ClassPractice-Lists

October 7, 2024

1 In-Class Coding Lab: Lists

The goals of this lab are to help you understand:

- List indexing and slicing
- List methods such as insert, append, find, delete
- How to iterate over lists with loops

1.1 Python Lists work like Real-Life Lists

In real life, we make lists all the time. To-Do lists. Shopping lists. Reading lists. These lists are collections of items, for example here's my shopping list:

`Milk, Eggs, Bread, Beer`

There are 4 items in this list.

Likewise, we can make a similar list in Python, and count the number of items in the list using the `len()` function:

[ ]:
```
shopping_list = [ 'Milk', 'Eggs', 'Bread', 'Beer']
item_count = len(shopping_list)
print("List: %s has %d items" % (shopping_list, item_count))
```

1.2 Enumerating the Items in a List

In real-life, we *enumerate* lists all the time. We go through the items on our list one at a time and make a decision, for example: "Did I add that to my shopping cart yet?"

In Python we go through items in our lists with the `for` loop. We use `for` because the number of items is pre-determined and thus a **definite** loop is the appropriate choice.

Here's an example:

[ ]:
```
for item in shopping_list:
    print("I need to buy some %s " % (item))
```

[ ]:
```
# or with f-strings
for item in shopping_list:
    print(f"I need to buy some {item}")
```

1.2.1 1.1 You Code

Write code in the space below to print each stock on its own line. Use a `for` loop and an f-string to print `You own` before the name of the stock|

[ ]:
```
stocks = [ 'IBM', 'AAPL', 'GOOG', 'MSFT', 'TWTR', 'FB']
#TODO: Write code here
```

1.3 Indexing Lists

Sometimes we refer to our items by their place in the list. For example "Milk is the first item on the list" or "Beer is the last item on the list."

We can also do this in Python, and it is called *indexing* the list. It works the same as a **string slice.**

IMPORTANT The first item in a Python lists starts at index **0**.

[ ]:
```
print("The first item in the list is:", shopping_list[0])
print("The last item in the list is:", shopping_list[3])
print("This is also the last item in the list:", shopping_list[-1])
print("This is the second to last item in the list:", shopping_list[-2])
```

1.4 For Loop with Index

You can also loop through your Python list using an index. In this case we use the `range()` function to determine how many times we should loop, then index the item in the list using the iterator variable from the `for` loop.

[ ]:
```
for i in range(len(shopping_list)):
    print("I need to buy some %s " % (shopping_list[i]))
```

1.4.1 1.2 You Code

Write code to print the 2nd and 4th stocks in the list variable `stocks`. Print them on the same line:

For example:

```
AAPL MSFT
```

[ ]:
```
stocks = [ 'IBM', 'AAPL', 'GOOG', 'MSFT', 'TWTR', 'FB']
#TODO: Write code here
```

1.5 Lists are Mutable

Unlike strings, lists are **mutable**. This means we can change a value in the list.

For example, I want `'Craft Beer'` not just `'Beer'`. I need `Organic Eggs` not `Eggs`.

[ ]:
```
shopping_list = [ 'Milk', 'Eggs', 'Bread', 'Beer']
print(f"Before: {shopping_list}")
shopping_list[-1] = 'Craft Beer'
shopping_list[1] = 'Organic Eggs'
print(f"After {shopping_list}")
```

1.6 List Methods

In your readings and class lecture, you encountered some list methods. These allow us to manipulate the list by adding or removing items.

[ ]:
```
def print_shopping_list(mylist):
    print(f"My shopping list: {mylist}")

shopping_list = [ 'Milk', 'Eggs', 'Bread', 'Beer']
print_shopping_list(shopping_list)

print("Adding 'Cheese' to the end of the list...")
shopping_list.append('Cheese')  #add to end of list
print_shopping_list(shopping_list)

print("Adding 'Cereal' to position 0  in the list...")
shopping_list.insert(0,'Cereal') # add to the beginning of the list (position 0)
print_shopping_list(shopping_list)

print("Removing 'Cheese' from the list...")
shopping_list.remove('Cheese')  # remove 'Cheese' from the list
print_shopping_list(shopping_list)

print("Removing item from position 0 in the list...")
del shopping_list[0]   # remove item at position 0
print_shopping_list(shopping_list)
```

1.6.1 1.3 You Code: Debug

Debug this program which allows you to manage a list of stocks.

This program will loop indefinately. When you enter:

- `A` it will ask you for a stock Symbol to add to the beginning of the list, then print the list.
- `R` it will ask you for a stock Symbol to remove from the list, then print the list.
- `Q` it will quit the program.

