

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:

Разворотников Денис Сергеевич  
2 курс, группа ПИЖ-б-о-21-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Проверил:

---

(подпись)

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

Ставрополь 2022

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

1. Был создан общедоступного репозиторий в GitHub в котором были добавлены gitignore, правила для работы с IDE PyCharm с ЯП Python и лицензия MIT, репозиторий был клонировал на локальный сервер и организован в соответствии с моделью ветвления git-flow.

Owner \* / Repository name \*

DenisRazvorotnikov / lab2\_3 ✓

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

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:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

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

License: MIT License ▾

**Create repository**

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

Рисунок 1.1 – Создание общедоступного репозитория

```
denisrazvorotnikov@Air-Denis lab2_3 % git branch -a
* develop
  main
remotes/origin/HEAD -> origin/main
remotes/origin/main
```

Рисунок 1.2 – Репозиторий был организован в соответствии с моделью ветвления git-flow

2. Была создана папка rucharm в которую были помещены примеры.

```
/Users/denisrazvorotnikov/Desktop/lab2_2/PY/bin/python /Users/d
Введите предложение: hy my friends
Предложение после замены: {'hy_my_friends'}

Process finished with exit code 0
```

Рисунок 2.1 – Результат работы первого примера

```
/Users/denisrazvorotnikov/Desktop/lab2_2/PY/bin/python
Введите слово: denis
deis

Process finished with exit code 0
```

Рисунок 2.2 – Результат работы второго примера

```
/Users/denisrazvorotnikov/Desktop/lab2_2/PY/bin/python /Users/den
Введите предложение: hi friends
Введите длину: 12
hi friends

Process finished with exit code 0
```

Рисунок 2.3 – Результат работы третьего примера

### 3. Было выполнено три индивидуальных задания

#### Задание 1

Дан текст. Подсчитать общее число вхождений в него символов «+» и «-».

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

if __name__ == '__main__':
    s = input()
    a = s.count("+")
    b = s.count("-")
    print(a + b)
```

```
/Users/denisrazvorotnikov/Desktop/lab2_2/PY/bin/python
+fds- +fds
3

Process finished with exit code 0
```

Рисунок 3.1 – Результат выполнения программы

#### Задание 2

Дано слово из четного числа букв. Поменять местами первую букву со второй, третью – с четвертой и т. д.

```
#!/usr/bin/env python3
# -- coding: utf-8 --
import sys

if __name__ == '__main__':
    s = input()
    a = list(s)
    if len(a) % 2 == 0:
        a[0], a[1] = s[1], s[0]
        a[2], a[3] = s[3], s[2]
        print(''.join(a))
    else:
        print(
            "Нечетное число букв в слове",
            file=sys.stderr
        )
```

```
/Users/denisrazvorotnikov/Desktop2/bin/  
Киношка  
Нечетное число букв в слове  
  
Process finished with exit code 0
```

Рисунок 3.2 – Результат работы программы во втором случае

### Задание 3

Дано слово из 12 букв. Переставить его буквы следующим способом: первая, двенадцатая, вторая, одиннадцатая, ..., пятая, восьмая, шестая, седьмая.

```
#!/usr/bin/env python3  
# -- coding: utf-8 -  
  
if __name__ == '__main__':  
a = input()  
print(a[: :-1])
```

```
йцукенгшщзф  
фзщшгнекуцй  
  
Process finished with exit code 0
```

Рисунок 3.3 – Результат работы программы с предложением на русском

### Задание повышенной сложности

24. Дано предложение. Напечатать все его различные слова.

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

if __name__ == '__main__':
    stroka = input('Введите предложение:\n')
    if stroka[-1] == '.':
        stroka = stroka[0:-1]
        stroka_set = set(stroka.split(' '))
        print(stroka_set)
    else:
        stroka_set = set(stroka.split(' '))
        print(stroka_set)
```

```
/Users/denisrazvorotnikov/Desktop2/011
Введите предложение:
Предложение из букв
{'из', 'Предложение', 'букв'}
```

