Курс «Алгоритмы и структуры данных на Python»

Регламент курса

- Время урока - 1,5-2 часа
- Подготовленные листинги с примерами кода
- Практические задания (с разбором и размещением примеров выполнения)
- Видеозапись на следующий день после урока
- Вопросы – в ходе урока
- Большой упор на внеурочную работу (обсуждение материала с преподавателем в чате, в ЛС телеграма) – логин преподавателя @cdi999

Внимание!

Курс примерно на 50% отличается от материалов, которые представлены в методичках, кроме того отличается порядок уроков и названия некоторых тем.

Почему?

Преподаватель считает, что материалы нужно постоянно актуализировать, но согласование изменений с поддержкой – долгий процесс, поэтому у вас будут «старые» методичка и презентация, НО ИХ МЫ НЕ БЕРЕМ. В раздел Материалы к уроку будут выкладываться актуальные материалы для курса.

Внимание!

Поэтому со старой версией курса есть расхождения.
НО! Ориентируемся мы именно на новую версию.
По отзывам слушателей, курс от изменений стал лучше:

Хороший курс. В силу обстоятельств проходил его дважды(пришлось перенести из-за болезни) и во второй раз было ещё интереснее, чем в первый. Преподаватель старается актуализировать материал, сделать задачи и темы лучше и интереснее. Как минус такого подхода - не на все темы есть сразу полностью готовые методички, но всё, что задаёт, преподаватель разбирает на уроках. Объясняет всё понятно и доходчиво, часто по несколько раз. Так что недостаток методичек решается пересмотром уроков и, в крайнем случае, сайтами с документацией питона. Всем доволен =)

Преподаватель Дмитрий Читалов • 5 Курс «Алгоритмы и структуры данных на Python. Базовый курс» • 5

Преподаватель в начале курса предупредил: так как курс многие недолюбливали, поскольку он почти полностью повторял курс Основы Python, преподаватель перестроил программу курсов, изменил материалы и дз. По началу я отнесся к этому скептически, а потом курс по-настоящему понравился. По ходу курса получил кучу полезных навыков, от замерки времени до оптимизации памяти и построения деревьев. Преподавателю поставил бы 7 за старания, но максимальная оценка, к сожалению, только 5

Преподаватель Дмитрий Читалов • 5 Курс «Алгоритмы и структуры данных на Python. Базовый курс» • 5

Практические задания

- В начале каждого урока − обсуждаем очередное задание.
- Выполненные задания ЖЕЛАТЕЛЬНО! сдавать к началу очередного урока, а лучше раньше (идеально приступать к выполнению в ближайшее время после урока).
- Хотя сейчас дэдлайн существенно увеличен и составляет не 3, а 7 дней, преподаватель настоятельно рекомендует сдавать именно в три дня, а не через неделю, месяц, год. Делать нужно по «свежим следам», оперативно, а не когда курс завершился.
- Текст каждого задания и пример его выполнения задания будет размещаться в разделе «Материалы» соответствующего урока.

Практические задания

- Сдача задания ТОЛЬКО в виде ссылки на pull-request! Эту ссылку вы прикладываете к форме сдачи ДЗ. Подчеркиваю – ссылка на ПР, а не на ваш репозиторий!
- Задания, сданные как-то по другому (в виде архива, файлов, скриншотов, ссылок на репозитории и т.д.) получают оценку не выше Удовлетворительно!!!!! Понимаю недовольство некоторых слушателей курса, но видимо другого пути убедить в необходимости освоения Гита нет!

Практические задания

Почему такая строгость с форматом сдачи ДЗ:

- Разработчик в современных реалиях совершенно обязан уметь пользоваться системами контроля версий и сервисами хостинга IT-проектов.
- Можно сколько угодно давать себе послабления и не изучать эти инструменты, но рано или поздно они понадобятся. Поэтому лучше их освоить прямо сейчас.
- По идее эти инструменты вы должны были освоить на курсе Основы Python, а если не освоили, значит где-то поленились. Придется срочно осваивать сейчас.

