

## Лекция 6

Здравствуйте, уважаемые слушатели!

Тема нашей лекции – Строки и методы строк в Python

План лекции:

1. Основы работы со строками в Python
2. Доступ к символам и срезы
3. Методы строк и форматирования
4. Методы проверки содержимого, поиска и замены
5. Заключение

### Введение

Строки в Python являются одними из наиболее важных и широко используемых типов данных, представляющих собой последовательности символов, заключенных в кавычки. Благодаря своей универсальности и богатой встроенной поддержке методов, строки в Python позволяют выполнять широкий спектр операций: от простого форматирования до сложных манипуляций с текстом. В данной лекции рассмотрены основные особенности строк в Python, включая создание, доступ к элементам, основные методы и практические примеры их использования.

### Основы работы со строками в Python

#### Создание строк

Строки в Python можно создавать, заключая последовательность символов в одинарные, двойные или тройные кавычки. Тройные кавычки используются для многострочных строк:

```
python
Копировать код
single_quoted = 'Привет, мир!'
double_quoted = "Привет, мир!"
triple_quoted = """Это
многострочная
строка"""
```

#### Иммутабельность строк

Строки в Python являются неизменяемыми (immutable), что означает, что после создания строку нельзя изменить. Попытки изменения содержимого строки приведут к возникновению ошибки.

#### Доступ к символам и срезы

Доступ к символам строки осуществляется с помощью индексации, начиная с нуля. Срезы позволяют получить подстроку, используя начальный и конечный индексы:

```
python
Копировать код
s = "Hello, world!"
print(s[1]) # e
print(s[0:5]) # Hello
```

## Методы строк

Python предоставляет богатый набор методов для работы со строками, позволяющий выполнять различные типы манипуляций и проверок.

## Методы форматирования

- `upper()`, `lower()`: преобразование строки к верхнему или нижнему регистру.

```
python
Копировать код
"python".upper() # PYTHON
"PYTHON".lower() # python
```

- `title()`: преобразование первых символов всех слов в строке к верхнему регистру.

```
python
Копировать код
"hello world".title() # Hello World
```

- `strip()`, `rstrip()`, `lstrip()`: удаление пробельных символов с начала и/или конца строки.

```
python
Копировать код
" hello world ".strip() # 'hello world'
```

## Методы поиска и замены

- `find()`, `rfind()`: поиск подстроки в строке с возвратом индекса первого вхождения (или последнего для `rfind`).

```
python
Копировать код
```

```
"hello world".find("world") # 6
```

- `replace(old, new)`: замена всех вхождений подстроки на новую подстроку.

```
python
Копировать код
"hello world".replace("world", "Python") # hello Python
```

## Методы проверки содержимого

- `isdigit()`, `isalpha()`, `isspace()`, `islower()`, `isupper()`: проверка, состоит ли строка из цифр, букв, пробелов, или все символы в нижнем/верхнем регистре.

```
python
Копировать код
"123".isdigit() # True
"abc".isalpha() # True
```

## Методы разбиения и соединения

- `split(sep=None)`: разбиение строки по разделителю `sep` на список подстрок.

```
python
Копировать код
"hello world".split() # ['hello', 'world']
```

- `join(iterable)`: соединение элементов итерируемого объекта в одну строку, используя строку, на которой вызван метод, как разделитель.

```
python
Копировать код
", ".join(["apple", "banana", "cherry"]) # apple, banana, cherry
```

## Продвинутое использование строк

### Форматирование строк

Форматирование строк в Python можно выполнить несколькими способами, включая оператор `%`, метод `format()` и f-строки (с версии Python 3.6).

```
python
Копировать код
name = "Alice"
age = 30
```

```
print(f"{name} is {age} years old.") # Alice is 30 years old.
```

## **Работа с юникодом**

Строки в Python 3.x являются Unicode строками по умолчанию, что упрощает обработку текста на различных языках:

```
python
Копировать код
s = "Привет, мир!"
print(s) # Привет, мир!
```

## **Заключение**

Строки в Python обладают множеством методов для их обработки, что делает их неотъемлемой частью любой программы на Python. Знание и умелое использование методов строк позволяет значительно упрощать и ускорять разработку программ, делая код более читаемым и эффективным. Владение строками и их методами является ключевым навыком для любого разработчика, работающего с Python.