алгоритмика

Международная школа
программирования для детей

Модуль 2. Урок 5.

Сортировки

алгоритмика

Повторим

Сегодня на занятии:

- Сортировки — что это такое и «с чем это едят»?
- Внесение порядка в хаос списка.
- Выбирай и сортируй, или «сортировка выбором».
- Сложность алгоритма — что это такое и как это оценить?

Демонстрация

(заполнение списка)

алгоритмика

Что получилось?

Сортировка —

это алгоритм для упорядочивания множества объектов по какому-либо признаку.

Сортировка выбором

1. Выбираем элемент, который по умолчанию считаем самым наименьшим в списке
2. Сравниваем его со всеми остальными. Если среди них находится элемент меньше, мы выбираем его в качестве нового наименьшего и меняем местами с прошлым.

Демонстрация

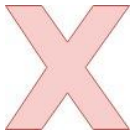
(заполнение списка (пишем алгоритм сортировки))

Важное замечание!!!

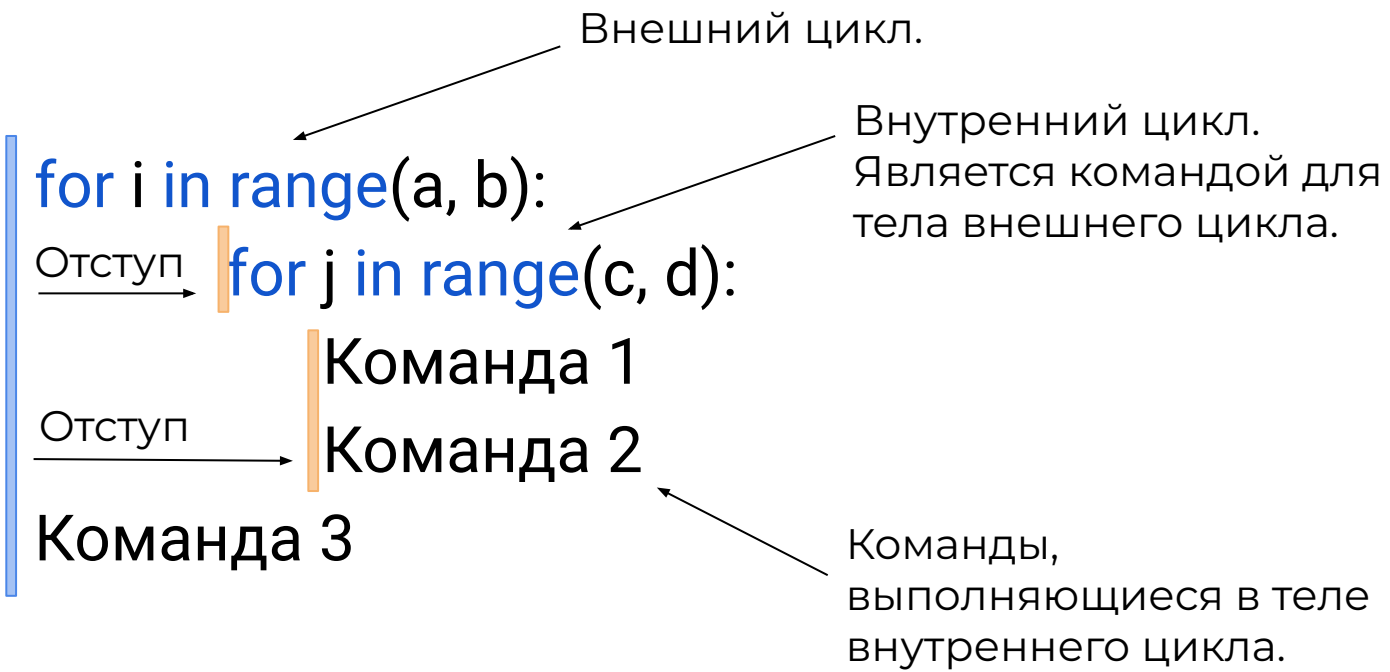
Применять сортировку можно только к элементам, которые можно сравнивать друг с другом.

checklist = [56, 2, 0, -3, 123] 

