Basic Data Structures

Queues

Updated version of Miller's slides

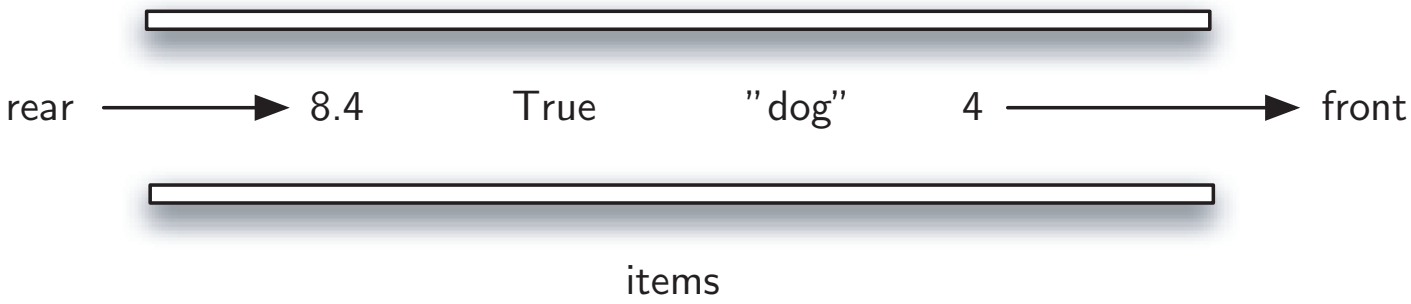
Outline

1. Queues
   - What Is a Queue?
   - The Queue Abstract Data Type
   - Implementing a Queue in Python
   - Simulation: Printing Tasks
   -
2.
   -
   -
   -
   -

Outline

1. Queues
   - What Is a Queue?
   - The Queue Abstract Data Type
   - Implementing a Queue in Python
   - Simulation: Printing Tasks

A Queue of Python Data Objects



Outline

1 Queues
- What Is a Queue?
- The Queue Abstract Data Type
- Implementing a Queue in Python
- Simulation: Printing Tasks

- `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.

Outline

1 Queues
- What Is a Queue?
- The Queue Abstract Data Type
- Implementing a Queue in Python
- Simulation: Printing Tasks

Queue Implementation in Python

```
1  class Queue:
2      def __init__(self):
3          self.items = []
4
5      def isEmpty(self):
6          return self.items == []
7
8      def enqueue(self, item):
9          self.items.insert(0,item)
10
11     def dequeue(self):
12         return self.items.pop()
13
14     def size(self):
15         return len(self.items)
```

