

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 №6**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Магомедов Имран Борисович  
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., кандидат технических  
наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

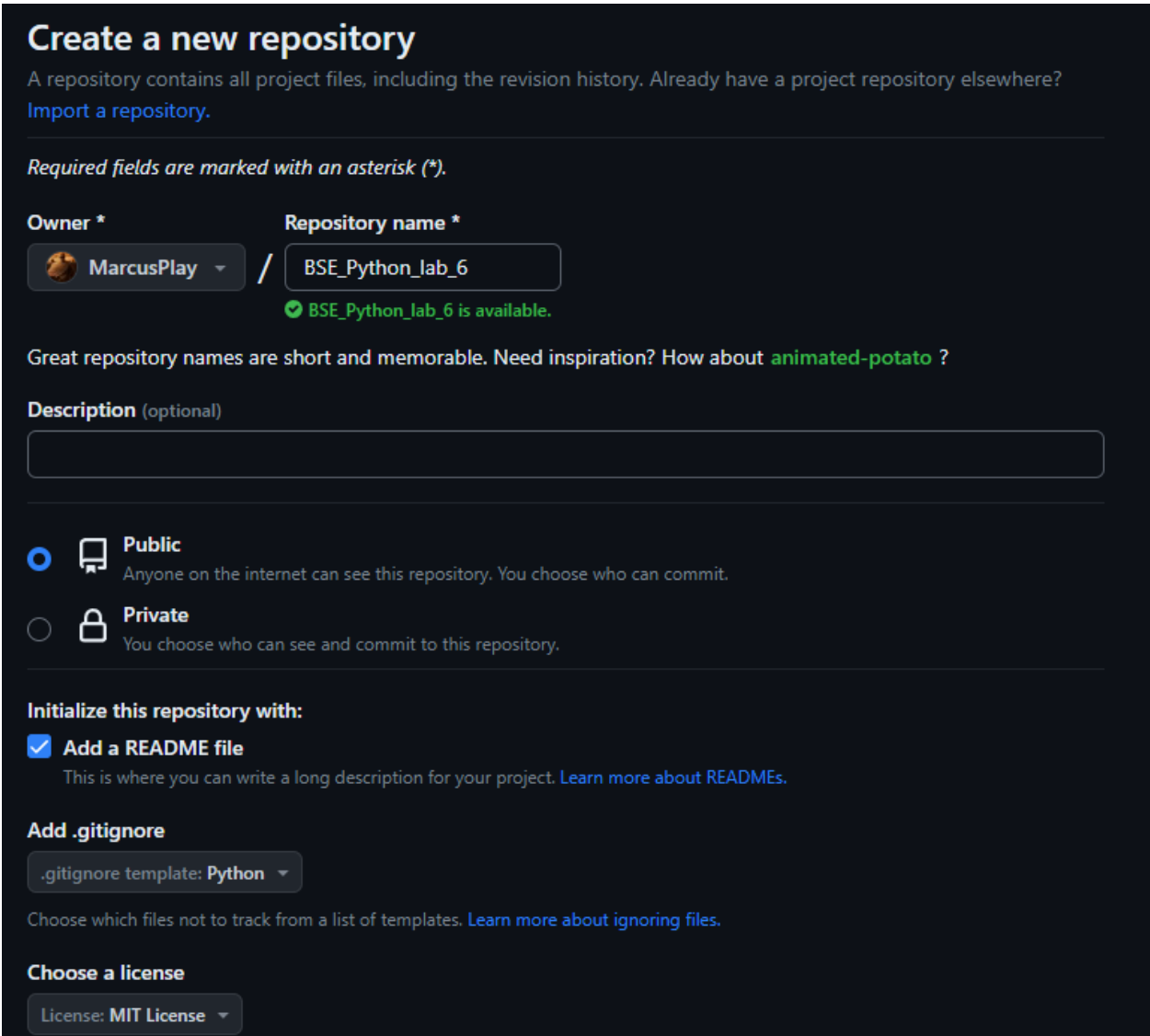
Ставрополь, 2023 г.

## 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.

### Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, it states 'Required fields are marked with an asterisk (\*)'. The 'Owner' field is set to 'MarcusPlay' and the 'Repository name' field is 'BSE\_Python\_lab\_6'. A green checkmark indicates 'BSE\_Python\_lab\_6 is available.'. Below the repository name, it says 'Great repository names are short and memorable. Need inspiration? How about [animated-potato](#) ?'. The 'Description' field is optional and empty. The 'Visibility' section has two options: 'Public' (selected) and 'Private'. The 'Initialize this repository with:' section has a checked box for 'Add a README file'. The 'Add .gitignore' section has a dropdown menu set to 'Python'. The 'Choose a license' section has a dropdown menu set to 'MIT License'.

