Computer Science — Lists

1. **Lists**
   The `list` type in Python is used to store mutable, collections of data in a known order. The list itself is surrounded by square brackets `[ ]`, and individual items in a list are separated by commas. Examples include the list `[ 2, 3, 5, 7, 11, 13, 17, 19]`, the list `["Gary", "Dana", "Ruthie"]`, and the list `[ "Richard", "White", 1960, 57, "626-845-1235"]`.

2. **Storing data in a list**

   a. **Initializing a list with known values**

```
primes = [2, 3, 5, 7, 11, 13, 17, 19]
```

   b. **Filling a list using the `.append()` method in a loop**

```
friends = []    # initialize empty list
for i in range(num_of_friends):
    friends.append(input("Enter friend name: "))
```

3. **Accessing data in a list**

   a. **Accessing individual elements in a list**

```
print("My first friend is",friends[0])
```

```
# printing contents of list using an index
for i in range(len(friends)):
    print("Friend #", i, "is", friends[i])
```

```
# printing contents of list using an iterator
for person in friends:
    print(person)
```

   b. **Accessing a range of elements by "slicing" a list (not common)**

```
print("All my friends:" , friends[0:len(friends)]
print("All of my friends but the first and last:" , friends[1:-1]
```

4. **Working through a list to find values**

   a. **Finding an item on a list**

```
names = ["Gary", "Dana", "Robbie"]
search_value = "Richard"
found = False     # boolean flag
for i in range(len(names)):
    if names[i] = search_value:
        found = True    # we found it
        break           # no need to keep looking
# By the time we get down here, we’ve either found it, or we haven’t. :)
if (found):
    return i            # Return the position we found it at
else:
    return -1           # Return a non-valid position that indicates failure
```

b. Finding the highest value in a list

```
random_nums = [1, 3, -5, 17, 6, 42, -5]
max_value = random_nums[0]
for num in random_nums:
    if num > max_value:
        max_value = num
print("The largest value was", max_value)
```

c. Finding the position of the highest value in a list

```
random_nums = [1, 3, -5, 17, 6, 42, -5]
max_index = 0
for i in range(len(random_nums)):
    if random_nums[i] > random_nums[max_index]:
        max_index = i
print("The location of the largest value is",i)
```

d. Identifying specific values in a list

```
random_nums = [1, 3, -5, 17, 6, 42, -5]
even_numbers = 0
odd_numbers = 0
for i in range(len(random_nums)):
    if random_nums[i] % 2 == 0:
        even_numbers += 1
    else:
        odd_numbers += 1
print("There were",even_numbers,"even values and",odd_numbers,"odd values.")
```

5. Functions and methods that are useful with lists

```
shopping_list = []       # initializes an empty list that can be used to store items
listA + list_b           # concatenates two lists into one
listA * <integer>        # repeats a list a number of times
listA[n]                 # gets the item located at position n
listA[i:j]               # slices a subset from the list
len(listA)               # determines the length of a list
if (j in listA):         # boolean expression, true if j is in the list
listA.append(x)          # add an element to end of listA
listA.sort()             # sorts the list
listA.reverse()          # reverses listA
listA.index(x)           # finds the first location of x in listA
listA.insert(i,x)        # insert the value x into listA at index i
listA.count(x)           # count how many occurrences of x in listA
listA.remove(x)          # remove the first occurrence of x from listA
listA.pop(i)             # returns element i from listA and removes it from list
listA.pop()              # pops the last item of the list.
```

6. **Lists of lists**
The elements of a `list` can include any type of data, including another `list`. Here are two examples to demonstrate this strategy.

a. **Contacts Manager**
In this example, a list maintains a list of friends, and each element in the list is in turn a list which contains a person's name, birthday, cellphone, and email.

```
friends = []          # keeps track of each friend's name, birthday,
                      # cellphone, and email
a_friend = ["Alex Brady", "6/17/92", "626-845-6348", "alexb@gmail.com"]
friends.append(a_friend)
another_friend = ["Brian McGoldrick", "", "", "brian@mcgoldrick.com"]
friends.append(another_friend)

# printing out all the data
for i in range(len(friends)):
    for j in range(len(friends[i])):
        print(friends[i][j])
```

b. **A 2-Dimensional Grid of Values**
A map, a Cartesian coordinate system, a chessboard, a table of values, a spreadsheet... all can be represented as pieces of data located on a two-dimensional grid, with each piece of data accessible by its column and row in the table.

```
num_of_rows = 10
num_of_cols = 12

# initialize the grid
grid = []
for row in range(num_of_rows):
    grid.append([])          # append a list (a row)
    # Now go through and put elements as a list in that row
    for col in range(num_of_cols):
        grid[row].append(0)

# print out the grid
for row in range(num_of_rows):
    for col in range(num_of_cols):
        print(grid[row][col], end=' ')
    print()     # space down at the end of a line
```

Note that in this example we are using a "row-first" address system. It's also possible to use a "column-first" address system, but be sure to error test the model extensively.

EXERCISES

1. Write a function `list_100()` that creates and returns a list of the values from 0 (inclusive) to 100 (exclusive).

2. Write a function called `randnums()` that takes three parameters: `size`, `low`, and `high` . The function should fill a list with `size` random numbers in the range `low` (inclusive) to `high` (exclusive) and return that list.

3. Write a function called `find_index()` that takes two parameters: a list and a search term. The function should look through the list to find the search term, and return its index it if is in the list. If the item is not in the list, it should return a value of `-1`.

4. Write a function called `is_in_list()` that takes two parameters: a list and a search term. The function should return `True` if the item is in the list. Otherwise, it should return `False`.

5. Write a function called `get_odds()` that takes a list of integers as a parameter. The function should go through the list, find all the odd integers, and return a list with just those values in it.

6. Write a function called `sum_values` that takes a list of numbers as a parameter and returns the sum of all the values in the list.

7. Write a function `censor()` that takes a `sentence` (as a string) and a `bad_words` list as parameters. The function should return the sentence with all of the bad words replaced by asterisks (`*`).

8. Write a function `card_name()` that takes a card number from a deck of 52 playing cards (numbered 0 - 51), and returns a string indicating the name of the card. The cards have four suits—Hearts, Diamonds, Spades, and Clubs—and 13 values: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King. So calling `card_name(0)` would return the string "`Ace of Hearts`," `card_name(1)` would return "`2 of Hearts`," `card_name(12)` would return "`King of Hearts`," `card_name(13)` would return "`Ace of Diamonds`," and `card_name(51)` would return "`King of Clubs`." [Hint: Use integer division to identify the card number with the names of the suits identified in a list. Use the mod function (%) to identify the value of the card.]