Тема урока: строки

1. Методы `count()`
2. Метод `startswith()`
3. Метод `endswith()`
4. Метод `find()`
5. Метод `rfind()`
6. Метод `index()`
7. Метод `rindex()`
8. Метод `strip()`
9. Метод `lstrip()`
10. Метод `rstrip()`
11. Метод `replace()`

Аннотация. Строковый тип данных, основные методы поиска и замены.

Поиск и замена

Методы поиска и замены строк внутри других строк.

Каждый метод в этой группе поддерживает необязательные аргументы `<start>` и `<end>`. Как и в строковых срезах, действие метода ограничено частью исходной строки, начинающейся с позиции символа `<start>` и продолжающейся вплоть до позиции символа `<end>`, но не включающей её. Если параметр `<start>` указан, а параметр `<end>` нет, то метод применяется к части исходной строки от `<start>` до конца строки. Если параметры не заданы, то подразумевается, что `<start> = 0`, `<end> = len(s)`.

Метод count()

Метод `count(<sub>, <start>, <end>)` считает количество **непересекающихся** вхождений подстроки `<sub>` в исходную строку `s`.

Приведённый ниже код:

```
s = 'foo goo moo'
print(s.count('oo'))
print(s.count('oo', 0, 8))  # подсчет с 0 по 7 символ
```

выводит:

```
3
2
```

Метод startswith()

Метод `startswith(<prefix>, <start>, <end>)` определяет, **начинается** ли исходная строка `s` подстрокой `<prefix>`. Метод возвращает значение `True`, если исходная строка начинается с подстроки `<prefix>`, или `False` в противном случае.

Приведённый ниже код:

```
s = 'foobar'
print(s.startswith('foo'))
print(s.startswith('baz'))
```

выводит:

```
True
False
```

Метод endswith()

Метод `endswith(<suffix>, <start>, <end>)` определяет, **оканчивается** ли исходная строка `s` подстрокой `<suffix>`. Метод возвращает значение `True`, если исходная строка оканчивается на подстроку `<suffix>`, или `False` в противном случае.

Приведённый ниже код:

```
s = 'foobar'
print(s.endswith('bar'))
print(s.endswith('baz'))
```

выводит:

```
True
False
```

Методы find(), rfind()

Метод `find(<sub>, <start>, <end>)` находит **индекс первого вхождения** подстроки `<sub>` в исходной строке `s`. Если строка `s` не содержит подстроки `<sub>`, то метод возвращает значение `-1`. Мы можем использовать данный метод наравне с оператором `in` для проверки: содержит ли заданная строка некоторую подстроку или нет.

Приведённый ниже код:

```
s = 'foo bar foo baz foo qux'
print(s.find('foo'))
print(s.find('bar'))
print(s.find('qu'))
print(s.find('python'))
```

выводит:

```
0
4
20
-1
```

Метод `rfind(<sub>, <start>, <end>)` идентичен методу `find(<sub>, <start>, <end>)`, за тем исключением, что он ищет первое вхождение подстроки `<sub>`, начиная с конца строки `s`.

Методы index(), rindex()

Метод `index(<sub>, <start>, <end>)` идентичен методу `find(<sub>, <start>, <end>)`, за тем исключением, что он **вызывает ошибку** `ValueError: substring not found` во время выполнения программы, если подстрока `<sub>` не найдена.

Метод `rindex(<sub>, <start>, <end>)` идентичен методу `index(<sub>, <start>, <end>)`, за тем исключением, что он ищет первое вхождение подстроки `<sub>`, начиная с конца строки `s`.

Методы `find()` и `rfind()` являются более безопасными, чем `index()` и `rindex()`, так как не приводят к возникновению ошибки во время выполнения программы.

Метод strip()

Метод `strip()` возвращает копию строки `s`, у которой удалены все пробелы, стоящие **в начале и конце** строки.

Приведённый ниже код:

```
s = '    foo bar foo baz foo qux    '
print(s.strip())
```

выводит:

```
foo bar foo baz foo qux
```

Метод lstrip()

Метод `lstrip()` возвращает копию строки `s`, у которой удалены все пробелы, стоящие **в начале** строки.

Приведённый ниже код:

```
s = '    foo bar foo baz foo qux    '
print(s.lstrip())
```

выводит (символом `_` обозначены пробелы):

```
foo bar foo baz foo qux____
```

Метод rstrip()

Метод `rstrip()` возвращает копию строки `s`, у которой удалены все пробелы, стоящие **в конце** строки.

Приведённый ниже код:

```
s = '    foo bar foo baz foo qux    '
print(s.rstrip())
```

выводит (символом `_` обозначены пробелы):

```
____foo bar foo baz foo qux
```

Метод replace()

Метод `replace(<old>, <new>)` возвращает копию `s` **со всеми** вхождениями подстроки `<old>`, заменёнными на `<new>`.

Приведённый ниже код:

```
s = 'foo bar foo baz foo qux'
print(s.replace('foo', 'grault'))
```

выводит:

```
grault bar grault baz grault qux
```

Метод `replace()` может принимать необязательный третий аргумент `<count>`, который определяет количество замен.

Приведённый ниже код:

```
s = 'foo bar foo baz foo qux'
print(s.replace('foo', 'grault', 2))
```

выводит:

```
grault bar grault baz foo qux
```

Примечания

Примечание. Методы `strip()`, `lstrip()`, `rstrip()` могут принимать на вход необязательный аргумент `<chars>`. Необязательный аргумент `<chars>` – это строка, которая определяет набор символов для удаления.

Приведённый ниже код:

```
s = '-+-+abc+-+-'

print(s.strip('+-'))
print(s.rstrip('+-'))
print(s.lstrip('+-'))
```

выводит:

```
abc
-+-+abc
abc+-+-
```

Happy Pythoning!