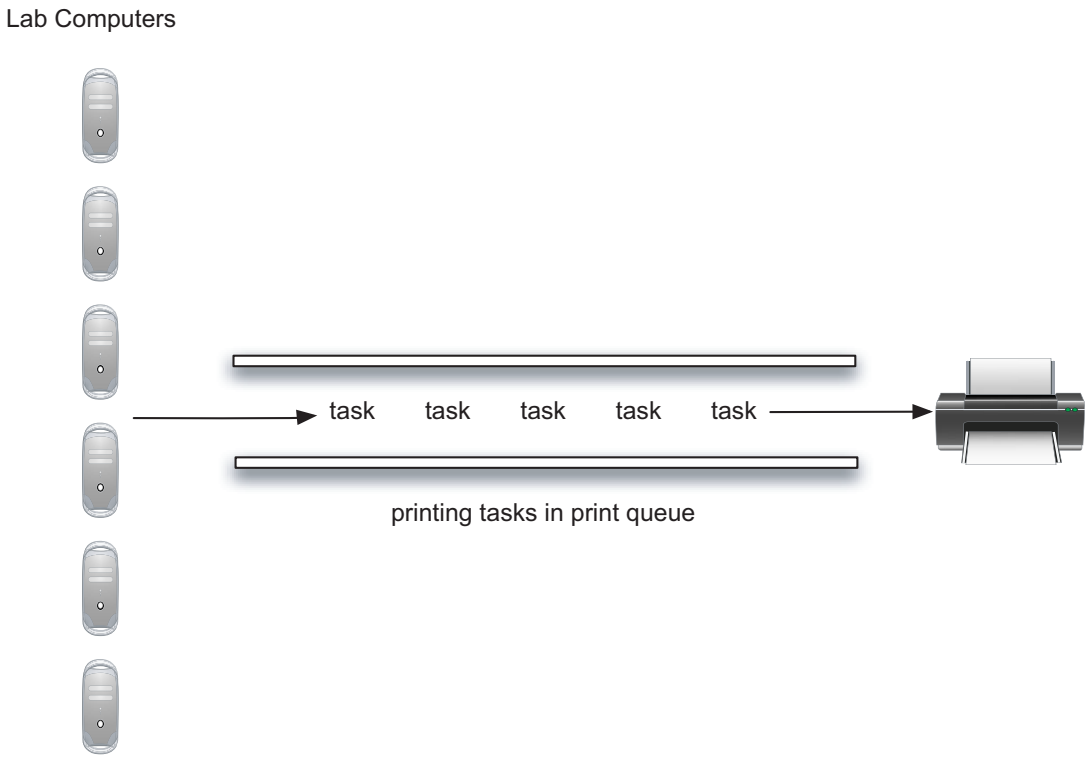
Queue Applications

- Many servers use a queue to keep track of client requests.
- Whenever you see reference to something being called "event-driven" it is probably using a queue.
- GUIs are often event-driven, that is they accumulate the mouse clicks and keyboard presses, etc.
- That's why sometimes when a computer hangs on some operation and you click the mouse a bunch of times when it un-hangs a ton of actions might take place, like context menus popping up all over the screen, etc.
- That's because the GUI can finally flush the backlogged (المتأخرات المتراكمة) queue.

Outline

1 Queues
- What Is a Queue?
- The Queue Abstract Data Type
- Implementing a Queue in Python
- Simulation: Printing Tasks

Computer Science Laboratory Printing Queue



Simulation: Printing Tasks (1)

- Students send printing tasks to the shared printer
- The tasks are placed in a queue to be processed in a first-come first-served manner.
- On any average day about 10 students are working in the lab at any given hour.
- These students typically print twice during that time, and the length of these tasks ranges from 1 to 20 pages.
- The printer in the lab is older, capable of processing 10 pages per minute of draft quality.
- The printer could be switched to give better quality, but then it would produce only five pages per minute. The slower printing speed could make students wait too long.

Simulation: Printing Tasks (2)

- We could decide by building a simulation that models the laboratory.
- As students submit printing tasks, we will add them to a waiting list, **a queue of print tasks attached to the printer.**
- When the printer completes a task, it will look at the queue to see if there are any remaining tasks to process.
- Of interest for us is the average amount of time students will wait for their papers to be printed (**= the average amount of time a task waits in the queue**).
- To model this situation we need to use some **probabilities**.
- If each length from 1 to 20 is equally likely, the actual length for a print task can be simulated by using a **random number** between 1 and 20 inclusive.
- This means that there is equal chance of any length from 1 to 20 appearing.

Simulation: Printing Tasks (3)

- If there are 10 students in the lab and each prints twice, then there are 20 print tasks per hour on average.
- Twenty tasks per hour means that on average there will be one task every 180 seconds:
  - 20 tasks/1 hour = 20 tasks/3600 sec =1 task/180 sec (0.5%)
- For every **second,** we can simulate the chance that a print task occurs by generating a random number between 1 and 180 inclusive.
- If the number is 180, we say a task has been created.

Main Simulation Steps

1. Create a queue of print tasks. Each task will be given a timestamp upon its arrival. The queue is empty to start.
2. For each second (`currentSecond`):
    - Does a new print task get created? If so, add it to the queue with the `currentSecond` as the timestamp.
    - If the printer is not busy and if a task is waiting,
        - Remove the next task from the print queue and assign it to the printer.
        - Subtract the timestamp from the `currentSecond` to compute the waiting time for that task.
        - Append the waiting time for that task to a list for later processing.
        - Based on the number of pages in the print task, figure out how much time will be required(remaining printing time).
    - The printer now does one second of printing if necessary. It also subtracts one second from the time required for that task.
    - If the task has been completed, in other words the time required has reached zero, the printer is no longer busy.
3. After the simulation is complete, compute the average waiting time from the list of waiting times generated.

