Стеки и очереди в Python

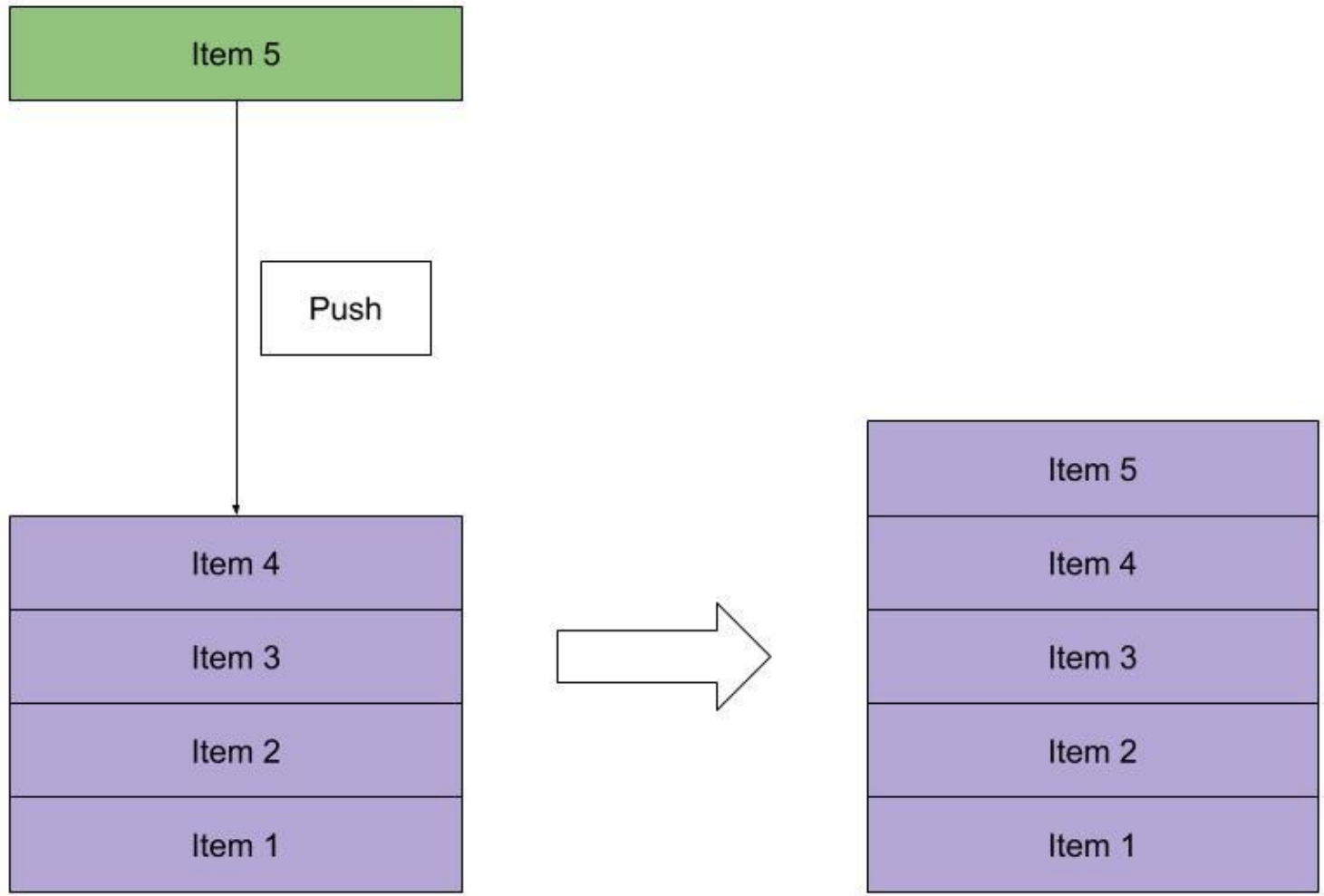
- Структуры данных организуют хранение в компьютерах, чтобы мы могли эффективно получать доступ к данным и изменять их.
- *Стеки* и *Очереди* являются одними из самых ранних структур данных, определенных в информатике.

