**Конспект Яндекс.Учебника по основам Python**

Contents

[Циклы 2](#_Toc195406801)

[Вложенные циклы 3](#_Toc195406802)

[Кортежи, списки, массивы, строки 5](#_Toc195406803)

[Множества и словари в Python 7](#_Toc195406804)

[List comprehension (списочные выражения) 9](#_Toc195406805)

[Возможности для работы с коллекциями 10](#_Toc195406806)

[Потоковый ввод-вывод, работа с файлами и JSON 12](#_Toc195406807)

[Функции, области видимости, передача параметров 14](#_Toc195406808)

[Функции высшего порядка. Lambda-функции. 16](#_Toc195406809)

[Рекурсия, декораторы и генераторы 18](#_Toc195406810)

# Циклы

Циклы позволяют выполнять повторяющиеся действия. В Python есть два основных типа циклов: while и for.

**Цикл**while

**Используется, когда количество итераций заранее неизвестно.**

**Цикл**for

**Используется для перебора последовательностей (строк, списков, диапазонов и т. д.).**

**Функция**range() - генерирует последовательность чисел

break**и**continue

* break – досрочно выходит из цикла.
* continue – пропускает текущую итерацию.

# Вложенные циклы

**1. Основная концепция**

* Вложенный цикл выполняется **для каждой итерации внешнего цикла**.
* Могут использоваться любые комбинации (for в for, while в for, for в while и т. д.).

A black background with numbers and symbols

AI-generated content may be incorrect.



**2. Управление вложенными циклами**

* break — прерывает **текущий** (внутренний) цикл.
* continue — пропускает итерацию **текущего** цикла.
* Чтобы выйти из всех вложенных циклов, используют **флаги** или **исключения**.

A black background with white text

AI-generated content may be incorrect.



**3. Оптимизация вложенных циклов**

* Вложенные циклы могут замедлять выполнение кода (сложность O(n²) и выше).
* Альтернативы для оптимизации:
  + Использование **списковых включений** (list comprehension).
  + Применение функций из библиотек (numpy, itertools).

# Кортежи, списки, массивы, строки

**1. Строки (**str**)**

* **Неизменяемый** тип данных (последовательность символов).
* **Индексация и срезы**:

A group of white text on a black background

AI-generated content may be incorrect.

* **Методы строк**:

A screenshot of a computer

AI-generated content may be incorrect.

* **Форматирование:**



**2. Кортежи (**tuple**)**

* **Неизменяемый** упорядоченный тип данных (аналог списка, но без изменений).
* **Создание**:



* **Особенности**:
  + Можно использовать как ключ в словаре.
  + Занимает меньше памяти, чем список.
  + Поддерживает индексацию и срезы (t[0], t[1:2]).

**3. Списки (**list**)**

* **Изменяемый** упорядоченный тип данных.
* **Создание**:

A close up of a smiley face

AI-generated content may be incorrect.

* **Основные операции**:

A computer screen shot of a number

AI-generated content may be incorrect.

* **Срезы и индексация:**

A number and hashtag on a black background

AI-generated content may be incorrect.

**4. Массивы (array)**

* **Может содержать только элементы одного типа данных**
* **Ограниченная функциональность по сравнению со списками**
* **Изменяемые**
* **Метод .append() добавляет один элемент, метод .extend() добавляет несколько элементов, метод .insert() добавляет новый элемент на определенную позицию, метод .remove() удаляет заданный элемент, метод .pop() удаляет элемент с заданным индексом**

**A close-up of a computer code

AI-generated content may be incorrect.**

# Множества и словари в Python

**1. Множества (**set**)**

* **Неупорядоченная** коллекция **уникальных** элементов.
* **Изменяемый** тип данных (можно добавлять/удалять элементы).
* **Эффективные** операции поиска и удаления дубликатов.

Создание:

A number and symbol on a dark background

AI-generated content may be incorrect.

Основные операции:

A screen shot of a computer code

AI-generated content may be incorrect.

Операции с множествами:

A screenshot of a computer program

AI-generated content may be incorrect.

Неизменяемое множество (frozenset):



**2. Словари (**dict**)**

* **Неупорядоченная** коллекция пар **ключ: значение**.
* **Ключи** уникальны и **неизменяемы** (числа, строки, кортежи).
* **Значения** могут быть любыми.

Создание:

A black background with white text

AI-generated content may be incorrect.

Основные операции:

A screen shot of a computer

AI-generated content may be incorrect.

И у множеств, и у словарей нет индексов, но имеются итерации, что позволяет выполнять эффективный поиск (in работает за O(1)).

# List comprehension (списочные выражения)

Списочные выражения возвращают объект, который можно присвоить переменной или сразу отправить в функцию. По сути – являются генераторами списков.

Синтаксис:



Пример:

A screenshot of a computer

AI-generated content may be incorrect.

