Using the Right Data Structures

Dan Tofan

Software Engineer, PhD

@dan_tofan | www.programmingwithdan.com

Module Overview

Which data structure is faster?

Comparing lists and arrays

Comparing sets and tuples

Comparing queues and deques

Using dictionaries

Comparing data classes, dictionaries and named tuples

Evaluating Performance of Operations

Python data structures

- Lists
- Dictionaries
- Sets

Operations

- Adding
- Finding
- Deleting

Big O Notation

Machine independent

Widely accepted

Performance for large inputs

```
amounts = [3,11,25]
```

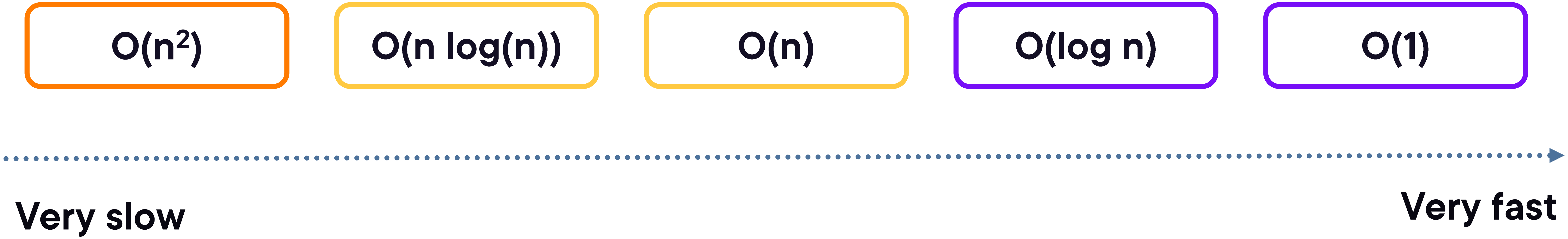
```
double_top = amounts[0] * 2
```

◀ **Constant, O(1)**

```
sum = 0
for amount in amounts:
     sum += amount
```

◀ **Linear, O(n)**

Performance Hierarchy with Big O Notation



$O(n^2)$

$O(n \log(n))$

$O(n)$

$O(\log n)$

$O(1)$

Very slow

Very fast

Demo

Write O(1) code

Write O(n) code

Compare performance

Lists

Ordered collections of items

Allow mixed types

Implemented as resizable arrays

Lists Performance

Very fast – O(1):

- Getting
- Setting
- Appending

Slow – O(n):

- Finding
- Removing

Memory allocation

- Extra room for future appends
- Old list is copied to the new list

Arrays

Built-in arrays

Compact data storage
Only for certain types
Less popular

NumPy arrays

Numeric computations
Items of different types
Very popular

Demo

Run: pip install numpy

Double some numbers

- Stored in a list
- Stored in a NumPy array

Compare performance

Sets

Unordered collections

Unique items

Immutable items

Sets are mutable

Sets Performance

Very fast – O(1):

- Adding
- Deleting
- Membership checking

Slow – O(n):

- Remove duplicates

Tuples

Immutable lists

Lightweight

Faster than lists

Sets vs Tuples

Sets	VS	Tuples
Mutable		Immutable
Unordered collection		Ordered collection
Unique, immutable items		Fixed content
Fast membership check		Memory efficient
Sets-specific operations		