Практические задания

Почему такая строгость с форматом сдачи ДЗ:

- Убедительная просьба, не нужно задавать преподавателю вопросы, как создать pull-request, как прикрепить ссылку и т.д. Эти вопросы не имеют отношения к курсу Алгоритмов. Это вспомогательные темы, которыми вы должны уже владеть, хотя бы на базовом уровне. Эти темы в идеале должны быть освоены на курсе Основ Python.
- Если же вы не знаете как создавать репозитории, ветки и ПР, вашему вниманию курс: Git. Базовый курс (https://geekbrains.ru/courses/1117)

Как создать ссылку на ПР

Если же вы все-таки испытываете сложности с созданием ПР, небольшая подсказка:

- ❑ Сделать форк репозитория преподавателя - https://github.com/DmitryChitalov/algorithms_2022
- ❑ Создать локальную копию репозитория у себя на машине.
- ❑ Создавать под каждый урок локальную ветку, например, lesson_1.
- ❑ Из этой локальной ветки коммитите и пушите изменения в такую же, но удаленную ветку сделанного форка.
- ❑ Переходим на сайт github
- ❑ Из каждой удаленной ветки делаете ПР в мой (А НЕ В СВОЙ) репозиторий.
- ❑ Получаете требуемую ссылку на ПР.

Возможны ли пересдачи и сдачи позднее регламента?

Преподаватель всегда готов идти навстречу. НО! Тема очень болезненная, потому что слушатели упорно не хотят услышать как нужно пересдавать. И шлют сообщения вроде «Я хочу пересдать задание за пятый урок»

Как преподаватель должен узнать за какой курс, от какой даты его старт, за какой урок, от какого слушателя??

ПОЭТОМУ ПРОШУ ОБРАТИТЬ ВНИМАНИЕ НА ИНСТРУКЦИЮ ПО ПЕРЕСДАЧАМ И СДАЧАМ С ОПОЗДАНИЕМ

Возможны ли пересдачи и сдачи позднее регламента?

1. Сдача позднее установленного дедлайна!

Ничего страшного, всякое бывает, заболели (не дай бог), уехали. Пишем сразу в поддержку (А НЕ МНЕ), что вы хотите пересдать с указанием названия курса, даты его старта, номера урока, ваших ФИО. Преподавателю писать не нужно, он не против поздних сдач.

Возможны ли пересдачи и сдачи позднее регламента?

2. Пересдача для повышения оценки! Например, вы сдали заглушку и получили Удовл. (кстати, заглушки – формальные сдачи, принимаются и за них ставятся тройки) или вас просто не устраивает оценка. Вы пишете СРАЗУ! Преподавателю в телеграм (@cdi999) и говорите название курса, дату его старта, номер урока, ваши ФИО! Пересдавать конечно требуется после доработок!

Возможны ли пересдачи и сдачи позднее регламента?

Теперь самое главное по досдачам и пересдачам! Как они делаются, написано в презентации и рассказано на первом уроке! Если обращение о досдаче/пересдаче подается с нарушением указанных выше требований, ОНО БУДЕТ ОСТАВЛЕНО БЕЗ ОТВЕТА!

Поэтому просьба в этом случае не писать негодования в поддержку и отзывы, потому как пересдачи и досдачи в Гикбреинс не являются обязанностью преподавателя, а делаются исключительно по его желанию. Я не против пересдач, но прошу соблюдать требования по их выполнению.

Возможны ли пересдачи и сдачи позднее регламента?

Почему так строго?

Потому что группы большие и преподавателю приходится тратить рабочее время на выяснения с какого курса слушатель, от какой даты и т.д.

Прошу отнестись с пониманием и экономить время друг друга.

Возможны ли пересдачи и сдачи позднее регламента?

ПОЭТОМУ, УВАЖАЕМЫЕ СЛУШАТЕЛИ КУРСА, ПРОШУ ВАС ЕЩЕ РАЗ ИЗУЧИТЬ ПРЕЗЕНТАЦИЮ, ЧТО-ТО ЗАПИСАТЬ И НЕ ЗАДАВАТЬ ПРЕПОДАВАТЕЛЮ ВОПРОСЫ, КОТОРЫЕ УЖЕ БЫЛИ РАЗОБРАНЫ.

