Introduction to Programming

Spring 2022

Data Collection

- **Simple Statistics**
- **Applying Lists**
- Lists of Records
- Designing with Lists and Classes
- Case Study: Python Calculator
- Case Study: Better Cannonball Animation
- Non-Sequential Collections

Example Problem: Simple Statistics

- Many programs deal with large collections of similar information.
  - Words in a document
  - Students in a course
  - Customers of a business
  - Graphics objects drawn on the screen

Sample Problem: Simple Statistics

```
# average4.py
#    A program to average a set of numbers
#    Illustrates sentinel loop using empty string as sentinel

def main():
    sum = 0.0
    count = 0
    xStr = input("Enter a number (<Enter> to quit) >> ")
    while xStr != "":
        x = float(xStr)
        sum = sum + x
        count = count + 1
        xStr = input("Enter a number (<Enter> to quit) >> ")
    print("\nThe average of the numbers is", sum / count)
```

Sample Problem: Simple Statistics

- This program allows the user to enter a sequence of numbers
  - but the program itself doesn't keep track of the numbers that were entered – it only keeps a running total.
- Suppose we want to extend the program to compute not only the mean, but also the median and standard deviation.

Sample Problem: Simple Statistics

- The median is the data value that splits the data into equal-sized parts.
- For the data 2, 4, 6, 9, 13, the median is 6, since there are two values greater than 6 and two values that are smaller.
- One way to determine the median is to store all the numbers, sort them, and identify the middle value.

Sample Problem: Simple Statistics

- The standard deviation is a measure of how spread out the data is relative to the mean.
  - If the data is tightly clustered around the mean, then the standard deviation is small.
  - If the data is more spread out, the standard deviation is larger.
- The standard deviation is a yardstick to measure/express how exceptional a value is.

Sample Problem: Simple Statistics

- The standard deviation is

$$s = \sqrt{\frac{\sum(\bar{x} - x_i)^2}{n-1}}$$

- Here $\bar{x}$ is the mean, x_i represents the i^{th} data value and n is the number of data values.
- The expression $(\bar{x} - x_i)^2$ is the square of the "deviation" of an individual item from the mean.

Sample Problem: Simple Statistics

- The numerator is the sum of these squared "deviations" across all the data.
- Suppose our data was 2, 4, 6, 9, and 13.
  - The mean is 6.8
  - The numerator of the standard deviation is

$$(6.8-2)^2+(6.8-4)^2+(6.8-6)^2+(6.8-9)^2+(6.8-13)^2=74.8$$

$$s=\sqrt{\frac{74.8}{5-1}}=\sqrt{18.7}=4.32$$

Sample Problem: Simple Statistics

- As you can see, calculating the standard deviation not only requires the mean
  - which can't be calculated until all the data is entered
  - but also each individual data element!
- We need some way to remember these values as they are entered.

Applying Lists

- We need a way to store and manipulate an entire collection of numbers.
- We can't just use a bunch of variables, because we don't know many numbers there will be.
- What do we need? Some way of combining an entire collection of values into one object.

Lists and Arrays

- Python lists are ordered sequences of items. For instance, a sequence of n numbers might be called S:

S = s0, s1, s2, s3, …, sn-1

- Specific values in the sequence can be referenced using subscripts.
- By using numbers as subscripts, mathematicians can succinctly summarize computations over items in a sequence using subscript variables.

Lists and Arrays

- Suppose the sequence is stored in a variable s. We could write a loop to calculate the sum of the items in the sequence like this:

```
sum = [1, 5, 10, 15]
for i in range(n):
    sum = sum + s[i]
```

- Almost all computer languages have a sequence structure like this, sometimes called an array.

Lists and Arrays

- A list or array is a sequence of items where:
  - the entire sequence is referred to by a single name (i.e. s) and
  - individual items can be selected by indexing (i.e. s[i]).
- In other programming languages, arrays are generally a fixed size, meaning that when you create the array, you have to specify how many items it can hold.
- Arrays are generally also homogeneous, meaning they can hold only one data type.

Lists and Arrays

- Python does not have an array
  - We can use list for the same purpose
- Python lists are dynamic. They can grow and shrink on demand.
- Python lists are also heterogeneous, a single list can hold arbitrary data types.
- Python lists are mutable sequences of arbitrary objects.

List Operations

Operator	Meaning
<seq> + <seq>	Concatenation
<seq> * <int-expr>	Repetition
<seq>[]	Indexing
len(<seq>)	Length
<seq>[:]	Slicing
for <var> in <seq>:	Iteration
<expr> in <seq>	Membership (Boolean)

List Operations

Except for the membership check, we've used these operations before on strings.

