MORE ON LISTS

Brief Recap

- Discussion on Lab Problems
- Lists
- Accessing Elements of a List
- Modifying List Elements

Menu for Today!

- Iterating over lists
- Methods for Manipulating Lists
- Methods for Manipulating Strings

Iterating over Lists

- Simple way to iterate through a list is to use index

```
sum = 0
for i in range(0, len(myList)):
    sum = sum + myList[i]
```

- A better way is to use a list iterator

```
sum = 0
for element in myList:
    sum = sum + element
```

Iterating over Lists

- Note the difference
  - *In the first case, we are iterating through indices of the list*
  - *In the second case, we are iterating through the elements of the list*
- Note also:
  - *list elements are indexed from* `0` *to* `len(L) - 1`
  - *range(n) goes from* `0` *to* `n-1`
- In both cases, semantics is the same
- List is an **iterable** objects
  - *We can use* `for` *to go through iterable objects*
  - *A notion of* ***initial*** *item*
  - *A notion of* ***next*** *item*
  - *A notion of* ***done***

List Operators

- + and * work the same as with strings
  - `[1 , 2, 3] + [4, 5]` gives `[1, 2, 3, 4, 5]`
  - `[1, 2, 3] * 3` gives `[1, 2, 3, 1, 2, 3, 1, 2, 3]`
  - *Note that* `+` *and* `*` *create new objects: no aliasing*
- in checks for membership, returns True or False
  - `1 in [1, 2, 3]` gives `True`
  - `4 in [1, 2, 3]` gives `False`
  - `[1, 2] in [1, 2, 3]` gives `False`
  - `[1, 2] in [[1, 2], 3, 4]` gives `True`

Adding Elements to a List

- Use + operator

```
x = [1, 2, 3]
y = [4, 5, 6]
z = x + y
```

z will get the value `[1,2,3,4,5,6]`

- x and y are unchanged

- Use extend or append methods

```
x = [1, 2, 3]
y = [4, 5, 6]
x.extend(y)
```

x will get the value `[1,2,3,4,5,6]`

```
x.append(7)
```

x will get the value `[1,2,3,4,5,6,7]`

- Both `extend` and `append` change the value of x

What is the Dot Doing

- Notice that the syntax for append is list.append(elem) and not append(list, elem)
- This means that append is not a function in the sense that we have studied functions
- Strings and Lists are objects
- Append / Extend are methods to manipulate these objects

More on this in a later class!

More on Methods

- Note the syntax: object.method(x1, …, xn)
  - *The method just like a function with n + 1 arguments*
  - *In addition to the n arguments we specify explicitly, the value of the object itself is also an implicit argument to the function*
- Similarly, we have methods on lists

```
x = [1, 2, 3]

x.append(4)
```

- x is now [1, 2, 3, 4]
  - *Note* x.append *does not return* [1, 2, 3, 4], *it modifies the value of* x *itself*

Changing Money

- In a country, there are notes of seven denominations, namely Rs. 1, Rs. 2, Rs. 5, Rs. 10, Rs. 50 and Rs. 200.
- If an amount Rs N is entered through the keyboard, we need to compute the smallest number of notes that will combine to give Rs N.

```
amount = int(input('How much money do you need: '))
denominations = [200, 100, 50, 10, 5, 2, 1]
change = []
for i in denominations:
    change.append(amount//i)
    amount = amount % i
for i in range(len (change)):
    print(f'You get {change[i]} notes of value {denominations[i]}')
```

Some Useful List Methods

- *Append :* `list.append(element)`
  - *Adds an* `element` *to end of* `list`
- *Extend :* `list1.extend(list2)`
  - *Merges* `list2` *to the end of* `list1`
  - `list1 + list2` *generates a new object*
  - `Extend` *modifies the existing object*
- *Count:* `list.count(elem)`
  - *Returns the number of times* `elem` *occurs in* `list`
- *Index:* `list.index(elem)`
  - *Returns the index of the first occurrence of* `elem` *in* `list`
- *Insert:* `list.insert(index, elem)`
  - *Insert* `elem` *into the* `list` *at* `index`
- *Remove:* `list.remove(elem)`
  - *Remove the first occurrence of* `elem` *from the* `list`
- *Reverse*: `list.reverse()`
  - Reverses the order of the `list`
- `Sort: list.sort(reverse=True|False, key=myFunc)`
  - *Sorts the* `list` *in ascending order by default*

List Functions

- Functions:
  - `len(list)`*: returns the length of a* `list`
  - `max(list)`*: returns the largest element of a* `list`
  - `min(list)`*: returns the smallest element of a* `list`
  - `del list[index]`*: delete the element at* `index` *from* `list`
  - `sorted(myList): returns sorted list, does not mutate myList`
    - `mylist.sort() will mutate the list`

Working With Lists: Summary

- Several ways to work with List data
  - *Operators* (`+`, `*`, `in`)
  - *Methods* (`append`, `extend`, `count`, `index`, …)
  - *Functions* (`len`, `max`, `min`, `del`, `sorted`)

String Methods

- Just as Python has built in methods to manipulate lists, there is a variety of methods to manipulate strings

```
x.capitalize()
x.upper()
x.lower()
x.isalpha()
x.isdecimal()
x.isprintable()
x.center(length, character)
```

Taking Multiple Inputs on a Single Line

```
x,y,z = input().split()
```

```
str.split(sep=None, maxsplit=-1)
```

- *used to break down a bigger string into several smaller strings*
- *returns a list of strings*
- `sep` *is the character where the string is to be broken*
- `maxsplit` *is the number of times a given string can be split up*

```
x,y,z = map(int, input().split())
```

- *More on what map does in a couple of classes*

Formatting Strings

```
x = 32.678
fstring = ”The price is {price:.2f}”
y = fstring.format(price = x)
```

- Format
  - *string.format(value1, value2…) replaces the placeholders in the string with the corresponding values*