Аналоги для других коллекций:

A computer code with text

AI-generated content may be incorrect.

# Возможности для работы с коллекциями

**1. Основные встроенные функции для коллекций**

1. len() — возвращает количество элементов:



1. sum() — сумма элементов (для чисел):



1. min()**/**max() — минимальный/максимальный элемент:

A group of numbers on a black background

AI-generated content may be incorrect.

1. sorted() — возвращает новый отсортированный список:



1. reversed() — возвращает обратный итератор:



**2. Полезные методы для работы с коллекциями**

**Для списков (**list**)**

* .append(x) — добавить элемент в конец.
* .extend([x, y]) — добавить несколько элементов.
* .pop() — удалить и вернуть последний элемент.
* .index(x) — найти индекс элемента.
* .count(x) — подсчитать количество вхождений.

**Для множеств (**set**)**

* .add(x) — добавить элемент.
* .remove(x) — удалить элемент (ошибка, если нет).
* .discard(x) — удалить элемент (без ошибки).
* .union()**/**.intersection() — объединение/пересечение.

**Для словарей (**dict**)**

* .keys() — все ключи.
* .values() — все значения.
* .items() — пары ключ-значение.
* .get(key, default) — безопасное получение значения.

**3. Функции**map()**,**filter()**,**zip()

1. map(func, коллекция) — применяет функцию к каждому элементу:



1. filter(func, коллекция) — фильтрует элементы по условию:



1. zip() — объединяет несколько коллекций попарно:



# Потоковый ввод-вывод, работа с файлами и JSON

**1. Работа с файлами**

**Открытие файлов**

Используется конструкция with open(...) as (автоматически закрывает файл):



**A screenshot of a phone

AI-generated content may be incorrect.**

A black screen with white text

AI-generated content may be incorrect.

**2. JSON (сериализация данных)**

Формат для хранения и передачи структурированных данных (словари, списки).

A screenshot of a computer program

AI-generated content may be incorrect.

**3. Потоковый ввод-вывод (**sys.stdin**,**sys.stdout**)**

Используется для чтения/записи данных через стандартные потоки.

A screen shot of a computer code

AI-generated content may be incorrect.

# Функции, области видимости, передача параметров

**Функции (определение и вызов)**

**Функция** — это блок кода, который можно вызывать по имени.

**Области видимости**

Переменные бывают:

* **Локальные** (внутри функции)
* **Глобальные** (вне функций)

**Ключевые слова:**

* **global — позволяет изменять глобальную переменную внутри функции.**

**Передача параметров в функции**

**Параметры можно передавать:**

1. **По позиции (positional):**

**A computer screen shot of a number

AI-generated content may be incorrect.**

1. **По имени (keyword):**

****

1. **Смешанный вариант (сначала позиционные, потом именованные):**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**Возврат значений (return)**

* **Функция может возвращать любое количество значений (через кортеж).**
* **Если return нет, возвращается None.**

# Функции высшего порядка. Lambda-функции.

**Функции высшего порядка**

Функции, которые принимают другие функции в качестве аргументов или возвращают их.

**Примеры**

**Функция как аргумент:**

A computer screen shot of a program code

AI-generated content may be incorrect.

**Функция как возвращаемое значение:**

A screen shot of a computer code

AI-generated content may be incorrect.

**Встроенные функции высшего порядка**

* map(func, iterable) — применяет функцию к каждому элементу коллекции.
* filter(func, iterable) — фильтрует элементы по условию.
* sorted(iterable, key=func) — сортирует коллекцию по ключу.

**Lambda-функции**

Анонимные функции, которые можно использовать **без объявления**def.

**Синтаксис**



**Примеры:**

A computer screen with text

AI-generated content may be incorrect.

**Особенности**

* Не имеют имени (анонимные).
* Могут принимать любое количество аргументов.
* Не могут содержать сложные конструкции (только одно выражение).

# Рекурсия, декораторы и генераторы

**Рекурсия**

**Рекурсия** — когда функция вызывает саму себя.

**Особенности:**

* Должно быть **базовое условие** (когда рекурсия останавливается)
* Каждый вызов создает новый уровень в **стеке вызовов**
* Может привести к **Stack Overflow (переполнению буфера)** при слишком большой глубине

Пример:

A black background with white text

AI-generated content may be incorrect.

**Плюсы:**  
✓ Удобна для задач с рекурсивной природой (обход деревьев)  
✓ Читаемый код для математических определений

**Минусы:**  
✗ Менее эффективна по памяти  
✗ Риск переполнения стека

**Декораторы**

**Декораторы** — функции, которые изменяют поведение других функций.

Синтаксис:

A computer screen shot of a code

AI-generated content may be incorrect.

