LECTURE 14

LISTS AS STACKS AND QUEUES

MCS 260 Fall 2020

Emily Dumas

REMINDERS

- Work on:
  - Worksheet 5
  - Quiz 5
- Now posted
  - Project 2 description (read it!)

More on this example: lettervalues.py, vals.txt

```
"""Compute the "value" of a word, if each letter is
worth a different number of points specified in a
file.
MCS 260 Fall 2020 Lecture 13 - Emily Dumas
"""

import sys

if len(sys.argv) != 3:
    print("Usage:",sys.argv[0],"valuefile word")
    print("Read values for each alphabet letter from `valuefile`")
    print("in the format letter,value, e.g.:")
    print("a,1\nb,3\nc,3\nd,2")
    print("Then, compute the value of `word` and print it.")
    sys.exit()  # exits the program immediately

fin = open(sys.argv[1],"r")
vals = dict()
for line in fin:
    ltr,valstr = line.split(",")
    vals[ltr] = int(valstr)
fin.close()
```

TWO DATA STRUCTURES

- **Stack** - LIFO (last in, first out) storage of items. Like a physical stack. Operations:
  - push - add element to the "top" of the stack
  - pop - remove and return element on the "top" of the stack
- **Queue** - FIFO (first in, first out) storage of items. Like a line or waiting list. Operations:
  - Enqueue - add an element to the back of the line
  - Dequeue - remove and return the element at the front of the line

STACK

Common applications:

- **Undo** a sequence of actions.
- **Parsing**: Which block are we in?
- **Function calls**: Which function are we in?

Can use a Python list with:

- push becomes `list.append`
- pop becomes `list.pop`

This implementation is efficient.

Stack example: winter.py

```
"""Simulate getting ready to go out in winter"""

print("Enter items worn, in order put on:")
L = []
while True:
    s = input().strip()
    if s == "":
        break
    L.append(s)

print("Ok. Press Enter when ready to remove winter gear.")
input()
while len(L)>0:
    s = L.pop()
    print("Remove",s,"and press Enter when ready.")
    input()
```

QUEUE

Common applications:

- **Work** to be done / data to be processed.
- **Buffer** for communication method.

Can use a Python list with:

- enqueue becomes `list.append(item)`
- dequeue becomes `list.pop(0)`

Using a list as a queue is NOT efficient. Time to remove an item grows with the size of the queue.

More efficient: deque from the collections module

```
import collections
Q = collections.deque()
Q.append("first in")    # enqueue
Q.append(260)
Q.append("last in")
while len(Q)>0:
    x = Q.popleft()     # dequeue
    print(x)
```

Output:

```
first in
260
last in
```

Notice deque implements queue operations:

- enqueue becomes `deque.append(item)`
- dequeue becomes `deque.popleft()`

Efficiency means time to add or remove an item is independent of how many items are present (like stacks).

ANOTHER STACK EXAMPLE

Checking parenthesis matching (example of parsing)

This expression is ok:

- `((2+3) - (4*5))`

These are not:

- `((5*7))) - ((2)`
- `((2+3)-5))`

Goal: Decide if ok, give useful error if not.

parens.py

```
"""Check arithmetic expression for balanced parentheses"""

print("Enter an arithmetic expression in parentheses:")
s = input().strip()

paren_stack = []
for i,c in enumerate(s):
    if c == "(":
        paren_stack.append(i)
    elif c == ")":
        if len(paren_stack) == 0:
            # Too many right parentheses
            print("ERROR: Extra right parenthesis")
            print(s)
            print(i*" " + "^")
            break
        paren_stack.pop()

if len(paren_stack) > 0:
    # Unclosed left parenthesis
    i = paren_stack.pop()  # Where was the left paren that's open?
    print("ERROR: Unclosed parenthesis")
    print(s)
```

REFERENCES

- Optional text Brookshear & Brylow discuss stacks and queues in Section 8.1
- Downey does not discusses stacks and queues in general
- *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser discusses stacks and queues in Chapter 6.

REVISION HISTORY

- 2020-09-24 Initial publication