Пример сообщения: «Я Иван Иванов, с курса Алгоритмов (от 11 января), хочу пересдать задание к уроку 1, прикладываю ссылку на ПР с доработкой»

Пример сообщения, на которое не будет ответа: «Я с курса Алгоритмов, хочу пересдать задание к уроку 1, прикладываю ссылку на ПР с доработкой»

План обновленного курса:

- Урок 1. Введение в алгоритмы и структуры данных на Python
- Урок 2. Рекурсия
- Урок 3. Хеширование
- Урок 4. Профилирование времени работы алгоритмов
- Урок 5. Специализированные коллекции
- Урок 6. Профилирование памяти
- Урок 7. Алгоритмы сортировки
- Урок 8. Бинарные деревья

Почему нужно проходить этот курс:

- ✔ Правильно составленный алгоритм позволяет ускорить работу кода.
- ✔ Применение алгоритмов при решении реальных задач повышает статус разработчика.
- ✔ Алгоритмы – это структурные блоки систем искусственного интеллекта, машинного обучения и data science.

Цели курса

1. Научиться сравнивать различные варианты решения задач по ключевым критериям и выбирать наиболее эффективный в текущей ситуации вариант.
2. Развитие алгоритмического мышления, т.е. научиться представлять весь ход решения задачи от ее постановки до получения итогового результата.
3. Научиться перекладывать сформулированный алгоритм на язык реализации.
4. Освоить фундаментальные алгоритмы, не зависящие от языка реализации, например, решето Эратосфена, алгоритмы сортировки и т.д.
5. Изучить возможностей применения приемов алгоритмизации для реальных задач.

Напутствие!

Уважаемые студенты, просьба с пониманием отнестись к оценкам, мы ведь не в школе, главное – знания.

Этот курс обязательно повысит ваш опыт и принесет новые знания!

Но если вы не согласны с оценкой и комментариями по коду, сразу пишите преподавателю в телеграм, вместе мы все решим. Преподаватель тоже может что-то не заметить или сделать ошибку.