Demo

Check membership of some items

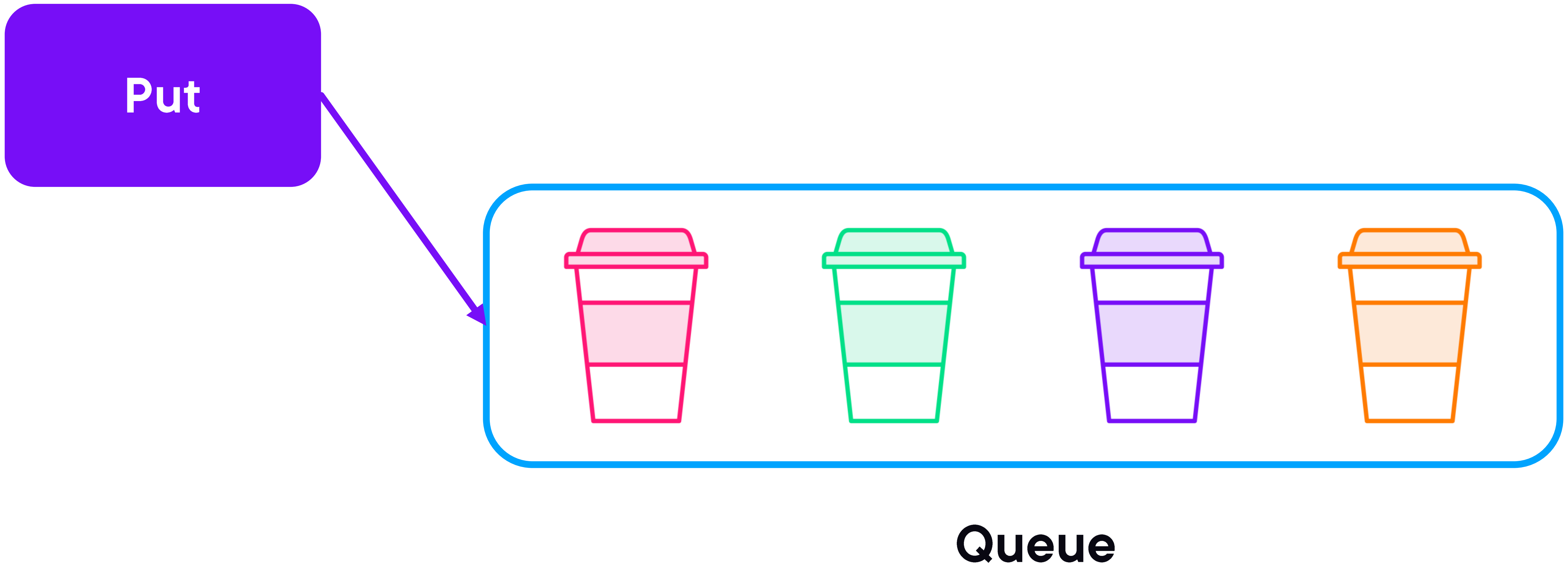
- In a list
- In a set
- In a tuple

Compare results

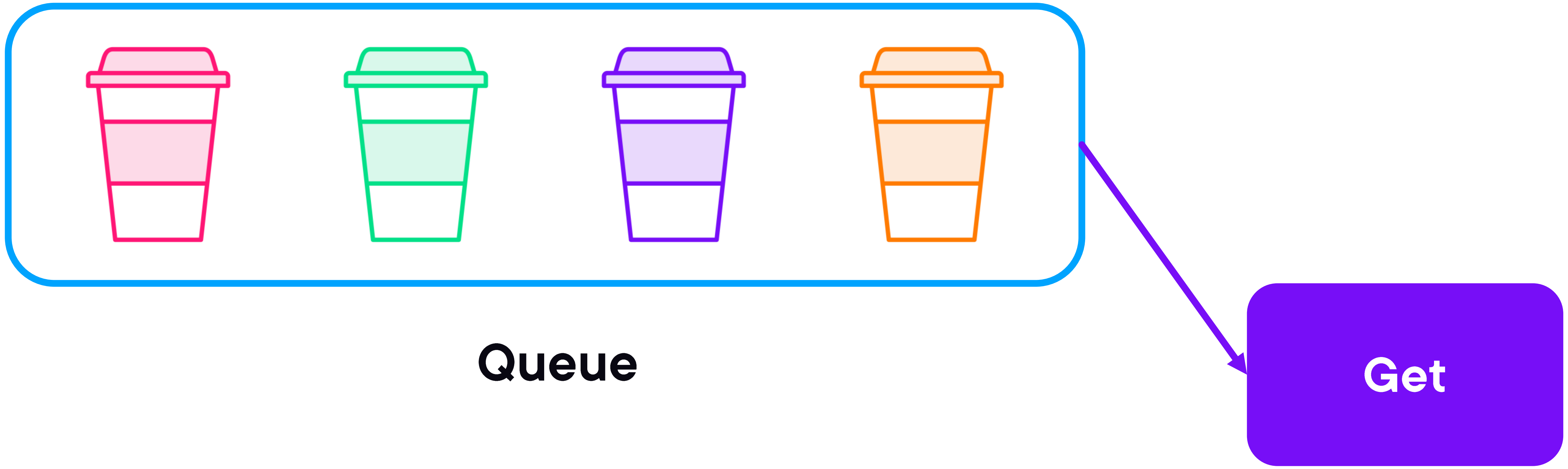
Python Queue Implementations

Queues	VS	Deques
From queue module		From collections module
Simple queue		Double-ended queue
First-In-First-Out		FIFO, Last-In-First-Out
Specialized for multithreading		Multithreading support
Few operations		More operations