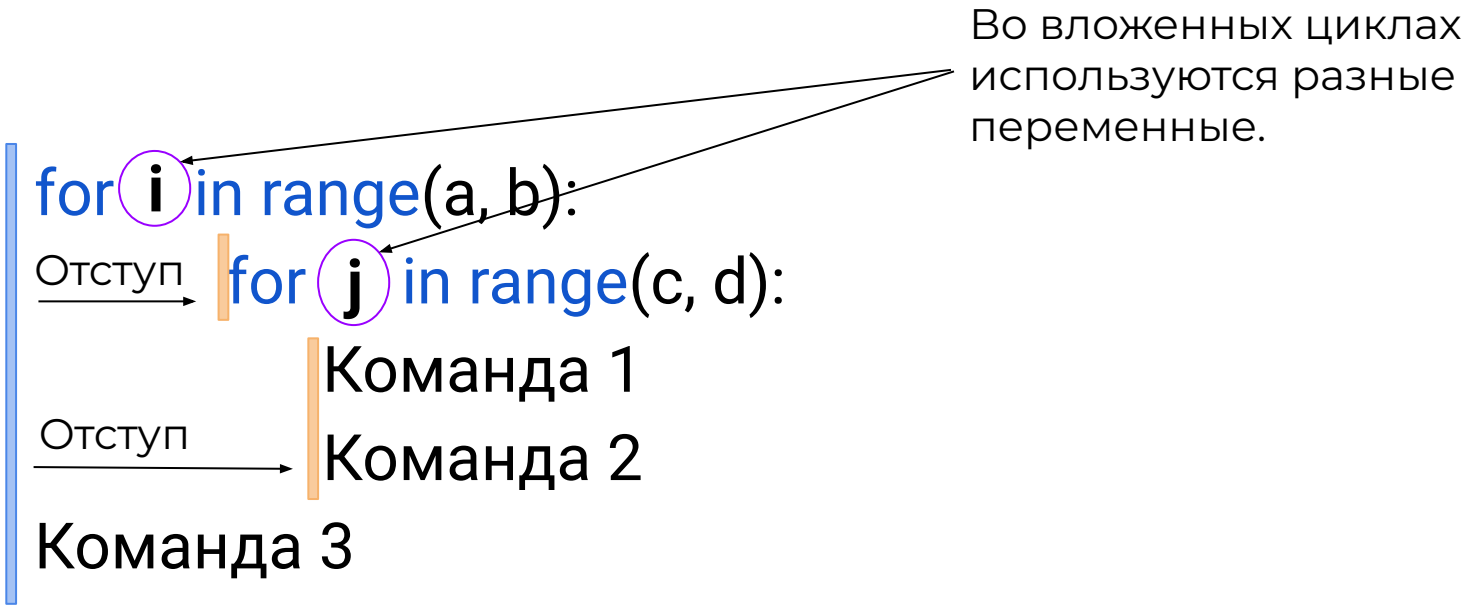
checklist = ["к", "с", "а", "б", "у", "е"] 

checklist = ["к", 3, "н", -4, "у", 153, -9, "г"] 

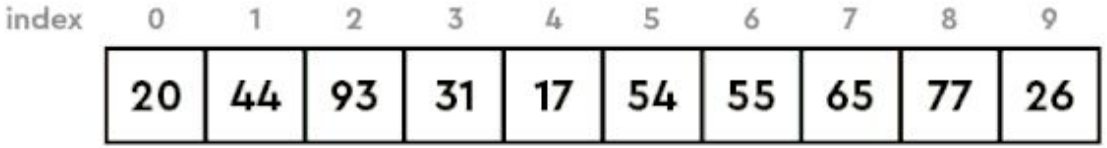
Вложенные циклы:



Обрати внимание!!!



