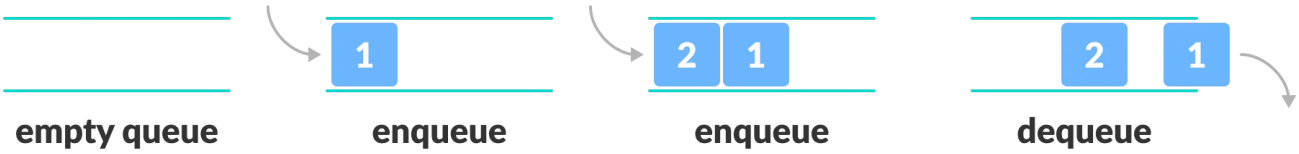
Author: Kamrul Islam Shahin 🇧🇩

🚀 Queue Data Structure (15 days study plan)

- Linear data structure
- Follows the principle of **First In First Out FIFO)**
- First element inserted inside the queue is removed first.



🚀 Basic Operations of Queue

- **Enqueue** : Add an element to the end of the queue
- **Dequeue** : Remove an element from the front of the queue
- **IsEmpty** : Check if the queue is empty
- **IsFull** : Check if the queue is full
- **Peek** : Get the value of the front of the queue without removing it

🚀 Implementing Queue using list

* Method 1

```
queue = []

# Adding elements to the queue
queue.append('a')
queue.append('b')
queue.append('c')

print("Initial queue")
print(queue)

# Removing elements from the queue
print("\nElements dequeued from queue")
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))

print("\nQueue after removing elements")
print(queue)

# it will raise and IndexError as the queue is now empty
print(queue.pop(0))
```

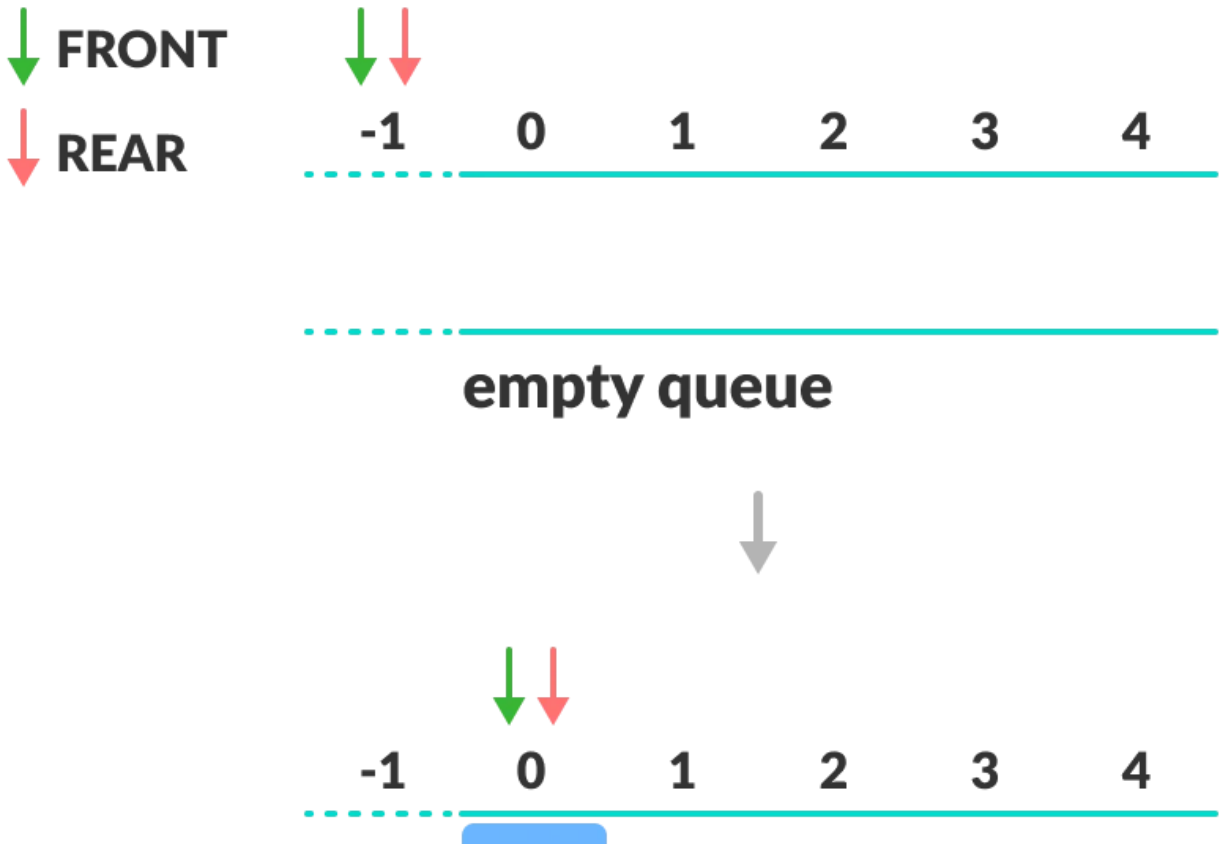
Output:

