

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
«Работа со строками в языке Python»**

**Отчет по лабораторной работе № 2.3
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Гасанов Г. М « » 2022г.

Подпись студента_____

Работа защищена « »_____2022г.

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:

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

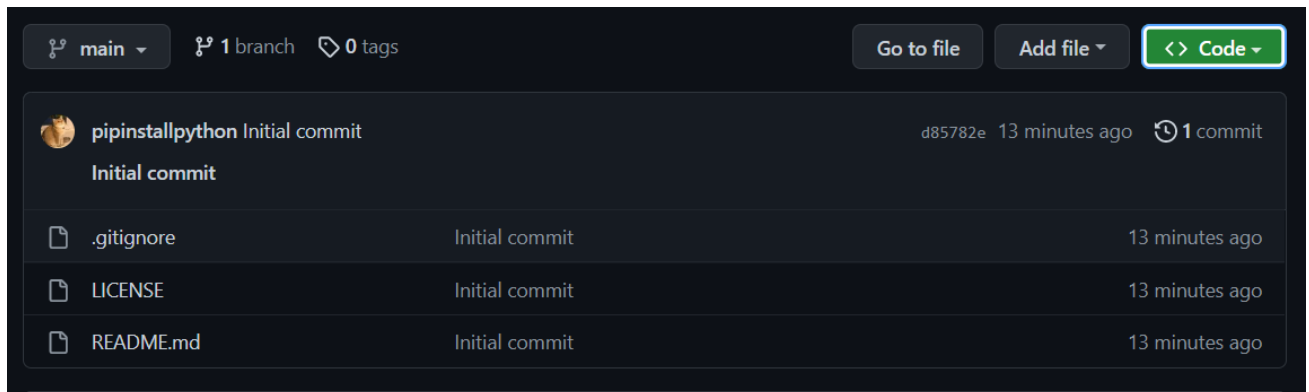


Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
C:\Users\dimu7\Desktop\ОПИ>git clone https://github.com/pipinstallpython/Laba_6.git
Cloning into 'Laba_6'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\dimu7\Desktop\ОПИ\Laba_6>git branch  
develop  
* main
```

Рисунок 3 – организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

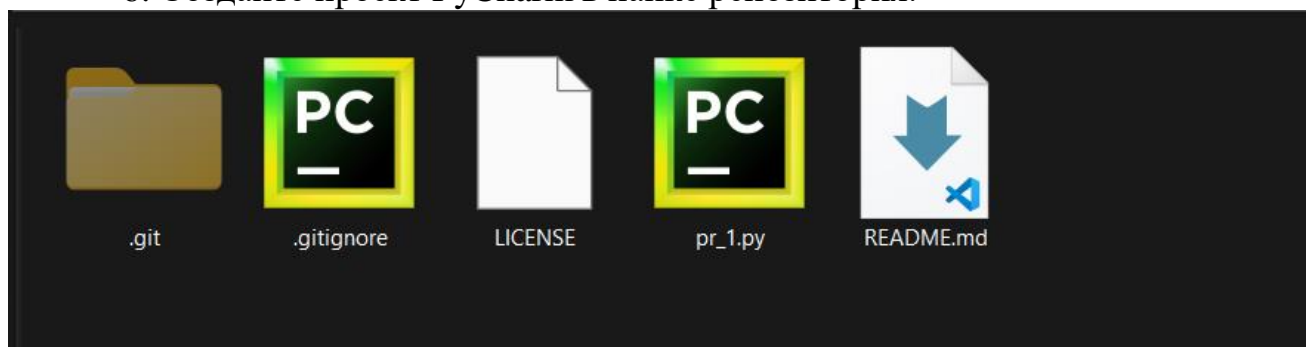


Рисунок 4 – Создание проекта PyCharm в папке репозитория

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

Пример 1. Дано предложение. Все пробелы в нем заменить СИМВОЛОМ «_».

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

if __name__ == '__main__':
    s = input("Введите предложение: ")
    r = s.replace(' ', '_')
    print(f"Предложение после замены: {r}")
```

```
C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe
Введите предложение: мейби бейби лучше твоей девки
Предложение после замены: мейби_бейби_лучше_твоей_девки
```

Рисунок 5 – Результат работы программы

Пример 2. Дано слово. Если его длина нечетная, то удалить среднюю букву, в противном случае – две средние буквы.

