**Data Structure**

**&**

**Algorithm**

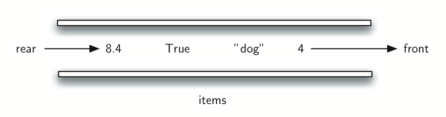
**Class 8**

**Lab 13**

|  |
| --- |
| Lab Objectives:QueueAbstract Data TypeHot Potato |

# What Is a Queue?

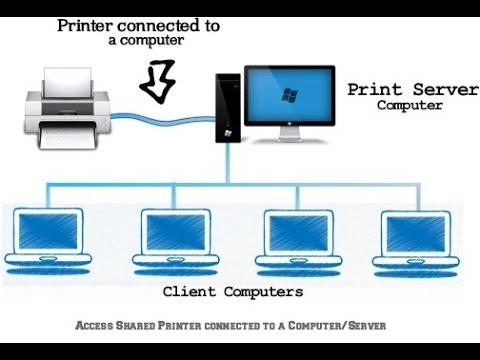
## A queue is an ordered collection of items where the addition of new items happens at one end, called the “rear,” and the removal of existing items occurs at the other end, commonly called the “front.” As an element enters the queue it starts at the rear and makes its way toward the front, waiting until that time when it is the next element to be removed.



## The most recently added item in the queue must wait at the end of the collection. The item that has been in the collection the longest is at the front.

## IMG_256This ordering principle is sometimes called FIFO, first-in first-out. It is also known as “first-come first-served.”

Your computer laboratory has 30 computers networked with a single printer. When students want to print, their print tasks “get in line” with all the other printing tasks that are waiting. The first task in is the next to be completed. If you are last in line, you must wait for all the other tasks to print ahead of you.



# The Queue Abstract Data Type

## The queue abstract data type is defined by the following structure and operations.Queues maintain a FIFO ordering property. The queue operations are given below.

## *Queue()* creates a new queue that is empty. It needs no parameters and returns an empty queue.

## *enqueue(item)* adds a new item to the rear of the queue. It needs the item and returns nothing.

## *dequeue(*) removes the front item from the queue. It needs no parameters and returns the item. The queue is modified.

## *isEmpty()* tests to see whether the queue is empty. It needs no parameters and returns a boolean value.

## *size()* returns the number of items in the queue. It needs no parameters and returns an integer.



## As an example, if we assume that q is a queue that has been created and is currently empty, then [Table 1](https://interactivepython.org/runestone/static/pythonds/BasicDS/TheQueueAbstractDataType.html" \l "tbl-queueoperations) shows the results of a sequence of queue operations. The queue contents are shown such that the front is on the right. 4 was the first item enqueued so it is the first item returned by dequeue.



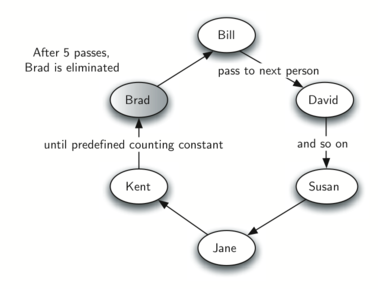
# Hot Potato

One of the typical applications for showing a queue in action is to simulate a real situation that requires data to be managed in a FIFO manner. To begin, let’s consider the children’s game Hot Potato.

## children line up in a circle and pass an item from neighbor to neighbor as fast as they can. At a certain point in the game, the action is stopped and the child who has the item (the potato) is removed from the circle. Play continues until only one child is left.

## IMG_256

## We will implement a general simulation of Hot Potato. Our program will input a list of names and a constant, call it “num,” to be used for counting. It will return the name of the last person remaining after repetitive counting by num. What happens at that point is up to you.



## To simulate the circle, we will use a queue. Assume that the child holding the potato will be at the front of the queue. Upon passing the potato, the simulation will simply dequeue and then immediately enqueue that child, putting her at the end of the line.

## She will then wait until all the others have been at the front before it will be her turn again. After num dequeue/enqueue operations, the child at the front will be removed permanently and another cycle will begin. This process will continue until only one name remains (the size of the queue is 1).

## IMG_256