Printer Queue Simulation–The Printer Class I

```
1  class Printer:
2      def __init__(self, pages):
3          self.pagerate = pages
4          self.currentTask = None
5          self.timeRemaining = 0
6
7      def tick(self):
8          if self.currentTask != None:
9              self.timeRemaining = self.timeRemaining - 1
10             if self.timeRemaining == 0:
11                 self.currentTask = None
12
13
14
15
```

Printer Queue Simulation–The Printer Class II

```
16      def busy(self):
17          if self.currentTask != None:
18              return True
19          else:
20              return False
21
22      def startNext(self,newtask):
23          self.currentTask = newtask
24          self.timeRemaining = newtask.getPages() \
25                               * 60/self.pagerate
```

Printer Queue Simulation–The Task Class

```
1  import random
2  class Task:
3      def __init__(self,time):
4          self.timestamp = time
5          self.pages = random.randrange(1,21)
6
7      def getStamp(self):
8          return self.timestamp
9
10     def getPages(self):
11         return self.pages
12
13     def waitTime(self, currenttime):
14         return currenttime - self.timestamp
```

Printer Queue Simulation–The Main Simulation I

```
1  from pythonds.basic.queue import Queue
2  from printer import *
3  from task import *
4
5  import random
6
7  def simulation(numSeconds, pagesPerMinute):
8
9      labprinter = Printer(pagesPerMinute)
10     printQueue = Queue()
11     waitingtimes = []
12
13     for currentSecond in range(numSeconds):
14
15
```

Printer Queue Simulation–The Main Simulation II

```
16          if random.randrange(1,181) == 180:
17              task = Task(currentSecond)
18              printQueue.enqueue(task)
19
20          if (not labprinter.busy()) and \
21                      (not printQueue.isEmpty()):
22              nexttask = printQueue.dequeue()
23              waitingtimes.append( \
24                  nexttask.waitTime(currentSecond))
25              labprinter.startNext(nexttask)
26
27          labprinter.tick()
28
29      averageWait=sum(waitingtimes)/len(waitingtimes)
30      print ("Average Wait " , averageWait ," secs" ,printQueue.size() , " tasks remaining.")
31
```

Simulation Result at 5 ppm

```
>>>for i in range(10):
      simulation(3600,5)

Average Wait 165.38 secs 2 tasks remaining.
Average Wait  95.07 secs 1 tasks remaining.
Average Wait  65.05 secs 2 tasks remaining.
Average Wait  99.74 secs 1 tasks remaining.
Average Wait  17.27 secs 0 tasks remaining.
Average Wait 239.61 secs 5 tasks remaining.
Average Wait  75.11 secs 1 tasks remaining.
Average Wait  48.33 secs 0 tasks remaining.
Average Wait  39.31 secs 3 tasks remaining.
Average Wait 376.05 secs 1 tasks remaining.
```

- After running our 10 trials we can see that there is a large variation in the average wait time with a minimum average of 17.27 seconds and a maximum of 376.05 seconds (about 6 minutes).
- You may also notice that in only two of the cases were all the tasks completed. In 8 out of 10 runs, there were print tasks still waiting in the queue at the end of the hour.

Simulation Result at 10 ppm

```
>>>for i in range(10):
      simulation(3600,10)

Average Wait   1.29 secs 0 tasks remaining.
Average Wait   7.00 secs 0 tasks remaining.
Average Wait  28.96 secs 1 tasks remaining.
Average Wait  13.55 secs 0 tasks remaining.
Average Wait  12.67 secs 0 tasks remaining.
Average Wait   6.46 secs 0 tasks remaining.
Average Wait  22.33 secs 0 tasks remaining.
Average Wait  12.39 secs 0 tasks remaining.
Average Wait   7.27 secs 0 tasks remaining.
Average Wait  18.17 secs 0 tasks remaining.
```

- With a faster printing rate (10 ppm), the low value was 1.29 second with a high of only 28.96.
- At low ppm, students cannot afford to wait that long for their papers, especially when they need to be getting on to their next class. A six-minute wait would simply be too long.

- What if the average number of students increases by 20?
- What if it is Saturday and students are not needing to get to class? Can they afford to wait?
-