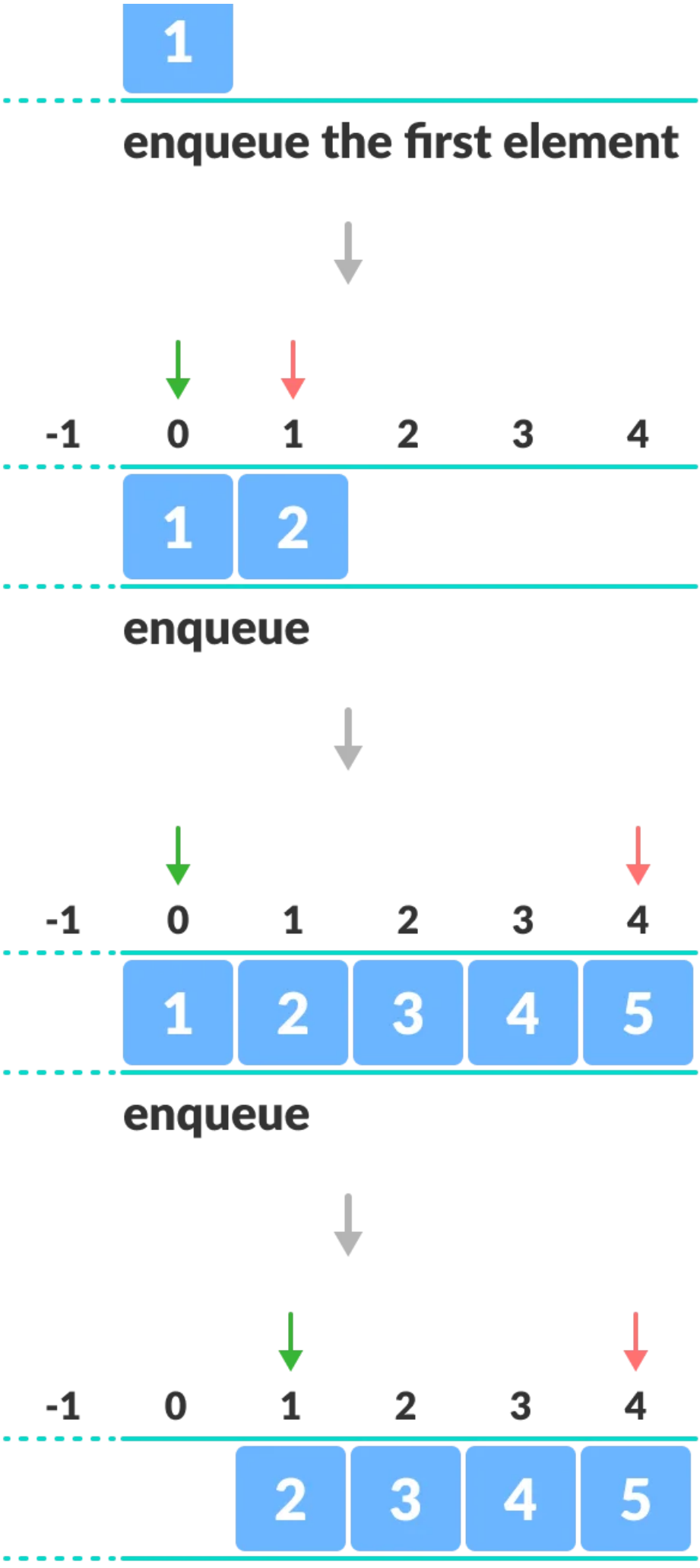
```
Initial queue
['a', 'b', 'c']

Elements dequeued from queue
a
b
c

Queue after removing elements
[]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipykernel_27562/3751923502.py in <module>
     21 # will raise and IndexError
     22 # as the queue is now empty
---> 23 print(queue.pop(0))

IndexError: pop from empty list
```

