

COMP 141

Lists II



1

Announcements

Program 7 – due Thursday, Nov. 16th

Reminder:

Midterm 2 on Nov. 8th



2

Practice from Last Time

Get the file Nov1.py from my Box.com code directory. It has the main function written for you and stubs for 2 other functions that you will need to write.

findAverage(numbers) – will return the average of all the numbers in the list

countNumbers(numbers, average) - will return 2 values; it counts the number of above average and below average numbers in a list



3

Other String Methods

- **Programs commonly need to search for substrings**
- **Several methods to accomplish this:**
 - `endswith(substring)`: checks if the string ends with `substring`
 - Returns True or False
 - `startswith(substring)`: checks if the string starts with `substring`
 - Returns True or False



4

More String Methods

- Several methods to accomplish this (cont'd):
 - `find(substring)`: searches for *substring* within the string
 - Returns lowest index of the substring, or if the substring is not contained in the string, returns -1
 - `replace(substring, new_string)`:
 - Returns a copy of the string where every occurrence of *substring* is replaced with *new_string*



5

Using the find method

```
def main():
    filename = "First_Last_assignsubmission_file_lastname_firstname_prgr6.py"
    print(renameFile(filename))

def renameFile(filename):
    ind = filename.find("file_")
    fileName = filename[ind+5:]
    return fileName

main()
```

Output:
lastname_firstname_prgr6.py



6

String Methods

Table 9-3 Search and replace methods

Method	Description
<code>endswith(substring)</code>	The <i>substring</i> argument is a string. The method returns true if the string ends with <i>substring</i> .
<code>find(substring)</code>	The <i>substring</i> argument is a string. The method returns the lowest index in the string where <i>substring</i> is found. If <i>substring</i> is not found, the method returns -1.
<code>replace(old, new)</code>	The <i>old</i> and <i>new</i> arguments are both strings. The method returns a copy of the string with all instances of <i>old</i> replaced by <i>new</i> .
<code>startswith(substring)</code>	The <i>substring</i> argument is a string. The method returns true if the string starts with <i>substring</i> .



7

Testing, Searching, and Manipulating Strings

- You can use the `in` operator to determine whether one string is contained in another string
 - General format: `string1 in string2`
 - *string1* and *string2* can be string literals or variables referencing strings
- Similarly you can use the `not in` operator to determine whether one string is not contained in another string



8

Class Practice

- Write a function called `count_unique` that counts the number of unique characters in a string.
 - `count_unique("abracadabra")` returns 5.
- Write a function called `count_dups` that counts the number of back-to-back duplicated characters in a string.
 - `count_dups("balloon")` returns 2

Finding Items in Lists with the `in` Operator

- You can use the `in` operator to determine whether an item is contained in a list
 - General format: `item in list`
 - Returns `True` if the item is in the list, or `False` if it is not in the list
- Similarly you can use the `not in` operator to determine whether an item is not in a list



10

Example Using `in` Operator

```
# This program demonstrates the in operator
# used with a list.

def main():
    # Create a list of product numbers.
    prod_nums = ['V475', 'F987', 'Q143', 'R688']

    # Get a product number to search for.
    search = input('Enter a product number: ')

    # Determine whether the product number is in the list.
    if search in prod_nums:
        print(search, 'was found in the list.')
    else:
        print(search, 'was not found in the list.')

    # Call the main function.
    main()
```



11

List Methods and Useful Built-in Functions

- `append(item)`: used to add items to a list – `item` is appended to the end of the existing list
- `index(item)`: used to determine where an item is located in a list
 - Returns the index of the first element in the list containing `item`
 - Raises `ValueError` exception if `item` not in the list



12

find() doesn't exist for lists

- `list_var.index(item)`
- Searches left to right, returns position where found, but crashes if not found.
- Let's build an algorithm that replicates `find()`, but works for lists (returns -1 if not found).

