



Урок 15. Python: Сортировка пузырьком

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 15 из 34 · 45 минут · Модуль «Сортировка»

Уровень: начинающий программист; Python 3.

Программа: <https://schoolotzyv.ru/blog/python-cpp-9-klass-programma>

После одного прохода большой элемент оказывается справа, но весь список ещё не обязательно отсортирован. Проследим, что гарантирует каждый проход.

Результат урока: Реализовать пузырьковую сортировку с ранней остановкой, объяснить границу прохода и проверить сохранность всех элементов.

01. Проблемная ситуация

0–5 минут. Для [3,1,2,1] выполните слева направо три сравнения соседей, меняя их при левом большем. Запишите список после каждого сравнения. Готов ли весь результат?

02. Понятия и разобранный пример

5–14 минут: 4 минуты на понятия, 3 на пример, 2 на объяснение способа партнёру.

Соседний обмен и правый суффикс

Пузырьковая сортировка сравнивает соседей и обменивает их, если они стоят неверно. После первого полного прохода максимум находится справа. Следующий проход не трогает уже закреплённый правый элемент. Переменная `end` задаёт правую границу сравнений: `i` от 0 до `end-1`, сравниваются `i` и `i+1`.

Если за целый проход не было обменов, все соседи уже не убывают: можно завершить работу. Отсутствие обмена только в одной паре ещё ничего не доказывает. Код работает с копией `data`, возвращает отсортированный список, число сравнений и число обменов.

Контракт: до 100 целых от -1000 до 1000. Пустота и один элемент уже отсортированы, сравнений ноль. Худшее число сравнений $n*(n-1)/2$; для уже отсортированного списка длины $n \geq 2$ достаточно $n-1$. Обмен только при $>$ сохраняет относительный порядок равных ключей.

Отступы набирайте обычными пробелами: 4 на уровень. При копировании с веб-страницы замените неразрывные пробелы обычными; полный код с обычными пробелами есть в PDF.

[3,1,2,1]: первый проход даёт [1,3,2,1]→[1,2,3,1]→[1,2,1,3]; второй — [1,1,2,3]; третий проверяет первую пару и завершается без обмена. Всего 6 сравнений и 4 обмена.

```
def bubble(data):
    a = data.copy()
    comparisons = swaps = 0
    for end in range(len(a)-1, 0, -1):
        changed = False
        for i in range(end):
            comparisons += 1
            if a[i] > a[i+1]:
                a[i], a[i+1] = a[i+1], a[i]
                swaps += 1
                changed = True
        if not changed:
            break
    return a, comparisons, swaps

for a in [[3,1,2,1],[],[5],[1,2,3],[3,2,1]]:
    print(a, bubble(a))
```



Начало	3 1 2 1
Проход end=3	1 2 1 3
Проход end=2	1 1 2 3
Проход end=1	1 1 2 3; обменов нет

Правый отсортированный суффикс растёт после каждого прохода.

03. Практика 1. Проверяем материал

Считаем сравнения и обмены

14–25 минут. Для шести списков составьте журнал проходов и счётчиков, затем запустите код.

Вход	Результат; сравнения; обмены
[3,1,2,1]	[1,1,2,3]; 6; 4
[]; [5]	[]; 0; 0 и [5]; 0; 0
[1,2,3]; [3,2,1]	[1,2,3]; 2; 0 и [1,2,3]; 3; 3
[2,2,2]	[2,2,2]; 2; 0

2 минуты — прогноз; 4 — запуски; 3 — сравнение с sorted и исходной копией; 2 — объяснение теста [3,1,2,1]. Критерии: сортировка, сохранение повторов, неизменность входа, правильные счётчики.

04. Практика 2. Перенос способа

Сортируем по убыванию

25–33 минуты. Сделайте bubble_down для убывания. За 2 минуты поменяйте сравнение, за 4 проверьте [1,3,2,3], [] и [2,2], за 2 сопоставьте со sorted(...,reverse=True). Критерии: знак <, сохранение повторов, четыре сравнения и три обмена для первого входа? Проверьте это предположение журналом.