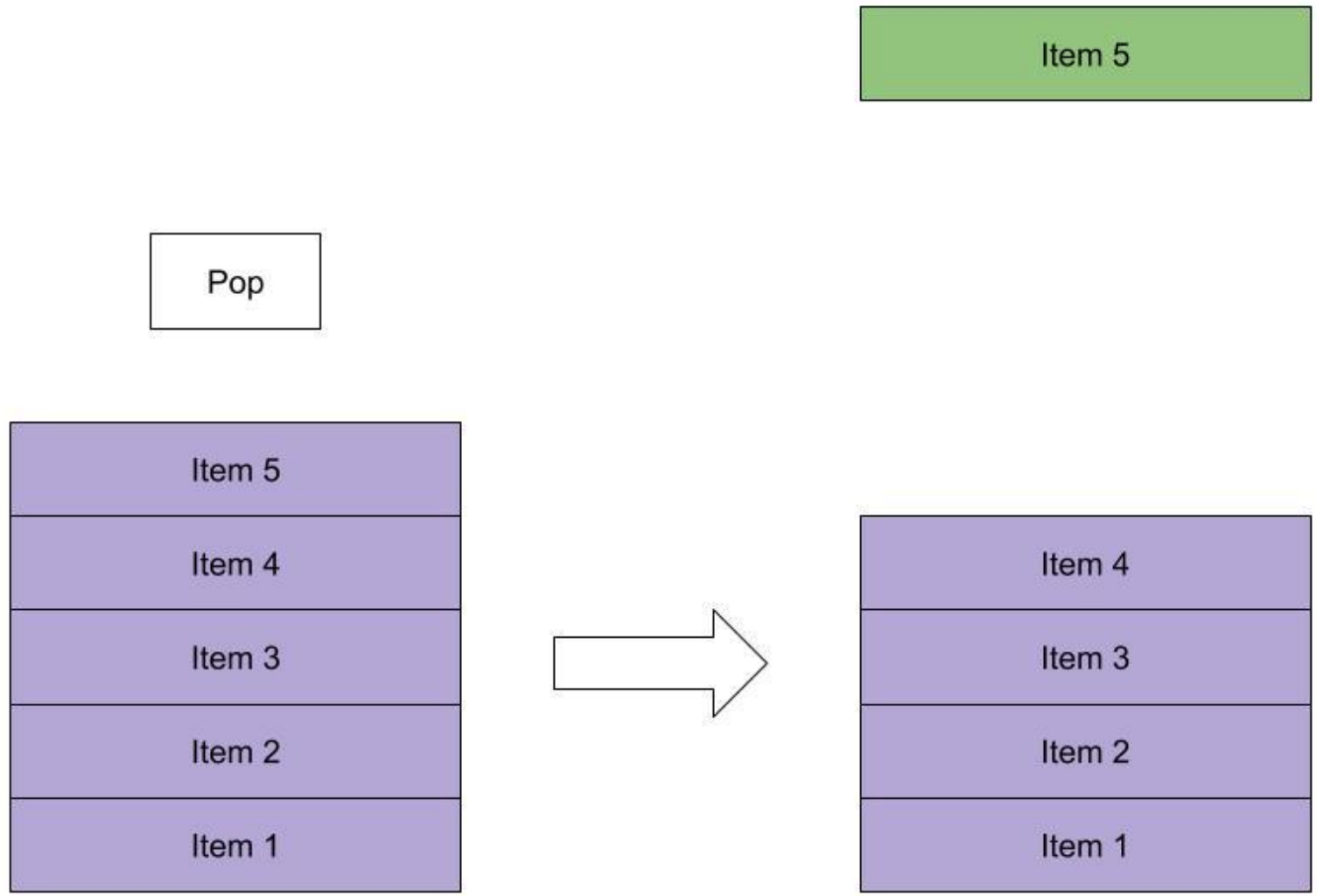
Стек

- Стеки, как следует из названия, следуют принципу Last-in-First-Out (LIFO). Как если бы мы складывали монеты одну на другую, последняя монета, которую мы кладем сверху, – это та, которая будет первой извлечена из стопки позже.
- Мы имеем доступ только к последнему(верхнему) элементу

Поэтому для реализации стека нам нужны две простые операции:

- push – добавляет элемент в верхнюю часть стека
- pop – удаляет элемент в верхней части стека


