input('Enter sentence: ')
6     length = int(input('Enter length: '))
7
8     if len(string) >= length:
9         print('Input length must be greater than length of sentence!', file=sys.stderr)
10        exit(1)
11
12    words = string.split()
13    if len(words) < 2:
14        print('Sentence must contain at least several words!', file=sys.stderr)
15        exit(1)
16
17    delta = length - sum([len(word) for word in words])
18    w, r = delta // (len(words) - 1), delta % (len(words) - 1)
19
20    lst = []
21    for i, word in enumerate(words):
22        lst.append(word)
23        if i < len(words) - 1:
24            width = w
25            if r > 0:
26                width += 1
27                r -= 1
28            if width > 0:
29                lst.append(' ' * width)
30    print(''.join(lst))
```

Run task3 x

"C:\Study\СКФУ\Основы программной инженерии\Лаборатория 1\Task3.py"
Enter sentence: hello
Enter length: 10
Sentence must contain at least several words!
Process finished with exit code 1

Рисунок 12 – Вставка пробелов между словами, пока не будет достигнута нужная длина (пример №3) (2)

```
1 import sys
2
3
4 if __name__ == '__main__':
5     string = input('Enter sentence: ')
6     length = int(input('Enter length: '))
7
8     if len(string) >= length:
9         print('Input length must be greater than length of sentence!', file=sys.stderr)
10        exit(1)
11
12    words = string.split()
13    if len(words) < 2:
14        print('Sentence must contain at least several words!', file=sys.stderr)
15        exit(1)
16
17    delta = length - sum([len(word) for word in words])
18    w, r = delta // (len(words) - 1), delta % (len(words) - 1)
19
20    lst = []
21    for i, word in enumerate(words):
22        lst.append(word)
23        if i < len(words) - 1:
24            width = w
25            if r > 0:
26                width += 1
27                r -= 1
28            if width > 0:
29                lst.append(' ' * width)
30    print(' '.join(lst))
```

Run task3

"C:\Study\СКФУ\Основы программной инженерии\Лабораторная

Enter sentence: *hello world*

Enter length: *2*

Input length must be greater than length of sentence!

Process finished with exit code 1

Рисунок 13 – Вставка пробелов между словами, пока не будет достигнута нужная длина (пример №3) (3)

3. Выполним индивидуальные задания и задание повышенной сложности (вариант №7):

```
1 if __name__ == '__main__':
2     sentence = input('Enter sentence: ')
3     [print(letter, end=' ') for letter in sentence if letter == 'и']
```

Run individual_task1

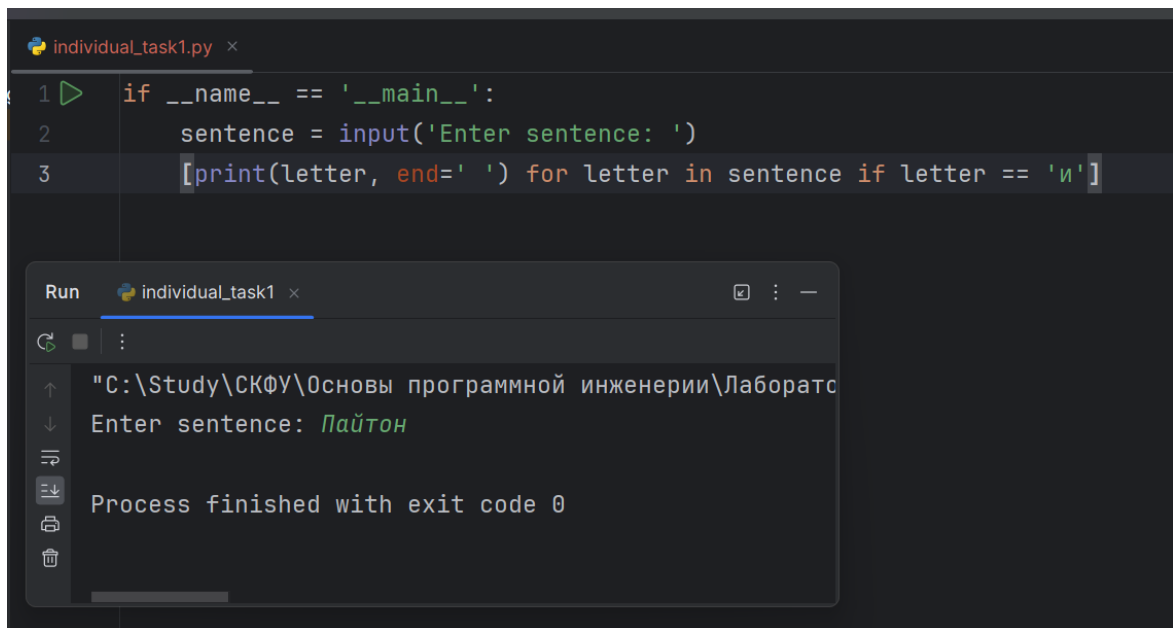
"C:\Study\СКФУ\Основы программной инженерии\Лабораторная

Enter sentence: *Привет мир!*

и и

Process finished with exit code 0

Рисунок 14 – Напечатать все буквы «и» предложения (задание №1) (1)



