4.13. Simulation: 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. In this game (see Figure 2) 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.

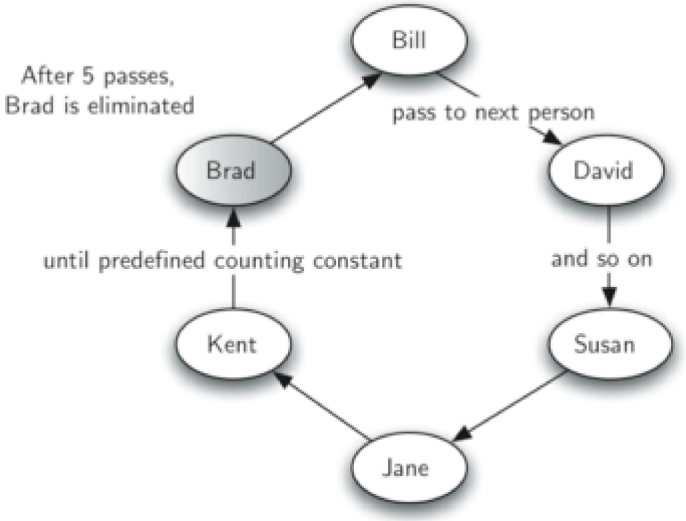

Figure 2: A Six Person Game of Hot Potato

This game is a modern-day equivalent of the famous Josephus problem. Based on a legend about the famous first-century historian Flavius Josephus, the story is told that in the Jewish revolt against Rome, Josephus and 39 of his comrades held out against the Romans in a cave. With defeat imminent, they decided that they would rather die than be slaves to the Romans. They arranged themselves in a circle. One man was designated as number one, and proceeding clockwise they killed every seventh man. Josephus, according to the legend, was among other things an accomplished mathematician. He instantly figured out where he ought to sit in order to be the last to go. When the time came, instead of killing himself, he joined the Roman side. You can find many different versions of this story. Some count every third man and some allow the last man to escape on a horse. In any case, the idea is the same.

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 (see Figure 3). 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).

(ImplementingaQueueinPython.html) (SimulationPrintin

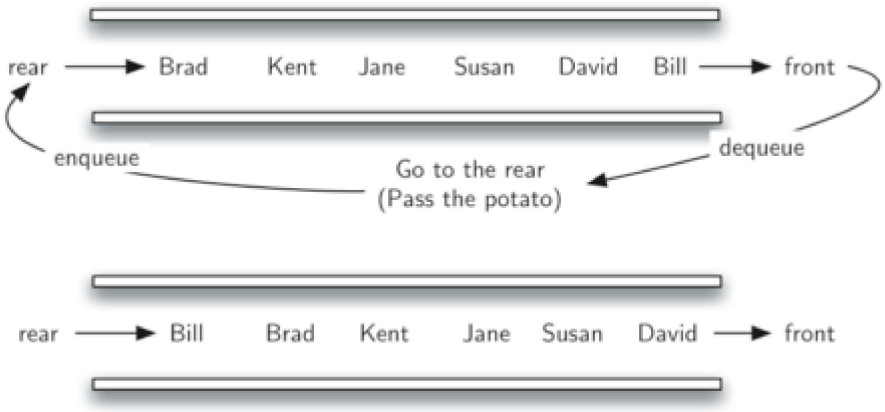

Figure 3: A Queue Implementation of Hot Potato

The program is shown in ActiveCode 1. A call to the `hotPotato` function using 7 as the counting constant returns `Susan` .

Run Load History

```
from pythonds.basic import Queue

def hotPotato(namelist, num):
    simqueue = Queue()
    for name in namelist:
        simqueue.enqueue(name)

    while simqueue.size() > 1:
        for i in range(num):
            simqueue.enqueue(simqueue.dequeue())

        simqueue.dequeue()

    return simqueue.dequeue()
```

Activity: 4.13.1 Hot Potato Simulation (qujosephussim)

Note that in this example the value of the counting constant is greater than the number of names in the list. This is not a problem since the queue acts like a circle and counting continues back at the beginning until the value is reached. Also, notice that the list is loaded into the queue such that the first name on the list will be at the front of the queue. `Bill` in this case is the first item in the list and therefore moves to the front of the queue. A variation of this implementation, described in the exercises, allows for a random counter.

You have attempted 1 of 2 activities on this page

(ImplementingaQueueinPython.html) (SimulationPrintin

user not logged in

(ImplementingaQueueinPython.html) (SimulationPrintin