COMP 141

Lists

November 1, 2017

Announcements

Guest Lecture

Bill Berghel
M.S., Georgia Tech
B.S., Washington & Lee
Retired data scientist with FedEx

Slides source:
Catie Welsh's Fall 2016-2017 Lecture 25 slides for COMP141

Other Announcements

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

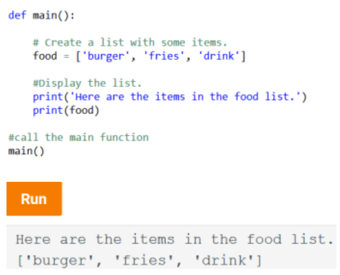
Introduction to Lists

A list of integers:
```
even_numbers = [2, 4, 6, 8, 10]
```

A list of strings:
```
Name=['Emma', 'Sophia', 'Isabella', 'Emily']
```

A list holding different types:
```
fastfood = ['egg mcmuffin', 290, 2.79]
```

Example Using Lists



```
def main():

    # Create a list with some items.
    food = ['burger', 'fries', 'drink']

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

#call the main function
main()
```

Run

```
Here are the items in the food list.
['burger', 'fries', 'drink']
```

Why Use Lists?

Lists exist so that 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.

Basic List Operations

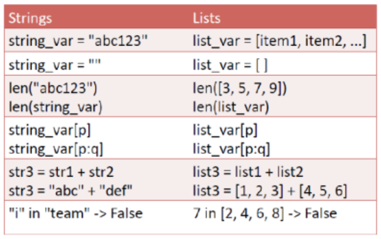
Lists are created using square brackets around items (elements) separated by commas.

```
    mylist = [1, 2, 3]
    numbers = [-9.1, 4.77, 3.14]
    fedexpsp = ['people', 'service', 'profit']
```

Lists are accessed using indices/positions just like strings.

Most – but not all – string functions also exist for lists.

Basic List Operations



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

One Important Difference

Strings are <u>immutable</u> .
- You can't change a string without making a copy of it.

```
s = 'abc'
s[0] = 'A' # definitely not legal!
s = 'A' + s[1:] # legal
```

Lists are <u>mutable</u>.
- You can change lists in-place without explicit copying.

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

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?

Compare Immutable and Mutable

How can we switch the first and last letter in a string?

```
ltrs = 'ABCDE'
print('Original string is', ltrs)
ltrs = ltrs [len(ltrs)-1] + ltrs [1:len(ltrs)-1] + ltrs [0]
print('New string is', ltrs)
```

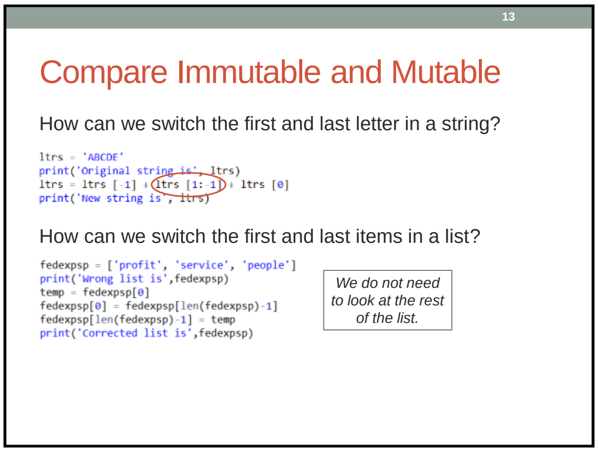
How can we switch the first and last items in a list?

Compare Immutable and Mutable

How can we switch the first and last letter in a string?

```
ltrs = 'ABCDE'
print('Original string is', ltrs)
ltrs = ltrs [-1] + ltrs [1:-1] + ltrs [0]
print('New string is', ltrs)
```

How can we switch the first and last items in a list?



Compare Immutable and Mutable

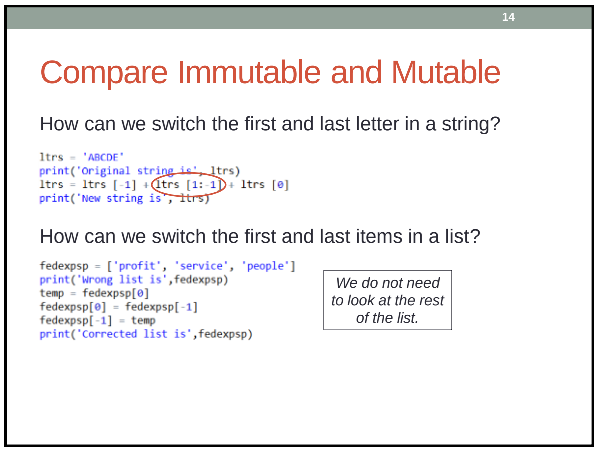
How can we switch the first and last letter in a string?

```
ltrs = 'ABCDE'
print('Original string is', ltrs)
ltrs = ltrs [-1] + ltrs [1:-1] + ltrs [0]
print('New string is', ltrs)
```

How can we switch the first and last items in a list?

```
fedexpsp = ['profit', 'service', 'people']
print('Wrong list is',fedexpsp)
temp = fedexpsp[0]
fedexpsp[0] = fedexpsp[len(fedexpsp)-1]
fedexpsp[len(fedexpsp)-1] = temp
print('Corrected list is',fedexpsp)
```

We do not need to look at the rest of the list.