05. Обсуждение и границы вывода

33–40 минут. За 2 минуты сравните число сравнений на возрастающем и обратном списках, за 3 объясните стоимость для 100 элементов, за 2 решите, где полезен этот алгоритм.

06. Индивидуальная проверка

40–45 минут. Для [2,1,1] дайте результат bubble, сравнения, обмены и исходный список после вызова. Четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Один проход для [3,1,2,1] оставляет нарушение $2 > 1$, поэтому ранний выход после первого обмена неверен. `sorted(a)` проверяет результат, `a==исходная_копия` — отсутствие изменения входа. Одной проверки неубывания мало: [1,2,3] не содержит обе единицы исходного списка.

Практика 2: разбор и критерии

Предположение о четырёх сравнениях неверно: для [1,3,2,3] будет 6 сравнений и 4 обмена, результат [3,3,2,1]. Переименуйте `bubble` в `bubble_down` и замените единственное условие `a[i]>a[i+1]` на `a[i]<a[i+1]`; остальные строки функции сохраняются. Для [] счётчики 0,0; для [2,2] — 1,0. Полный исходник основной функции дан выше, изменения ровно две строки.

Обсуждение: возможный вывод

Упорядоченный список длины 100 требует 99 сравнений и 0 обменов. Обратный с различными элементами — 4950 сравнений и 4950 обменов. Пузырёк полезен для обучения инвариантам и отладке, но для обычной прикладной сортировки Python предпочтителен `sorted` или `list.sort`.

Индивидуальная проверка: ответ и критерии

[1,1,2]; 3; 2; [2,1,1]. Каждый ответ — один балл. Равные единицы друг с другом не меняются.

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, редактор, папку с правом записи, распечатку ученической части PDF, бумагу и ручку. Python 3.11+ и IDLE или редактор; запуск `python lesson.py` (либо `python3 lesson.py`). Внешние библиотеки не нужны. Установку и проверку запуска учитель выполняет заранее. Все программы, данные и ожидаемые ответы даны; регистрация детей и интернет на занятии не нужны.

Уровень — начинающий программист, 9 класс. Предпосылки: списки, индексы, вложенные циклы, обмен двух значений. В паре автор прогнозирует и вводит код, проверяющий ведёт журнал, затем роли меняются. Без компьютера выполните трассировку полного кода на бумаге и сопоставьте с готовыми результатами; учитель демонстрирует подготовленный запуск. Бумажный вариант не считается фактическим запуском ученика. Перед практикой учитель выдаёт готовый файл разобранного примера, чтобы дети тратили время на изменение и проверку, а не длинное перепечатывание. Матрица программы предназначена учителю.

Минуты	Работа	Результат
0–5	Проблема	Вопрос и первое решение
5–14	Понятия и пример	Способ проверки
14–25	Практика 1	Готовый продукт и проверка
25–33	Практика 2	Перенос способа
33–40	Обсуждение	Аргумент и ограничение
40–45	Индивидуальная проверка	Самостоятельный ответ



07. Соответствие исходной программе

Пункт программы	Действие ученика	Проверяемый результат
Алгоритмы сортировки	Считаем сравнения и обмены	Реализовать пузырьковую сортировку с ранней остановкой, объяснить границу прохода и проверить сохранность всех элементов.
Структурирование решения и отладка	Сортируем по убыванию	Исходник, журнал обычных и граничных тестов; индивидуальный ответ

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-9-klass-programma>

Python: функции и управление выполнением

<https://docs.python.org/3/tutorial/controlflow.html>

Тексты, учебные данные, реплики, программы и схемы подготовлены для уроков портала «Школы России». Числовые серии — учебные модели, не результаты проведённых классом опытов. Схемы авторские, сторонние фотографии не используются. Методические ссылки приведены для учителя; ученик получает все материалы здесь. Проверка первичных источников: 18.10.2026.