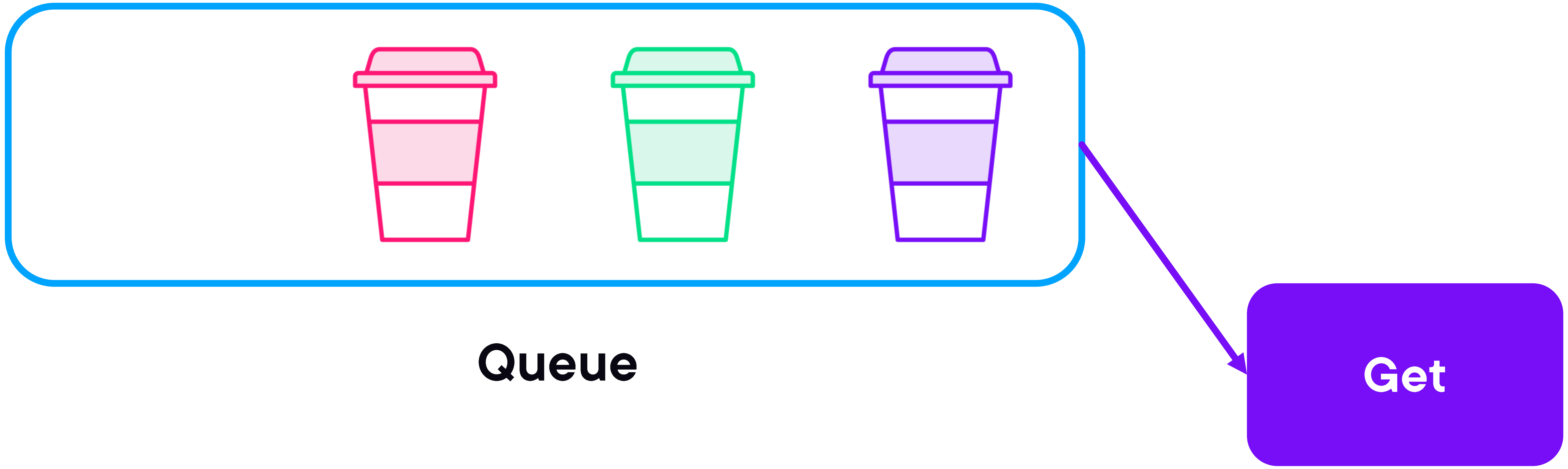
Adding Items to a Queue



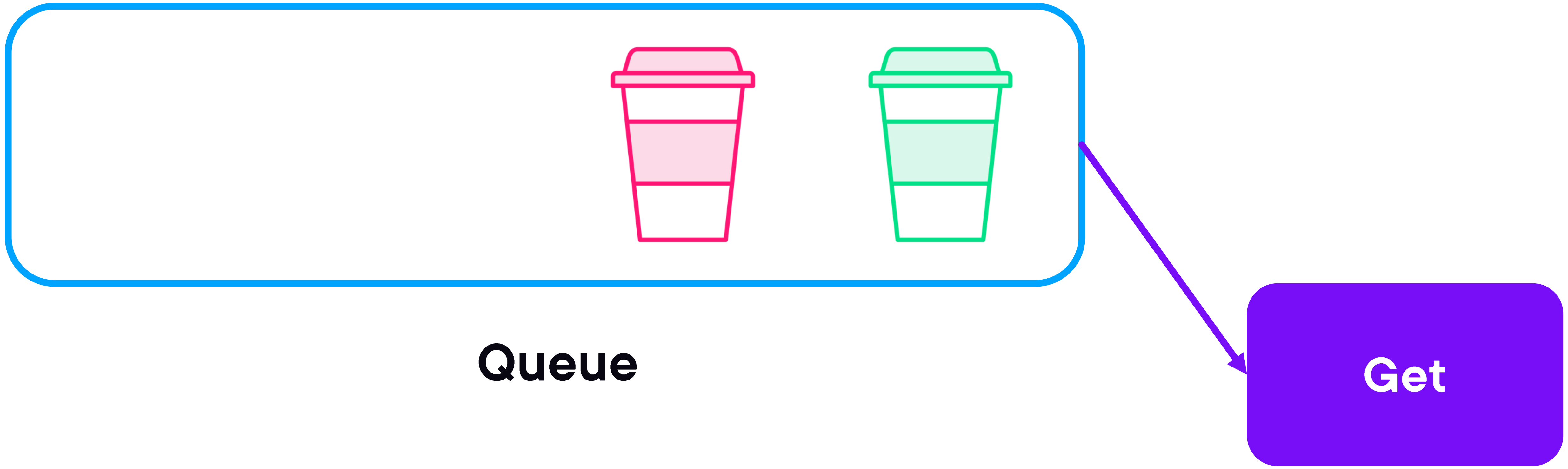
Removing Items from a Queue



Removing Items from a Queue



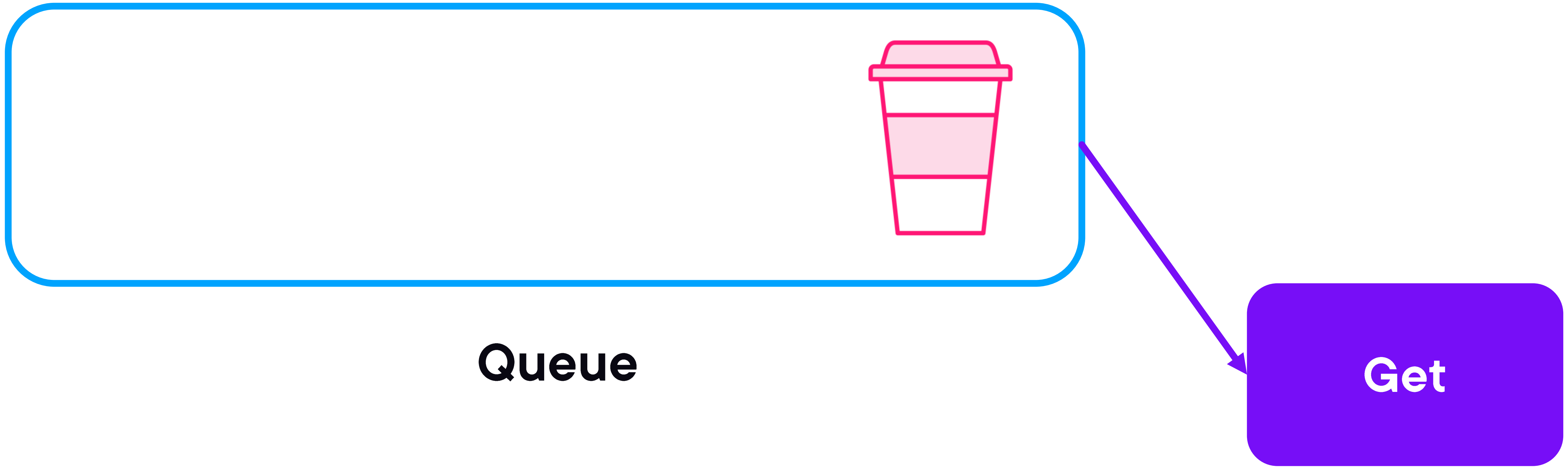
Removing Items from a Queue



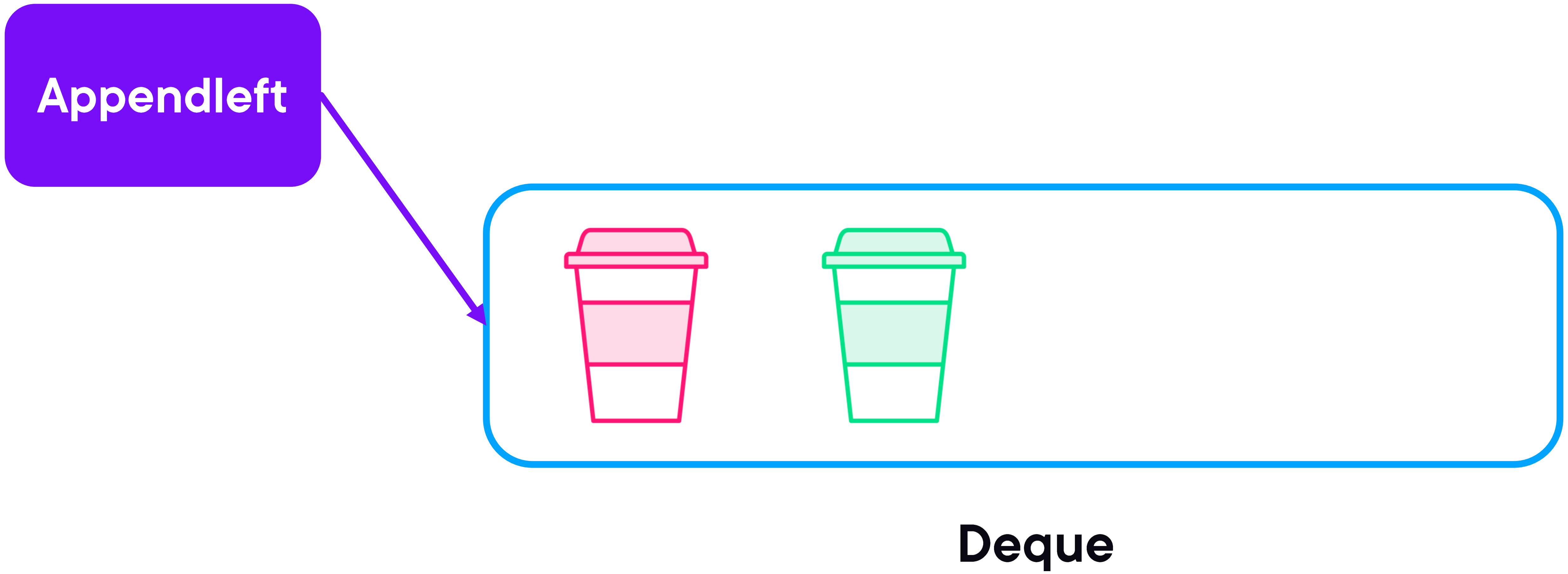
Queue

Get

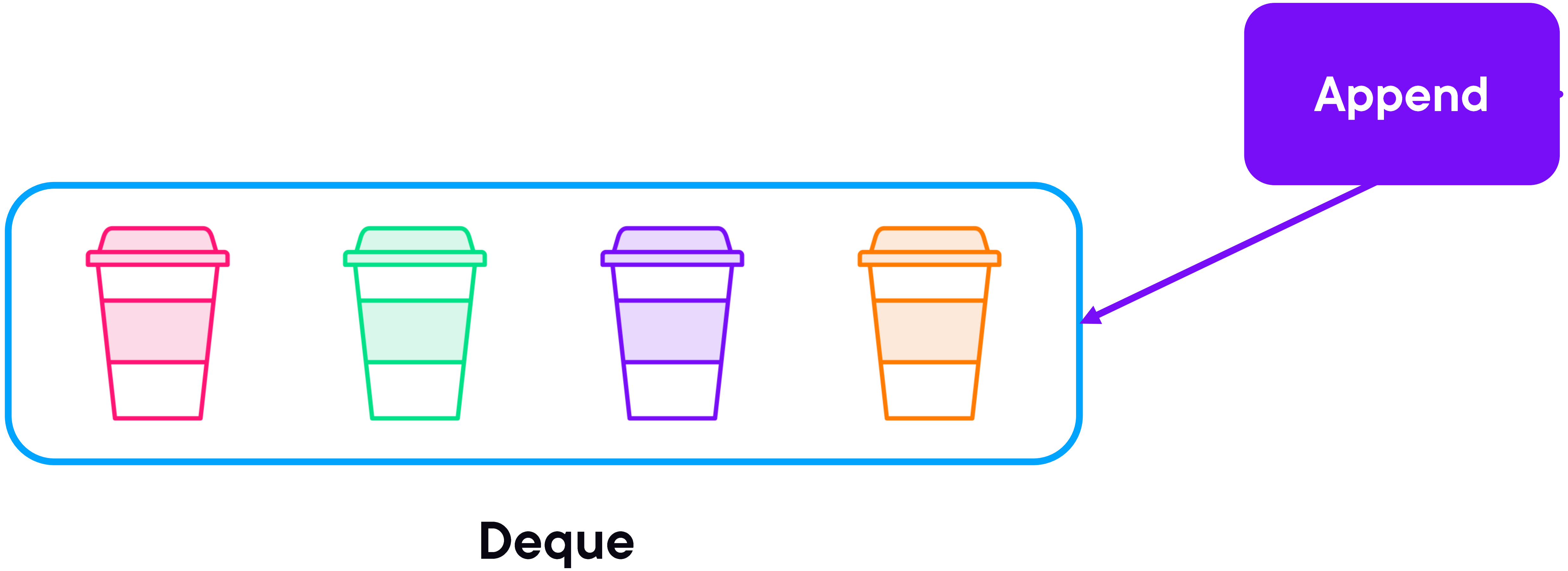
Removing Items from a Queue



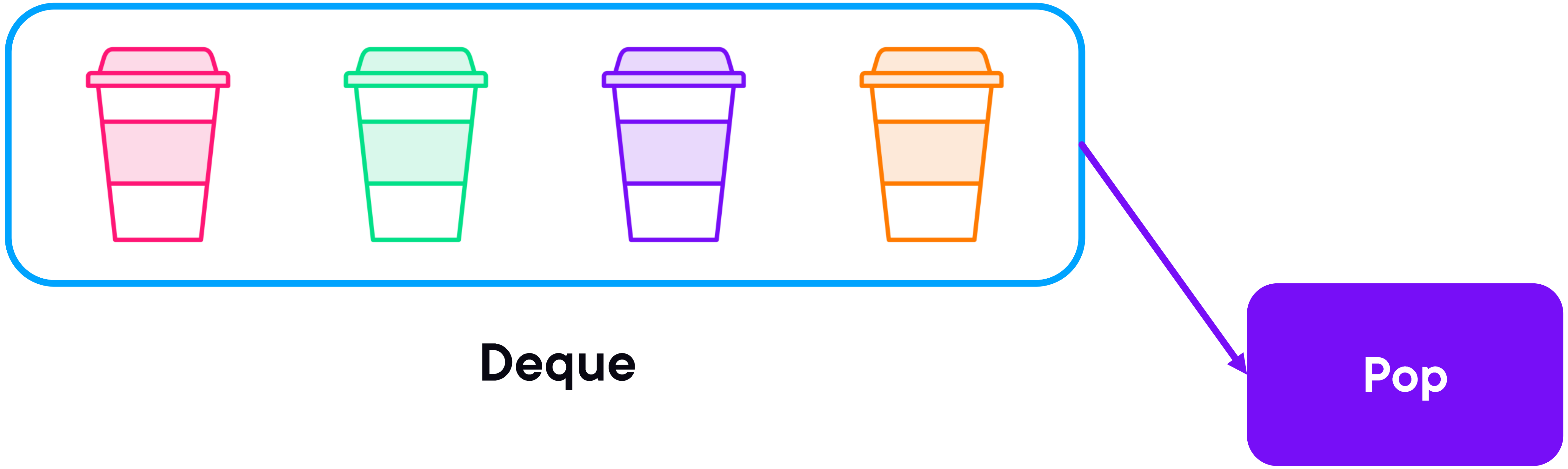
Adding Items to a Deque



