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

Ans. Various type of names and functions of string objects in python are:

<string>.capitalize() Converts the first character of the string to a capital (uppercase) letter

<string>.casefold() Implements caseless string matching

<string>.center() Pad the string with the specified character.

<string>.count() Returns the number of occurrences of a substring in the string.

<string>.encode() Encodes strings with the specified encoded scheme

<string>.endswith() Returns “True” if a string ends with the given suffix

<string>.expandtabs() Specifies the amount of space to be substituted with the “\t” symbol in the string

<string>.find() Returns the lowest index of the substring if it is found

<string>.format() Formats the string for printing it to console

<string>.format\_map() Formats specified values in a string using a dictionary

<string>.index() Returns the position of the first occurrence of a substring in a string

<string>.isalnum() Checks whether all the characters in a given string is alphanumeric or not

<string>.isalpha() Returns “True” if all characters in the string are alphabets

<string>.isdecimal() Returns true if all characters in a string are decimal

<string>.isdigit() Returns “True” if all characters in the string are digits

<string>.isidentifier() Check whether a string is a valid identifier or not

<string>.islower() Checks if all characters in the string are lowercase

<string>.isnumeric() Returns “True” if all characters in the string are numeric characters

<string>.isprintable() Returns “True” if all characters in the string are printable or the string is empty

<string>.isspace() Returns “True” if all characters in the string are whitespace characters

<string>.istitle() Returns “True” if the string is a title cased string

<string>.isupper() Checks if all characters in the string are uppercase

<string>.join() Returns a concatenated String

<string>.ljust() Left aligns the string according to the width specified

<string>.lower() Converts all uppercase characters in a string into lowercase

<string>.lstrip() Returns the string with leading characters removed

<string>.maketrans() Returns a translation table

<string>.partition() Splits the string at the first occurrence of the separator

<string>.replace() Replaces all occurrences of a substring with another substring

<string>.rfind() Returns the highest index of the substring

<string>.rindex() Returns the highest index of the substring inside the string

<string>.rjust() Right aligns the string according to the width specified

<string>.rpartition() Split the given string into three parts

<string>.rsplit() Split the string from the right by the specified separator

<string>.rstrip() Removes trailing characters

<string>.splitlines() Split the lines at line boundaries

<string>.startswith() Returns “True” if a string starts with the given prefix

<string>.strip() Returns the string with both leading and trailing characters

<string>.swapcase() Converts all uppercase characters to lowercase and vice versa

<string>.title() Convert string to title case

<string>.translate() Modify string according to given translation mappings

<string>.upper() Converts all lowercase characters in a string into uppercase

<string>.zfill() Returns a copy of the string with ‘0’ characters padded to the left side of the string

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

Ans. In Python 3.X, default format of strings is Unicode.

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

Ans. In Python 3.x unidecode() method from unidecode library can be used to put non-ASCII Unicode Characters in a string.

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

Ans. When we try to read or write files in our program, usually there are two modes to use. Text mode, usually by default, and binary mode. Obviously, in text mode, the program writes data to file as text characters, and in binary mode, the program writes data to files as 0/1 bits.

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

Ans. Use of encode() and decode() method can be used to you interpret a Unicode text file containing text encoded in a different encoding than your platform's default, by default encoding parameter is UTF-8.

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

Ans. Use str.encode() and file.write() to make a Unicode text file in a particular encoding format, default encoding format is UTF-18.

Call str.encode(encoding) with encoding set to utf8 to encode str.

Call open(file, mode) to open a file with mode set to wb . wb writes to files in binary mode & preserves UTF-8format.

Call file.write(data) to write data to the file.

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

Ans. Unicode represents most written languages in the world. ASCII has its equivalent in Unicode.

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

Ans. Python 3.X stores strings as Unicode by default whereas Python 2 requires you to mark a string with a u if you want to store it as Unicode. Unicode strings are more versatile than ASCII strings, which are the Python 3.X default, as they can store letters from foreign languages as well as emoji and the standard Roman letters and numerals.