CSc 110, Sprint 2017

Lecture 16: Lists (cont.) and File Input

okay dad. the science fair is tomorrow. let's make up some data.

Toothpaste For Dinner.com
Weather question

• Use a list to solve the weather problem:

How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average.
def main():
    days = int(input("How many days' temperatures? "))

    temps = [0] * days
    # list to store days' temperatures
    sum = 0

    for i in range(0, days):
        # read/store each day's temperature
        temps[i] = int(input("Day " + (i + 1) + "'s high temp: "))
        sum = sum + temps[i]

    average = sum / days

    count = 0
    # see if each day is above average
    for i in range(0, days):
        if (temps[i] > average):
            count = count + 1

    # report results
    print("Average temp = " + str(average))
    print(str(count) + " days above average")
Weather question 2

• Modify the weather program to print the following output:

  Type in a temperature or "done" to finish
  Day 1's high temp: 45
  Day 2's high temp: 44
  Day 3's high temp: 39
  Day 4's high temp: 48
  Day 5's high temp: 37
  Day 6's high temp: 46
  Day 7's high temp: 53
  Day 7's high temp: done
  Average temp = 44.6
  4 days were above average.
Problem

• We don't know how many elements the list will have
• We need a way to build a list while processing the input.

• New method:
  append(x) - add an item to the end of a list
def main():
    temps = []
    avg = 0
    day = 1
    temp = input("Day " + str(day) + ", high temperature: ")
    while(temp != "done"):
        avg = avg + int(temp)
        temps.append(int(temp))
        day = day + 1
        temp = input("Day " + str(day) + ", high temperature: ")

    avg = avg / len(temps)
    above = 0
    for number in temps:
        if(number > avg):
            above = above + 1

    print("Average temperature = " + str(round(avg, 1)))
    print(str(above) + " days above average.")
    print()
    print("Temperatures: " + str(temps))
Weather question 3

• Modify the weather program to print the following output:

How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average.

Temperatures: [45, 44, 39, 48, 37, 46, 53]
Two coldest days: 37, 39
Two hottest days: 53, 48
## List functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>append(x)</code></td>
<td>Add an item to the end of the list. Equivalent to <code>a[len(a):] = [x]</code>.</td>
</tr>
<tr>
<td><code>extend(L)</code></td>
<td>Extend the list by appending all the items in the given list. Equivalent to <code>a[len(a):] = L</code></td>
</tr>
<tr>
<td><code>insert(i, x)</code></td>
<td>Inserts an item at a given position. <code>i</code> is the index of the element before which to insert, so <code>a.insert(0, x)</code> inserts at the front of the list.</td>
</tr>
<tr>
<td><code>remove(x)</code></td>
<td>Removes the first item from the list whose value is <code>x</code>. Errs if there is no such item.</td>
</tr>
<tr>
<td><code>pop(i)</code></td>
<td>Removes the item at the given position in the list, and returns it. <code>a.pop()</code> removes and returns the last item in the list.</td>
</tr>
<tr>
<td><code>clear()</code></td>
<td>Remove all items from the list.</td>
</tr>
<tr>
<td><code>index(x)</code></td>
<td>Returns the index in the list of the first item whose value is <code>x</code>. Errs if there is no such item.</td>
</tr>
<tr>
<td><code>count(x)</code></td>
<td>Returns the number of times <code>x</code> appears in the list.</td>
</tr>
<tr>
<td><code>sort()</code></td>
<td>Sort the items of the list</td>
</tr>
<tr>
<td><code>reverse()</code></td>
<td>Reverses the elements of the list</td>
</tr>
<tr>
<td><code>copy()</code></td>
<td>Return a copy of the list.</td>
</tr>
</tbody>
</table>
Weather answer 3

# Reads temperatures from the user, computes average and # days above average.
def main():
    temps = []
    avg = 0
    day = 1
    temp = input("Day " + str(day) + "'s high temperature: ")
    while(temp != "done"):
        avg = avg + int(temp)
        temps.append(int(temp))
        day = day + 1
        temp = input("Day " + str(day) + "'s high temperature: ")

    # counts days above average
    avg = avg / len(temps)
    above = 0
    for number in temps:
        if(number > avg):
            above = above + 1

    print("Average temperature = " + str(round(avg, 1)))
    print(str(above) + " days above average.")
    print()
    print("Temperatures: " + str(temps))
    temps.sort()
    print("Two coldest: " + str(temps[0]) + ", " + str(temps[1]))
    print("Two hottest: " + str(temps[-1]) + ", " + str(temps[-2]))
File Input

- **open(name)** – a built-in function that opens the specified file and returns a file object. The type of name is `str`.

