Strings, Lists, and Files

Professor Hugh C. Lauer

CS-1004 — Introduction to Programming for Non-Majors

(Slides include materials from *Python Programming: An Introduction to Computer Science*, 2nd edition, by John Zelle and copyright notes by Prof. George Heineman of Worcester Polytechnic Institute)

Reading Assignment

- **Chapter 5:– Sequences**
  - I.e., Strings, Lists, and Files

What is a String?

- **String *literal:–***
  - Any piece of text enclosed in *matching* quotes
  - May be single or double quotes

- **“This sentence, of course, is written in English.”**
- **‘Esta frase se escrita en español.’**
- **‘Wir können auch in Deutsch und Französisch zu schreiben.‘**
- **“En fait, nous pouvons écrire les chaînes en Python en japonais, aussi.”**
- **‘Pythonはあっても、私たちは日本語などのアジアの言語で書くことができます。’**

- **Anything that you can type in *ANY* language can be represented and stored as a string in Python!**
  - Unicode standard

String literal (continued)

- **A string literal is a constant, …**
- **… just like a numerical value**
- **Example:–**

```
S = 'This is a string!'
```

- **The name S refers to the value ‘This is a string!’**
  - Without the quotes
- **Quotes must be paired**
  - Either single or double

Operations on Strings

- **+ — concatenation**

```
S = "This is a string"
T = "and this is a another string"
U = S + ', ' + T + '.'
```

  - What is the value of **U**?

- *** — repetition**

```
Y = "yuck, yuckity, "
Y * 5
```

- **[] — indexing**
  - Like lists

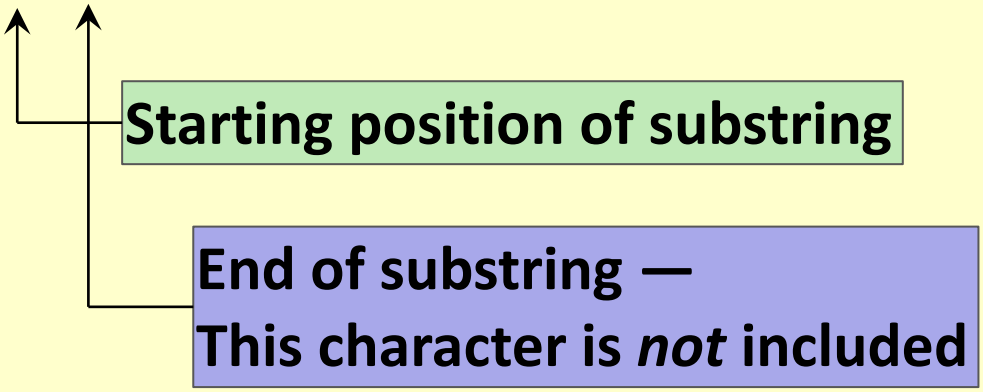
```
S[0]
T[1]
U[2]
U[-1]
U[-2]
```

Note negative indexing for lists and strings counts from end!

Operations on Strings (continued)

- **Slicing**
  - I.e., taking a subset of a string

```
S = "This is a string"
T = S[5:7]
```

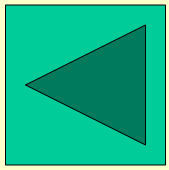


- **Special cases**

```
U1 = S[-7:-1]
U2 = S[-7:]
U3 = S[:-7]
...
```

Operations on Strings (continued again)

- **`len(S)` — length of string in characters**



- **`for c in S:`**
  - Iterate thru the characters of the string S
- **`ord('c')`**
  - Get the Unicode character number of c
  - 'c' *must* be a string of length 1
- **`chr(n)`**
  - Return a string containing a single character, the **`ord`** of which is integer **`n`**

More about substrings and slicing

- **Simple slices:–**
  - `S[2:3]`
  - `S[5:-1]`
  - `S[7:]`
- **Slices with *strides***
  - `S[:-1:2]` # selects alternate characters
  - `S[1: :3]` # selects every third character
- **Cannot find a way to extract specific characters from a string**
- **I.e.,**
  - `S[1, 3, 7]` #Does not work in Python 3
  - There are deep, intricate methods for slicing
  - but beyond scope of this course

Useful string methods

- **s.capitalize()**
- **s.count(sub)**
  - Count occurrences of a substring
- **s.find(sub)**
  - Find a substring
- **s.strip([chars])**
  - Removes characters from *beginning* and *end* of string.
  - Defaults to white space
- **s.replace(old, new)**
  - Replaces instances of old substring with new substring
- **… # a zillion more**
  - Some later in the course
- **All return a *new* string**
  - I.e., a modified copy of the original
  - The methods do *not* update original strings

Lists have methods, too!

- **L.append()**
  - Our friend
- **L.sort()**
  - Sorts elements *in place*. May be high-to-low or low-to-high
- **L.reverse()**
  - Reverses the list
- **L.index(x)**
  L.count(x)
  - Returns index of first occurrence of x in list or count of x's in list
- **L.pop(i)**
  L.remove(x)
  - Removes *ith* element (pop) or first occurrence of x
- **See p. 345, also *Python documentation***

Reminder about lists and strings

- **A list can be updated in place!**
  - L1.append adds to end of L1
- **Assignment creates another name for same list!**
  - **`M1`** = **`L1`** $\Rightarrow$ **`L1`** and **`M1`** are same list
  - Changes to one are visible in other
- **A string can never be modified!**
  - All methods return entirely new string
  - (Partial) copy of original

Slicing lists

- **Simple slices:–**
  - `L[2:3]`
  - `L[5:-1]`
  - `L[7:]`
- **Slices with *strides***
  - `L[:-1:2]` # selects alternate list items
  - `L[1: :3]` # selects every third item
- **Special note about slicing lists:–**
  - `L[start:end:stride]` # creates a *new list*
  - # Same members as old list
  - … but a separate, list
- **E.g.,**
  - `M = L[ : ]` # creates *clone* of **L**, assigns to **M**
- **Counter-intuitive**
  - Based on what we know about assignment of lists!

Questions?

Definition — File

- **A (potentially) large amount of information that lives a (potentially) very long time**
- **May be (much) larger than the amount of RAM in your computer**
- **(Usually) expected to outlive the running of your program**
- **(May be) expected to outlive the computer itself!**
- **Stored on**
  - Hard drive
  - Flash drive
  - Spread out across multiple disks
  - Somewhere in the “cloud”
  - On some other medium
  - …

The rest of the *File* topic is postponed to Homework #5