ОТДЕЛЬНЫЙ УЧЕБНЫЙ КОМПЛЕКТ · PYTHON 3

Строки: методы и их применение

Учебный материал · три занятия и дополнительная практика

Как работать с темой

Ты уже умеешь вводить данные, писать условия и циклы. Теперь научишься обрабатывать текст: приводить названия к одному виду, находить фрагменты, проверять коды и разбирать сообщения. Главная цель — **объяснять, почему выбран именно этот метод**.

Занятие	Что изучаем	Практика
1 · Обрабатываем текст	Строка и метод; регистр; очистка; замена	D1–D3
2 · Ищем и проверяем	Поиск, подсчёт, начало/конец; проверки символов	D4–D8
3 · Разбираем и собираем	Список после split; join; сочетания методов	D9–D12
Дальше / для сильных	Расширение имени, команды, очистка списка	D13–D16

> Это ориентир на три занятия по 45 минут, а не обещание успеть весь комплект. Сначала предскажи вывод примера, затем запусти его в IDLE, измени входные данные и объясни результат.

В IDLE: **File → New File → сохранить .py → F5**. В примерах нет символов приглашения `>>>`: копируй код в редактор целиком. Короткие выражения можно отдельно исследовать в Shell.

---

1. Строка, символ и метод

Строка (`str`) — текст в кавычках. Число `7` и строка `"7"` имеют разные типы. `input()` возвращает строку, даже если введены только цифры. Поэтому код пропуска `"0072"` сохраняем как текст: преобразование в число потеряет ведущие нули.

Пример · Python 3

```
s = "Робот"
print(len(s))
print(s[0])
print(s[-1])
print(s[1:4])
```

Вывод

```
5
Р
т
обо
```

`len(s)` — количество символов; первый индекс равен `0`, последний можно записать как `-1`. Срез `s[1:4]` берёт позиции 1, 2, 3: правая граница не включается. Пустая строка записывается `""`, её длина равна 0. У неё нет первого и последнего символа: перед обращением к индексу проверь длину.

Метод — действие, которое вызывают у объекта через точку: `строка.метод(аргументы)`. Например, `s.upper()` возвращает строку в верхнем регистре. Скобки обязательны, даже когда аргументов нет.

Пример · Python 3

```
s = "Робот"
result = s.upper()
print(result)
print(s)
s = s.upper()
print(s)
```

Вывод

```
РОБОТ
Робот
РОБОТ
```

Строки неизменяемы. Методы преобразования возвращают результат, исходная строка сама не меняется. Его нужно присвоить переменной, передать дальше или сразу напечатать. Запись `s[0] = "р"` для строки вызовет ошибку.

Запись	Что возвращает
`len(s)`	Число; это функция, не метод
`s.lower()`	Строку
`s.find("о")`	Число — индекс или −1
`s.isalpha()`	Логическое значение True/False
`s.split()`	Список строк

Попробуй сам. Что выведет код `s = "кот"; s.upper(); print(s)`?

Подсказка / самопроверка

`кот`. Результат `upper()` не сохранён.

---

2. Регистр: lower, upper, capitalize

Регистр — различие между строчными и прописными буквами. Методы работают с латиницей и кириллицей. Цифры и знаки обычно остаются как есть.

Метод	Зачем	Пример → результат
`lower()`	Привести текст к нижнему регистру	`"PyThOn 7".lower()` → `"python 7"`
`upper()`	Сделать буквы прописными	`"Робот 2".upper()` → `"РОБОТ 2"`

Метод	Зачем	Пример → результат
`capitalize()`	Первая буква прописная, остальные строчные	`"пРИВЕТ, МИР".capitalize()` → `"Привет, мир"`

Разобранная задача. Пользователь написал ответ в произвольном регистре. Программа должна распознать слово «да». Сначала приводим текст к одному регистру, затем сравниваем.