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

if __name__ == '__main__':
    word = input("Введите слово: ")

    idx = len(word) // 2
    if len(word) % 2 == 1:
        # Длина слова нечетная.
        r = word[:idx] + word[idx+1:]
    else:
        # Длина слова четная.
        r = word[:idx-1] + word[idx+1:]
    print(r)
```

```
C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe
Введите слово: пион
пион

Process finished with exit code 0
|
```

Рисунок 6 – Результат работы программы

```
C:\Users\dimu7\AppData\Local\Progra
Введите слово: удав
ув

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

Пример 3. Дана строка текста, в котором нет начальных и конечных пробелов. Необходимо изменить ее так, чтобы длина строки стала равна заданной длине (предполагается, что требуемая длина не меньше исходной). Это следует сделать путем вставки между словами дополнительных пробелов. Количество пробелов между отдельными словами должно отличаться не более чем на 1.

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

import sys

if __name__ == '__main__':
    s = input("Введите предложение: ")
    n = int(input("Введите длину: "))

    # Проверить требуемую длину.
    if len(s) >= n:
        print(
            "Заданная длина должна быть больше длины предложения",
            file=sys.stderr
        )
        exit(1)

    # Разделить предложение на слова.
    words = s.split(' ')
    # Проверить количество слов в предложении.
    if len(words) < 2:
        print(
            "Предложение должно содержать несколько слов",
            file=sys.stderr
        )
        exit(1)

    # Количество пробелов для добавления.
    delta = n
    for word in words:
        delta -= len(word)

    # Количество пробелов на каждое слово.
    w, r = delta // (len(words) - 1), delta % (len(words) - 1)

    # Сформировать список для хранения слов и пробелов.
    lst = []
```

```

# Пронумеровать все слова в списке и перебрать их.
for i, word in enumerate(words):
    lst.append(word)

# 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))

```

```

C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python
Введите предложение: Hello World
Введите длину: 5
Заданная длина должна быть больше длины предложения

```

Рисунок 8 – Результат работы программы

```

C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python
Введите предложение: Hello World
Введите длину: 23
Hello           World

Process finished with exit code 0

```

Рисунок 9 – Результат работы программы

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

Задание 1.

9. Дано предложение. Вывести «столбиком» его третий, шестой и т. д. символы.

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

if __name__ == '__main__':
    s = input("Введите предложение: ")
    symb = list(s[2::3])
    print('\n'.join(symb))
```

```
C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe
Введите предложение: fsdafasfsdf
d
a
s
Process finished with exit code 0
```

Рисунок 10 – Результат работы программы

Задание 2.

9. Дано предложение. Определить, есть ли в нем буквосочетания чу или шу. В случае положительного ответа найти также порядковый номер первой буквы первого из них.

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

import sys

if __name__ == '__main__':
    s = input("Введите предложение: ")
    if 'чу' not in s and 'шу' not in s:
        print("Нет слов с буквосочетаниями чу или шу.", file=sys.stderr)
        exit(1)
    else:
        print(s.find('чу'), s.find('шу'))
```

