More About Lists and a Little About Strings

September 21, 2020

Today’s Topics

- Reminder that Homework #2 is due tonight at Midnight
- Homework #3 is out and is due next Monday, September 28 at Midnight
- Test #1 is scheduled for Wednesday, October 7
  - Monday, October 5 will be a review day
  - The test will be on Blackboard, in real time, from 4 - 5:30 on the 7th
  - There will be a “practice test” on Blackboard about a week beforehand
- Today, more about Lists
- Strings are not lists, but they can sometimes be treated like lists. We’ll talk about that
- Wednesday’s lecture is on For loops

Lists - from Last time

List is a fundamental type in Python. It contains zero or more values

- The elements of a list do not have to be of the same type!!!
  - But you should carefully think about that when writing your program - you can make it a lot harder to understand and debug if you arbitrarily mix types

Declare a new list using square brackets:

- Declare a new empty list: new_empty_list = []
- Declare a new list with initial values: new_grade_list = [98, 94, 152, 93]
-

Add a new value to a list with .append or .insert

Two different methods that apply to a list and return a modified list

- .append puts the new value at the end of the existing list
- .insert puts the new value at the index you specify, and moves everything after that down one place

List Indices and len

- Remember that the first element in a list ALWAYS has index 0. So the element with index 4 is actually the 5th element in the list!!
- len is a function that returns the “length” of a list - the number of elements in the list - this will be an integer
  - The LAST element in the list will have index = len(list)-1
- You can always count indices backward
  - The last item in the list has index -1
  - The next to last has index -2, the third from last has index -3, and so on

Removing, popping, and deleting items from the list

I botched this last time, so let's do it right this time

- To remove an item from the list by its value, use the .remove method
  - It removes the ***first*** item in the list that has the specified value
  - If no item in the list has that value you get an error message and your program crashes
- To remove an item from the list by its index, use the .pop method
  - mylist = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘a’]
    - If you want to remove the first ‘a’, you would use mylist.remove(‘a’)
    - If you want to remove the third item in the list, you would use mylist.pop(2)
      - That's a trick. The third item in the list has index 2!!!

Removing, popping, and deleting items from the list

If you want to remove a bunch of items in the list at once you can use the del function

- del mylist[1 : 4]
  - Pay attention to this if you use del. This deletes the list elements starting at the first number - 1, here - and stopping just before the second number - the 4, here. What was mylist[4] is NOT deleted; it is STILL IN THE LIST
  - If you don’t give a first number, deletion starts at the beginning of the list, with the element with index 0.
  - If you don’t give a last number, deletion goes to the end of the list, with the element with index len(mylist) - 1 or just -1.
  - If you don’t give either number - you just say del mylist [:] - everything is deleted. You still have your variable but it is now an empty list

in - a Boolean function

in - used to determine if a specific value is an element in a list

- Returns True if the value is an element in the list, and False otherwise
- You can use it to stop your program from crashing if you try to remove a value from a list that isn't there
  - mylist = ['a', 'b', 'c', 'd', 'e', 'a']
  - mylist.remove('f') #will cause your program to die a screaming death leaving you with only an error message
    - That not usually desirable behavior from a program
  - Use "in"
    - if 'f' in mylist:
      - mylist.remove('f')
    - else:
      - print("Couldn't remove non-existent value from list")

List slicing (this works for strings, too)

What do you do if you want some elements from a list, but not all of them?

```
states = ["Alabama","Alaska","Arizona","Arkansas","California","Colorado",
"Connecticut","Delaware","Florida","Georgia","Hawaii","Idaho","Illinois",
  "Indiana","Iowa","Kansas","Kentucky","Louisiana","Maine","Maryland",
  "Massachusetts","Michigan","Minnesota","Mississippi","Missouri","Montana",
  "Nebraska","Nevada","New Hampshire","New Jersey","New Mexico","New York",
  "North Carolina","North Dakota","Ohio","Oklahoma","Oregon","Pennsylvania",
  "Rhode Island","South Carolina","South Dakota","Tennessee","Texas","Utah",
  "Vermont","Virginia","Washington","West Virginia","Wisconsin","Wyoming"]
```

How do you get the first ten states? States 21 - 30? The last 20?

You can “slice” the list using subscripts

Examples of slicing a list

```
print(states[:10])
```

['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut', 'Delaware', 'Florida', 'Georgia']

```
print(states[20:30])
```

['Massachusetts', 'Michigan', 'Minnesota', 'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire', 'New Jersey']

```
print(states[-20:])
```

['New Mexico', 'New York', 'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington', 'West Virginia', 'Wisconsin', 'Wyoming']

This is called “slicing” the list, or taking a “slice”. Here’s how it works:

- Give the list variable name, then square brackets hold the subscript(s) of the element(s) you want
- Separate the first subscript from the second with a colon :
- If there is no first subscript, start at the beginning. If there is no last subscript, go to the end
- Negative numbers can be used in subscripts

- The first element you get is the first subscript. Remember that you start counting at 0! The element at subscript 20 is actually the 21st element in the list.
- The second subscript is where you stop. You do not get element 30 (the 31st element in the list)

Now, About Those Strings

Strings are NOT Lists!!

Strings are sets of zero or more characters that are treated as unitary items

BUT - you can do some of the same things to strings that you can do to lists

- But not everything!! We'll be dealing with some of those details throughout the semester
- For now
- You can get the value of an particular character or set of characters from a string by using an index, the same as you can with a list
  - Just like with lists, you start counting indices at 0
  - stringvar = 'abcde'
  - stringvar[0] = 'a'
  - stringvar[1:3] = 'bc'

Other functions and methods for strings

- Does ‘in’ work?
  - ‘a’ in stringvar # yep
- Do .append or .insert work?
  - Nope - not allowed
- How about remove, pop, or del?
  - Nope - not allowed

String operations

Split - splits a string up into a list of its component parts - e.g., a sentence into words

“Hey diddle diddle the cat and the fiddle” becomes [“Hey”, “diddle”, “diddle”, “the”, “cat”, “and”, “the”, “fiddle”]

Strip - removes leading and/or trailing white space

“   Kansas City 31 San Francisco 20   “ becomes “Kansas City 31 San Francisco 20”

Splitting Strings

A method to break up a string into its component parts

- A method operates on the object to which it is linked

Can split on any character or position Python can recognize, but the default is to split on whitespace

“Whitespace” = spaces, tabs, newlines, formfeeds

Examples of splitting strings

```
str = “Hey diddle diddle the cat and the fiddle”
str_list = str.split() #this will split on whitespace, and the whitespace will be deleted
str_list = str.split(“e”) #what does this do? What happened to the “e” ?
int_str = “3 1 4 5 6 7 8 9 10”
int_list = int_str.split()
actual_int_list = []
for num in int_list:
        actual_int_list.append(int(num))
```

Strip

Removing whitespace from a string - get rid of leading and/or trailing tabs, spaces,...

str.lstrip() - gets rid of whitespace on the left side of the string; before any other characters

str.rstrip() - gets rid of whitespace on the right side of the string; after the last other character

str.strip() - gets rid of whitespace on both the left and the right sides

Note that this does not get rid of blank spaces between words in a string