Python Exercises, Part 2: String Manipulation!

Eckel, TJHSST AI1, Fall 2020

Background & Explanation

For the second half of your Python exercises, we'll be focusing on string manipulation. The following pointers still apply:

- Refer often to the list of built-in commands (https://docs.python.org/3/library/functions.html) or Google it!
- You may NOT import ANYTHING besides "sys" for this assignment. Plenty of that coming later, but for now we're just learning basics!

Required Work

Python makes available a number of helpful string manipulation commands which you should be familiar with. To make sure you learn them all, you are required to know how to accomplish each of the following tasks **in only one line of code each**. If that seems impossible, you probably need to search "Python string slicing" and/or "Python string join".

This can all be done with one .py file.

First, write this line of code at the top of your program:

```
s = sys.argv[1]
```

*Then, assume that the console input makes sense (ie, the string is long enough) and write a **one-line expression** for each problem that prints to the console…*

1) …the character at position 2. (Which I don't need to remind you is *not* the second character.)
2) …the fifth character. (Which I don't need to remind you is *not* the character at position 5.)
3) …the number of characters in the string.
4) …the first character.
5) …the last character.
6) …the penultimate character.
7) …the five character long substring starting at position 3.
8) …a substring consisting of the last five characters of the string.
9) …a substring starting at the third character and continuing to the end of the string.
10) …a string containing every other character from the input string.
11) …a string consisting of every third character from the input string, starting from its second character.
12) …the input string reversed. (One line!)
13) …the position of the first space in the input string.
14) …the string shifted to the right by one (ie, the original string with the last character removed).
15) …the string shifted to the left by one (ie, the original string with the first character removed).
16) …the string all in lower case.
17) …a list of all the space delimited substrings of the input string. Examples:
    `"234"` ➔ `["234"]`
    `"12 35"` ➔ `["12", "35"]`
    `"The  quick    fox"` ➔ `["The", "quick", "fox"]`
18) …the number of space delimited words there are in your input string.
19) …a list of all characters, including duplicates, in the string
    (eg, `"foo"` ➔ `["f", "o", "o"]`).
20) …a new string consisting of the characters of the input string rearranged in ascending ascii order (eg, `"quick"` ➔ `"cikqu"`).

21) …a new string consisting of the substring of your input string starting at the beginning and going up to, but not including, the first space. If there is no space at all, it should give the entire input.
22) …whether or not the input string is a palindrome. (For this question, you can assume the input string is only lower case letters with no grammatical symbols.)

Sample Run

NOTE: With ANY assignment like this, you should also **make your own runs** and compare your output to other students' (or output you work out by hand) to make sure that your program is catching all the details. For instance, in the sample run below, #22 outputs "True". If you write incorrect code that *always* outputs "True", you won't be able to tell unless you try other sample output as well. The sample run here is a good way to check your progress, but *not sufficient* by itself!

```
Enter input: abcde  edcba
#1: c
#2: e
#3: 12
#4: a
#5: a
#6: b
#7: de  e
#8: edcba
#9: cde  edcba
#10: ace db
#11: beeb
#12: abcde  edcba
#13: 5
#14: abcde  edcb
#15: bcde  edcba
#16: abcde  edcba
#17: ['abcde', 'edcba']
#18: 2
#19: ['a', 'b', 'c', 'd', 'e', ' ', ' ', 'e', 'd', 'c', 'b', 'a']
#20:   aabbccddee
#21: abcde
#22: True
```

Specification

Submit your Python file to the link given on the course website.

This assignment is **complete** if:

- Every problem is either **solved** or **prints a written explanation to the console** explaining why it doesn't work / how you're stuck.
- Output for each problem **clearly indicates** which question goes with which line of output (see sample output above for example).
- The "Name" field on the Dropbox submission form contains your **class period**, then your **last name**, then your **first name**, in that order.

For **resubmission**:

- Since answers will be given in class, solving the problems is insufficient for a resubmission, so instead you will need to **copy the correct answers and write a comment under each one explaining how it works** (in other words, pretend someone wasn't in class when I went over it and write explanations to them for each solution).