Другие сортировки



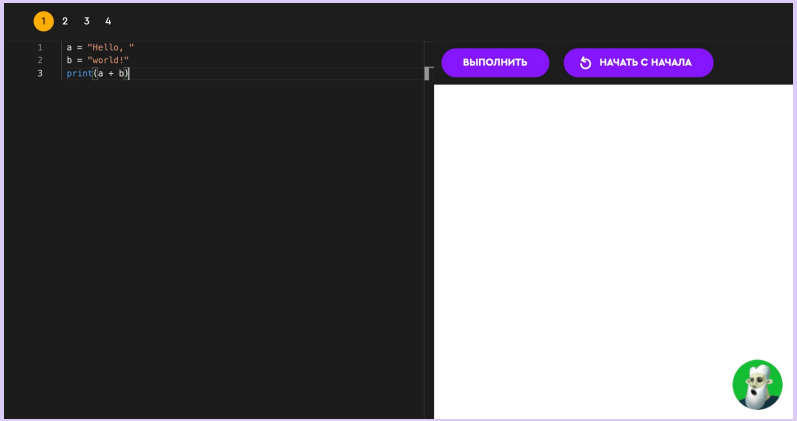
← Пузырьковая

6 5 3 1 8 7 2 4

← Сортировка вставками

Заходим на платформу

Сортировки



Задание на платформе

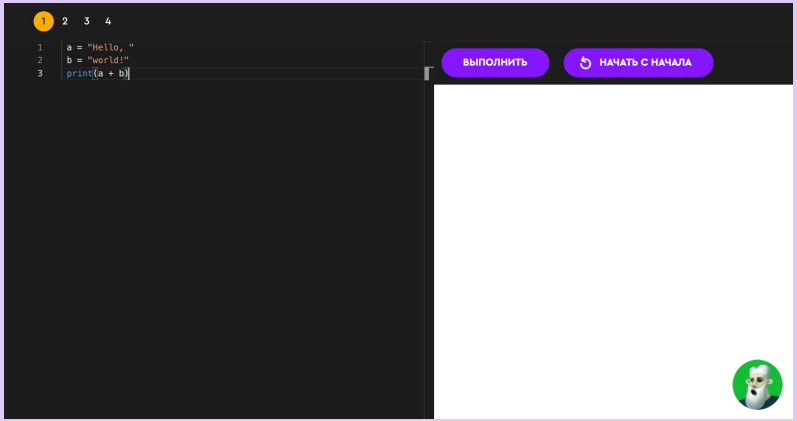
Итог первой половины урока

алгоритмика

Давайте отдохнём!

Заходим на платформу

Сортировки



Задание на платформе

алгоритмика

Сложность алгоритмов

На какие два фактора обращают внимание программисты при написании алгоритмов?

Демонстрация

(увеличение входных данных)

Скорость работы алгоритмов различной сложности

		Размер входных данных					
		10	20	30	40	50	
Сложность алгоритма	n	0,00001 сек.	0,00002 сек.	0.00003 сек.	0,00004 сек.	0,00005 сек.	**Время работы**
	n^3	0,001 сек.	0,008 сек.	0,027 сек.	0,0064 сек.	0,125 сек.	
	n^4	0,01 сек.	0,16 сек.	0,81 сек.	2,56 сек.	6,25 сек.	

Теория

Посчитаем, сколько времени затратит алгоритм сортировки «выбором»

Задача:

Сложность алгоритма сортировки выбором: N^2.
Размеры данных в списках: 10, 20 и 30 соответственно.
Зная эти данные, вычислите время работы алгоритма сортировки для каждого, из 3-х списков и оформите результат в виде таблицы.

Сложность и время работы алгоритма сортировки выбором

		Размер входных данных			
		10	20	30	
Сложность алгоритма	n^2	0,0001 сек.	0,0004 сек.	0,0009 сек.	**Время работы**

Теория

алгоритмика

Как прошло занятие?