**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 \* / Repository name \*

MarcusPlay / BSE\_Python\_lab\_6

✓ BSE\_Python\_lab\_6 is available.

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

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

3. Выполните клонирование созданного репозитория.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

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

8. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры.

### Пример 1.

```
example_1.py > ...
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}")
```

### Вывод:

```
m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6/example_tasks develop python3 example_1.py
Введите предложение: Сегодняшний день прекрасен для прогулки в парке.
Предложение после замены: Сегодняшний_день_прекрасен_для_прогулки_в_парке.
```

### Пример 2.

```

example_2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      word = input("Введите слово: ")
6
7      idx = len(word) // 2
8      if len(word) % 2 == 1:
9          # Длина слова нечетная.
10         r = word[:idx] + word[idx+1:]
11     else:
12         # Длина слова четная.
13         r = word[:idx-1] + word[idx+1:]
14
15     print(r)

```

### Вывод:

```

m4gomedovaKaskad ~ /work/BSE/lab_6/BSE_Python_lab_6/example_tasks  python3 example_2.py
Введите слово: встреча
встреча
m4gomedovaKaskad ~ /work/BSE/lab_6/BSE_Python_lab_6/example_tasks  python3 example_2.py
Введите слово: холод
холод

```

### Пример 3.

```

example_3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      s = input("Введите предложение: ")
9      n = int(input("Введите длину: "))
10
11     # Проверить требуемую длину.
12     if len(s) >= n:
13         print(
14             "Заданная длина должна быть больше длины предложения",
15             file=sys.stderr
16         )
17         exit(1)
18
19     # Разделить предложение на слова.
20     words = s.split(' ')
21     # Проверить количество слов в предложении.
22     if len(words) < 2:
23         print(
24             "Предложение должно содержать несколько слов",
25             file=sys.stderr
26         )

```

### Вывод:

```

x m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6/example_tasks > python3 example_3.py
Введите предложение: Вчера вечером мы посетили новый ресторан в центре города.
Введите длину: 50
Заданная длина должна быть больше длины предложения
x m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6/example_tasks > python3 example_3.py
Введите предложение: Вчера вечером мы посетили новый ресторан в центре города.
Введите длину: 100
Вчера          вечером          мы          посетили          новый          ресторан          в          центре          города.

```

Зафиксировал изменения в репозиторий.

```

m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6/example_tasks > git commit -m "added example tasks"
[develop 390b2aa] added example tasks
3 files changed, 79 insertions(+)
create mode 100644 example_tasks/example_1.py
create mode 100644 example_tasks/example_2.py
create mode 100644 example_tasks/example_3.py

```

9. Выполните индивидуальные задания, согласно своему варианту. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

### Индивидуальное задание 1 (12 – вариант).

Дано предложение. Вывести все имеющиеся в нем буквосочетания нн.

```
individual_task_1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Дано предложение. Вывести все имеющиеся в нем буквосочетания нн.
5
6  if __name__=="__main__":
7      substring = 'Ваше предложение с буквосочетанием нн и еще немного нн.'
8
9      for i in range(len(substring) - 1):
10         if substring[i:i+2] == "нн":
11             print(f"{i}:{i+2} == нн" , end="\n")
```

### Вывод:

```
m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6$ python3 individual_task_1.py
35:37 == нн
52:54 == нн
```

### Индивидуальное задание 2 (12 – вариант).

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.

```

individual_task_2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Дано предложение. Напечатать все символы, расположенные между первой и второй
5  # запятыми. Если второй запятой нет, то должны быть напечатаны все символы,
6  # расположенные после единственной имеющейся запятой.
7
8  def find_comma(s):
9      start_comma = 0
10     sentence = ""
11     first_comma = False
12     for i in range(len(s)):
13         if s[i] == ',':
14             if first_comma == False:
15                 first_comma = True
16                 start_comma = i
17             else:
18                 first_comma = False
19                 sentence += s[start_comma + 1:i] + "; "
20                 start_comma = 0
21     if first_comma == True:
22         sentence += s[start_comma + 1:-1]
23         start_comma = 0
24
25     return (sentence)
26
27
28 if __name__ == "__main__":
29     s_1 = "Он пришел домой, снял свою куртку, и сразу пошел в спальню."
30     s_2 = "На столе лежала книга, она была открыта на последней странице."
31
32     print(find_comma(s_1))
33     print(find_comma(s_2))

```

## Вывод:

```

m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6 develop python3 individual_task_2.py
снял свою куртку;
она была открыта на последней странице

```

## Индивидуальное задание 3 (12 – вариант).

Путем вставок и удаления символов исправить ошибки:

- в слове процессор;
- во фразе текстовый файл;
- во фразе программа и алгоритм;

