

**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  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2023

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 с использованием лицензии MIT и язык 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 (\*).

Owner \*

 MrPlatynum ▾

Repository name \*

/ ProgrammEngineering6

✓ ProgrammEngineering6 is available.

Great repository names are short and memorable. Need inspiration? How about [refactored-rotary-phone](#) ?

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.](#)



You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering6.git
Cloning into 'ProgrammEngineering6'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер

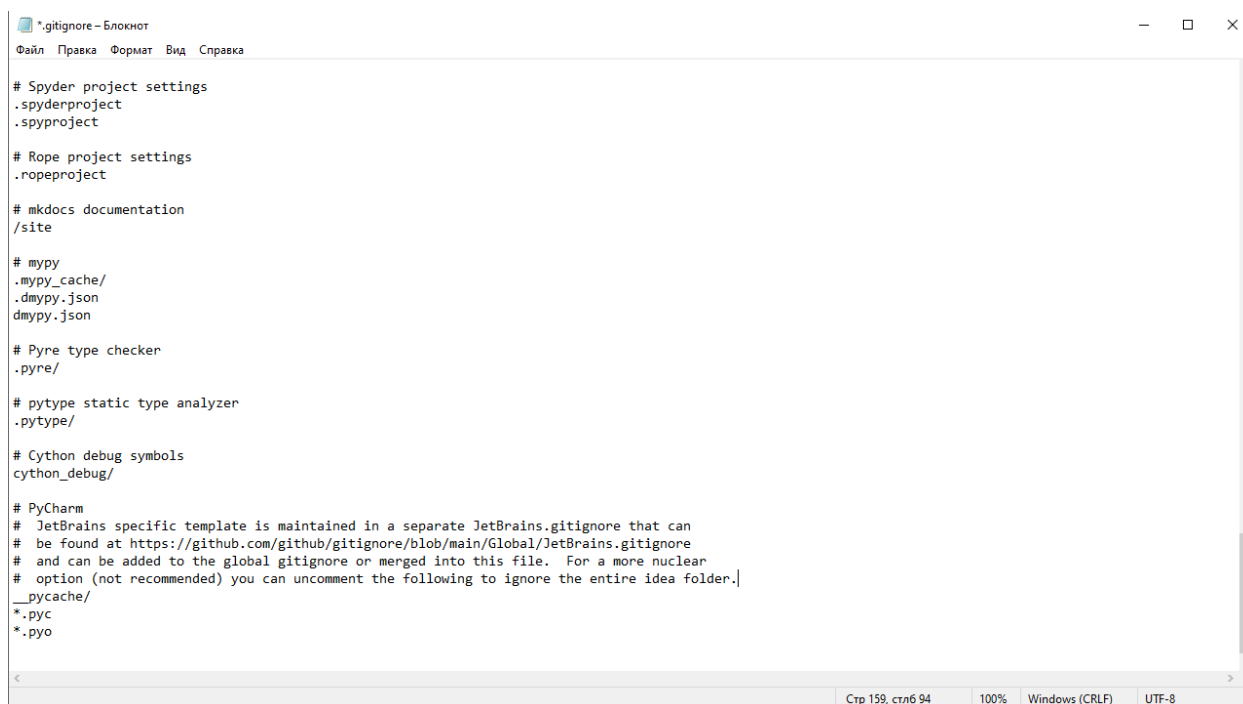


Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering6 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

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

```

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 – Замена пробелов (задание №1)

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexand
Введите предложение: Hello world !
Предложение после замены: Hello_world!

```

Рисунок 6 – Вывод программы (задание №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
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)
16

```

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

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering6/example2.py
Введите слово: abde
abde

```

Рисунок 8 – Вывод программы (задание №2)

```
example3.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import sys
5
6  ▶  if __name__ == '__main__':
7      s = input("Введите предложение: ")
8      n = int(input("Введите длину: "))
9
10     # Проверить требуемую длину.
11     if len(s) >= n:
12         print("Заданная длина должна быть больше длины предложения", file=sys.stderr)
13         exit(1)
14
15     # Разделить предложение на слова.
16     words = s.split(' ')
17
18     # Проверить количество слов в предложении.
19     if len(words) < 2:
20         print("Предложение должно содержать несколько слов", file=sys.stderr)
21         exit(1)
22
23     # Количество пробелов для добавления.
24     delta = n
25     for word in words:
26         delta -= len(word)
27
28     # Количество пробелов на каждое слово.
29     w, r = delta // (len(words) - 1), delta % (len(words) - 1)
30
31     # Сформировать список для хранения слов и пробелов.
32     lst = []
33
34     # Пронумеровать все слова в списке и перебрать их.
35     for i, word in enumerate(words):
36         lst.append(word)
```

