Q1. In Python 3.X, what are the names and functions of string object types?

Python string is a sequence of Unicode characters that is enclosed in quotation marks. In this article, we will discuss the in-built function i.e. the functions provided by Python to operate on strings.

The below functions are used to change the case of the strings.

lower(): Converts all uppercase characters in a string into lowercase

upper(): Converts all lowercase characters in a string into uppercase

title(): Convert string to title case

swapcase(): Swap the cases of all characters in a string

capitalize(): Convert the first character of a string to uppercase

Q2. How do the string forms in Python 3.X vary in terms of operations?

Python does not support a character type; these are treated as strings of length one, thus also considered a substring.

To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring

You can "update" an existing string by (re)assigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether.

Q3. In 3.X, how do you put non-ASCII Unicode characters in a string?

Transliteration is a process of writing the word of one language using similarly pronounced alphabets in other languages. It deals with the pronunciation of words in other languages. Similarly, in computer language, the computer can handle ASCII characters but has problems with non-ASCII characters. There are some times when we are unable to skip non-ASCII characters as it can lead to loss of information.

Q4. In Python 3.X, what are the key differences between text-mode and binary-mode files?

There are mainly two types of data files — text file and binary file. A text file consists of human readable characters, which can be opened by any text editor. On the other hand, binary files are made up of non-human readable characters and symbols, which require specific programs to access its contents.

Q5. How can you interpret a Unicode text file containing text encoded in a different encoding than your platform's default?

Handling character encodings in Python or any other language can at times seem painful. Places such as Stack Overflow have thousands of questions stemming from confusion over exceptions like UnicodeDecodeError and UnicodeEncodeError. This tutorial is designed to clear the Exception fog and illustrate that working with text and binary data in Python 3 can be a smooth experience. Python’s Unicode support is strong and robust, but it takes some time to master.

Q6. What is the best way to make a Unicode text file in a particular encoding format?

The default mode for the open() function is to read text files: mode = 'r'.

But you can use the open() function to write files, too. Simply set the mode to write: mode = 'w'

open('a-new-file.txt', mode='w', encoding='utf-8')

To write something to this newly opened text fle, you can use the .write() method.

open('a-new-file.txt', mode='w', encoding='utf-8').write('I just wrote this to a text file. Alright!')

If we read this newly created text file, we can see that the .write() method worked correctly:

open('a-new-file.txt', mode='r', encoding='utf-8').read()

Q7. What qualifies ASCII text as a form of Unicode text?

SCII encodes symbols, digits, letters, etc., whereas Unicode encodes special texts from different languages, letters, symbols, etc.

It can be said that ASCII is a subset of the Unicode encoding scheme.

Q8. How much of an effect does the change in string types in Python 3.X have on your code?

In Python, individual characters of a String can be accessed by using the method of Indexing. Indexing allows negative address references to access characters from the back of the String, e.g. -1 refers to the last character, -2 refers to the second last character, and so on.

String1 = "GeeksForGeeks"

print("Initial String: ")

print(String1)

# Printing First character

print("\nFirst character of String is: ")

print(String1[0])

# Printing Last character

print("\nLast character of String is: ")

print(String1[-1])