Урок 1

Введение в алгоритмы и структуры данных на Python

Классические варианты функций

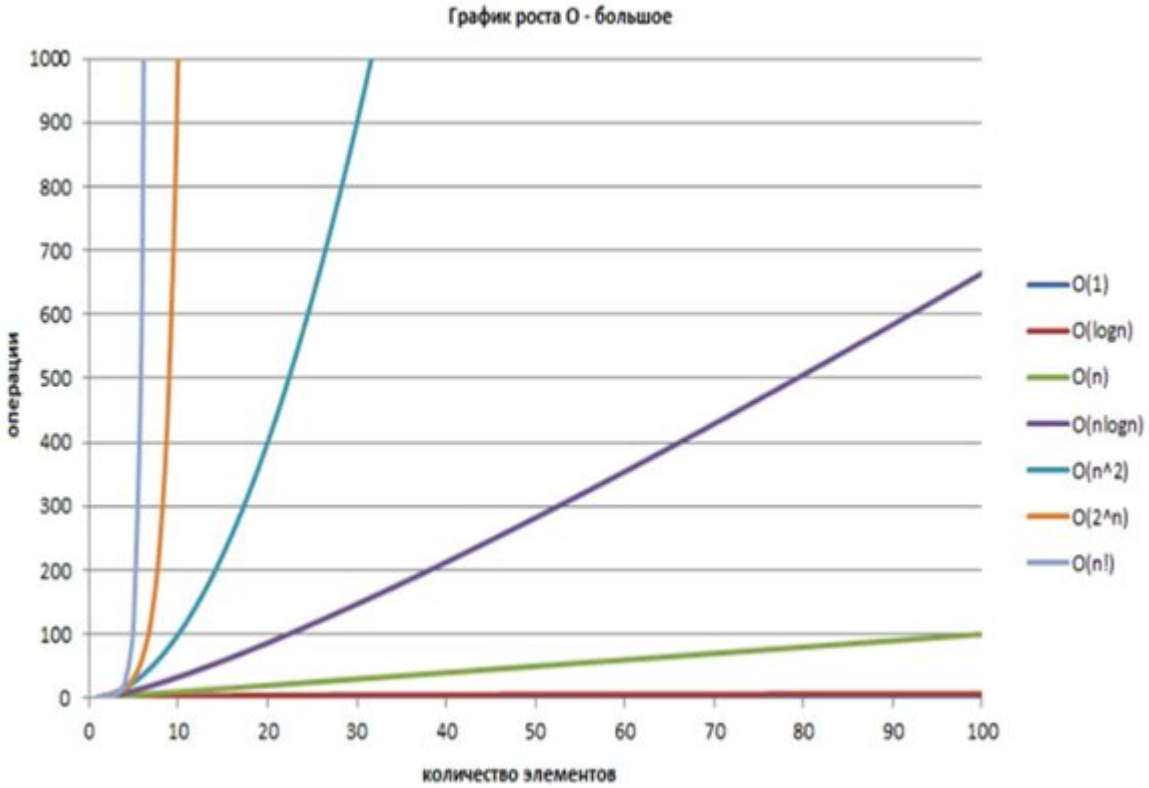
$$T(n) = 5n^2 + 27n + 1005.$$

Таблица 1: Наиболее распространённые функции для “большого O”

f(n)	Название
1	Константная
$\log n$	Логарифмическая
n	Линейная
$n \log n$	Линейно-логарифмическая
n^2	Квадратичная
n^3	Кубическая
2^n	Экспоненциальная