Compare Immutable and Mutable

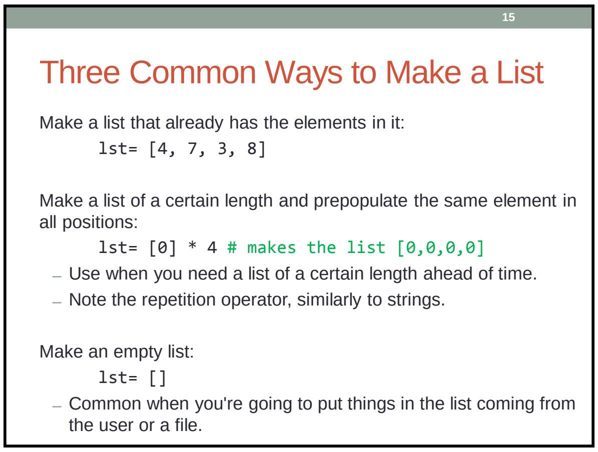
How can we switch the first and last letter in a string?

```
ltrs = 'ABCDE'
print('Original string is', ltrs)
ltrs = ltrs [-1] + ltrs [1:-1] + ltrs [0]
print('New string is', ltrs)
```

How can we switch the first and last items in a list?

```
fedexpsp = ['profit', 'service', 'people']
print('Wrong list is',fedexpsp)
temp = fedexpsp[0]
fedexpsp[0] = fedexpsp[-1]
fedexpsp[-1] = temp
print('Corrected list is',fedexpsp)
```

We do not need to look at the rest of the list.



Three Common Ways to Make a List

Make a list that already has the elements in it:

```
lst= [4, 7, 3, 8]
```

Make a list of a certain length and prepopulate the same element in all positions:

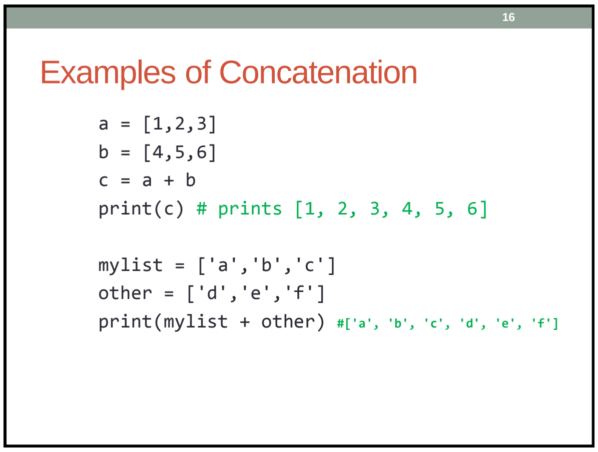
```
lst= [0] * 4 # makes the list [0,0,0,0]
```

- Use when you need a list of a certain length ahead of time.
- Note 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.



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']
```

Simple List Problems

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):
```

Simple List Problems

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):
    lst = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
    lst = lst + ['Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
    print('Month',month,'is',lst[month-1])

getmonth(11)
```

Run

```
Month 11 is Nov
```

Simple List Problems

What does this code do?

```
lst1 = [2] * 3
lst2 = [4] * 2
lst3 = lst1 + lst2
for x in range(0, len(lst3), 2):
   lst3[x] = -1
```

Simple List Problems

What does this code do?

```
lst1 = [2] * 3
lst2 = [4] * 2
lst3 = lst1 + lst2
for x in range(0, len(lst3), 2):
    lst3[x] = -1
print('lst1 is',lst1)
print('lst2 is',lst2)
print('lst3 is',lst3)
```

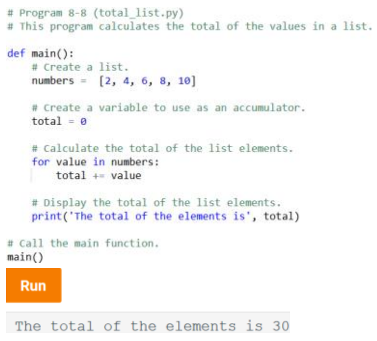
Run

```
lst1 is [2, 2, 2]
lst2 is [4, 4]
lst3 is [-1, 2, -1, 4, -1]
```

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]
```

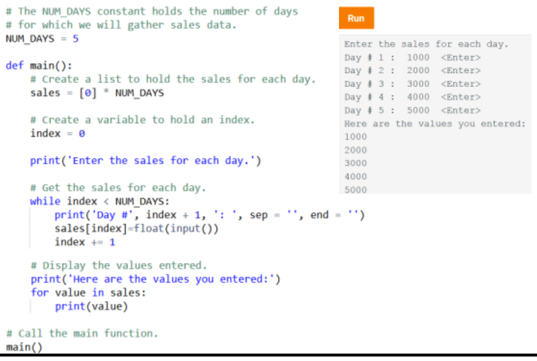
Problem – Total of the Values in a List



```
# Program 8-8 (total_list.py)
# This program calculates the total of the values in a list.

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

    # Create a variable to use as an accumulator.
    total = 0

    # Calculate the total of the list elements.
    for value in numbers:
        total += value

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

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

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

Problem – Total of Sales Data



```
# The NUM_DAYS constant holds the number of days
# for which we will gather sales data.
NUM_DAYS = 5

def main():
    # Create a list to hold the sales for each day.
    sales = [0] * NUM_DAYS

    # Create a variable to hold an index.
    index = 0

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

    # Get the sales for each day.
    while index < NUM_DAYS:
        print('Day #', index + 1, ': ', sep = '', end = '')
        sales[index]=float(input())
        index += 1

    # Display the values entered.
    print('Here are the values you entered:')
    for value in sales:
        print(value)

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

```
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
2000
3000
4000
5000
```

Practice

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.