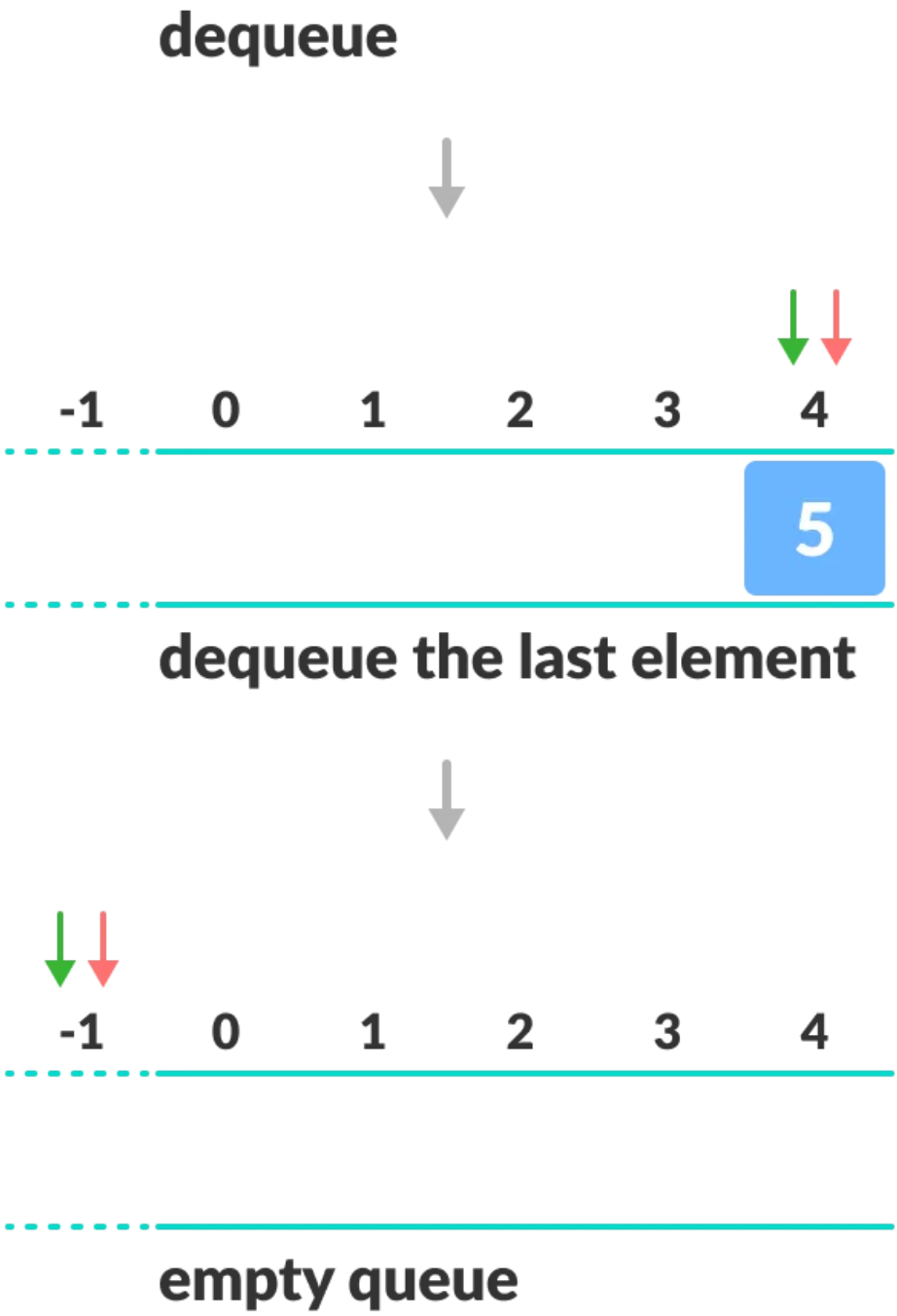
Drawbacks:

