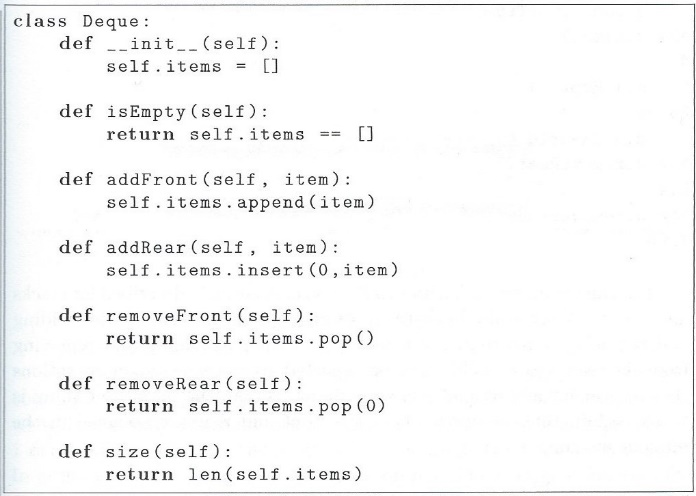
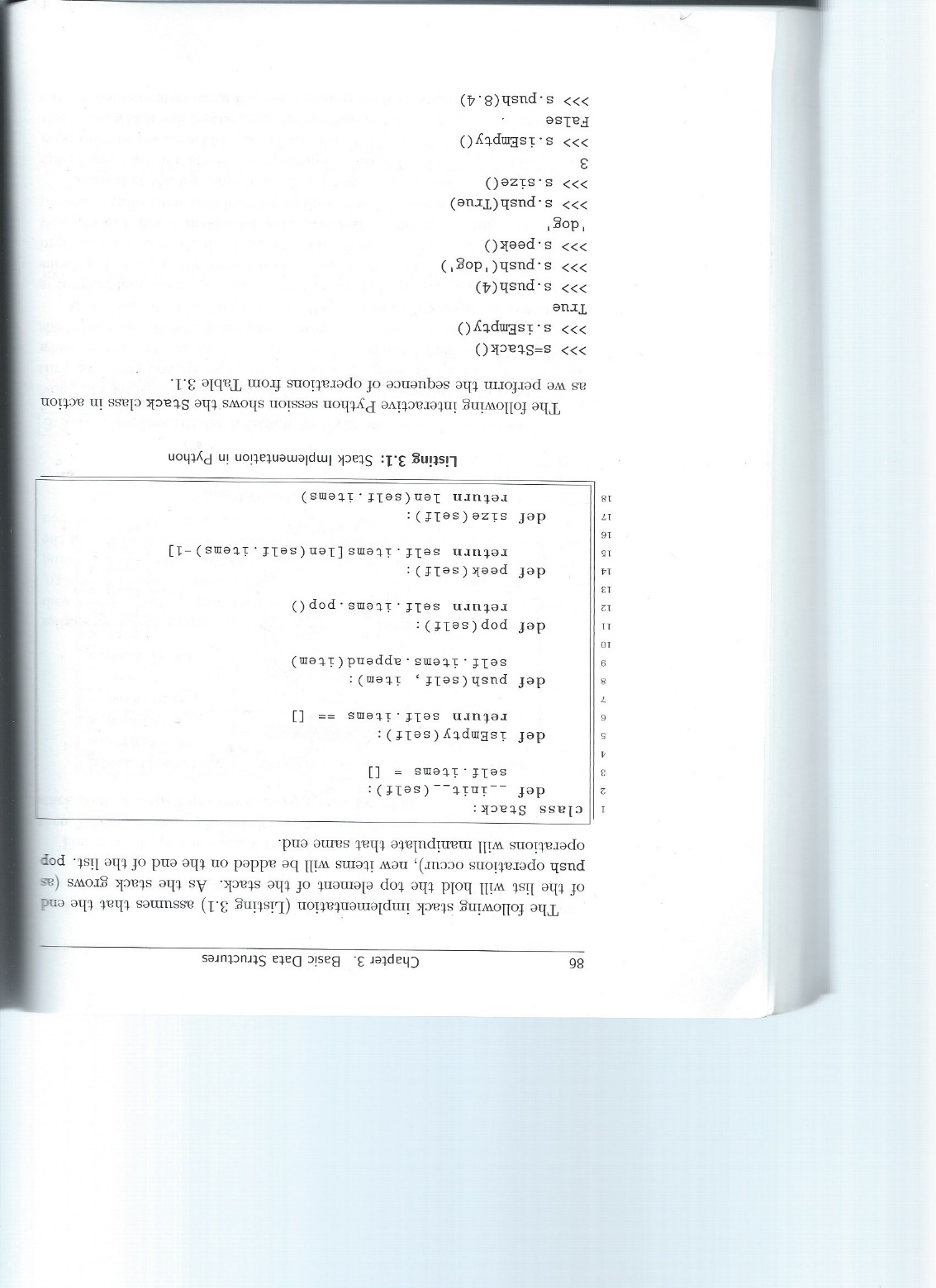
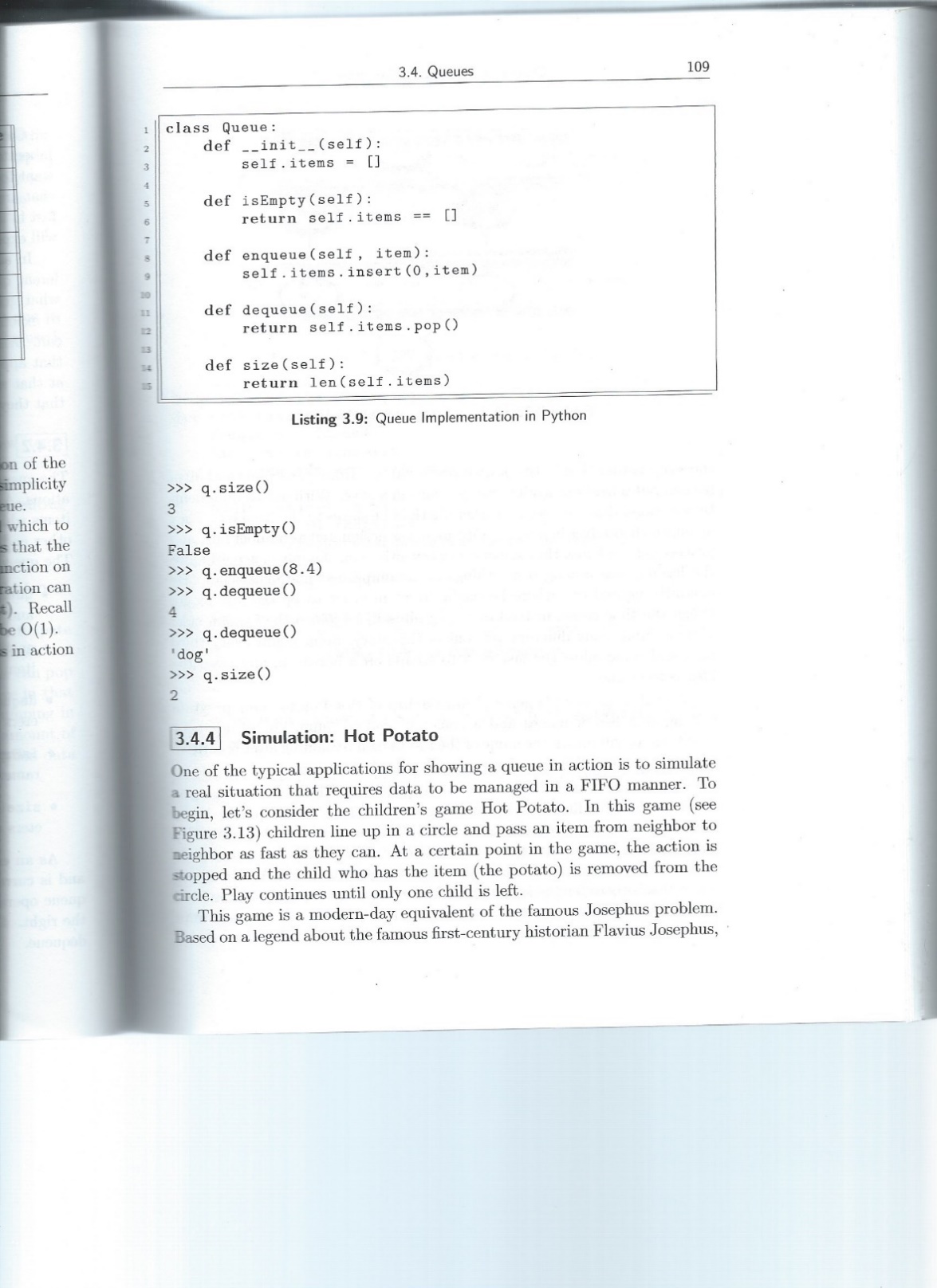
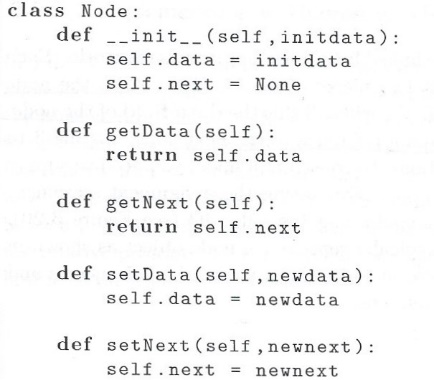
|  |  |  |  |
| --- | --- | --- | --- |
| Type | Method | | |
| All types | def **\_\_init\_\_**(self):  self.head = None | def isEmpty (self):  return self.head == None | |
| Ordered  Unordered  Doubly  Circular | def **length** (self):  current = self.head  count = 0  while current != None/self.head:  count += 1  current = current.getNext()  return count | | \*Doubly: in Node class add self.previous = None to \_\_init\_\_(), create getPrevious(), setPrevious() methods |
| Unordered  Doubly  Ordered  Circular | def **search** (self,item):  current = self.head  found = False  stop = False  while current != None/self.head and not found and not stop:  if current.getData() == item:  found = True  elif current.getData()>item:  stop = true  else:  current = current.getNext()  return found | | |
