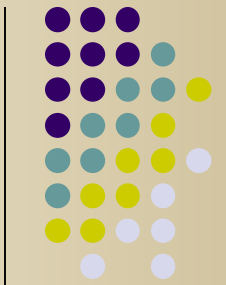
ENGR/CS 101 CS Session Lecture 13

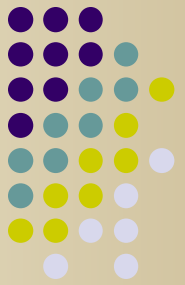


- Log into Windows/ACENET (reboot if in Linux)
- Start Python, open program from last time.

Outline

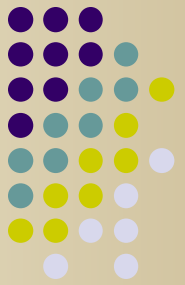
- Problem: Computing the mean of a collection of numbers
- Problem: Computing the median of a collection of numbers

Problem: Computing the Mean



- ***Central tendency*** is one of the most often used measures of a collection of data. This measure estimates where the "center" of a collection is. Two common ways to measure central tendency that we will look at today are the ***mean*** and the ***median***. (A third way is the ***mode*** which we will look at next class.)

Computing the Mean

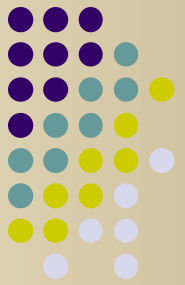


- The mean of a list of *N* values named $\{x_0, x_1, \ldots x_i, \ldots x_{n-1}\}$ is defined by the following formula:

$$mean = \frac{\sum_{i=0}^{N-1} x_i}{N}$$

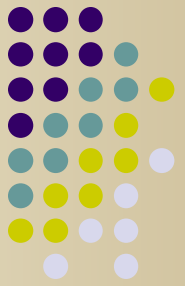
- The basic idea is to sum up all the numbers in a collection, then divide by the number of elements. The summation is implemented using a for-loop.

computeMean Function



- Analysis
  - Received: a list of data values
  - Returned: mean of data values, a real number
- Design
  1. Initialize total to 0.0
  2. For each element of the list
     a. Add the element to total
  3. Compute mean = total/number of items
  4. Return the mean

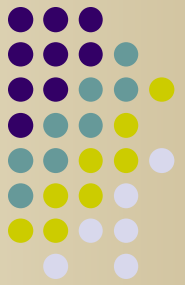
For-loops for counting



- Another way to access each element of list is to have a for-loop count from 0 to one less than the number of elements and use the counter as an index into the list
- The range(n) function returns a list of values [0..n) that we can to count

```
n = len(aList)
for i in range(n) # [0..n)
  total = total + aList[i]
```

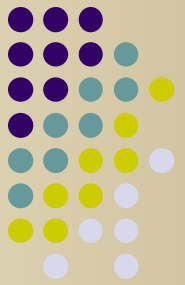
In-class Exercise, Part 1



- Write the Python code for the **`computeMean`** function. Note: we want to do a real division.
- Run the module and test your function in the interpreter

```
>>> mylist = [35, 67, 12, 48, 55, 29]
>>> computeMean(mylist)
41.0
```

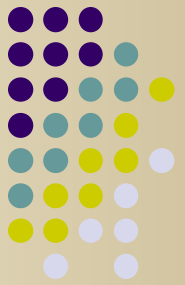
In-class Exercise, Part 1



- Write code in the main program function to call **`computeMean`** and display the mean of the **`items`** list. Run and test the program.

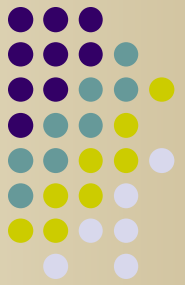
```
>>> main()
Enter the name of the input file: numbers.dat
Range of this collection is 94
Mean of this collection is 49.84
```

Computing the Median



- The median is the data value that occurs in the exact middle of the collection. That is, the same number of values are less than the median as are greater than the median. If there are an even number of values, the median is the average of the two middle values.
- One way to find the middle value is to sort the list and compute the middle index.

computeMedian Function

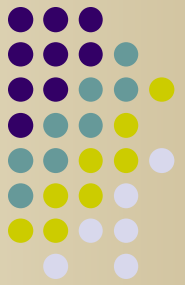


- Analysis – same as computeMean
- Design

1. Make copy of the list and sort the copy
2. Find midIdx, the index of the middle element
3. If there are an even number of elements
   a. The index of the other middle element is midIdx-1
   b. Compute median = average of two middle elements of the copy

   Else
   c. Compute median = middle element of the copy
4. Return the median

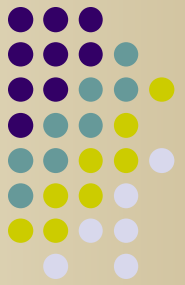
In-class Exercise, Part 2



- Write the Python code for the **`computeMedian`** function. Note: we want to do a real division when there are 2 middle elements.
- Run the module and test your function in the interpreter

```
>>> mylist = [35, 67, 12, 48, 55, 29]
>>> computeMedian(mylist)
41.5
>>> mylist = [35, 67, 12, 48, 55]
>>> computeMedian(mylist)
48
```

In-class Exercise, Part 2



- Write code in the main program function to call **`computeMedian`** and display the median of the **`items`** list. Run and test the program.

```
>>> main()
Enter the name of the input file: numbers.dat
Range of this collection is 94
Mean of this collection is 49.84
Median of this collection is 44
```

- The **`numbers.dat`** file has an odd number of values (25). Create a data file with an even number of values and run/test the program again.