

ПЗ №7 Работа со строками в Python

1. Тема работы

Работа со строками в Python. Создание, модификация и обработка строк. Использование строковых методов и операций.

2. Цель работы

Изучить основы работы со строками в Python, освоить методы создания и модификации строк, научиться применять встроенные строковые методы, изучить форматирование строк и работу с Юникод.

3. Задачи работы

- Научиться создавать и форматировать строки различными способами
- Освоить основные строковые методы и операции
- Изучить способы поиска и замены в строках
- Написать 4 программы для работы со строками согласно индивидуальному варианту

4. Необходимое оборудование и ПО

- Компьютер с ОС Windows 10
 - Установленные: VS Code, Python
 - Установленные расширения VS Code: Python, Python Debugger
-

5. Ход работы

Часть 1: Повторение и подготовка

1. Откройте вашу папку проекта в VS Code (например, `Иванов_CA-501`)
2. Создайте новую папку `Practice_7` внутри вашей основной папки
3. Создайте файлы для заданий: `string1.py` , `string2.py` , `string3.py` , `string4.py`

Часть 2: Разбор примера программы

Пример 1: Базовые операции со строками

```
# Демонстрация основных операций со строками

# Создание строк
str1 = "Привет, мир!"
str2 = 'Python programming'
str3 = """Многострочная
строка"""
str4 = "12345"

print("Базовые операции:")
print(f"str1: {str1}")

# Конкатенация и повторение
combined = str1 + " " + str2
repeated = "Python! " * 3
print(f"\nКонкатенация: {combined}")
print(f"Повторение: {repeated}")

# Индексация и срезы
print(f"\nИндексация:")
print(f"Первый символ str1: '{str1[0]}'")
print(f"Последний символ str1: '{str1[-1]}'")
print(f"Срез str1[7:11]: '{str1[7:11]}'")
print(f"Срез с шагом str4[::2]: '{str4[::2]}'")
```

Пример 2: Строковые методы

```
# Демонстрация строковых методов

text = "  Привет, мир! Сегодня прекрасный день.  "
print(f"Исходный текст: '{text}'")

# Базовые методы
print(f"\nstrip(): '{text.strip()}'")
print(f"upper(): '{text.upper()}'")
print(f"lower(): '{text.lower()}'")
print(f"title(): '{text.title()}'")

# Поиск и замена
print(f"\nПоиск и замена:")
print(f"Найти 'мир': позиция {text.find('мир')}")
```

```

print(f"Найти 'Python': позиция {text.find('Python')}")
print(f"Замена 'прекрасный' на 'хороший': {text.replace('прекрасный',
'хороший')}")

# Разделение и соединение
words = text.strip().split()
print(f"\nРазделение на слова: {words}")
print(f"Количество слов: {len(words)}")
print(f"Объединение с '-': {'-'.join(words)}")

# Проверки
print(f"\nПроверки:")
print(f"isalpha(): {'hello'.isalpha()}")
print(f"isdigit(): {'123'.isdigit()}")
print(f"startswith('Привет'): {text.startswith('Привет')}")

```

Пример 3: Форматирование строк

```

# Различные способы форматирования строк

name = "Анна"
age = 25
salary = 50000.50

# Метод format()
print("\nМетод format():")
message2 = "Имя: {}, Возраст: {}, Зарплата: {:.2f}".format(name, age,
salary)
print(message2)

# f-строки (Python 3.6+)
print("\nf-строки:")
message3 = f"Имя: {name}, Возраст: {age}, Зарплата: {salary:.2f}"
print(message3)

# Выражения в f-строках
a, b = 10, 20
print(f"\nВыражения: {a} + {b} = {a + b}")
print(f"Верхний регистр: {name.upper()}")

```

Пошаговый разбор:

1. **Создание строк** - одинарные, двойные, тройные кавычки
2. **Конкатенация** - объединение строк `+`
3. **Повторение** - умножение строк `*`
4. **Индексация** - доступ к символам по индексу
5. **Срезы** - получение подстрок `[start:end:step]`
6. **Строковые методы** - `upper()`, `lower()`, `strip()`, `split()` и др.
7. **Форматирование** - `%`, `format()`, f-строки

Запуск программы:

- Сохраните файл (Ctrl+S)
- Запустите через Ctrl+F5 или в терминале:

```
python string1.py
```