Рост времени

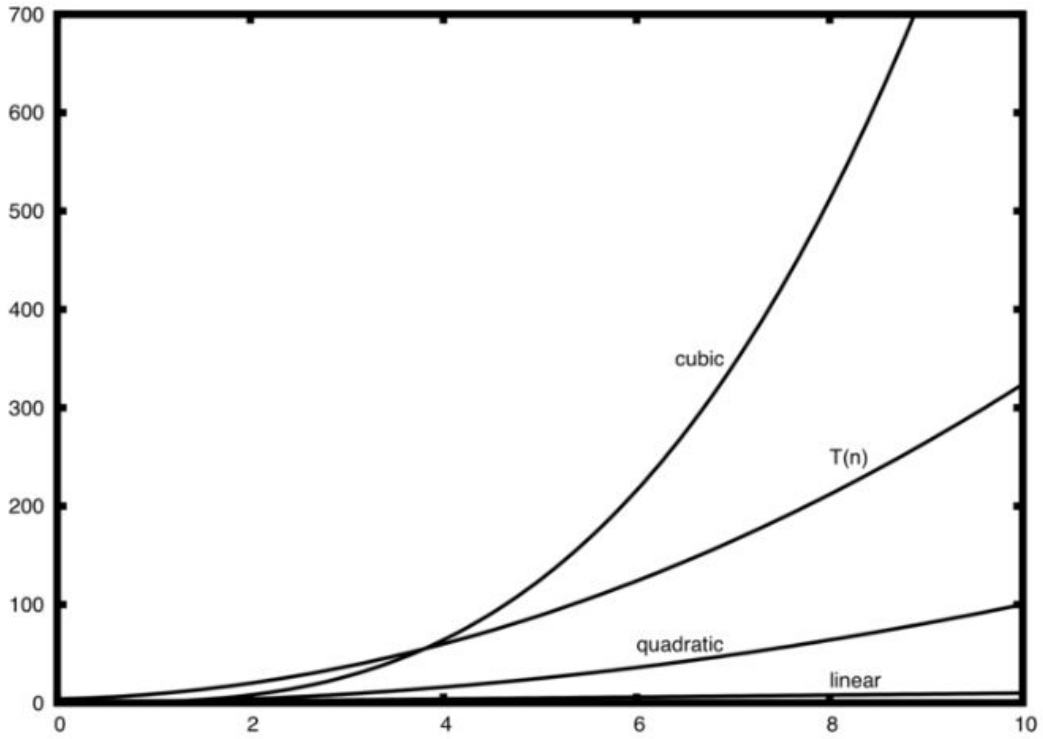
Графики функций



Классические примеры

- O(log n) – логарифмическое время. Пример: двоичный поиск.
- O(n) – линейное время. Пример: простой поиск.
- O(n * log n). Пример: быстрый алгоритм сортировки, такой как quicksort (быстрая сортировка).
- O(n2) – квадратичное время. Пример: медленный алгоритм сортировки, такой как сортировка выбором.
- O(n!) – факториальное время. Пример: очень медленный алгоритм, такой как в задаче коммивояжера.

Как узнать сложность своего алгоритма через график



Сложность базовых операций и функций заранее известна

Таблица 6.1.1 - Асимптотические сложности для списка или кортежа¶

Операция	Сложность	Примечание
`len(lst)`	$O(1)$	
`lst.append(5)`	$O(1)$	
`lst.pop()`	$O(1)$	Аналогично `lst.pop(-1)`
`lst.clear()`	$O(1)$	Аналогично `lst = []`
`lst[a:b]`	$O(b-a)$	
`lst.extend(...)`	$O(len(...))$	Зависит от длины аргумента
`list(...)`	$O(len(...))$	Зависит от длины аргумента
`lst1 == lst2`	$O(N)$	
`lst[a:b] = ...`	$O(N)$	
`del lst[i]`	$O(N)$	
`lst.remove(...)`	$O(N)$	
`x in/not in lst`	$O(N)$	Поиск в списке
`lst.copy()`	$O(N)$	Аналогично `lst[:]`
`lst.pop(i)`	$O(N)$	Точнее $O(N)$
`min(lst)/max(lst)`	$O(N)$	
`lst.reverse()`	$O(N)$	
`for v in lst:`	$O(N)$	
`lst.sort()`	$O(N \log N)$	Направление сортировки не играет роли
`k * lst`	$O(k \cdot N)$	

Таблица 6.1.2 - Асимптотические сложности для множества¶

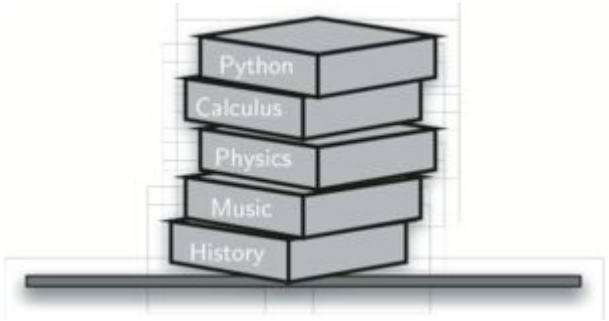
Операция	Сложность	Примечание
`len(s)`	$O(1)$	
`s.add(5)`	$O(1)$	
`x in/not in s`	$O(1)$	В отличие от списка, где $O(N)$
`s.remove(5)`	$O(1)$	В отличие от списка, где $O(N)$
`s.discard(5)`	$O(1)$	
`s.pop(i)`	$O(1)$	В отличие от списка, где $O(N)$
`s.clear()`	$O(1)$	Аналогично `s = set()`
`set(...)`	$O(len(\dots))$	Зависит от длины аргумента
`s != t`	$O((len(s))$	Аналогично `len(t)`
`s <= t`	$O(len(s))$	
`s >= t`	$O(len(t))$	
`s | t`	$O(len(s) + len(t))$	
`s & t`	$O(len(s) + len(t))$	
`s - t`	$O(len(s) + len(t))$	
`s ^ t`	$O(len(s) + len(t))$	
`for v in s:`	$O(N)$	
`s.copy()`	$O(N)$	

