- Python lists are similar to arrays in other programming languages. Python lists are more flexible because their size can vary and they are heterogeneous. Python lists also support a number of useful methods.
- One particularly important data-processing operation is sorting. Python lists have a `sort` method that can be customized by supplying a suitable key function. This allows programs to sort lists of arbitrary objects.
- Classes can use lists to maintain collections stored as instance variables. Oftentimes using a list is more flexible than using separate instance variables. For example, a GUI application might use a list of buttons instead of an instance variable for each button.
- An entire program can be viewed as a collection of data and a set of operations—an object. This is a common approach to structuring GUI applications.
- A Python dictionary implements an arbitrary mapping from keys into values. This is very useful for representing non-sequential collections.

11.9 Exercises

Review Questions

True/False

1. The median is the average of a set of data.
2. Standard deviation measures how spread out a data set is.
3. Arrays are usually heterogeneous, but lists are homogeneous.
4. A Python list cannot grow and shrink in size.
5. Unlike strings, Python lists are not mutable.
6. A list must contain at least one item.
7. Items can be removed from a list with the `del` operator.
8. A tuple is similar to an immutable list.
9. A Python dictionary is a kind of sequence.

Multiple Choice

1. Where mathematicians use subscripting, computer programmers use
   a) slicing b) indexing c) Python d) caffeine

2. Which of the following is not a built-in sequence operation in Python?
   a) sorting b) concatenation c) slicing d) repetition

3. The method that adds a single item to the end of a list is
   a) `extend` b) `add` c) `plus` d) `append`

4. Which of the following is not a Python list method?
   a) `index` b) `insert` c) `get` d) `pop`

5. Which of the following is not a characteristic of a Python list?
   a) It is an object. b) It is a sequence.
   c) It can hold objects. d) It is immutable.

6. Which of the following expressions correctly tests if `x` is even?
   a) `x % 2 == 0` b) `even(x)` c) `not odd(x)` d) `x % 2 == x`

7. The parameter `xbar` in `stdDev` is what?
   a) median b) mode c) spread d) mean

8. What keyword parameter is used to send a key-function to the `sort` method?
   a) reverse b) reversed c) cmp d) key

9. Which of the following is *not* a dictionary method?
   a) `get` b) `keys` c) `sort` d) `clear`

10. The `items` dictionary method returns a(n)
    a) int b) sequence of tuples c) bool d) dictionary

Discussion

1. Given the initial statements

```
s1 = [2,1,4,3]
s2 = ['c','a','b']
```

   show the result of evaluating each of the following sequence expressions:

a) `s1 + s2`

b) `3 * s1 + 2 * s2`

c) `s1[1]`

d) `s1[1:3]`

e) `s1 + s2[-1]`

2. Given the same initial statements as in the previous problem, show the values of `s1` and `s2` after executing each of the following statements. Treat each part independently (i.e., assume that `s1` and `s2` start with their original values each time).

a) `s1.remove(2)`

b) `s1.sort()`

c) `s1.append([s2.index('b')])`

d) `s2.pop(s1.pop(2))`

e) `s2.insert(s1[0], 'd')`

Programming Exercises

1. Modify the statistics program from this chapter so that client programs have more flexibility in computing the mean and/or standard deviation. Specifically, redesign the library to have the following functions:

   `mean(nums)` Returns the mean of numbers in `nums`.

   `stdDev(nums)` Returns the standard deviation of `nums`.

   `meanStdDev(nums)` Returns both the mean and standard deviation of `nums`.

2. Extend the `gpasort` program so that it allows the user to sort a file of students based on GPA, name, or credits. Your program should prompt for the input file, the field to sort on, and the output file.

3. Extend your solution to the previous problem by adding an option to sort the list in either ascending or descending order.

4. Give the program from the previous exercise(s) a graphical interface. You should have `Entrys` for the input and output file names and a button for each sorting order. Bonus: Allow the user to do multiple sorts and add a button for quitting.

5. Most languages do not have the flexible built-in list (array) operations that Python has. Write an algorithm for each of the following Python operations and test your algorithm by writing it up in a suitable function. For example, as a function, `reverse(myList)` should do the same as `myList.reverse()`. Obviously, you are not allowed to use the corresponding Python method to implement your function.

   a) `count(myList, x)` (like `myList.count(x)`)

   b) `isin(myList, x)` (like `x in myList`))

   c) `index(myList, x)` (like `myList.index(x)`)

   d) `reverse(myList)` (like `myList.reverse()`)

   e) `sort(myList)` (like `myList.sort()`)

6. Write and test a function `shuffle(myList)` that scrambles a list into a random order, like shuffling a deck of cards.

7. Write and test a function `innerProd(x,y)` that computes the inner product of two (same length) lists. The inner product of x and y is computed as:

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

8. Write and test a function `removeDuplicates(somelist)` that removes duplicate values from a list.

9. One disadvantage of passing a function to the list `sort` method is that it makes the sorting slower, since this function is called repeatedly as Python compares various items.

   An alternative to creating a special key function is to create a "decorated" list that will sort in the desired order using the standard Python ordering. For example, to sort `Student` objects by GPA, we could first create a list of tuples `[(gpa0, Student0), (gpa1,Student1), ..]` and then sort this list without passing a key function. These tuples will get sorted into GPA order. The resulting list can then be traversed to rebuild a list of student objects in GPA order. Redo the `gpasort` program using this approach.

10. The Sieve of Eratosthenes is an elegant algorithm for finding all of the prime numbers up to some limit n. The basic idea is to first create a list of numbers from 2 to n. The first number is removed from the list, and announced as a prime number, and all multiples of this number up to n are removed from the list. This process continues until the list is empty.