```
individual_task1.py x
1 if __name__ == '__main__':
2     sentence = input('Enter sentence: ')
3     [print(letter, end=' ') for letter in sentence if letter == 'и']
```

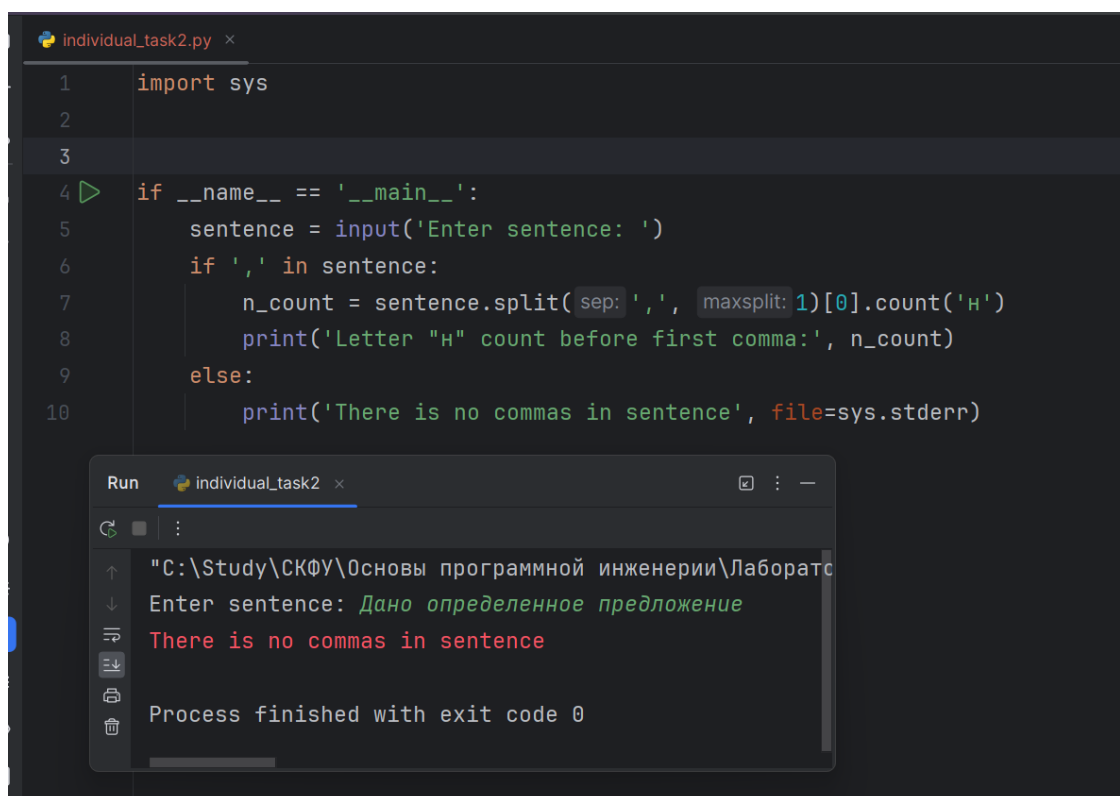
Run individual_task1 x

"C:\Study\СКФУ\Основы программной инженерии\Лаборато

Enter sentence: *Пайтон*

Process finished with exit code 0

Рисунок 15 – Напечатать все буквы «и» предложения (задание №1) (2)



