Sequences: Lists and Strings

CMSC 104 Section 2
September 30, 2025

Administrative Notes

What you should take away from this lecture

Strings

- Treating characters individually
- Manipulating strings
- Built-in methods
  - len, upper, lower, …
- Indexing
  - Emphasize that you always start counting at 0
  - You can use negative index values to start counting from the end instead of the beginning
- Slicing
- Concatenating
- Repeating via “multiplication”

Lists

- Multi-valued data type
  - Mutable; can change individual list values
- Adding values (append, insert)
- Removing values (remove, del)
- len
- Indexing
  - You always start counting at 0
  - You can use negative index values to start counting from the end instead of the beginning
- Often used with a different type of “for” loop that processes each list element exactly once
- List elements don’t have to be the same type
- You can have lists of lists

Strings

See the file “strings.py” on Github

String operations

Method	What it does
.upper()	Converts each letter in the string to uppercase
.lower()	Converts each letter in the string to lowercase
.title()	Converts the entire string to Title Case
.strip()	Removes leading and/or trailing blank spaces
.split()	Splits a string into a list of strings, and removes all instances of the character(s) in the method
.replace()	Replaces all instances of a substring with a new substring

Lists

See the file “lists.py” on GitHub

Lists

- Fundamental “Pythonic” data structure
- Similar to strings, but different
- 0 or more elements
- Start indexing at 0
- Built-in methods & functions
  - len,...
- List slicing
- Not all elements have to be the same type
- Lists of lists

List Operations

Method	What it does
append	Adds one new element to the end of the list
remove	Removes the first instance of a given value from the list - crashes if that value is not in the list
insert	Adds one new element to the list at the designated index spot
del	Removes an element from the list by index
len	Returns the number of elements in the list - note that the last element is always one less than the len
join	Combines elements of a list of strings and returns a single string consisting of those elements separated by the designated character
Combining lists w/ +	Creates a new list consisting of the elements of each original list, in the order specified

Why lists are different than strings

Mutability - ability to directly change a value

- Lists are mutable - you can modify them by inserting, deleting, or changing one element of the lists
- Strings are immutable - you can’t change them directly - by inserting, deleting, or changing a letter
  - You have to make the change indirectly - by assigning a new value to the string variable
- If we have time:
  - How this is handled in memory

A “for each” loop

Previously, we used what are called “for i” loops - go through a loop based on the value of an index variable

Suppose we have a list, and we want to do something with each element in the list, exactly one time

There’s a simpler “for” loop, called “for each”

Syntax:

```
for var in list_var:

  #do whatever you want as part of the loop body
```