- It can run into speed issues as list are quite slow.
- The last command will raise an IndexError after calling .pop() on an empty queue.

* Method 2




1
enqueue the first element
-1 0 1 2 3 4
1 2
enqueue
-1 0 1 2 3 4
1 2 3 4 5
enqueue
-1 0 1 2 3 4
2 3 4 5



{:height="36px" width="36px"}

```
# Custom queue implementation in Python
class Queue:

    # Initialize queue
    def __init__(self, size=1000):
        self.q = [None] * size      # list to store queue elements
        self.capacity = size        # maximum capacity of the queue
        self.front = 0              # front points to the front element in the
queue
        self.rear = -1              # rear points to the last element in the queue
        self.count = 0              # current size of the queue

    # Function to dequeue the front element
```

```
    def dequeue(self):
        # check for queue underflow
        if self.isEmpty():
            print('Queue Underflow!! Terminating process.')
            exit(-1)
        x = self.q[self.front]
        print('Removing element…', x)
        self.front = (self.front + 1) % self.capacity
        self.count = self.count - 1
        return x

    # Function to add an element to the queue
    def enqueue(self, value):
        # check for queue overflow
        if self.isFull():
            print('Overflow!! Terminating process.')
            exit(-1)
        print('Inserting element…', value)
        self.rear = (self.rear + 1) % self.capacity
        self.q[self.rear] = value
        self.count = self.count + 1

    # Function to return the front element of the queue
    def peek(self):
        if self.isEmpty():
            print('Queue UnderFlow!! Terminating process.')
            exit(-1)
        return self.q[self.front]

    # Function to return the size of the queue
    def size(self):
        return self.count

    # Function to check if the queue is empty or not
    def isEmpty(self):
        return self.size() == 0

    # Function to check if the queue is full or not
    def isFull(self):
        return self.size() == self.capacity


if __name__ == '__main__':

    # create a queue of capacity 5
    q = Queue(5)

    q.enqueue(1)
    q.enqueue(2)
    q.enqueue(3)

    print('The queue size is', q.size())
    print('The front element is', q.peek())
    q.dequeue()
```

```
    print('The front element is', q.peek())

    q.dequeue()
    q.dequeue()

    if q.isEmpty():
        print('The queue is empty')
    else:
        print('The queue is not empty')
```

Output:

```
[None, None, None]
Enqueue: 1
[1, None, None]
Enqueue: 2
[1, 2, None]
Enqueue: 3
[1, 2, 3]
Overflow!! Terminating process.
[1, 2, 3]
Queue size: 3
Front element: 1
Dequeue: 1
[None, 2, 3]
Front element: 2
Dequeue: 2
[None, None, 3]
Dequeue: 3
[None, None, None]
The queue is empty
```