```
individual_task2.py x
1 import sys
2
3
4 if __name__ == '__main__':
5     sentence = input('Enter sentence: ')
6     if ',' in sentence:
7         n_count = sentence.split(sep=',', maxsplit=1)[0].count('н')
8         print('Letter "н" count before first comma:', n_count)
9     else:
10        print('There is no commas in sentence', file=sys.stderr)
```

Run individual_task2 x

"C:\Study\СКФУ\Основы программной инженерии\Лаборато

Enter sentence: *Дано определенное предложение*

There is no commas in sentence

Process finished with exit code 0

Рисунок 16 – Посчитать количество букв «н», предшествующих первой запятой предложения (задание №2) (1)


```
individual_task2.py x
1 import sys
2
3
4 if __name__ == '__main__':
5     sentence = input('Enter sentence: ')
6     if ',' in sentence:
7         n_count = sentence.split(sep=',', maxsplit=1)[0].count('н')
8         print('Letter "н" count before first comma:', n_count)
9     else:
10        print('There is no commas in sentence', file=sys.stderr)

Run individual_task2 x
"C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа
Enter sentence: Дано определенное предложение, рандомной длины
Letter "н" count before first comma: 4
Process finished with exit code 0
```

Рисунок 17 – Посчитать количество букв «н», предшествующих первой запятой предложения (задание №2) (2)

```
individual_task3.py x
1 if __name__ == '__main__':
2     word = input('Enter word: ')
3
4     result_word, letter_count = '', {}
5     for letter in word:
6         if letter not in letter_count:
7             letter_count[letter] = 0
8             letter_count[letter] += 1
9             if letter_count[letter] > 1:
10                continue
11            result_word += letter
12
13    print(result_word)
14

Run individual_task3 x
"C:\Study\СКФУ\Основы программной инженер
Enter word: Python
Python
Process finished with exit code 0
```

Рисунок 18 – Удалить из слова повторяющиеся буквы, оставив их первые вхождения (задание №3) (1)

```
individual_task3.py x
1 if __name__ == '__main__':
2     word = input('Enter word: ')
3
4     result_word, letter_count = '', {}
5     for letter in word:
6         if letter not in letter_count:
7             letter_count[letter] = 0
8             letter_count[letter] += 1
9             if letter_count[letter] > 1:
10                 continue
11         result_word += letter
12
13     print(result_word)
14
```

Run individual_task3 x

"C:\Study\СКФУ\Основы программной инженер

Enter word: *Successfully*

Sucesfly

Process finished with exit code 0

Рисунок 19 – Удалить из слова повторяющиеся буквы, оставив их первые вхождения (задание №3) (2)

```
hard_task.py x
1 if __name__ == '__main__':
2     word1, word2 = [input(f'Enter word{i+1}: ') for i in range(2)]
3     result = ''
4     for letter in word1:
5         if letter not in word2:
6             result += letter
7     for letter in word2:
8         if letter not in word1:
9             result += letter
10    print(result)
11
```

Run hard_task x

"C:\Study\СКФУ\Основы программной инженер

Enter word1: *процессор*

Enter word2: *информация*

пессинфмаия

Process finished with exit code 0

Рисунок 20 – Напечатать уникальные буквы слов, которые есть лишь в одном из них (повторяющиеся тоже). Задание повышенной сложности (1)

```

hard_task.py x
1 if __name__ == '__main__':
2     word1, word2 = [input(f'Enter word{i+1}: ') for i in range(2)]
3     result = ''
4     for letter in word1:
5         if letter not in word2:
6             result += letter
7     for letter in word2:
8         if letter not in word1:
9             result += letter
10    print(result)
11

```

Run hard_task x

```

"C:\Study\СКФУ\Основы программной инженер
Enter word1: питон
Enter word2: железа
питонжелеза