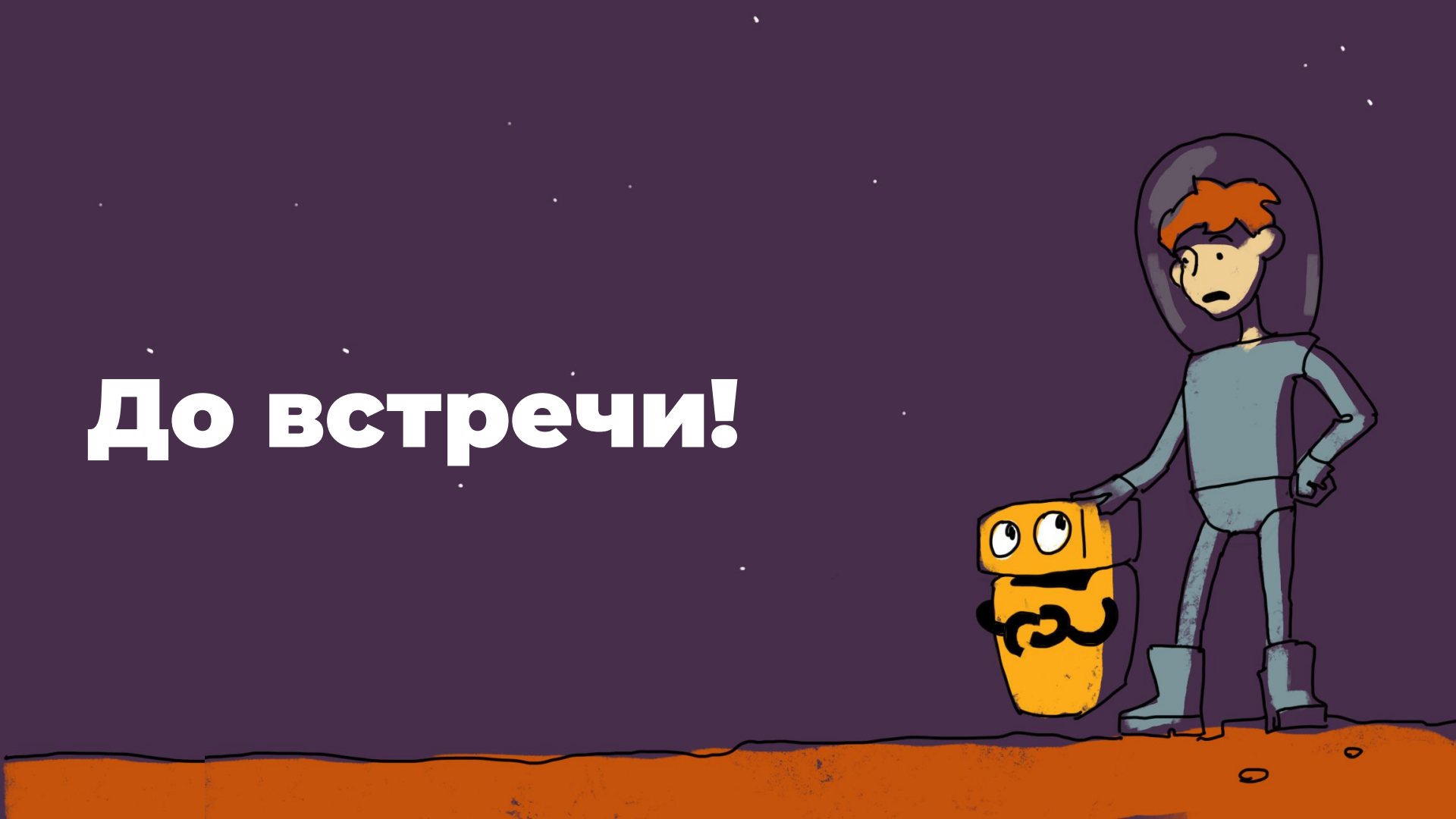
Проверь себя

- Что такое сортировка?
- Как реализовать сортировку?
- Как работает алгоритм сортировки выбором?
- Что такое вложенные циклы и как они работают?
- Какие ещё виды сортировок бывают?
- Что такое сложность алгоритма?
- На какие факторы она влияет?
- Как вычислить сложность алгоритма и время его работы? От чего это зависит?

На следующем занятии:

- Словари и множества — в чём взаимосвязь и на что они способны?

До встречи!



Сортировка «пузырьком»

Простой алгоритм сортировки, эффективный только для небольших списков.

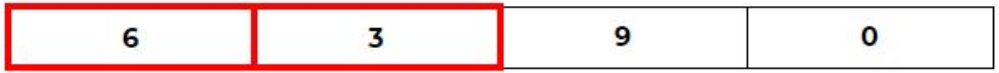
Реализация для сортировки по возрастанию: алгоритм проходит по всем элементам списка слева направо и сравнивает их попарно. Если элемент слева больше, чем элемент справа — они меняются местами.

Для реализации сортировки по убыванию условие сравнения противоположное.

Сортировка «пузырьком» (схема)

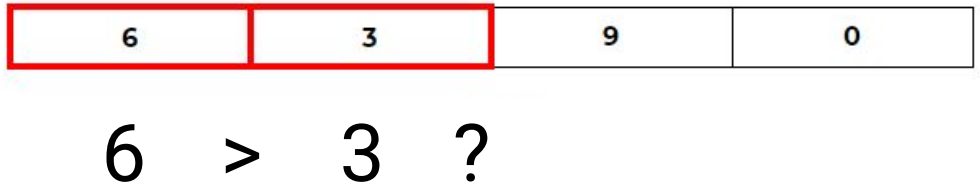
Дан список:

Выбираем первые два элемента:



Сортировка «пузырьком» (схема)

