Python Exercise 5: String Manipulation and List Comprehensions

These exercises will teach you about various techniques to manipulate strings using the many built-in methods of string objects, as well as list comprehensions for efficient coding.

# String Manipulation

1. Create two strings with variables forename and surname, and initialise them with your own name.
2. Add forename and surname to each other and place the result into a third variable, fullName. (You will be using this variable through the exercise, so make sure to retain it.)
3. What is the commonly used word for what you did in 2.?
4. Alter your answer in 2. to tell Python to insert a space between the forename and surname.
5. A string can be regarded as a list. What is meant by this statement?
6. Given that a string is a list , write the syntax that would return the third character from your surname.
7. Similarly, extract the first character of both your forename and surname variables and concatenate them into a new variable called initials.
8. Create a variable holding the number 123 and write the syntax that converts this number into a string.
9. Create a variable called username that is the concatenation of your forename and the number 123, making sure to convert this number to a string. Do this one line of code.
10. An alternative way to concentrate strings is to build a string using placeholders. Run the following line and explain what has happened.

fullName = “{0} {1}”.format(forename, surname)

1. Put your forename, age and school/college into variables and build a sentence explaining who you are, how old you are and where you went to school, by using placeholders. Print this sentence.
2. Print the length of fullName.
3. Use the count() function to print the number of occurrences of the letter “e” in your full name (or a letter of your choice).
4. Use the find() or index() function print the location of the first letter “e” in your name (or a letter of your choice).
5. Print the first three characters of fullName.
6. Print the third to sixth characters of fullName, inclusive.
7. Print the final five characters of fullName.
8. Use the split() function to split fullName back into its component names, using the space character as the delimiter.
9. Experiment with the lower(), upper() and title() functions to change the case of fullName.
10. Similarly, experiment with the capitalize() and swapcase() functions.
11. Use the replace() function to replace all spaces in fullName with hashes (#) and then swap them back.
12. Append some spaces to the front and end of fullName, then strip them out using the strip() function. Also experiment with lstrip() and rstrip().
13. Use the list() function to explicitly convert fullName into a list of characters.
14. Use the join() function to join the characters of fullName back, but this time separated by dashes (-). (After you’ve done this, remove the dashes using the replace() function.)

# List Comprehensions

1. Create a list of the numbers 1 to 10 inclusive, i.e. nums = list(range(1, 11)). Now execute the following instruction. What do you think this is doing?

nums = [2 \* i for i in nums]

1. List comprehensions, such as that above, allow for code vectorisation. Research what this means.
2. Create another range of numbers from 1 to 10, and use a list comprehension to create and another variable containing their squares. (E.g. 1 becomes 1, 2 becomes 4, 3 becomes 9, and so on.) Recall that the exponentiation operator is \*\* in Python.
3. Use a list comprehension to put each letter of fullName into uppercase.
4. Type the following, run it and explain what it is doing:

list1 = [1, 2, 3, 4, 5]

list2 = [6, 7, 8, 9, 10]

list3 = [i + j for i in list1 for j in list2]

print(list3)

1. Type the following and explain what is happening.

list1 = [6, 1, 9, 8, 2, 7]

list2 = [2 \* i for i in list1 if i > 5]

1. Create a range of numbers 1 to 100, and another list containing only Boolean values that correspond to whether the respective value in the first list is even (True) or odd (False).
2. Copy and paste a paragraph of text of your choice into a Python string variable, then use a list comprehension and any relevant string manipulation functions to extract the first character of each word and join those character together into one string.
3. Similar to the previous question, take your paragraph and reverse the characters in each and every word, while keeping the order of words the same. You could create a simple cipher using this technique. See if you can do some more complex manipulation to achieve this.

# Answers

1. forename = “Neil”

Surname = “Buckley”

1. fullName = forename + surname
2. concatenation is the joining of strings, characters and/or numbers together.
3. fullName = forename + “ “ + surname
4. Python simply views a string as a list of characters joined together in order.
5. surname[2]
6. initials = forename[0] + surname[0]
7. num = 123

str(num)

1. username = forename + str(123)
2. {0} and {1} and ordered placeholders that are replaced with the variables in the argument list of the format() function.
3. forename = “Shreya” ; age = 23 ; school = “Lancers Convent”

print(“My name is {0}, I am {1} years old, and I went to {2}.”.format(forename, age, school))

1. print(len(fullName))
2. print(fullName.count(“e”))
3. print(fullName.find(“e”)) (note that fullName.index(“e”) does the same thing)
4. print(fullName[0:3]) or fullName[:3]
5. print(fullName[3:7])
6. print(fullName[-5:])
7. fullName.split(“ “)
8. print(fullName.upper())

print(fullName.lower())

print(fullName.title())

(note that title()puts each letter at the start of a word in the string into upper case and the remainder of the letters into lower case)

1. capitalize() capitalizes the first letter only of the string, and swapcase() swaps all character cases.
2. fullName = fullName.replace(“ “, “#”) ; fullName = fullName.replace(“#“, “ ”)
3. fullName = “ {0} “.format(fullName)

fullName = fullName.strip()

1. fullName = list(fullName)
2. fullName = “-“.join(fullName)

fullName = fullName.replace(“-“, “”)

1. This is a list comprehension that iterates through each element in the list, aliases it into a counting variables (in this case, i) and multiplies it by 2.
2. Code vectorisation is a technique in programming that avoids loops and performs calculations much faster.
3. squares = [i \*\* 2 for i in nums]
4. fullName = [upper(i) for i in fullName]

1. This is double list comprehension, which executes the operation on every element against every other element.

1. This is a conditional list comprehension applying the calculation only to those elements for which the condition at the end returns true, i.e. only numbers in the list that are greater than 5.
2. nums = list(range(0, 1001))

isEvens = [i%2==0 for i in nums]

1. paragraph = “[some long paragraph of your choice]”

words = paragraph.split(“ “)

firstLetters = [i[0] for i in words]

firstLetters = “”.join(firstLetters)

1. This is a very complex question that we will go through at some stage.