Пример · Python 3

```
answer = "Да"
if answer.lower() == "да":
    print("Продолжаем")
else:
    print("Останавливаемся")
```

Вывод

```
Продолжаем
```

`lower()` не удаляет пробелы: `" Да ".lower()` даст `" да "`. Регистр и лишние пробелы — две разные проблемы.

Попробуй сам. Получи из `"мОСКВА"` строку `"Москва"`.

Подсказка / самопроверка

`"мОСКВА".capitalize()`. Этот метод меняет регистр остальных букв тоже, поэтому не подходит для сохранения написания брендов вроде iPhone.

---

3. Очистка краёв: strip, lstrip, rstrip

При вводе человек может случайно добавить пробелы перед именем или после него. `strip()` убирает пробельные символы с **обоих краёв**, `lstrip()` — слева, `rstrip()` — справа. Без аргументов убираются также табуляции и другие пробельные символы. Внутри текста ничего не меняется.

Пример · Python 3

```
s = "  робот  кот  "
print(repr(s.strip()))
print(repr(s.lstrip()))
print(repr(s.rstrip()))
```

Вывод

```
'робот  кот'
'робот  кот  '
'  робот  кот'
```

`repr()` здесь нужен только для наблюдения: он показывает кавычки вокруг строки, чтобы были заметны пробелы. В решениях задач печатай обычный результат, если условие не просит другое.

Цепочка методов. Результат одного вызова можно сразу обработать следующим. Читать и выполнять такую цепочку нужно слева направо:



Пример · Python 3

```
name = "  PyThOn  "
clean = name.strip().lower()
print(clean)
```

Вывод

```
python
```

Аргумент `strip(chars)` — **набор удаляемых символов**, а не точное слово или окончание. `"abbaXba".strip("ab")` даст `"X"`. Поэтому расширение файла не удаляют через `strip(".py")`. Для точного префикса/суффикса есть дополнительные методы в памятке.

Попробуй сам. Нужно убрать пробелы вокруг `"  7 Д  "`, сохранив пробел между 7 и Д. Какой метод выбрать?

Подсказка / самопроверка

`s.strip()` вернёт `"7 Д"`. `replace(" ", "")` здесь испортит нужный пробел.

---

4. Замена: replace

`s.replace(old, new)` заменяет **все** вхождения фрагмента `old` на `new`. Третий аргумент ограничивает число замен, начиная слева. Метод чувствителен к регистру.

Пример · Python 3

```
s = "кот, кот и Кот"
print(s.replace("кот", "робот"))
print(s.replace("кот", "робот", 1))
print("A B C".replace(" ", ""))
```

Вывод

```
робот, робот и Кот
робот, кот и Кот
ABC
```

Разобранная задача. Датчик прислал десятичную дробь с запятой. Чтобы получить число Python, сначала меняем разделитель, затем преобразуем тип. По условию датчик гарантированно присылает корректную запись одного числа.

Пример · Python 3

```
raw = " 23,5 "
temperature = float(raw.strip().replace(",", "."))
print(temperature + 1)
```

Вывод

```
24.5
```

Одна замена двойного пробела не сжимает произвольную длинную цепочку пробелов до одного: в `"A  B"` четыре пробела превратятся в два. Для нормализации пробельных промежутков позже используем `" ".join(s.split())`.

Попробуй сам. Заменить только первый дефис в `"7-Д-2"` на пробел.

Подсказка / самопроверка

`s.replace("-", " ", 1)` → `"7 Д-2"`.

---

5. Поиск и подсчёт: in, find, rfind, index, count

Перед поиском уточни вопрос: нужно узнать **наличие**, **позицию** или **количество**? Для этих целей нужны разные инструменты. Все перечисленные поиски чувствительны к регистру.