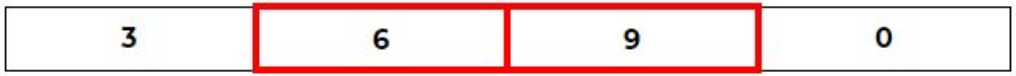
Сравниваем их между собой:



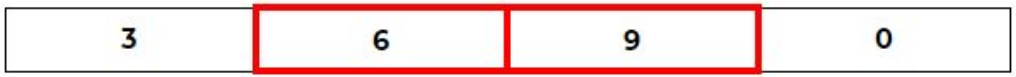
Если условие истинно (6 больше, чем 3), то меняем элементы местами:

Сортировка «пузырьком» (схема)

Берём следующие два элемента:



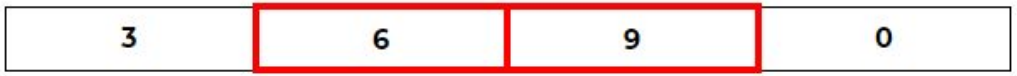
Также сравниваем их между собой:



6 > 9 ?

Сортировка «пузырьком» (схема)

Условие ложно (6 не больше, чем 9), значит элементы остаются на своих местах:



Обрати внимание!!!

Если перебраны все элементы списка, но он всё ещё не отсортирован, алгоритм действий начинается сначала, то есть вновь сравниваются первый и второй, затем второй и третий элементы и так далее, пока список не будет отсортирован.

Сортировка «выбором»

Реализация для сортировки по возрастанию:

1-й шаг. Выбираем наименьший элемент списка. Для удобства, считаем, что самый первый элемент — самый маленький.

2-й шаг. Перебираем все оставшиеся элементы и сравниваем с выбранным. Если нашёлся элемент меньше, запоминаем его.

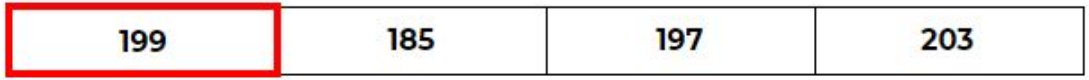
3-й шаг. Найденный наименьший элемент ставится в начало списка. Возвращаемся к первому шагу.

Для реализации сортировки по убыванию условие сравнения противоположное.

Сортировка «выбором» (схема)

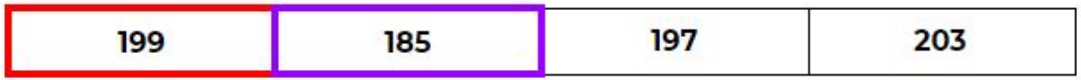
Дан список:

Запоминаем первый элемент:



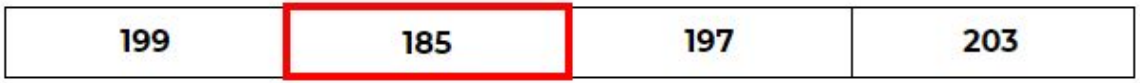
Сортировка «выбором» (схема)

Сравниваем с ним следующий элемент:



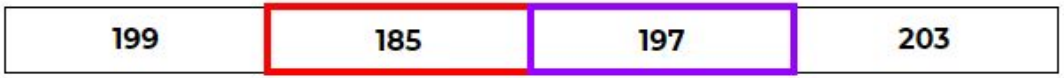
199 > 185 ?

Если условие истинно (199 больше, чем 185), то запоминаем новый элемент:



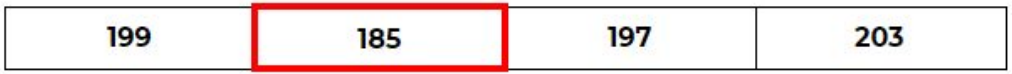
Сортировка «выбором» (схема)

Сравниваем с ним следующий элемент:



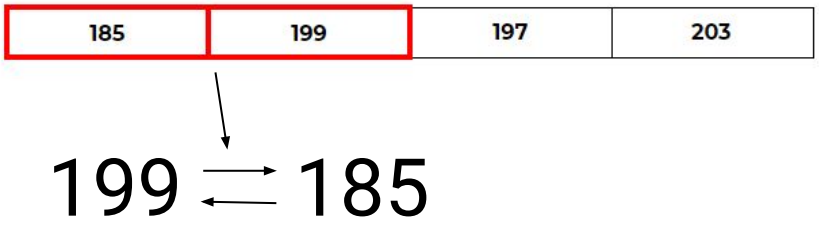
185 > 197 ?