- во фразе процессор и паммать.

```
individual_task_3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Путем вставок и удаления символов исправить ошибки:
5  # в слове прроцессор;
6  # во фразе теекстовыйфайл;
7  # во фразе програма и аллгоритм;
8  # во фразе процессор и паммать.
9
10 if __name__=="__main__":
11     s_1 = "прроцессор"
12     s_2 = "теекстовыйфайл"
13     s_3 = "програма и аллгоритм"
14     s_4 = "процессор и паммать"
15
16     print(s_1[0:2] + s_1[3:6] + 'c' + s_1[6:])
17     print(s_2[0:2] + s_2[3:10] + ' ' + s_2[10:])
18     print(s_3[0:7] + 'м' + s_3[7:13] + s_3[14:])
19     print(s_4[0:6] + 'c' + s_4[6:13] + s_4[14:])
```

### Вывод:

```
m4gomedova@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6 develop ± python3 individual_task_3.py
процессор
текстовый файл
программа и алгоритм
процессор и память
```

### Задание повышенной сложности (12 – вариант).

Даны три слова. Напечатать неповторяющиеся в них буквы.



```

hard_task.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Даны три слова. Напечатать неповторяющиеся в них буквы.
5
6  if __name__=="__main__":
7      word_1 = input()
8      word_2 = input()
9      word_3 = input()
10
11     words_list = []
12     for i in word_1:
13         if i not in word_2 and i not in word_3:
14             words_list.append(i)
15     for i in word_2:
16         if i not in word_1 and i not in word_3:
17             words_list.append(i)
18     for i in word_3:
19         if i not in word_2 and i not in word_1:
20             words_list.append(i)
21
22     print(set(words_list))

```

**Вывод:**

```

m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6 develop python3 hard_task.py
imran
djamal
idris
{'n', 's', 'l', 'j'}

```

10. Зафиксируйте сделанные изменения в репозитории.

```

m4gomedov@Kaskad ~/work/BSE/lab_6/BSE_Python_lab_6 develop + git commit -m "added new basic tasks"
[develop 59db302] added new basic tasks
5 files changed, 80 insertions(+), 1 deletion(-)
create mode 100644 hard_task.py
create mode 100644 individual_task_1.py
create mode 100644 individual_task_2.py
create mode 100644 individual_task_3.py

```

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

12. Выполните слияние ветки для разработки с веткой main / master.
13. Отправьте сделанные изменения на сервер GitHub.
14. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

### **Контрольные вопросы:**

1. **Что такое строки в языке Python?** Строки в Python — это последовательности символов, используемые для хранения текстовой информации. Они могут содержать буквы, цифры, символы и пробелы.
2. **Какие существуют способы задания строковых литералов в языке Python?** Строки можно задавать в Python с использованием одинарных (' '), двойных (" ") или тройных (''' ' ' ' или """" """" ) кавычек. Например: `строка = "Пример строки"`.
3. **Какие операции и функции существуют для строк?** Для строк существует множество операций и методов, таких как конкатенация (+), умножение (\*), доступ по индексу ([]), срезы, методы для поиска, замены, преобразования регистра и многие другие.
4. **Как осуществляется индексирование строк?** Строки индексируются с использованием квадратных скобок, например: `символ = строка[индекс]`. Индексы начинаются с 0 для первого символа.
5. **Как осуществляется работа со срезами для строк?** Срезы в строках задаются с использованием квадратных скобок и двоеточия, например: `подстрока = строка[начало:конец]`. Это позволяет извлекать подстроку из строки.
6. **Почему строки Python относятся к неизменяемому типу данных?** Строки в Python являются неизменяемыми, что означает, что их содержимое нельзя изменить после создания. Можно создать новую строку на основе старой, но не изменять существующую строку.

**7. Как проверить то, что каждое слово в строке начинается с заглавной буквы?** Это можно сделать с помощью метода `istitle()`. Например: `строка.istitle()` вернет `True`, если каждое слово начинается с заглавной буквы.

**8. Как проверить строку на вхождение в неё другой строки?** Это можно сделать с помощью оператора `in`. Например: `подстрока in строка` вернет `True`, если `подстрока` содержится в `строке`.

**9. Как найти индекс первого вхождения подстроки в строку?** Это можно сделать с помощью метода `find()`. Например: `индекс = строка.find(подстрока)` вернет индекс первого вхождения `подстрока` в `строку`, или -1, если подстрока не найдена.

**10. Как подсчитать количество символов в строке?** Это можно сделать с помощью функции `len()`. Например: `длина = len(строка)` вернет количество символов в строке.

**11. Как подсчитать то, сколько раз определённый символ встречается в строке?** Это можно сделать с помощью метода `count()`. Например: `количество = строка.count(символ)` вернет количество вхождений `символа` в `строку`.