Инструмент	Результат	Если фрагмента нет
`part in s`	True/False — есть ли фрагмент	False
`s.find(part)`	Индекс первого вхождения	−1
`s.rfind(part)`	Индекс последнего вхождения	−1
`s.index(part)`	Индекс первого вхождения	Ошибка ValueError
`s.count(part)`	Количество неперекрывающихся вхождений	0

Пример · Python 3

```
s = "банан"
print("ан" in s)
print(s.find("ан"))
print(s.rfind("ан"))
print(s.count("ан"))
print(s.find("кот"))
print(s.count("кот"))
print("aaa".count("aa"))
```

Вывод

```
True
1
3
2
-1
0
1
```

`"aaa".count("aa")` даёт 1: найденные фрагменты не перекрываются. Поиск подстроки не понимает границы слов: `"робототехника".count("робот")` тоже даст 1. Для поиска целых слов сначала потребуется определить, как отделяются слова и знаки препинания.

Пример · Python 3

```
message = "РОБОТ готов"
if "робот" in message.lower():
    print("Найден фрагмент")
pos = message.find("Р")
if pos != -1:
    print(pos)
```

Вывод

```
Найден фрагмент
0
```

Не пиши `if s.find(part):`. Найденный индекс 0 трактуется как False, а −1 — как True. Для наличия используй `in`, для найденной позиции проверяй `pos != -1`. Поиск непустого фрагмента — основной случай этого урока; пустую подстроку здесь не используем.

Попробуй сам. Для `s = "cat.cat"` предскажи `s.find("cat")` и `s.rfind("cat")`.

Подсказка / самопроверка

0 и 4. Если нужно только наличие, достаточно `"cat" in s`.

6. Начало и конец: startswith, endswith

`startswith(prefix)` проверяет начало строки, `endswith(suffix)` — конец. Результат — `True` или `False`. Это удобнее ручного вычисления длины среза, когда нужно проверить известный фрагмент.

Пример · Python 3

```
command = "/start robot"
filename = "  LESSON.PY  "
print(command.startswith("/start"))
print(filename.strip().lower().endswith(".py"))
print("lesson.py.txt".endswith(".py"))
```

Вывод

```
True
True
False
```

Проверка окончания `".py"` отвечает на учебный вопрос о названии, но не изучает содержимое файла. Аналогично начало `"/start"` совпадёт и у `"/starter"`. Если нужна именно команда, следует сначала выделить первое слово и сравнить его целиком — это сделаем в D14.

Попробуй сам. Почему поиск `".py" in name` не подходит для проверки окончания?

Подсказка / самопроверка

Он примет `"task.py.txt"`. `endswith(".py")` проверит именно конец.

---

7. Проверки: isalpha, isdigit, isalnum, isspace

Методы с началом `is` возвращают логический результат. Эти четыре метода требуют **непустую строку**: каждый из них вернёт False для `""`. Они проверяют все символы, а не наличие хотя бы одного подходящего.

Метод	Какие символы подходят	Примеры
`isalpha()`	Только буквы	`"Робот"` → True; `"Робот 7"` → False
`isdigit()`	Только цифры	`"007"` → True; `"-7"` → False
`isalnum()`	Буквы и/или цифры	`"Бот7"` → True; `"bot_7"` → False
`isspace()`	Только пробельные символы	`" "` → True; `""` → False

Пример · Python 3

```
code_text = "0072"
if len(code_text) == 4 and code_text.isdigit():
    print("Код подходит")
else:
    print("Ошибка")
print("Иван Петров".isalpha())
print("ИванПетров".isalpha())
print("".isdigit())
```

Вывод

```
Код подходит
False
True
False
```

`isdigit()` не означает «можно безопасно вызвать int». Минус, плюс и десятичная точка не считаются цифрами; кроме того, Python признаёт некоторые цифры Unicode, например `"²".isdigit()` → True, а `int("²")` вызовет ошибку. В задачах на цифровой код отдельно гарантируем алфавит 0–9. Для строгой проверки именно ASCII-цифр можно дополнить условие `s.isascii()`; этот метод есть в памятке.