Если условие ложно (185 меньше, чем 199), то наименьший элемент остаётся тем же:



Сортировка «выбором» (схема)

Сравниваем все оставшиеся элементы таким образом. Если элемента меньше не нашлось, то ставим наименьший, который мы запомнили, в начало списка. Элемент, стоявший там, ставим на место наименьшего:



Конец сортировки

Следуя такому алгоритму действий, все элементы перебираются, сравниваются и меняются местами до тех пор, пока список не будет отсортирован.

Сортировка «вставками»

Реализация для сортировки по возрастанию:

1-й шаг. Выбираем наименьший элемент списка. Для удобства, считаем, что самый первый элемент — самый маленький.

2-й шаг. Перебираем все оставшиеся элементы и сравниваем с выбранным. Если нашёлся элемент меньше, ставим его на первое место в списке, а другие элементы сдвигаем на одну позицию вперёд.

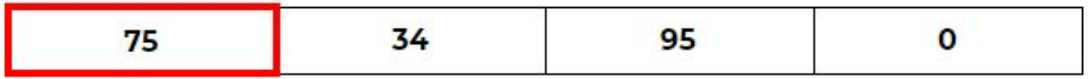
Повторяем эти действия до тех пор, пока список не будет отсортирован.

Для реализации сортировки по убыванию условие сравнения противоположное.

Сортировка «вставками» (схема)

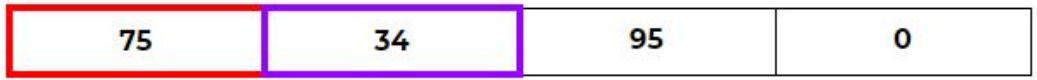
Дан список:

Запоминаем первый элемент:



Сортировка «вставками» (схема)

Сравниваем с ним следующий элемент:

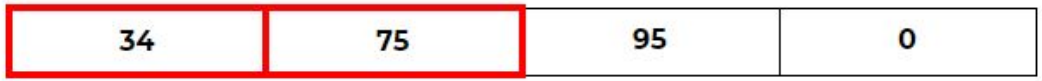


75 > 34 ?

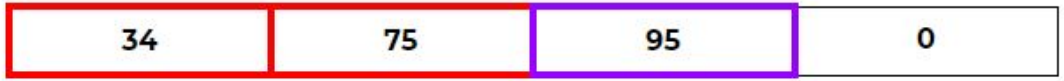
Если условие истинно (75 больше, чем 34), то найден элемент меньше. Ставим его в начало списка, а все остальные элементы сдвигаем на 1 позицию вперёд:

Сортировка «вставками» (схема)

Запоминаем два первых элемента:



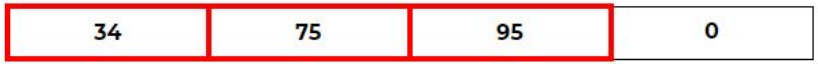
Сравниваем с ними по очереди следующий элемент:



34 > 95 ? 75 > 95 ?

Сортировка «вставками» (схема)

Если элемент оказался меньше одного из тех, что мы запомнили, то он ставится на его место, а все оставшиеся сдвигаются вперёд на одну позицию. В нашем случае элемент 95 больше, чем 75 и 34. Значит — он остаётся на своём месте:



Конец сортировки

Следуя такому алгоритму действий, все элементы сравниваются с теми, что мы запомнили ранее, и, если находится элемент меньше, он встаёт на своё место, а все остальные сдвигаются вперёд на одну позицию до тех пор, пока список не будет отсортирован.

Встроенная сортировка

В языке программирования Python есть встроенная функция для сортировки — sorted(). В качестве её аргумента указывается список, элементы которого необходимо отсортировать. Стандартно функция упорядочивает элементы в порядке возрастания. Если необходимо отсортировать список в порядке убывания, необходимо, через запятую, записать в функцию второй параметр — reverse и присвоить ему значение True.

Пример (сортировка по возрастанию):

```
checklist = [1, 56, 0, -5]
checklist = sorted(checklist)
print(checklist)
```

Вывод:

[-5, 0, 1, 56]

Пример (сортировка по убыванию):

```
checklist = [1, 56, 0, -5]
checklist = sorted(checklist, reverse = True)
print(checklist)
```

Вывод:

[56, 1, 0, -5]