Часть 3: Выполнение задания по варианту

1. **Получите задание** у преподавателя согласно вашему варианту
 2. **Реализуйте программы** в соответствующих файлах
 3. **Протестируйте программы** на разных входных данных
 4. **Убедитесь**, что работа со строками выполняется корректно
-

6. Варианты заданий

Вариант 1:

1. Напишите программу, которая принимает строку и выводит её в обратном порядке.
2. Напишите программу, которая подсчитывает количество гласных и согласных букв в строке.
3. Напишите программу, которая проверяет, является ли строка палиндромом.
4. * Напишите программу, которая реализует простой шифр Цезаря (сдвиг букв на N позиций).

Вариант 2:

1. Напишите программу, которая принимает предложение и выводит каждое слово с заглавной буквы.
2. Напишите программу, которая удаляет все пробелы из строки и проверяет, является ли результат палиндромом.
3. Напишите программу, которая находит самое длинное слово в предложении.

4. * Напишите программу, которая генерирует случайный пароль заданной длины из букв, цифр и специальных символов.

Вариант 3:

1. Напишите программу, которая подсчитывает количество вхождений каждого символа в строке.
2. Напишите программу, которая проверяет, является ли строка корректным email-адресом (содержит @ и точку после @).
3. Напишите программу, которая преобразует строку в "верблюжий регистр" (camelCase).
4. * Напишите программу, которая реализует поиск подстроки в строке (аналог метода find()).

Вариант 4:

1. Напишите программу, которая разделяет строку на список слов и выводит их в алфавитном порядке.
2. Напишите программу, которая проверяет, содержит ли строка только цифры.
3. Напишите программу, которая заменяет все вхождения подстроки в строке.
4. * Напишите программу, которая форматирует номер телефона в стандартный вид (+7 (XXX) XXX-XX-XX).

Вариант 5:

1. Напишите программу, которая переводит строку в верхний регистр и удаляет все знаки препинания.
2. Напишите программу, которая находит все позиции вхождения подстроки в строку.
3. Напишите программу, которая преобразует строку в "змеиный регистр" (snake_case).
4. * Напишите программу, которая реализует RLE-сжатие для строки (aaabcc → a3b1c2).

7. Отчёт по выполнению работы

Отчёт должен содержать:

1. **Цель работы**
2. **Задание по варианту** (текст всех 4 заданий)
3. **Листинги программ** (код всех 4 программ)
4. **Скриншоты выполнения программ** (результаты работы в терминале)

5. Выводы:

- С какими трудностями столкнулись при работе со строками?
 - Какие строковые методы оказались наиболее полезными?
 - В чём преимущество f-строк перед другими способами форматирования?
 - Какая программа была самой сложной и почему?
-

8. Контрольные вопросы

1. Чем строка отличается от списка в Python?
 2. Какие способы создания строк вы знаете?
 3. Как получить доступ к отдельному символу строки?
 4. Что такое срезы строк и как они работают?
 5. Какие основные строковые методы для изменения регистра вы знаете?
 6. Чем отличаются методы `find()` и `index()` ?
 7. Как разделить строку на список подстрок?
 8. Какие способы форматирования строк существуют в Python?
 9. Как проверить, состоит ли строка только из цифр или только из букв?
 10. Что такое экранирование символов и для чего оно нужно?
-

9. Дополнительные задания

1. Реализуйте алгоритм проверки орфографии с использованием словаря
 2. Реализуйте парсер простых арифметических выражений
-

10. Полезные строковые методы для справки

```
# Основные строковые методы
str.upper()      # Преобразовать в верхний регистр
str.lower()      # Преобразовать в нижний регистр
str.strip()      # Удалить пробелы с обоих концов
str.split()      # Разделить строку на список
str.join()       # Объединить список в строку
str.find()       # Найти подстроку (возвращает индекс)
str.replace()    # Заменить подстроку
str.startswith() # Проверить начало строки
str.endswith()   # Проверить конец строки
```

<code>str.isalpha()</code>	# Проверить, состоит ли только из букв
<code>str.isdigit()</code>	# Проверить, состоит ли только из цифр
<code>str.isalnum()</code>	# Проверить, состоит ли только из букв и цифр
<code>str.count()</code>	# Подсчитать количество вхождений