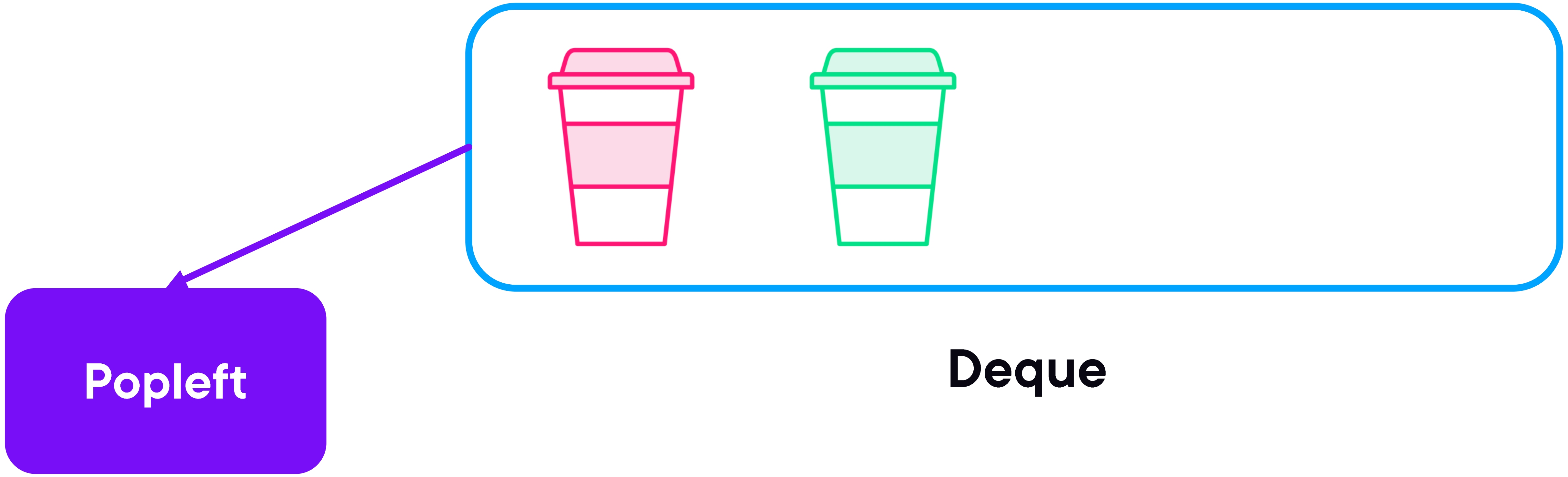
Adding Items to a Deque



Removing Items from a Deque



Removing Items from a Deque



Performance

Deques

Slow access by index – O(n)

Fast append and pop at the end

Fast append and pop at the start

VS

Lists

Fast access by index – O(1)

Fast append and pop at the end

Slow append and pop at the start

Demo

Order processing

Store orders

- In a list
- In a deque

Process orders

- From one side
- From the other side