**12. Что такое f-строки и как ими пользоваться?** F-строки - это способ форматирования строк, который позволяет вставлять значения переменных внутрь строк с использованием фигурных скобок и префикса `f` перед строкой. Например: `name = "Alice"; f_string = f"Привет, {name}!"`

**13. Как найти подстроку в заданной части строки?** Это можно сделать с помощью метода `find()` с указанием начальной и конечной позиции в строке.

**14. Как вставить содержимое переменной в строку, воспользовавшись методом `format()`?** Метод `format()` позволяет вставлять значения переменных внутрь строки, используя фигурные скобки в строке и

вызывая метод `format()` для строки. Например: `name = "Alice"; строка = "Привет, {}!".format(name)`

**15. Как узнать о том, что в строке содержатся только цифры?** Это можно сделать с помощью методов `isdigit()` или `isnumeric()`. Например: `строка.isdigit()` вернет `True`, если строка содержит только цифры.

**16. Как разделить строку по заданному символу?** Это можно сделать с помощью метода `split()`. Например: `список = строка.split(разделитель)` разделит строку на части, используя `разделитель`.

**17. Как проверить строку на то, что она составлена только из строчных букв?** Это можно сделать с помощью метода `islower()`. Например: `строка.islower()` вернет `True`, если строка состоит только из строчных букв.

**18. Как проверить то, что строка начинается со строчной буквы?** Это можно сделать с помощью метода `islower()` для первого символа строки. Например: `строка[0].islower()` вернет `True`, если строка начинается со строчной буквы.

**19. Можно ли в Python прибавить целое число к строке?** Нет, строки и числа являются разными типами данных, и их нельзя сложить напрямую. Необходимо сначала преобразовать число в строку с помощью функции `str()`, а затем выполнять операцию конкатенации.

**20. Как «перевернуть» строку?** Это можно сделать с помощью среза с отрицательным шагом. Например: `обратная_строка = строка[::-1]` вернет строку в обратном порядке.

**21. Как объединить список строк в одну строку, элементы которой разделены дефисами?** Это можно сделать с помощью метода `join()`. Например: `строка = "-".join(список)` объединит элементы списка в одну строку с дефисами между ними.

**22. Как привести всю строку к верхнему или нижнему регистру?** Это можно сделать с помощью методов ``upper()`` (верхний регистр) и ``lower()`` (нижний регистр). Например: ``строка_верхний_регистр = строка.upper()``, ``строка_нижний_регистр = строка.lower()``.

**23. Как преобразовать первый и последний символы строки к верхнему регистру?** Это можно сделать, создав новую строку с преобразованными символами. Например: ``новая_строка = строка[0].upper() + строка[1:-1] + строка[-1].upper()``.

**24. Как проверить строку на то, что она составлена только из прописных букв?** Это можно сделать с помощью метода ``isupper()``. Например: ``строка.isupper()`` вернет ``True``, если строка состоит только из прописных букв.

**25. В какой ситуации вы воспользовались бы методом ``splitlines()``?** Метод ``splitlines()`` используется для разделения строки на строки, разделенные символом новой строки (``\n``). Это может быть полезно, например, при чтении текстовых файлов с несколькими строками.

**26. Как в заданной строке заменить на что-либо все вхождения некоей подстроки?** Это можно сделать с помощью метода ``replace()``. Например: ``новая_строка = строка.replace(подстрока, замена)`` заменит все вхождения ``подстроки`` на ``замену``.

**27. Как проверить то, что строка начинается с заданной последовательности символов или заканчивается заданной последовательностью символов?** Это можно сделать с помощью методов ``startswith()`` и ``endswith()``. Например: ``строка.startswith(последовательность)`` вернет ``True``, если строка начинается с ``последовательности``.

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

Это можно сделать с помощью метода `isspace()`. Например: `строка.isspace()` вернет `True`, если строка состоит только из пробелов.

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

строки на число создаст новую строку, которая будет содержать исходную строку, повторенную 3 раза. Например: `"abc" * 3` вернет `"abcabcabc"`.

**30. Как привести к верхнему регистру первый символ каждого слова**

**в строке?** Это можно сделать с помощью метода `title()`. Например: `новая_строка = строка.title()` приведет к верхнему регистру первый символ каждого слова.

**31. Как пользоваться методом `partition()`?** Метод

`partition(разделитель)` разделяет строку на три части: часть до первого вхождения `разделителя`, сам `разделитель`, и часть после `разделителя`. Эти три части возвращаются в виде кортежа.

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

`rfind(подстрока)` используется для поиска последнего вхождения `подстроки` в строку. Это может быть полезно, когда необходимо найти последнее вхождение в строку.