Process finished with exit code 0

```

Рисунок 21 – Напечатать уникальные буквы слов, которые есть лишь в одном из них (повторяющиеся тоже). Задание повышенной сложности (2)

4. Сольем ветки develop и main / master и отправим на удаленный репозиторий:

```

Terminal Local x + v
7ff103fd (HEAD -> develop) hard_task.py is added
cf55cea individual_task3.py is added
589685f individual_task2.py is added
04b2186 individual_task1.py is added
21b301c task2.py and task3.py is added
8950ffd task1.py is added
0f108b8 (origin/main, origin/HEAD, main) Update README.md
0ad2c9a Initial commit
~

```

Рисунок 22 – История коммитов

```

individual_task3.py | 13 ++++++++
individual_task3.py | 13 ++++++++
individual_task3.py | 13 ++++++++
individual_task3.py | 13 ++++++++
individual_task3.py | 13 ++++++++
task1.py            | 4 +++
task2.py            | 9 ++++++
task3.py            | 30 ++++++
8 files changed, 81 insertions(+), 1 deletion(-)
create mode 100644 hard_task.py
create mode 100644 individual_task1.py
create mode 100644 individual_task2.py
create mode 100644 individual_task3.py
create mode 100644 task1.py
create mode 100644 task2.py
create mode 100644 task3.py
venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.3\fundamentals_of_software_engineering_lab2_3>

```

Рисунок 23 – Слияние ветки main с веткой develop

```
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.3\fundamentals_of_software_engineering_lab2_3> git push origin main
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 12 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 2.70 KiB | 2.71 MiB/s, done.
Total 21 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 1 local object.
To https://github.com/PoTtaTto/fundamentals\_of\_software\_engineering\_lab2\_3
 0f108b8..7f103fd  main -> main
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.3\fundamentals_of_software_engineering_lab2_3>
```

Рисунок 24 – Отправка изменений на удаленный репозиторий

Ответы на 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?

Через одинарные, двойные и тройные апострофы или кавычки.

3. Какие операции и функции существуют для строк?

Операции конкатенации (+) и дублирования/умножения (*), проверка вхождения (in). Встроенные функции строк: chr() число в символ, ord() символ в число, len() длина строки, str() преобразование объекта в строку.

4. Как осуществляется индексирование строк?

Часто в языках программирования, отдельные элементы в упорядоченном наборе данных могут быть доступны с помощью числового индекса или ключа. Этот процесс называется индексация. В Python строки являются упорядоченными последовательностями символьных данных и могут быть проиндексированы. Доступ к отдельным символам в строке можно получить, указав имя строки, за которым следует число в квадратных скобках []. Индексация строк начинается с нуля: у первого символа индекс 0, следующего 1 и так далее.

5. Как осуществляется работа со срезами для строк?

Python также допускает возможность извлечения подстроки из строки, известную как “string slice”. Если s это строка, выражение формы s[m:n] возвращает часть s, начинающуюся с позиции m, и до позиции n, но не включая позицию. Существует еще один вариант синтаксиса среза, о котором стоит упомянуть. Добавление дополнительного “:” и третьего индекса означает шаг, который указывает, сколько символов следует пропустить после извлечения каждого символа в срезе.

6. Почему строки Python относятся к неизменяемому типу данных?

Строки в Python считаются неизменяемыми типами данных по нескольким причинам. Во-первых, это обусловлено необходимостью хеширования строк для использования их в качестве ключей в словарях и элементов в множествах. Неизменяемость гарантирует постоянство хеша, что важно для эффективного доступа к данным. Кроме того, неизменяемость обеспечивает безопасность и целостность данных, предотвращая возможные изменения злоумышленниками. Она также поддерживает кэширование строк, оптимизируя использование памяти и повышая производительность. Кроме того, неизменяемость облегчает повторное использование строк в различных частях программы и обеспечивает потокобезопасность, упрощая работу с несколькими потоками. Все эти факторы объясняют выбор Python в пользу неизменяемости строк.

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

Методом `istitle()`.

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

С помощью оператора `in` или `not in`.

9. Как найти индекс первого вхождения подстроки в строку?

Методом `find()`.

10. Как подсчитать количество символов в строке?

Методом `len()`.

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

Методом `count()`.

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

В Python версии 3.6 был представлен новый способ форматирования строк. Эта функция официально названа литералом отформатированной строки, но обычно упоминается как f-строки (f-string). Возможности форматирования строк огромны и не будут подробно описаны здесь. Одной

простой особенностью f-строк, которые вы можете начать использовать сразу, является интерполяция переменной. Вы можете указать имя переменной непосредственно в f-строковом литерале (f'string'), и python заменит имя соответствующим значением.

13. Как найти подстроку в заданной части строки?

Методом `find(<start>, <end>)`. `start` – начальный индекс поиска, `end` – конечный индекс поиска (поиск происходит до него).

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

Иногда (а точнее, довольно часто) возникают ситуации, когда нужно сделать строку, подставив в неё некоторые данные, полученные в процессе выполнения программы (пользовательский ввод, данные из файлов и т. д.). Подстановку данных можно сделать с помощью форматирования строк. Форматирование можно сделать с помощью оператора `%`, либо с помощью метода `format`, либо с помощью так называемых f-строк. Форматирование с помощью оператора `%` относится к устаревшим способам форматирования.

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{} {}, {}'.format('a', 'b', 'c')
'a, b, c'
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
>>> '{2}, {1}, {0}'.format(*'abc')
'c, b, a'
>>> '{0}{1}{0}'.format('abra', 'cad')
'abracadabra'
>>> 'Coordinates: {latitude}, {longitude}'.format(latitude='37.24N',
longitude='-115.81W')
'Coordinates: 37.24N, -115.81W'
>>> coord = {'latitude': '37.24N', 'longitude': '-115.81W'}
>>> 'Coordinates: {latitude}, {longitude}'.format(**coord)
'Coordinates: 37.24N, -115.81W'
```