Compare performance

Dictionaries

Collections of key-value pairs

Mutable

Access by keys

Keys must be unique

Keys must be hashable

Very popular

Dictionaries Performance

Very fast – O(1):

- Getting
- Setting
- Deleting

Slow – O(n):

- Worst cases

Performance

Dictionaries	VS	Lists
Key-value pairs		Individual items
Restrictions on keys		No restrictions
Fast access by key – O(1)		Fast access by index – O(1)
Fast search		Slow search

Demo

Store order ids and their details

- In a list
- In a dictionary

Compare performance

- Adding items
- Searching items

Named Tuples

Tuples with named fields

Better readability

Immutable

Function arguments

Data Classes

Store data with a class, without boilerplate

Decorate class with @dataclass

Optional immutability

Type hints

Data Classes vs Named Tuples

Data Classes	VS	Named Tuples
Mutable by default		Always immutable
Supports class features		Tuples, not classes
Fast access		Memory efficient

Demo

Store order ids

- In a named tuple
- In a data class
- In a dictionary

Compare performance

- Creation
- Accessing order ids

Module Summary

Which data structure is faster?

Comparing lists and arrays

Comparing sets and tuples

Comparing queues and deques

Using dictionaries

Comparing data classes, dictionaries and named tuples

Up Next:

Optimizing Python Code