Item 5
Push
Item 4
Item 3
Item 2
Item 1
Item 5
Item 4
Item 3
Item 2
Item 1

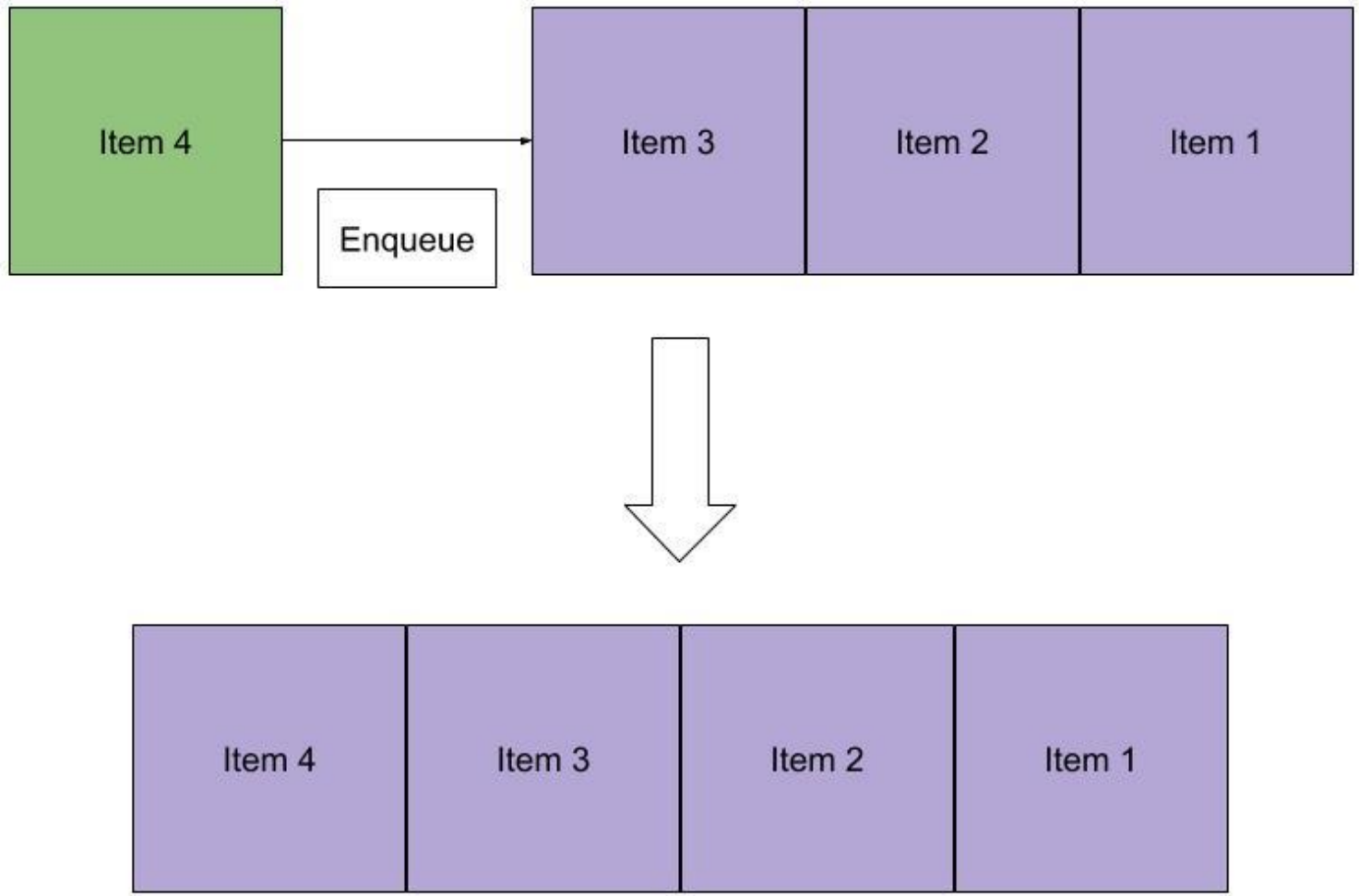

Item 5
Pop
Item 5
Item 4
Item 3
Item 2
Item 1
Item 4
Item 3
Item 2
Item 1

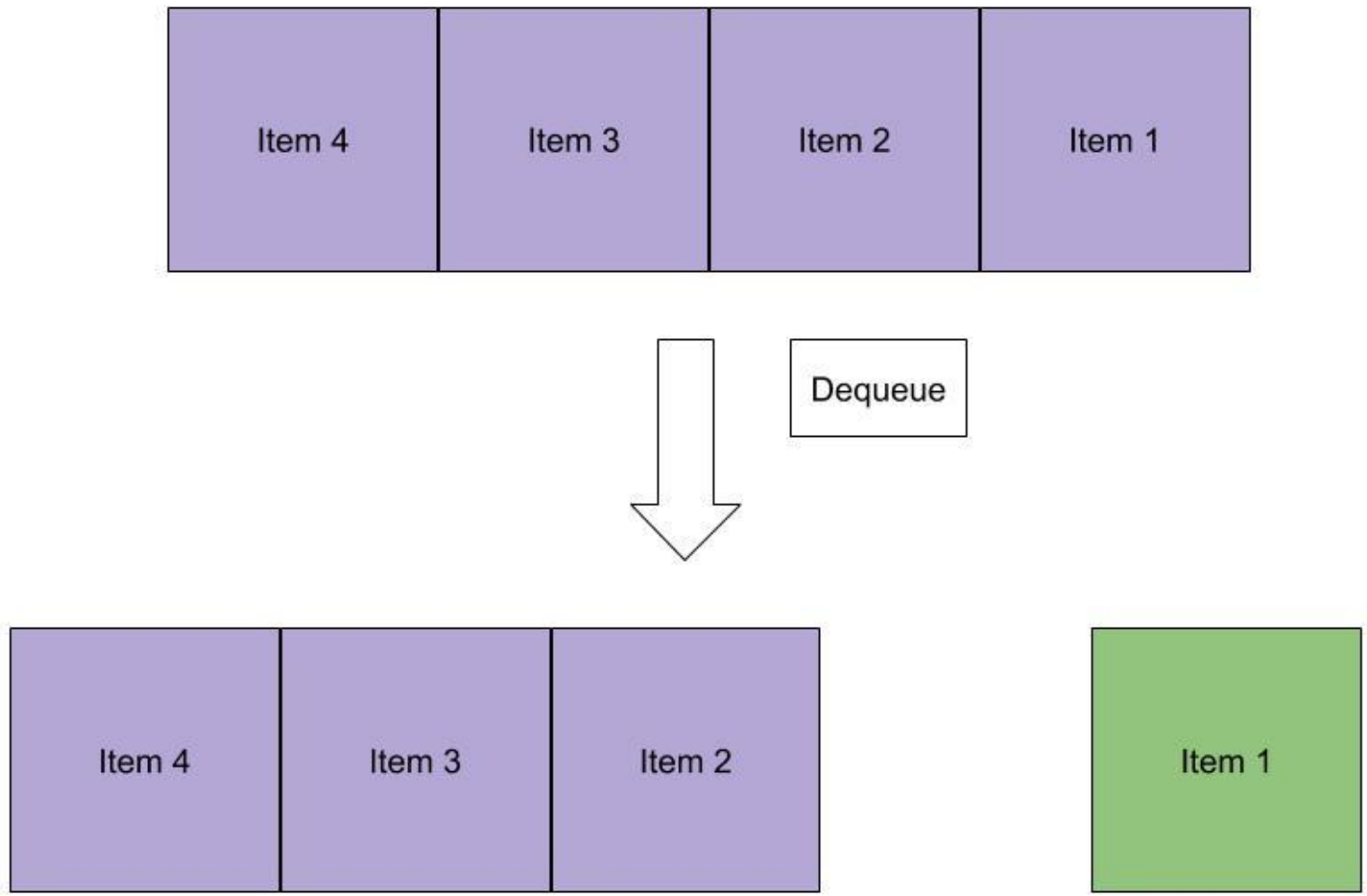
```
# Добавим буквы в list
letters.append('c')
letters.append('a')
letters.append('t')
letters.append('g')

# Теперь извлечем с помощью pop
last_item = letters.pop()
print(last_item)

last_item = letters.pop()
print(last_item)

print(letters) # ['c', 'a']
```

Очередь

- Очереди, как следует из названия, следуют принципу First-in-First-Out (FIFO). Как будто ожидая в очереди за билетами в кино, первый, кто встанет в очередь, первым купит билет и насладится фильмом.
- Мы имеем доступ только к первому и последнему элементу

Поэтому для реализации очереди нам нужны две простые операции:
enqueue – добавляет элемент в конец очереди
dequeue – удаляет элемент в начале очереди


Item 4
Enqueue
Item 3
Item 2
Item 1
Item 4
Item 3
Item 2
Item 1


Item 4
Item 3
Item 2
Item 1
Dequeue
Item 4
Item 3
Item 2
Item 1

```
letters = []
fruits = []
# Добавим уникальные фрукты в список
fruits.append('banana')
fruits.append('grapes')
fruits.append('mango')
fruits.append('orange')
#
first_item = fruits.pop(0)
print(first_item)
#
first_item = fruits.pop(0)
print(first_item)

#
print(fruits) # ['c', 'a']
```

Реализации, использующие стандартные методы List являются довольно медленными, если список будет большим

В Python есть встроенный тип данных deque, которая предоставляет последовательность с эффективными методами для работы в виде стека или очереди.

- `from collections import deque`

```python
from collections import deque
# создаем пустую deque
numbers = deque()

# добавляем элементы
numbers.append(99)
numbers.append(15)
numbers.append(82)
# извлекаем элемент как в стеке
last_item = numbers.pop()
print(last_item) # 47
print(numbers) # deque([99, 15, 82, 50])

# извлекаем элемент как в очереди
first_item = numbers.popleft()
print(first_item) # 99
print(numbers) # deque([15, 82, 50])
```

Сравним время

- Сгенерируем список из 1 000 000 элементов, а затем с помощью метода pop для списков и методов pop и popleft для deque извлечем 100 000 элементов
- Для списка
  - pop – 2.7999999998584713e-06
  - pop(0) – 2.7999999998584713e-06
- Для deque
  - pop -3.7999999999982492e-06
  - popleft – 0.0015673000000049342