- Example:

  ```python
  f = open("hours.txt")
  ```
File paths

- **absolute path**: specifies a drive or a top "/" folder
  
  C:/Documents/smith/hw6/input/data.csv

- Windows can also use backslashes to separate folders.

- **relative path**: does not specify any top-level folder

  names.dat  
  input/kinglear.txt

- Assumed to be relative to the *current directory*:

  ```python
  file = open("data/readme.txt")
  ```

  If our program is in H:/hw6,
  
  open will look for H:/hw6/data/readme.txt

NOTE: We will put files in the same directory as our Python programs.
File Input

• Now we need a way to access the contents of the file.
  
  \[
  f = \text{open}("hours.txt")
  \]

• \textbf{read()} – a method that reads a file and returns the contents as a string. Requires the "." notation for use.

• Example:
  
  \[
  f.\text{read}()
  \]

```plaintext
>>> f = open("hours.txt")
>>> f.read()
'123 Brett 12.5 8.1 7.6 3.2\n456 Sarina 4.0 11.6 6.5 2.7 12\n789 Nick 8.0 8.0 8.0 8.0 7.5\n'
```
More File methods

- **readline()** – Reads the next line of a file and returns it as a string.

- **readlines()** – Reads the contents of a file and returns it as a list.

- What if there are no more lines?

```python
>>> f = open("hours.txt")
>>> f.readline()
'123 Susan 12.5 8.1 7.6 3.2\n'

>>> f = open("hours.txt")
>>> f.readlines()
["123 Brett 12.5 8.1 7.6 3.2\n", "456 Sarina 4.0 11.6 6.5 2.7 12\n", "789 Nick 8.0 8.0 8.0 8.0 7.5\n"]
```
Process a file one line at a time

• Use `readlines()` to return the contents of the file as a list.

• Loop through the list:

```python
f = open("hours.txt")
hours = f.readlines()  # hours is a list
for i in range(0, len(hours)):
    print(hours[i])
```

Interesting output. Why?

```python
>>> ...
123 Brett 12.5 8.1 7.6 3.2
456 Sarina 4.0 11.6 6.5 2.7 12
789 Nick 8.0 8.0 8.0 8.0 7.5
```
Process a file one line at a time

• Use `readlines()` to return the contents of the file as a list

• `strip()` – a method that removes newlines \n
```python
f = open("hours.txt")
hours = f.readlines
for i in range(0, len(hours)):
    print(hours[i].strip())
```

```bash
>>>
for i in range(0, len(hours)):
    ...
    print(hours[i].strip())  # strip() removes \n
123  Brett  12.5  8.1  7.6  3.2
456  Sarina  4.0  11.6  6.5  2.7  12
789  Nick    8.0  8.0  8.0  8.0  7.5
```
File input question

- We have a file `weather.txt`:
  
  16.2
  23.5
  19.1
  7.4
  22.8
  18.5
  -1.8
  14.9

- Write a program that prints the change in temperature between each pair of neighboring days.

  16.2 to 23.5, change = 7.3
  23.5 to 19.1, change = -4.4
  19.1 to 7.4, change = -11.7
  7.4 to 22.8, change = 15.4
  22.8 to 18.5, change = -4.3
  18.5 to -1.8, change = -20.3
  -1.8 to 14.9, change = 16.7
# Displays changes in temperature from data in an input file.

def main():
    input = open("weather.txt")
    lines = input.readlines()
    prev = float(lines[0])  # fencepost

    for i in range(1, len(lines)):
        float(next) = lines[i]
        print(str(prev) + " to " + str(next) + ", change = " + str(next - prev))
        prev = next
Gas prices question

• Write a program that reads a file gasprices.txt
  • Format: Belgium $/gal
  US $/gal
  date

8.20
3.81
3/21/11
8.08
3.84
3/28/11
...

• The program should print the average gas price over all data in the file for both countries:

  Belgium average: 8.3 $/gal
  USA average: 3.9 $/gal
def main():
    file = open("gasprices.txt")
    belgium = 0
    usa = 0
    count = 0
    lines = file.readlines()

    for i in range(0, len(lines), 3):
        belgium = belguim + float(lines[i])
        usa = usa + float(lines[i + 1])

    print("Belgium average: " + str(belgium / count) + " $/gal")
    print("USA average: " + str(usa / count) + " $/gal")