Рисунок 25 – Примеры использования `.format()`

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

Методом `isdigit()`.

16. Как разделить строку по заданному символу?
Методом `split()`.
17. Как проверить строку на то, что она составлена только из строчных букв?
Методом `islower()`.
18. Как проверить то, что строка начинается со строчной буквы?
Можно использовать конструкцию: `<str>[0].islower()`.
19. Можно ли в Python прибавить целое число к строке?
Если предварительно число обернуть функцией `str()`.
20. Как «перевернуть» строку?
Можно воспользоваться конструкцией: `<str>[::-1]`.
21. Как объединить список строк в одну строку, элементы которой разделены дефисами?
С помощью конструкции: `'-'.join(<string_list>)`.
22. Как привести всю строку к верхнему или нижнему регистру?
Используя методы `upper()` и `lower()`.
23. Как преобразовать первый и последний символы строки к верхнему регистру?
`<str>[0].upper()` и `<str>[-1].upper()`
24. Как проверить строку на то, что она составлена только из прописных букв?
Методом `isupper()`.
25. В какой ситуации вы воспользовались бы методом `splitlines()`?
Метод `splitlines()` в Python используется для разделения строки на подстроки на основе следующих символов: `\n`, `\r`, `\r\n`, `\v` или же `\x0b`, `\f` или же `\x0c`, `\x1c`, `\x1d`, `\x1e`, `\x85`, `\u2028`, `\u2029`.

```
>>> 'foo\nbar\r\nbaz\fqux\u2028quux'.splitlines()
['foo', 'bar', 'baz', 'qux', 'quux']
```

Рисунок 26 – Пример, когда можно воспользоваться методом `splitlines()`

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

Методом `replace()`.

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

Методами `startswith()` или `endswith()`

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

Методом `isspace()`.

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

Если вы умножите строку на целое число, то она повторится указанное количество раз. Например, если у вас есть строка "abc" и вы умножите ее на 3, то получите новую строку "abcabcabc".

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

Методом `capitalize()`.

31. Как пользоваться методом `partition()`?

`s.partition()` отделяет от `s` подстроку длиной от начала до первого вхождения. Возвращаемое значение представляет собой кортеж из трех частей.

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

`rfind()` ищет в строке заданную подстроку, начиная с конца.