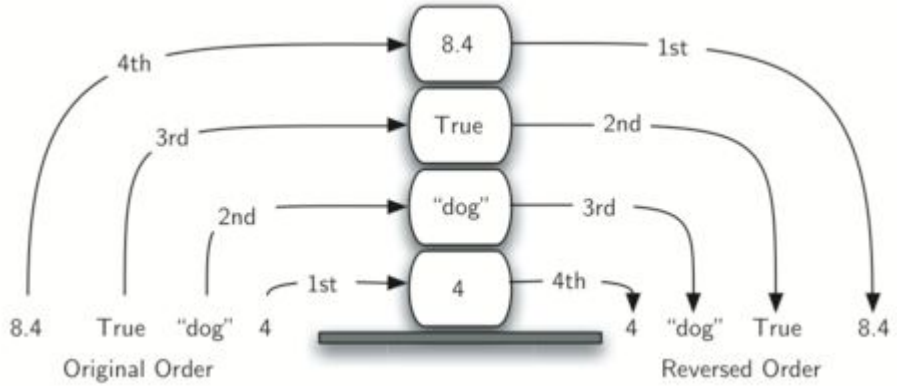
Таблица 6.1.3 - Асимптотические сложности для словаря¶

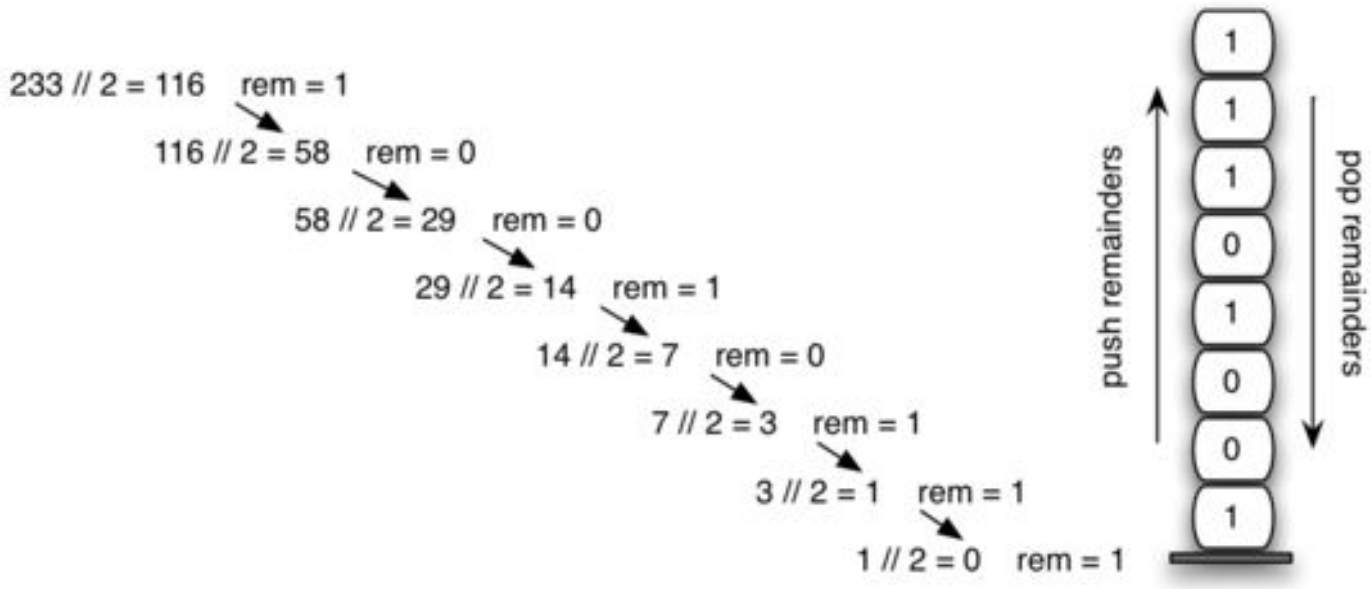
Операция	Сложность	Примечание
`d[k]`	$O(1)$	
`d[k] = v`	$O(1)$	
`len(d)`	$O(1)$	
`del d[k]`	$O(1)$	
`d.method`	$O(1)$	
`d.pop(k)`	$O(1)$	
`d.popitem()`	$O(1)$	
`d.clear()`	$O(1)$	Аналогично `s = {}` или `dict()`
`d.keys()`	$O(1)$	
`dict(...)`	$O(len(...))$	Зависит от длины аргумента
`for k in d:`	$O(N)$	Для всех методов: keys(), values(), items()

