COMP 141

Lists I

Rhodes College

1

1

Announcements

Program 7 assigned – due Sunday, April 19th

2

2

Introduction to Lists

- **List:** an object that contains multiple data items
  - Element: An item in a list
  - Format: `list = [item1, item2, etc.]`
  - Can hold items of different types
- **`print` function can be used to display an entire list**
- **`list()` function can convert certain types of objects to lists**

3

3

Introduction to Lists

A list of integers

```
even_numbers = [2, 4, 6, 8, 10]
```

A list of strings:

```
names = ['Molly', 'Steven', 'Will', 'Alicia']
```

A list holding different types:

```
info = ['Alicia', 27, 1550.87]
```

4

4

Example Using Lists

```
def main():
    # Create a list with some items.
    food = ['Pizza', 'Burgers', 'Chips']

    # Display the list.
    print('Here are the items in the food list:')
    print(food)


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

Program Output

```
Here are the items in the food list:
['Pizza', 'Burgers', 'Chips']
```

5

5

Why use lists?

- Lists exist so programmers can store multiple related variables together.
- Useful when we don't know ahead of time how many items we are going to store.
  - Lists solve this problem because a single list can hold from zero to practically any number of items in it.

6

Basic list operations

- Lists are created using square brackets around items separated by commas.

```
mylist = [1, 2, 3]
numbers = [-9.1, 4.77, 3.14]
fred = ["happy", "fun", "joy"]
```

- Lists are accessed using indices/positions just like strings.
- Most (but not all) string functions also exist for lists.

7

Strings	Lists
string_var = "abc123"	list_var = [item1, item2, ...]
string_var = ""	list_var = [ ]
len("abc123") len(string_var)	len([3, 5, 7, 9]) len(list_var)
string_var[p] string_var[p:q]	list_var[p] list_var[p:q]
str3 = str1 + str2 str3 = "abc" + "def"	list3 = list1 + list2 list3 = [1, 2, 3] + [4, 5, 6]
"i" in "team" -> False	7 in [2, 4, 6, 8] -> False

8

One important difference

Strings are *immutable*

- **You can't change a string without making a copy of it.**

```
s = "abc"
s[0] = "A"       # illegal!
s = "A" + s[1:] # legal
```

Lists are *mutable*

- **Can be changed "in-place" (without explicit copying)**

```
L = [2, 4, 6, 8, 10]
L[0] = 15        # legal
L.append(26)     # legal
```

9

9

Compare Immutable and Mutable

- **How can we switch the first and last letter in a string?**
- **How can we switch the first and last items in a list?**

10

10

Three common ways to make a list

- Make a list that already has stuff in it:
  `lst = [4, 7, 3, 8]`
- Make a list of a certain length that has the same element in all positions:
  `lst = [0] * 4  #makes the list [0,0,0,0]`
  - Common when you need a list of a certain length ahead of time.
  - Uses the repetition operator, similarly to strings
- Make an empty list:
  `lst = []`
  - Common when you're going to put things in the list coming from the user or a file.

11

Simple list problem

- How would we write a function to convert a number from 1-12 into the corresponding month of the year as a string?

```
def getmonth(month):
```

```
Ex: getmonth(2) should return
"February"
```

12

Examples of Concatenation

```
a = [1,2,3]
b = [4,5,6]
c = a + b
print(c)    # prints [1, 2, 3, 4, 5, 6]

mylist = ['a','b','c']
other = ['d','e','f']
print(mylist + other)  #['a', 'b', 'c', 'd', 'e', 'f']
```

13

13

Simple list problem

- What does this code do?

```
lst = [2] * 3
lst2 = [4] * 2
lst3 = lst + lst2
for x in range(0, len(lst3), 2):
    lst3[x] = -1
```

14

Examples of List Slices

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

numbers[2: ]   #[3, 4, 5, 6, 7, 8, 9, 10]

numbers[:-2]   #[1, 2, 3, 4, 5, 6, 7, 8]

numbers[1:8:2] #[2, 4, 6, 8]

numbers[5::-1] #[6, 5, 4, 3, 2, 1]

numbers[::-1]  #[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

15

15

```
#This program calculates the total of all the
#values in a list.

def main():
    #Create a list
    numbers = [2, 4, 6, 8, 10]

    #Create a variable to use as the accumulator
    total = 0

    #Calculate the total of the list elements
    for i in range(len(numbers)):
        value = numbers[i]
        total += value

    #Display the total of the list elements
    print('The total of the elements is', total)

#Call the main function
main()
```

Program Output

```
The total of the elements is 30
```

16

16

```
def main():
    #Number of days we will gather sales data for.
    NUM_DAYS = 5

    #Create a list to hold the sales for each day.
    sales = [0] * NUM_DAYS  #uses the repetition operator to initialize list

    print("Enter the sales for each day.")

    #Get the sales for each day
    for i in range(NUM_DAYS):
        print("Day #", i+1, ":", sep='', end='')
        sales[i] = float(input())


    #Display the values entered
    print("Here are the values you entered.")
    for i in range(len(sales)):
        print(sales[i])

#Call the main function
main()
```

Program Output (with input shown in bold)

```
Enter the sales for each day.
Day #1: 1000 Enter
Day #2: 2000 Enter
Day #3: 3000 Enter
Day #4: 4000 Enter
Day #5: 5000 Enter

Here are the values you entered:
1000.0
2000.0
3000.0
4000.0
5000.0
```

17

17

Practice

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

18

18