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

**Ans.**

. **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() ->** Retur

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

Ans.  In Python3 default format of strings is Unicode Whereas in Pyton2 we need to explicitly mention Unicode value using u.

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. The major difference between these two is that a text file contains textual information in the form of alphabets, digits and special characters or symbols. On the other hand, a binary file contains bytes or a compiled version of a text file.

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. The difference between ASCII and Unicode is that 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. mathematical symbols, historical scripts, emoji covering a wide range of characters than ASCII.

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

Ans. Python 3 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.