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

Ans:

- *<stirng>.isdecimal() -> Returns True if all characters in a string are decimal.*
- *<string>.isalnum() -> Returns True if all characters in the string are AlphaNumeric.*
- *<string>.istitle() -> Returns True if first character in a string is in Uppercase.*
- *<string>.partition(<sub_string>) -> Splits string at first occurance of sub string and returns a tuple of 3 elements.*
- *<string>.rpartition(<sub_string>) -> Splits string at last occurance of sub string and returns a tuple of 3 elements.*
- *<string>.isidentifier() -> Returns True if give string is a valid identifier name.*
- *len(<string>) -> Returns the length of the given string.*
- *<string>.index(<sub_string>) -> Returns the lowest index of substring if substring is found in the string.*
- *<string>.rindex(<sub_string>) -> Returns the highest index of substring if substring is found in the string.*
- *max(<string>) -> Returns the highest Alphabetical Character in the string as per ASCII.*
- *min(<string>) -> Returns the lowest Alphabetical Character in the string as per ASCII.*
- *<string>.splitlines() -> Returns a list of lines in the string.*
- *<string>.capitalize() -> Returns the string with first character capitalized.*
- *<string>.upper() -> Returns the string with all characters in uppercase.*
- *<string>.lower() -> Returns the string with all characters in lowercase*
- *<string>.casefold() -> Returns the string in lowercase which can be used for caseless comparisions.*
- *<string>.expandtabs(no_of_spaces) -> Replaces tabs in a string with specified no of spaces default is 8*
- *<string>.find(<sub_string>) -> Returns lowest index of substring if substring is found in the string else returns -1.*
- *<string>.rfind(<sub_string>) -> Returns highest index of substring if substring is found in the string else returns -1.*
- *<string>.count(<char>) -> Returns the no of occurances of the char in the given string.*
- *<string>.split(<sep>) -> Returns list of words seperated by given sep else seperated by whitespace.*
- *<string>.rsplit(<sep>) -> Returns list of words seperated by given sep else seperated by whitespace scanning from end.*
- *<string>.lstrip() -> Returns a copy of where leading whitespaces are removed.*
- *<string>.rstrip() -> Returns a copy of where trailed whitespaces are removed.*
- *<string>.strip() -> Returns a copy of where both leading and trailing whitespaces are removed.*
- *<string>.swapcase() -> Swaps lowercase characters with uppercase and vice versa.*
- *<sep>.join(<list>) -> Concatenates a list or tuple of words with intervening occuernces of sep.*
- *<string>.translate(<mapping_table>) -> translates the characters using table.*

- *<string>.maketrans(<dict>) -> Creating a mapping translation tbale usable for <string>.translate(<mapping_table>)*
- *<string>.replace(<char_1>,<char_2>) -> Replace all occurances of char_1 with char_2 in string.*
- *<string>.encode() -> Encodes string into any encoding supported by python. Default encoding is UTF-8.*
- *<string>.ljust(<no_of_spaces>) -> Left-justify in a field of given width.*
- *<string>.rjust(<no_of_spaces>) -> Right-justify in a field of given width.*
- *<string>.center(<no_of_spaces>) -> Center-justify in a field of given width.*
- *<stirng>.zfill(<length>) -> Zfill adds zeros to the begining of string until the specified length is reached.*

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

Ans: Default format of strings is Unicode in Python 3.X whereas in Python2 we need to explicitly mention Unicode value using u.

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

Ans: unidecode() method can be used to put non-ASCII Unicode Characters in a string.

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

Ans: The major difference between text-mode and binary-mode files is that a text file contains textual information in the form of alphabets, digits and special characters or symbols and a binary file contains bytes or a compiled version of a text file.
Text mode automatically decodes its content and returns it as a str and the files also support universal end-of-line translation, and encoding specification arguments. Binary mode does not decode data while reading and simply returns its content raw and unchanged.

5. 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

6. 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

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

Ans: Unicode represents most written languages in the world. ASCII represents lowercase letters (a-z), uppercase letters (A-Z), digits (0–9) and symbols such as punctuation marks while Unicode represents letters of English, Arabic, Greek etc. Unicode strings are more versatile than ASCII strings.

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

Ans: Python 3 always stores text strings as sequences of Unicode code points. default text processing behaviour in Python 3 aims to detect text encoding problems as early as possible unlike in Python 2 that allowed data corruption by default and strict correctness checks had to be requested explicitly.