COMP 141

Lists II

Rhodes College

1

1

Announcements

Reminders:

Program 7 assigned – due Sunday, April 19th

2

2

Quiz 8

```
def main():
    s = "mississippi"
    cnt = 0

    for i in range(0, len(s)):
        ch = s[i]
        if ch not in s[:i]:
            print(ch)
            cnt += 1
    print(cnt)

main()
```

Output

```
m
i
s
p
4
```

3

Practice from Last Time

Get the file list1practice.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

4

4

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

5

5

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()
```

6

6

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

7

7

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).

8

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]
```

9

9

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

10

10

Program 8-5 (insert_list.py)

```
# This program demonstrates the insert method.

def main():
    # Create a list with some names.
    names = ['James', 'Kathryn', 'Bill']

    # Display the list.
    print('The list before the insert:')
    print(names)

    # Insert a new name at element 0.
    names.insert(0, 'Joe')

    # Display the list again.
    print('The list after the insert:')
    print(names)

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

Program Output

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

11

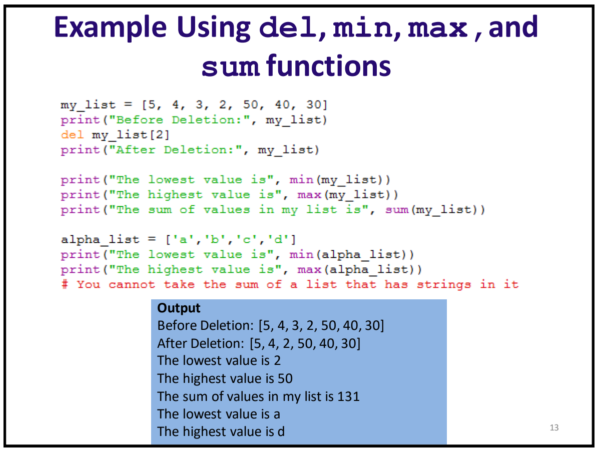
11

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

12

12



Example Using `del`, `min`, `max`, and `sum` functions

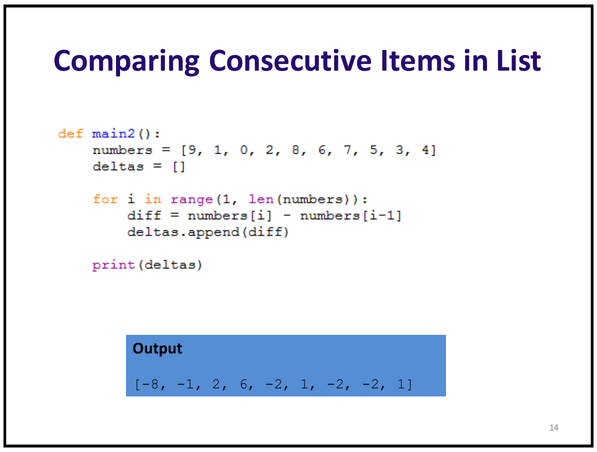
```
my_list = [5, 4, 3, 2, 50, 40, 30]
print("Before Deletion:", my_list)
del my_list[2]
print("After Deletion:", my_list)

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
```

13

13



Comparing Consecutive Items in List

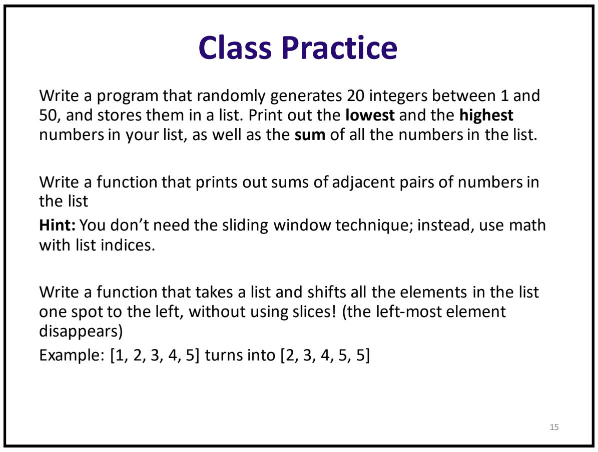
```
def main2():
    numbers = [9, 1, 0, 2, 8, 6, 7, 5, 3, 4]
    deltas = []

    for i in range(1, len(numbers)):
        diff = numbers[i] - numbers[i-1]
        deltas.append(diff)

    print(deltas)
```

Output

```
[-8, -1, 2, 6, -2, 1, -2, -2, 1]
```

14

14



Class 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]

15

15