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

Some of the names and functions of String object types of python 3.X are:

1. Capitalize(): To capitalize first letter of string only
2. Lower(): To convert all the letters of string to lowercase
3. Upper(): To convert all the letters of string to uppercase
4. Count(): To count the number of particular values in a string
5. Find(): To find the index number of the value in a string

And there are many more methods of the string manipulation in python 3.X

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

The string formed in python 3.X vary in terms of usability, operations to be performed. Earlier before defining a string in python we have to use the symbol “str” before hand while from python 3 we can simply define the python string by just using the inverted commas. Also, the number of operations in python 3.X are much more for manipulation and changing of rest of the things.

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

We can use ASCII characters by using escape(‘/’) before the character to print these characters.

For example: /’ 🡪 used to print **’**

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

The difference between text and binary files is that the text files consists of letters, alphabets, characters, numbers, symbols, etc. while the binary mode files are just the compiled version of the text file.

The characters in the text files are human readable while in binary are not human readable format.

Errors can be detected and rectified in the text files while the errors cannot be even seen in binary files

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

We can convert the file to **utf-8** format which is becoming extremely popular these days. This format is basically used for having the same type format to be able to run in different platforms.

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

The best way to make a Unicode text file in a particular encode format is by using the following command:

File.encode(“<encoding format>”)

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

The ASCII represents charaters from A-Z, a-z, numbers from 0-9, and some symbols which makes it qualify as a form of text

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

There will be no effect on the code as in python 3.X these things have been improved and all the code just runs on Unicode.