```
Run: ind_2 x
C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe
Введите предложение: ведро с шукой поставили у чулана
26 8
Process finished with exit code 0
```

Рисунок 11 – Результат работы программы

Задание 3.

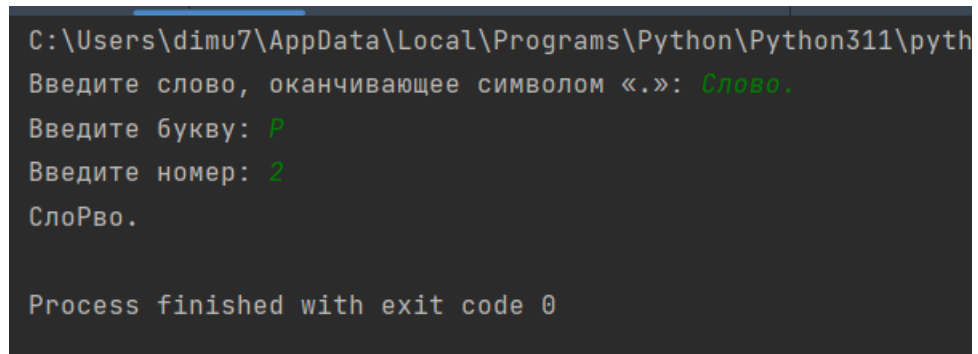
9. Дано слово, оканчивающее символом «.». Составить программу, которая вставляет некоторую заданную букву после буквы с заданным номером.

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

import sys

if __name__ == '__main__':
    s = input("Введите слово, оканчивающее символом «.»: ")
    symb = input("Введите букву: ")
    num = int(input("Введите номер: "))

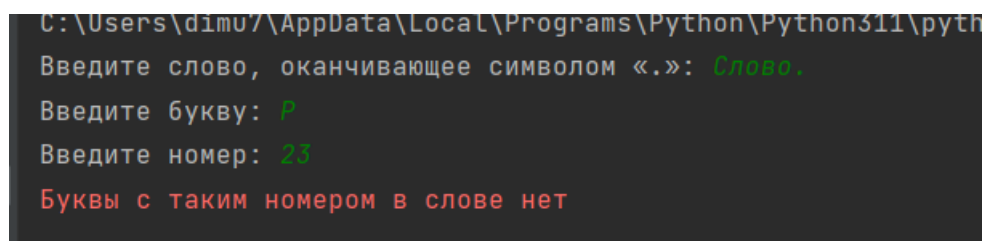
    if s[-1] != '.':
        print("Слово не оканчивается на «.»,", file=sys.stderr)
        exit(1)
    if num > len(s):
        print("Буквы с таким номером в слове нет", file=sys.stderr)
    else:
        print(s[:num+1] + symb + s[num+1:])
```



```
C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe
Введите слово, оканчивающее символом «.»: Слово.
Введите букву: Р
Введите номер: 2
СлоРво.

Process finished with exit code 0
```

Рисунок 12 – Результат работы программы



```
C:\Users\dimu7\AppData\Local\Programs\Python\Python311\python.exe
Введите слово, оканчивающее символом «.»: Слово.
Введите букву: Р
Введите номер: 23
Буквы с таким номером в слове нет
```

Рисунок 13 – Результат работы программы

Задание повышенной сложности.

9. Даны два слова. Определить, можно ли из букв первого из них получить второе.

Рассмотреть два варианта:

- повторяющиеся буквы второго слова могут в первом слове не повторяться;
- каждая буква второго слова должна входить в первое слово столько же раз, сколько и во второе.

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

if __name__ == '__main__':
    word1 = input("Введите первое слово: ")
    word2 = input("Введите второе слово: ")

    # рассмотрели первое условие:
    for symb in word2:
        if symb not in word1:
            print("NO")
            break
    else:
        print("YES")

    # рассмотрели второе условие
    for symb in word1:
        if word2.count(symb) != word1.count(symb):
            print("NO")
            break
        elif symb not in word2:
            print("NO")
            break
    else:
        print("YES")
```

```
C:\Users\dimu7\AppData\Local\Programs\Python
Введите первое слово: инстасамка
Введите второе слово: инстассамка
YES
NO
```

Рисунок 14 – Результат работы программы

Вопросы для защиты работы

1. Что такое строки в языке Python?

Строки в языке Python – это упорядоченные последовательности символов, используемые для хранения и представления текстовой информации.

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

Существует несколько литералов строк – это строки в апострофах(') и в кавычках("), что по сути одно и то же, и строки в тройных апострофах.

```
S = 'привет'
```

```
S = " привет"
```

Причина наличия первых двух вариантов в том, чтобы позволить вставлять в литералы строк символы кавычек или апострофов, не используя экранирование.

Последний способ позволяет для записывать многострочные блоки текста. Например:

```
>>> c = "это очень большая  
... строка, многострочный  
... блок текста"
```

3. Какие операции и функции существуют для строк?

Оператор сложения строк + – соединяет несколько строк в одну, например:

```
S = «Я люблю »
```

```
B = «роллы!»
```

```
S + B = Я люблю роллы!
```

Оператор умножения строк * – повторяет последовательность символов заданное количество раз, например:

```
S = «ма»; S * 2 = мама
```

Оператор принадлежности подстроки `in` – возвращает `true` или `false`, если подстрока входит в строку или нет соответственно.

Функция `chr()` – Преобразует целое число в символ (ASCII or Unicode)

Функция `ord()` – Преобразует символ в целое число (ASCII or Unicode)

Функция `len()` – Возвращает длину строки

Функция `str()` – Изменяет тип объекта на `string`

4. Как осуществляется индексирование строк?

Индексация строк начинается с нуля. Обращение к символу по индексу осуществляется путем записи `string[i]`, где `string` – имя строки, `i` – индекс, соответствующий порядковому номеру символа.

Попытка обращения по индексу большему чем длина строки приводит к ошибке.

Индексы могут быть как положительными, так и отрицательными (`-1` = последний символ, `-2` = предпоследний и т.д.)

Нет индекса, который применим к пустой строке.

5. Как осуществляется работа со срезами для строк?

Срез – это извлечение подстроки из строки.

Если `s` это строка, выражение формы `s[m:n:p]` возвращает часть `s`, начинающуюся с позиции `m`, и до позиции `n`, но не включая последнюю позицию. Последняя переменная означает шаг, означающий сколько символов следует пропустить после извлечения каждого символа в срезе.

Если не указан `m` – срез с начала, т.е. начиная с нулевого элемента.

Если не указан `n` – срез до конца строки

Если не указан `p` – шаг не осуществляется

Если `p < 0` – совершается шаг в обратном направлении

6. Почему строки Python относятся к неизменяемому типу данных?

Фактически работа с неизменяемыми типами данных осуществляется следующим образом: сначала создается объект с определенным значением, берется его адрес и присваивается переменной. При попытке изменить заданное ранее значение создается новый объект, в котором хранится новое значение, берется его адрес и этот адрес присваивается переменной.

В Python нельзя изменить некоторый одиночный символ в строке, например, через `s[2] = 'z'`, не говоря уже о том, чтобы вставить символ внутрь строки.

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

Для проверки того, что каждое слово начинается с заглавной буквы необходимо воспользоваться командой `string.istitle()`, которое возвращает `True` когда `s` не пустая строка и первый алфавитный символ каждого слова в верхнем регистре, а все остальные буквенные символы в каждом слове строчные.

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

Для этого можно воспользоваться `s.count()` возвращает количество точных вхождений подстроки в `s`.

9. Как найти индекс первого вхождения подстроки в строку?

Для этого необходимо воспользоваться командой `string.find(<sub>[, <start>[, <end>]])`, которая возвращает первый индекс в `s` который соответствует началу строки `<sub>` или `-1`, если указанная подстановка не найдена.

10. Как подсчитать количество символов в строке?

Для этого необходимо воспользоваться командой `len() + 1`

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

Для этого можно воспользоваться `string.count(<sub>[, <start> [, <end>]])`

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

f-строки – это один из способов форматирования текста в Python. Одной из отличительных особенностей f-строк, 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?

Для этого необходимо воспользоваться командой `string.find(<sub>[, <start>[, <end>]])`, указав в start и end индексы первого и последнего символа (последнего +1, т.к. он не включается) желаемого интервала.

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

Вводится строка, места, куда необходимо вставить переменную обозначаются как {[индекс переменной]}, за этим следует «.format()», в скобках – переменные через запятую. Если переменная одна {} можно оставить пустыми.

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

Для этого необходимо воспользоваться командой `string.isdigit()`

16. Как разделить строку по заданному символу?

Для этого необходимо воспользоваться командой `split()` (в скобках символ или последовательность символов)

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

Для этого необходимо воспользоваться командой `string.islower()`

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

Для этого необходимо воспользоваться вышеупомянутой командой для первого символа строки `string[0].islower()`

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

Нет

20. Как «перевернуть» строку?

Можно воспользоваться срезом: `имя_строки[::-1]`

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

Можно воспользоваться `'-'.join([список])`

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

`s.upper()` – к верхнему

`s.lower()` – к нижнему

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

С помощью вышеуказанных команд с обращением к индексам `[0] + [1:-1]` (срез середины) + `[-1]`

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

Можно воспользоваться `s.isupper()`

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

Он разделяет строки по символам разрыва строки, таким как «\n», «\r» и другие.

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

Можно воспользоваться `s.replace('что заменить', 'на что заменить')`

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

`s.startswith('...')` – начинается

`s.endswith('...')` – заканчивается

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

Можно воспользоваться `s.isspace()`

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

Она повторится три раза, например «лар» * 3 = «ларларлар»

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

Можно воспользоваться `s.title()`

31. Как пользоваться методом `partition()` ?

Делит строку на основе разделителя. Отделяет от `s` подстроку длиной от начала до первого вхождения. Возвращаемое значение представляет собой кортеж из трех частей: часть до, разделитель, часть после

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

Метод `rfind()` похож на метод `find()`, но он, в отличие от `find()`, ищет с конца.

Вывод: в ходе выполнения практической работы были приобретены навыки по работе со строками при написании программ с помощью языка программирования Python.