Для учебного логина с подчёркиванием можно проверить `s.replace("_", "").isalnum()`, **если по условию нужен хотя бы один символ-буква или цифра**. Логин из одних подчёркиваний тогда должен быть отклонён. Правило всегда сначала формулируем словами.

Попробуй сам. Проверь ответы для `"123".isalnum()`, `"a_b".isalnum()`, `"  ".isspace()`.

Подсказка / самопроверка

True, False, True. Буквы не обязательны для `isalnum()`; подчёркивание не считается буквой или цифрой.

---

8. Минимум о списке для split и join

Следующие методы связывают строки со списками. **Список** — несколько значений в определённом порядке. В записи Python его элементы стоят в квадратных скобках. Для этого урока достаточно создания списка методом `split()`, длины, индексов и уже знакомого цикла `for`.

Пример · Python 3

```
words = ["робот", "едет", "домой"]
print(len(words))
print(words[0])
print(words[-1])
for word in words:
    print(word.upper())
```

Вывод

```
3
робот
домой
РОБОТ
ЕДЕТ
ДОМОЙ
```

Число элементов списка и число символов строки — разные величины: у `words` три элемента, а у `words[0]` пять символов. Индексы списка тоже начинаются с 0. У пустого списка `[]` нет элемента с индексом 0: прежде чем брать первое слово, проверь длину.

> Это короткий мостик к обработке текста, отдельная большая тема о списках пока не требуется. Срез строки возвращает строку; результат `split()` — список строк.

---

9. Разбиение: split

`s.split()` без аргументов разбивает текст по промежуткам из пробельных символов. Несколько пробелов подряд и пробелы по краям не создают пустых элементов. Для пустой или полностью пробельной строки получается пустой список.

Пример · Python 3

```
s = "  робот   едет домой  "
words = s.split()
print(words)
print(len(words))
print("   ".split())
```

Вывод

```
['робот', 'едет', 'домой']
3
[]
```

Если разделитель указан явно, правило другое: `split(";")` делит по точке с запятой, а **пустые поля сохраняет**. Пробелы у полей автоматически не убираются. Явный разделитель может состоять из нескольких символов; пустой разделитель `""` недопустим.

Пример · Python 3

```
print("робот;;датчик;".split(";"))
print("a  b".split(" "))
print("a  b".split())
print("".split(";"))
print("key=value=extra".split("=", 1))
```

Вывод

```
['робот', '', 'датчик', '']
['a', '', 'b']
['a', 'b']
['']
['key', 'value=extra']
```

Второй аргумент ограничивает число разбиений. `split("=", 1)` удобен, когда всё после первого знака равно нужно оставить единым значением.

В учебной задаче слово — элемент, полученный после `split()`. Это соглашение: метод не убирает запятые и точки и не анализирует смысл текста. `"Привет, мир!".split()` содержит `"Привет,"` и `"мир!"`.

Попробуй сам. Сколько элементов в `"a;;b;".split(";")`?

Подсказка / самопроверка

Четыре: `["a", "", "b", ""]`. Непустых полей только два, но это другой вопрос.

---

10. Сборка: join и сочетания методов

`separator.join(parts)` соединяет строки из списка, помещая между ними разделитель. Метод вызывается **у строки-разделителя**. Разделитель не появляется по краям. Все соединяемые элементы должны быть строками.

Пример · Python 3

```
parts = ["робот", "едет", "домой"]
print(" ".join(parts))
print("-".join(parts))
print("".join(parts))
print(repr("-".join([])))
```

Вывод

```
робот едет домой
робот-едет-домой
роботедетдомой
''
```

Разобранная задача. Привести название кружка к единому виду: убрать лишние пробелы, перевести буквы в нижний регистр, слова соединить дефисами. Сначала понятный вариант с промежуточными переменными:

Пример · Python 3

```
raw = "  Клуб   Юных Роботов  "
words = raw.lower().split()
slug = "-".join(words)
print(words)
print(slug)
```

Вывод

```
['клуб', 'юных', 'роботов']
клуб-юных-роботов
```

Здесь `strip()` перед `split()` не нужен: разбиение без аргументов уже игнорирует пробельные края. Для выравнивания пробелов без смены регистра используй `" ".join(raw.split())`.