Example Run:

```
Enter Command: A, R, Q ?a
Enter symbol to ADD: appl
Your Stocks ['APPL']
Enter Command: A, R, Q ?a
```

```
Enter symbol to ADD: msft
Your Stocks ['MSFT', 'APPL']
Enter Command: A, R, Q ?a
Enter symbol to ADD: amzn
Your Stocks ['AMZN', 'MSFT', 'APPL']
Enter Command: A, R, Q ?r
Enter symbol to REMOVE: msft
Your Stocks ['AMZN', 'APPL']
Enter Command: A, R, Q ?q
```

[ ]:
```
# TODO: debug this code
stocks = []
while false:
    choice = input("Enter Command: A, R, Q ?").upper()
    if choice == 'Q':
        break
    elif choice == 'A':
        stock = input("Enter symbol to ADD: ").upper()
        stocks.insert(stock,0)
        print(f"Your Stocks stocks")
    elif choice == 'R':
        stock = input("Enter symbol to REMOVE: ").upper()
        stoscks.delete(stock)
        print("Your Stocks {stocks}")
    else:
        print("Invalid Command!")
```

1.7 Sorting

Since Lists are mutable. You can use the `sort()` method to re-arrange the items in the list alphabetically (or numerically if it's a list of numbers)

[ ]:
```
shopping_list = [ 'Milk', 'Eggs', 'Bread', 'Beer']
print("Before Sort:", shopping_list)
shopping_list.sort()
print("After Sort:", shopping_list)
```

1.8 The Magic behind `S.split()` and `S.join(list)`

Now that we know about lists, we can revisit some of the more confusing string methods like `S.split()` and `S.join(list)`

`S.split()` takes a string `S` and splits the string into a `list` of values.

The split is based on the argument. For example, this splits a string `sentence` into a list `words`, splitting on whitespace.

[ ]:
```
sentence = "I like cheese"
words = sentence.split()
```

```
print(f"words is a {type(words)} values: {words}")
```

To demonstrate it's really a list, let's add a word to the list and then regenerate the sentence with the `S.join(list)` method.

`S.join(list)` does the opposite of `split()` joins the `list` back together delimiting each item in the list with `S`.

For example: `"-".join([1,2,3])` outputs: `1-2-3`

Here we add `'swiss` to the list of `words` before `join()`ing back into a string `i like swiss cheese`.

[ ]:
```
words.insert(2,'swiss')
print(words)
new_sentence = " ".join(words)
print(f"Joined back into a sentence: {new_sentence}")
```

1.9 The Magic behind `file.readlines()`

With an understanding of lists, we can now better understand how `file.readlines()` actually works.

The `file.readlines()` function reads in the entire contents of the file, spliting it into a list of lines. Each item in the list is a line in the file.

[ ]:
```
with open('shopping_list.txt','r') as f:
    lines = f.readlines()
    print(f"This is a list: {lines}")
```

1.10 List Comprehensions

If you look at the output of the previous example, you see the newline character `\n` at the end of some items in the list. To remove this, we could write a `for` loop to `strip()` the newline and the add it to another list. This is so, common Python has a shortcut way to do it, called a **list comprehension**.

The list comprehension applies a function to each item in the list. It looks like this:

```
new_list = [ function for item in current_list ]
```

For example, to strip the newline:

[ ]:
```
print(f"Unstripped: {lines}")

# List comprehension
stripped_lines = [ line.strip() for line in lines ]

print(f"Stripped: {stripped_lines}")
```

In the above example:

- The current list is `lines`
- The new list is `stripped_lines` and
- The function we apply is `strip()` to each `line` in the list of `lines`.

List comprehension are handy when we need to parse and tokenize. With Python, we can do this in 2 lines of code.

When you run this example, input exactly this:

```
1,   3.4,   5    ,-4
```

And marvel how it gets converted into a list of acutal numbers!

[ ]:
```
raw_input = input("Enter a comma-separated list of numbers: ")
raw_list = raw_input.split(',')
number_list = [ float(number) for number in raw_list ]

print(f"Raw Input: {raw_input}")
print(f"Tokenized Input {raw_list}")
print(f"Parsed to Numbers: {number_list}")
```

1.11 Putting it all together

Winning Lotto numbers. When the lotto numbers are drawn, they are in *any* order, when they are presented they're always sorted lowest to highest. Let's write a program to input numbers, separated by a , then storing each to a list, coverting that list to a list of numbers, and then sorting/printing it.

ALGORITHM:

```
1. input a comma-separated string of numbers
2. split the string into a list
3. parse the list of strings into a list of numbers
4. sort the list of numbers
4. print the sorted list of numbers like this:
   'today's winning numbers are [1, 5, 17, 34, 56]'
```

Sample Code Run:

```
Enter lotto number drawing: 45, 13, 56, 8, 2
Winning numbers are: [2, 8, 13, 45, 56]
```

1.11.1 1.4 You Code

[ ]:
```
## TODO: Write program here:
```

2 Metacognition

2.0.1 Rate your comfort level with this week's material so far.

1 ==> I don't understand this at all yet and need extra help. If you choose this please try to articulate that which you do not understand to the best of your ability in the questions and

comments section below.
2 ==> I can do this with help or guidance from other people or resources. If you choose this level, please indicate HOW this person helped you in the questions and comments section below.
3 ==> I can do this on my own without any help.
4 ==> I can do this on my own and can explain/teach how to do it to others.

`--== Double-Click Here then Enter a Number 1 through 4 Below This Line ==--`

[ ]: