More on Lists and Some on Strings

CMSC 201 Section 60
September 28, 2024

Administrative Notes

Homework 2 is due next Monday

Getting items out of your lists: “pop” and “delete”

On Monday we learned about “remove”

```
grocery_list = [“milk”, “eggs”, “bread”]

grocery_list.remove(“milk”)
```

- Removes the item that matches the value - e.g,. Removes “milk” from the list
- ONLY removes the FIRST matching item, starting at index 0 and working up
- Results in an error if you try to remove an item that isn’t there

What if we want to remove an item by it’s index? E.g., remove the item in index 1

What if we want to remove multiple items at once?

That’s where “pop” and “del” come in

“pop” - removes the item at the specified index, and automatically shifts every item after that up one space.

```
grocery_list.pop(3)
```

“del” - removes all the items starting at the start index, and ending at the element before the end index (keeps the element at the end index in the list). Automatically shifts following elements up

```
del grocery_list[3:5]
```

List slicing

List slicing

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 s = “”

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
    - That “mutability” problem.
    - You can’t change the value of a string variable
      - You can re-assign it, but that’s it
- How about remove, pop, or del?
  - Nope - not allowed
    - Mutability. You can’t do anything that would change the value of a string

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

String slicing

- Like list slicing - take a “slice” or a part of each string by specifying the indices of the characters - the string elements - you want

  starting_string = “Today is President’s Day but only some people observe it”

- Let’s take some slices:

Other useful things to know:

Escape sequences:

\n a newline character - causes a newline to be printed on the screen

\t a tab character

\’ accepts an apostrophe, rather than a string delimiter

\\ prints a backslash character

- this turns out to be vital in Windows systems, because Windows uses the backslash \ as the path delimiter when you’re identifying a file - \\ is the only way to read a Windows file system in Python

Linux and Mac use the forward slash / as the path delimiter