| Left:  Ordered  Unordered  Doubly  Right:  Circular | def **remove** (self,item):  current = self.head  previous = None  found = False  while not found:  if current.getData() == item:  found = True  else:  previous = current  current = current.getNext()  if previous == None:  self.head = current.getNext()  current.setPrevious(None)  else:  previous.setNext(current.getNext()  current.getNext().setPrevious(previous | | |
| Left:  Unordered  Circular  Doubly  Right:  Ordered  Doubly | def addLast (self,item):  temp = Node(item)  if self.head == None:  self.head = temp  temp.setNext(self.head)  else:  current = self.head  previous = self.head  while current.getNext() != None:  previous = current  current = current.getNext()  temp.setNext(current/self.head)  previous.setNext(temp)  temp.setPrevious(current)  current.setNext(temp) | | |





|  |  |
| --- | --- |
| def unorderedListSum(u):  if self.head.getNext() == None:  return self.head.getData()  else:  item = self.head.getData()  u.remove(item)  return item + unorderedListSum(u) | def toStr(n,base):  convertString = “0123456789ABCDEF”  if n < base:  return convertString[n]  else:  toStr(n//base,base) + convertString[n%base] |

|  |  |  |
| --- | --- | --- |
| **Infix Expression** | **Prefix Expression** | **Postfix Ecpression** |
| A + B \* C + D | + + A \* B C D | A B C \* + D + |
| (A + B) \* (C + D) | \* + A B + C D | A B + C D +\* + |
| A \* B + C \* D | + \* A B \* C D | A B \* C D \* + |
| (A + B) \* C – (D – E) \* (F + G) | - \* + A B C \* - D E + F G | A B + C \* D E – F G + \* - |