    For example, if we wished to find all the primes up to 10, the list would originally contain 2, 3, 4, 5, 6, 7, 8, 9, 10. The 2 is removed and announced to be prime. Then 4, 6, 8, and 10 are removed, since they are multiples of 2. That leaves 3, 5, 7, 9. Repeating the process, 3 is announced as prime and removed, and 9 is removed because it is a multiple of 3. That leaves 5 and 7. The algorithm continues by announcing that 5 is prime and removing it from the list. Finally, 7 is announced and removed, and we're done.

    Write a program that prompts a user for n and then uses the sieve algorithm to find all the primes less than or equal to n.

11. Write an automated censor program that reads in the text from a file and creates a new file where all of the four-letter words have been replaced by `"****"`. You can ignore punctuation, and you may assume that no words in the file are split across multiple lines.

12. Extend the program from the previous exercise to accept a file of censored words as another input. The words in the original file that appear in the censored words file are replaced by a string of `"*"`s with length equal to the number of characters in the censored word.

13. Write a program that creates a list of card objects (see Programming Exercise 11 from Chapter 10) and prints out the cards grouped by suit and in rank order within suit. Your program should read the list of cards from a file, where each line in the file represents a single card with the rank and suit separated by a space. *Hint*: First sort by rank and then by suit.

14. Extend the previous program to analyze a list of five cards as a poker hand. After printing the cards, the program categorizes accordingly.

    Royal Flush 10, jack, queen, king, ace, all of the same suit.

    Straight Flush Five ranks in a row, all of the same suit.

    Four of a Kind Four of the same rank.

    Full House Three of one rank and two of another.

Flush Five cards of the same suit.

Straight Five ranks in a row.

Three of a kind Three of one rank (but not a full house or four of a kind).

Two pair Two each of two different ranks.

Pair Two of the same rank (but not two pair, three or four of a kind).

***X* High** If none of the previous categories fit, *X* is the value of the highest rank. For example, if the largest rank is 11, the hand is "Jack high."

15. Create a class `Deck` that represents a deck of cards. Your class should have the following methods:

constructor Creates a new deck of 52 cards in a standard order.

shuffle Randomizes the order of the cards.

dealCard Returns a single card from the top of the deck and removes the card from the deck.

cardsLeft Returns the number of cards remaining in the deck.

Test your program by having it deal out a sequence of n cards from a shuffled deck where n is a user input. You could also use your deck object to implement a Blackjack simulation where the pool of cards is finite. See Programming Exercises 8 and 9 in Chapter 9.

16. Create a class called `StatSet` that can be used to do simple statistical calculations. The methods for the class are:

`__init__(self)` Creates a `StatSet` with no data in it.

`addNumber(self,x)` `x` is a number. Adds the value `x` to the `statSet`.

`mean(self)` Returns the mean of the numbers in this `statSet`.

`median(self)` Returns the median of the numbers in this `statSet`.

`stdDev(self)` Returns the standard deviation of the numbers in this `statSet`.

`count(self)` Returns the count of numbers in this `statSet`.

`min(self)` Returns the smallest value in this `statSet`.

`max(self)` Returns the largest value in this `statSet`.

Test your class with a program similar to the simple statistics program from this chapter.

17. In graphics applications, it is often useful to group separate pieces of a drawing together into a single object. For example, a face might be drawn from individual shapes, but then positioned as a whole group. Create a new class `GraphicsGroup` that can be used for this purpose. A `GraphicsGroup` will manage a list of graphics objects and have the following methods:

    `__init__(self, anchor)` `anchor` is a `Point`. Creates an empty group with the given anchor point.

    `getAnchor(self)` Returns a clone of the anchor point.

    `addObject(self, gObject)` `gObject` is a graphics object. Adds `gObject` to the group.

    `move(self, dx,dy)` Moves all of the objects in the group (including the anchor point).

    `draw(self, win)` Draws all the objects in the group into `win`. The anchor point is not drawn.

    `undraw(self)` Undraws all the objects in the group.

    Use your new class to write a program that draws some simple picture with multiple components and moves it to wherever the user clicks.

18. Extend the random walk program from Chapter 9 (Programming Exercise 12). Consider the sidewalk as a sequence of squares, and each step moves the walker one square. Your program should keep track of how many times each square of the sidewalk is stepped on. Start your walker in the middle of a sidewalk of length n where n is a user input, and continue the simulation until it drops off one of the ends. Then print out the counts of how many times each square was landed on.

19. Create and test a `Set` class to represent a classical set. Your sets should support the following methods:

    `Set(elements)` Creates a set (`elements` is the initial list of items in the set).

    `addElement(x)` Adds `x` to the set.

    `deleteElement(x)` Removes `x` from the set, if present. If `x` is not in the set, the set is left unchanged.

    `member(x)` Returns true if `x` is in the set and false otherwise.

`intersection(set2)` Returns a new set containing just those elements that are common to this set and `set2`.

`union(set2)` Returns a new set containing all of elements that are in this set, `set2`, or both.

`subtract(set2)` Returns a new set containing all the elements of this set that are not in `set2`.

By the way, sets are so useful that Python actually has a built-in `set` datatype. While you may want to investigate Python's `set`, you should not use it here. The point of this exercise is to help you develop your skills in algorithm development using lists and dictionaries.

20. Extend the cannonball animation from the chapter to allow the user to adjust the initial height of the launcher. The height adjustment should be handled similar to the way angle and velocity are. Pick a pair of keys of your own choosing for adjusting the height up and down.

21. Extend the cannonball animation example to include target objects. A target is a randomly sized rectangle that is placed somewhere downrange in the animation. When a target is hit, it disappears and a new target is generated. Further extensions could involve moving targets and keeping track of the number of hits.