Рисунок 9 – Изменение длины строки (задание №3)

```

# 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 10 – Изменение длины строки (задание №3)

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering6/example3.py
Введите предложение: Hello world
Введите длину: 19
Hello          world

```

Рисунок 2 – Вывод программы (задание №3)

### 3. Выполним индивидуальные задания:

```

individual1.py x
1  ▶  #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == "__main__":
5      first_word = input("Enter the first word: ")
6      second_word = input("Enter the second word: ")
7      replaced_word2 = second_word[:len(second_word)] + first_word[len(second_word):]
8
9      print("Result:", replaced_word2)
10

```

Рисунок 3 – Замена букв в слове (задание №1)

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering6/individual1.py
Enter the first word: qwerty
Enter the second word: asdf
Result: asdfty

```

Рисунок 4 – Вывод программы (задание №1)

```
individual2.py x
1 ▶ #!/usr/bin/env python3
2   #- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     sentence = input("Введите предложение (без '-'): ")
6
7     sentence = sentence.lstrip()
8
9     first_space_index = sentence.find(' ')
10    if first_space_index == -1:
11        first_word = sentence
12    else:
13        first_word = sentence[:first_space_index]
14
15    count_o = first_word.lower().count('o')
16
17    print(f"Количество букв 'o' в первом слове: {count_o}")
18
```

Рисунок 14 – Определить количество букв в первом слове (задание №2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering6/individual2.py
Введите предложение (без '-'): аааааааа аааа
аааааааа
Количество букв 'o' в первом слове: 1
```

Рисунок 15 – Запуск программы (задание №2)

```
individual3.py x
1 ▶ #!/usr/bin/env python3
2   #- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     input_string = input("Enter a string: ")
6
7     string_without_spaces = input_string.replace(" ", "")
8
9     print(f"String without spaces: {string_without_spaces}")
10
```

Рисунок 16 – Удалить все пробелы (задание №3)

```
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/3 семестр/Основы программной инженерии/ЛР6_ДушинAB/individual3.py"
Enter a string: fdfsdf fdf f
String without spaces: fdfsdfsfdfsfdf
```

Рисунок 17 – Запуск программы (задание №3)

```
individual4.py x
1 ▶ 1 #!/usr/bin/env python3
2 2 # -*- coding: utf-8 -*-
3
4 ▶ 4 if __name__ == "__main__":
5     word1 = input("Введите первое слово: ")
6     word2 = input("Введите второе слово: ")
7
8     # Преобразование слов в нижний регистр для учёта регистра букв
9     word1 = word1.lower()
10    word2 = word2.lower()
11
12    # Список для хранения ответов (да/нет)
13    results = []
14
15    # Перебор букв первого слова
16    for letter in word1:
17        if letter in word2:
18            results.append("да")
19        else:
20            results.append("нет")
21
22    print(" ".join(results))
23
```

Рисунок 18 – Входят ли буквы первого слова во второе (задание №4)

```
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/De
Введите первое слово: asdf
Введите второе слово: ghja
да нет нет нет
```

Рисунок 19 – Запуск программы (задание №4)

4. Зафиксируем сделанные изменения, сольём ветки и отправим на удаленный репозиторий:



```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering6 (develop)
$ git log
commit 5905687d3a722671b14490749ce94050bf4e0f3d (HEAD -> develop)
Author: MrPlatynum <dushinsasha4@gmail.com>
Date:   Fri Nov 24 21:43:18 2023 +0300

    финальные изменения

commit c0f01bb31a9bea8654994ca79bd043f80713b1f4 (origin/main, origin/HEAD, main)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date:   Fri Nov 24 20:36:32 2023 +0300

    Initial commit
```

Рисунок 20 – Коммиты ветки develop во время выполнения лабораторной работы

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering6 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering6 (main)
$ git merge develop
Updating c0f01bb..5905687
Fast-forward
 example1.py   | 7 ++++++
 example2.py   | 15 ++++++
 example3.py   | 51 ++++++
 individual1.py | 9 ++++++
 individual2.py | 17 ++++++
 individual3.py | 9 ++++++
 individual4.py | 22 ++++++
7 files changed, 130 insertions(+)
create mode 100644 example1.py
create mode 100644 example2.py
create mode 100644 example3.py
create mode 100644 individual1.py
create mode 100644 individual2.py
create mode 100644 individual3.py
create mode 100644 individual4.py
```

Рисунок 21 – Слияние ветки develop в ветку main

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering6 (main)
$ git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 2.57 KiB | 2.57 MiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/MrPlatynum/ProgrammEngineering6.git
   c0f01bb..5905687  main -> main
```

Рисунок 22 – Отправка на удаленный репозиторий

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 последовательностью 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.