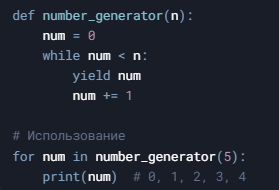
**Генераторы**

**Генераторы** — функции, которые возвращают итератор (генерируют значения по требованию).

**Ключевые особенности:**

* Используют yield вместо return
* Сохраняют состояние между вызовами
* Экономят память (не хранят всю последовательность)

Пример:



**Преимущества:**  
✓ Экономия памяти  
✓ Ленивые вычисления (вычисляются по мере необходимости)  
✓ Могут генерировать бесконечные последовательности

A screenshot of a computer

AI-generated content may be incorrect.

# ООП в Python (классы, поля, методы)

**1. Основы ООП в Python**

**Класс** — это шаблон для создания объектов.  
**Объект** — экземпляр класса с собственными данными.

A screenshot of a computer program

AI-generated content may be incorrect.

**2. Поля и методы**

**2.1. Поля (атрибуты)**

Могут быть:

* **Атрибутами класса** (общие для всех объектов)
* **Атрибутами экземпляра** (индивидуальные для каждого объекта)

A computer code with white text

AI-generated content may be incorrect.

**2.2. Методы**

Функции, принадлежащие классу. Бывают:

* **Методы экземпляра** (работают с конкретным объектом)
* **Методы класса** (@classmethod)
* **Статические методы** (@staticmethod)

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

**3. Наследование**

Позволяет создавать новые классы на основе существующих.

A computer screen shot of text

AI-generated content may be incorrect.

**4. Инкапсуляция**

Ограничение доступа к данным. В Python используются:

* **Публичные** атрибуты (без подчеркиваний)
* **Защищенные** (\_var) — соглашение
* **Приватные** (\_\_var) — реальное скрытие

A computer screen shot of a code

AI-generated content may be incorrect.

**5. Свойства (**@property**)**

Контролируемый доступ к атрибутам.

A screen shot of a computer code

AI-generated content may be incorrect.

Полиморфизм (polymorphism) — это понятие из объектно-ориентированного программирования, которое позволяет разным сущностям выполнять одни и те же действия. При этом неважно, как эти сущности устроены внутри и чем они различаются.

Абстрагирование — это способ выделить набор наиболее важных атрибутов и методов и исключить незначимые. Соответственно, абстракция — это использование всех таких характеристик для описания объекта. Важно представить объект минимальным набором полей и методов без ущерба для решаемой задачи.

# Исключения

**Обработка исключений (try-except)**

**Базовый синтаксис:**

A computer screen shot of a program code

AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

# Основные модули: math, numpy, pandas, requests

**1. Модуль**math

**Для базовых математических операций** (работает с отдельными числами).

**Основные функции:**

A screenshot of a computer

AI-generated content may be incorrect.

**2. Библиотека NumPy**

**Для научных вычислений** (работа с массивами и матрицами).

**2.1. Основные возможности**

* **Многомерные массивы** (ndarray)
* **Векторизованные операции** (быстрые вычисления без циклов)
* **Линейная алгебра**, статистика, Фурье-анализ

**2.2. Создание массивов**

A screenshot of a computer code

AI-generated content may be incorrect.

**2.3. Основные операции**

A screenshot of a computer

AI-generated content may be incorrect.

**2.4. Полезные функции NumPy**

A screenshot of a black screen

AI-generated content may be incorrect.

**Библиотека Pandas**

**1. Основные структуры данных**

**1.1. Series**

**Одномерный массив с индексами (аналог столбца таблицы).**

**A close-up of a computer code

AI-generated content may be incorrect.**

**1.2. DataFrame**

**Двумерная таблица (как Excel или SQL-таблица).**

**A screenshot of a computer

AI-generated content may be incorrect.**

**3. Основные операции с DataFrame**

* **Просмотр данных**
* **Выбор данных**
* **Добавление/удаление данных**
* **Группировка и агрегация**
* **Обработка пропусков**

**Библиотека Requests**

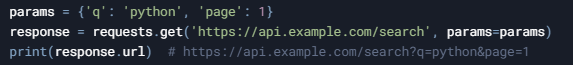
**requests — это популярная библиотека для отправки HTTP-запросов и работы с API. Она упрощает взаимодействие с веб-серверами по сравнению со стандартным модулем urllib.**

**Основные HTTP-методы**

* **GET-запрос -** запрос данных с сервера.
* **POST-запрос –** отправка данных на сервер.
* **PUT –** обновление данных.
* **DELETE –** удаление данных.
* **PATCH –** частичное обновление.

**Параметры запросов**

**Параметры URL (params):**

****

**Заголовки (headers)**

****

**Cookies (cookies)**

****

**HTTP-коды состояний:**

* 1хх – информационные
* 2хх – успешные
* 3хх – перенаправления
* 4хх – ошибки клиента
* 5хх – ошибки сервера