Шаг	Тип результата	Пример
`raw.lower()`	str	`"  клуб   юных роботов  "`
`... .split()`	list из str	`["клуб", "юных", "роботов"]`
`"-".join(words)`	str	`"клуб-юных-роботов"`

После `split()` получится список: у него нет строкового метода `lower()`. Сначала `s.lower().split()`, либо обрабатывай каждый элемент циклом. Ошибка `parts.join("-")`: вызывать `join` нужно у разделителя.

Попробуй сам. Из `"  урок   про   строки "` получи `"урок про строки"`.

Подсказка / самопроверка

`" ".join(s.split())`. Сравни с `s.strip()`: тот оставит промежутки внутри.

---

11. Как выбрать метод и проверить решение

Что требуется	Первый выбор	Уточнение
Убрать пробелы по краям	`strip()`	Пробелы внутри сохраняются
Сравнить без учёта регистра	`lower()` перед сравнением	Для нашего русского/английского текста
Заменить фрагмент	`replace()`	Все вхождения или первые n?
Узнать, есть ли фрагмент	`in`	Ищем подстроку или целое слово?

Что требуется	Первый выбор	Уточнение
Найти позицию	`find()` / `rfind()`	Что делать при −1?
Посчитать фрагменты	`count()`	Вхождения не перекрываются
Проверить начало/конец	`startswith()` / `endswith()`	Нужны ли очистка и смена регистра?
Проверить состав строки	`is...`	Пустая строка, пробелы, алфавит?
Выделить слова/поля	`split()`	Указан ли явный разделитель?
Собрать строки	`join()`	Выбери разделитель

Перед написанием кода ответь на четыре вопроса: **что вводится; что должно получиться; какие преобразования нужны; какие крайние случаи возможны**. Проверь обычный ввод и хотя бы два особых случая: пустую строку, отсутствие фрагмента, находку на позиции 0, повторные разделители или ведущие нули.

После решения задачи назови использованные методы, тип промежуточного результата и объясни один неудачный альтернативный выбор. Например: «Выбрал endswith, потому что .py должен быть в конце; in принял бы файл task.py.txt».

Перейти к практике D1–D16 → · Открыть краткую памятку →

---

12. Самопроверка: 10 вопросов

Выбери ответ в каждом вопросе, затем нажми «Проверить». Это самопроверка, результат никуда не отправляется.

1. Что выведет: s = "Кот"; s.lower(); print(s)?

- ○ `кот`
- ○ `Кот`
- ○ `Ошибка`

---

2. Что вернёт " a b ".strip()?

- ○ `"a b"`
- ○ `"ab"`
- ○ `"a  b"`

---

3. Чему равно "robot".find("r")?

- `0`
- `1`
- `-1`

---

4. Что вернёт "robot".find("x")?

- `0`
- `-1`
- `ValueError`

---

5. Чему равно "aaaa".count("aa")?

- `3`
- `4`
- `2`

---

6. Как проверить окончание имени .py?

- `".py" in name`
- `name.endswith(".py")`
- `name.startswith(".py")`

---

7. Что вернёт "".isdigit()?

- `True`
- `False`
- `Ошибка`

---

8. Что вернёт " a b ".split()?

- ○ `["a", "b"]`
- ○ `["", "a", "", "", "b", ""]`
- ○ `"a b"`

---

9. Сколько элементов в "a;;b;".split(";")?

- ○ `2`
- ○ `3`
- ○ `4`

---

10. Как получить a-b из parts = ["a", "b"]?

- ○ `parts.join("-")`
- ○ `"-".join(parts)`
- ○ `parts.split("-")`