Example Using Append

```
def main():
    infile = open("randomNums.txt", 'r')
    numbers = []
    for line in infile:
        numbers.append(int(line))
    print(numbers)
```

main()

Output

```
[62, 57, 35, 27, 45, 44, 46, 68, 86, 27, 88, 33, 11, 61, 64, 45,
56, 9, 33, 32, 56, 63, 24, 26, 100, 95, 62, 10, 87, 58, 69, 54, 75,
41, 22, 93, 82, 16, 92, 49, 6, 71, 85, 59, 56, 22, 3, 50, 1, 20, 54,
18, 27, 78, 17, 7, 41, 83, 92, 38, 5, 64, 60, 92, 15, 26, 57, 39,
80, 41, 67, 56, 24, 77, 28, 90, 24, 72, 2, 46, 75, 53, 58, 47, 50,
18, 40, 65, 24, 58, 4, 58, 81, 40, 6, 77, 85, 86, 68, 63]
```



14

List Methods and Useful Built-in Functions (cont'd.)

- `insert(index, item)`: used to insert *item* at position *index* in the list
- `sort()`: used to sort the elements of the list in ascending order
- `remove(item)`: removes the first occurrence of *item* in the list
- `reverse()`: reverses the order of the elements in the list



15

Program 8-5 (insert_list.py)

```
1 # This program demonstrates the insert method.
2
3 def main():
4     # Create a list with some names.
5     names = ['James', 'Kathryn', 'Bill']
6
7     # Display the list.
8     print('The list before the insert:')
9     print(names)
10
11     # Insert a new name at element 0.
12     names.insert(0, 'Joe')
13
14     # Display the list again.
15     print('The list after the insert:')
16     print(names)
17
18 # Call the main function.
19 main()
```

Program Output

```
The list before the insert:
['James', 'Kathryn', 'Bill']
The list after the insert:
['Joe', 'James', 'Kathryn', 'Bill']
```

16

Program 8-6 (remove_item.py)

```

1 # This program demonstrates how to use the remove
2 # method to remove an item from a list.
3
4 def main():
5     # Create a list with some items.
6     food = ['Pizza', 'Burgers', 'Chips']
7
8     # Display the list.
9     print('Here are the items in the food list:')
10    print(food)
11
12    # Get the item to change.
13    item = input('Which item should I remove? ')
14
15    try:
16        # Remove the item.
17        food.remove(item)
18
19        # Display the list.
20        print('Here is the revised list:')
21        print(food)
22
23    except ValueError:
24        print('That item was not found in the list.')
25
26 # Call the main function.
27 main()

```

Program Output (with input shown in bold)

```

Here are the items in the food list:
['Pizza', 'Burgers', 'Chips']
Which item should I remove? Burgers
Here is the revised list:
['Pizza', 'Chips']

```



17

List Methods and Useful Built-in Functions (cont'd.)

- **del statement**: removes an element from a specific index in a list
 - General format: `del list[i]`
- **min and max functions**: built-in functions that returns the item that has the lowest or highest value in a sequence
 - The sequence is passed as an argument
- **sum function**: built-in functions that returns the total of all the values in a sequence
 - The sequence is passed as an argument



18

Example Using del, min, max, and sum functions

```

my_list = [5, 4, 3, 2, 50, 40, 30]
print("Before Deletion:", my_list)
del my_list[2]
print("After Deletion:", my_list)

print("The lowest value is", min(my_list))
print("The highest value is", max(my_list))
print("The sum of values in my list is", sum(my_list))

alpha_list = ['a', 'b', 'c', 'd']
print("The lowest value is", min(alpha_list))
print("The highest value is", max(alpha_list))
# You cannot take the sum of a list that has strings in it

```

Output

```

Before Deletion: [5, 4, 3, 2, 50, 40, 30]
After Deletion: [5, 4, 2, 50, 40, 30]
The lowest value is 2
The highest value is 50
The sum of values in my list is 131
The lowest value is a
The highest value is d

```



19

Practice

Write a program that randomly generates 20 integers between 1 and 50, and stores them in a list. Print out the **lowest** and the **highest** numbers in your list, as well as the **sum** of all the numbers in the list.

Write a function that prints out sums of adjacent pairs of numbers in the list

Hint: You don't need the sliding window technique; instead, use math with list indices.

Write a function that takes a list and shifts all the elements in the list one spot to the left, without using slices! (the left-most element disappears)

Example: [1, 2, 3, 4, 5] turns into [2, 3, 4, 5, 5]



20