Рисунок 3.4 – Результат работы программы

```

denisrazvorotnikov@Air-Denis lab2_3 % git add .
denisrazvorotnikov@Air-Denis lab2_3 % git commit -m "tasks"
[develop d646a01] tasks
12 files changed, 130 insertions(+)
create mode 100644 PY/.idea/.gitignore
create mode 100644 PY/.idea/PY.iml
create mode 100644 PY/.idea/inspectionProfiles/profiles_settings.xml
create mode 100644 PY/.idea/misc.xml
create mode 100644 PY/.idea/modules.xml
create mode 100644 PY/.idea/vcs.xml
create mode 100644 PY/1.py
create mode 100644 PY/2.py
create mode 100644 PY/3.py
create mode 100644 PY/idz1.py
create mode 100644 PY/idz2.py
create mode 100644 PY/idz3.py
denisrazvorotnikov@Air-Denis lab2_3 % git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
denisrazvorotnikov@Air-Denis lab2_3 % git merge develop
Updating 29dc989..d646a01
Fast-forward
 PY/.idea/.gitignore           | 3 ++
 PY/.idea/PY.iml               | 8 ++++
 PY/.idea/inspectionProfiles/profiles_settings.xml | 6 +++
 PY/.idea/misc.xml             | 7 +++
 PY/.idea/modules.xml          | 8 ++++
 PY/.idea/vcs.xml              | 6 +++
 PY/1.py                       | 6 +++
 PY/2.py                       | 13 ++++++
 PY/3.py                       | 55 ++++++++++++++++++++++++++++++++++++++
 PY/idz1.py                    | 4 ++
 PY/idz2.py                    | 12 +++++
 PY/idz3.py                    | 2 +
12 files changed, 130 insertions(+)
create mode 100644 PY/.idea/.gitignore
create mode 100644 PY/.idea/PY.iml
create mode 100644 PY/.idea/inspectionProfiles/profiles_settings.xml
create mode 100644 PY/.idea/misc.xml
create mode 100644 PY/.idea/modules.xml
create mode 100644 PY/.idea/vcs.xml
create mode 100644 PY/1.py
create mode 100644 PY/2.py
create mode 100644 PY/3.py
create mode 100644 PY/idz1.py
create mode 100644 PY/idz2.py
create mode 100644 PY/idz3.py
denisrazvorotnikov@Air-Denis lab2_3 %

```

Рисунок 3.5 – Коммит изменений и слияние ветки develop с main

```
denisrazvorotnikov@Air-Denis lab2_3 % git push git@github.com:DenisRazvorotnikov/lab2_3.git
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (17/17), 3.05 KiB | 1.52 MiB/s, done.
Total 17 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:DenisRazvorotnikov/lab2_3.git
   29dc989..d646a01  main -> main
denisrazvorotnikov@Air-Denis lab2_3 %
```

Рисунок 3.6 – Пуш на удаленный сервер

Вывод: в результате лабораторной работы были приобретены навыки по работе со строками при написании программ с помощью языка программирования Python version .3.x

### 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 в кавычках, экранированные последовательности, "сырые" строки, строки в тройных апострофах или кавычках.

3. Какие операции и функции существуют для строк?

Сложение, дублирование, длина строки, длина строки, извлечение среза и т. д.



4. Как осуществляется индексирование строк?

Доступ к символам в строках основан на операции индексирования – после строки или имени переменной, ссылающейся на строку, в квадратных скобках указываются номера позиций необходимых символов.

5. Как осуществляется работа со срезами для строк?

Есть три формы срезов. Самая простая форма среза: взятие одного символа строки, а именно,  $S[i]$  — это срез, состоящий из одного символа, который имеет номер  $i$ , при этом считая, что нумерация начинается с числа 0. То есть если  $S = \text{'Hello'}$ , то  $S[0] == \text{'H'}$ ,  $S[1] == \text{'e'}$ ,  $S[2] == \text{'l'}$ ,  $S[3] == \text{'l'}$ ,  $S[4] == \text{'o'}$ .

Если указать отрицательное значение индекса, то номер будет отсчитываться с конца, начиная с номера -1.

Срез с двумя параметрами:  $S[a:b]$  возвращает подстроку из  $b$ -а символов, начиная с символа с индексом  $a$ , то есть до символа с индексом  $b$ , не включая его.

6. Почему строки Python относятся к неизменяемому типу данных?

Строки — один из типов данных, которые Python считает неизменяемыми, что означает невозможность их изменять. Python дает возможность изменять (заменять и перезаписывать) строки.

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

`string.istitle()`

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

`string.find()`

9. Как найти индекс первого вхождения подстроки в строку?

`s.partition(<sep>)`

10. Как подсчитать количество символов в строке?

`len(s)`

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

`s.count(<sub>)</sub>`

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

Эти строки улучшают читаемость кода, а также работают быстрее чем другие способы форматирования. F-строки задаются с помощью литерала «f» перед кавычками. Пример: `print(f"Меня зовут {name} Мне {age} лет.")`

13. Как найти подстроку в заданной части строки?

`s.find(значение, начало, конец)`

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

`print('{}'.format(s))`

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

`s.isdigit()`

16. Как разделить строку по заданному символу букв?

`str.split()`

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

`s.isalpha()`

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

`s.istitle()`

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

Нет

20. Как «перевернуть» строку?

`s.reverse()`

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

`str.split('-')`

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

`s.upper()` `s.lower`

23. Как преобразовать первый символ строки к верхнему регистру?

`s.capitalize()`

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

`s.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.

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

подстроки?

`s.replace(old, new)`

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

`str.startswith()` и `str.endswith()`

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

`s.isspace()`

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

`Asd*3 = AsdAsdAsd`

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

`s.title()`

31. Как пользоваться методом `partition()`?

Метод `partition()` разбивает строку при первом появлении строки аргумента и возвращает кортеж, содержащий часть перед разделителем, строку аргумента и часть после разделителя.

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

`s.rfind(<sub>)` возвращает индекс последнего вхождения подстроки `<sub>` в `s`, который соответствует началу `<sub>`.