Немного о фундаментальных структурах

Стек






233 // 2 = 116 rem = 1
116 // 2 = 58 rem = 0
58 // 2 = 29 rem = 0
29 // 2 = 14 rem = 1
14 // 2 = 7 rem = 0
7 // 2 = 3 rem = 1
3 // 2 = 1 rem = 1
1 // 2 = 0 rem = 1
push remainders
pop remainders
1
1
1
0
1
0
0
1

Очередь

Рисунок 1: Очередь из объектов данных Python

Дек

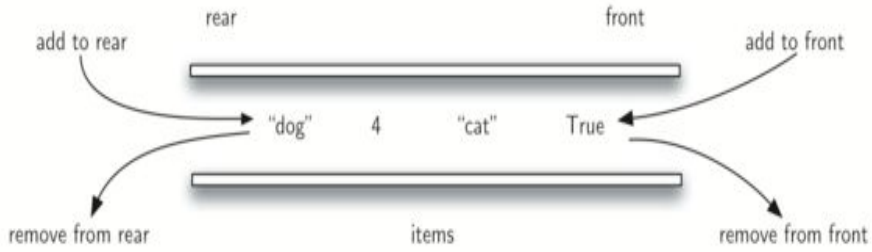


Рисунок 1: Дек из объектов данных Python

Дек

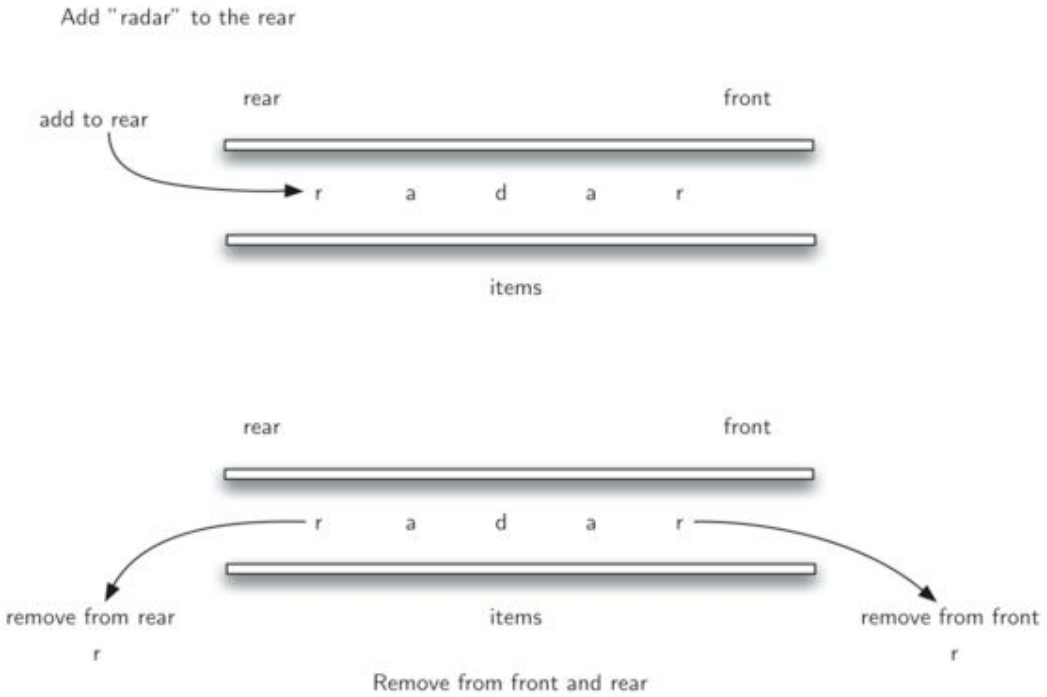


Рисунок 2: Дек