The membership operation can be used to see if a certain value appears anywhere in a sequence.

```
>>> lst = [1,2,3,4]
>>> 3 in lst
True
```

List Operations

The summing example from earlier can be written like this:

```
sum = 0
for x in s:
    sum = sum + x
```

Unlike strings, lists are mutable:

```
>>> lst = [1,2,3,4]
>>> lst[3]
4
>>> lst[3] = "Hello"
>>> lst
[1, 2, 3, 'Hello']
>>> lst[2] = 7
>>> lst
[1, 2, 7, 'Hello']
```

List Operations

- Except for the membership check, we've used these operations before on strings.
- The membership operation can be used to see if a certain value appears anywhere in a sequence.
- The summing example from earlier can be written like this:

```
sum = [1,5,10,15]
for x in s:
    sum = sum + x
```

List Operations

- A list of identical items can be created using the repetition operator. This command produces a list containing 50 zeroes:

```
zeroes = [0] * 50
```

List Operations

- Lists are often built up one piece at a time using append.

```
nums = []
x = float(input('Enter a number: '))
while x >= 0:
    nums.append(x)
    x = float(input('Enter a number: '))
```

- Here, `nums` is being used as an accumulator, starting out empty, and each time through the loop a new value is tacked on.

List Operations

Method	Meaning
<list>.append(x)	Add element x to end of list.
<list>.sort()	Sort (order) the list. A comparison function may be passed as a parameter.
<list>.reverse()	Reverse the list.
<list>.index(x)	Returns index of first occurrence of x.
<list>.insert(i, x)	Insert x into list at index i.
<list>.count(x)	Returns the number of occurrences of x in list.
<list>.remove(x)	Deletes the first occurrence of x in list.
<list>.pop(i)	Deletes the i^{th} element of the list and returns its value.

List Operations

```
>>> lst = [3, 1, 4, 1, 5, 9]
>>> lst.append(2)
>>> lst
[3, 1, 4, 1, 5, 9, 2]
>>> lst.sort()
>>> lst
[1, 1, 2, 3, 4, 5, 9]
>>> lst.reverse()
>>> lst
[9, 5, 4, 3, 2, 1, 1]
>>> lst.index(4)
2
```

```
>>> lst.insert(4, "Hello")
>>> lst
[9, 5, 4, 3, 'Hello', 2, 1, 1]
>>> lst.count(1)s
2
>>> lst.remove(1)
>>> lst
[9, 5, 4, 3, 'Hello', 2, 1]
>>> lst.pop(3)
3
>>> lst
[9, 5, 4, 'Hello', 2, 1]
```

List Operations

- Most of these methods don't return a value – they change the contents of the list in some way.
- Lists can grow by appending new items, and shrink when items are deleted.
- Individual items or entire slices can be removed from a list using the `del` operator.
- `del` isn't a list method, but a built-in operation that can be used on list items.

List Operations - del

```
>>> myList=[34, 26, 0, 10]
>>> del myList[1]
>>> myList
[34, 0, 10]
>>> del myList[1:3]
>>> myList
[34]
```

`del` isn't a list method, but a built-in operation that can be used on list items.

List Operations

Basic list principles

- A list is a sequence of items stored as a single object.
- Items in a list can be accessed by indexing, and sublists can be accessed by slicing.
- Lists are mutable; individual items or entire slices can be replaced through assignment statements.
- Lists support a number of convenient and frequently used methods.
- Lists will grow and shrink as needed.

Statistics with Lists

- One way we can solve our statistics problem is to store the data in a list.
- We could then write a series of functions that take a list of numbers and calculates the mean and other function to calculate the median.
- Let's rewrite our earlier program to use lists to find the mean and the median.

Statistics with Lists

Let's write a function called `getNumbers` that gets numbers from the user.

- We'll implement the sentinel loop to get the numbers.
- An initially empty list is used as an accumulator to collect the numbers.
- The list is returned once all values have been entered.

Statistics with Lists

```
def getNumbers():
    nums = []      # start with an empty list

    # sentinel loop to get numbers
    xStr = input("Enter a number (<Enter> to quit) >> ")
    while xStr != "":
        x = float(xStr)
        nums.append(x)   # add this value to the list
        xStr = input("Enter a number (<Enter> to quit) >> ")
    return nums
```

Using this code, we can get a list of numbers from the user with a single line of code:

```
data = getNumbers()
```

Statistics with Lists

Now we need a function that will calculate the mean of the numbers in a list.

- Input: a list of numbers
- Output: the mean of the input list

```
def mean(nums):
    sum = 0.0
    for num in nums:
        sum = sum + num
    return sum / len(nums)
```

Statistics with Lists

- We don't have a formula to calculate the median. We'll need to come up with an algorithm to pick out the middle value.
- First, we need to arrange the numbers in ascending order.
- Second, the middle value in the list is the median.
- If the list has an even length, the median is the average of the middle two values.

Statistics with Lists

```
def median(nums):
    nums.sort()
    size = len(nums)
    midPos = size // 2
    if size % 2 == 0:
        median = (nums[midPos] + nums[midPos-1]) / 2
    else:
        median = nums[midPos]
    return median
```

Class Work

- Write a function inner_prod (x, y) that computes the inner product of two (same size) lists. The inner product of x and y is computed as:

$$\sum_{i=0}^{n-1} x_i \, y_i$$