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

***Ans:***

1. ‘Str’ type: this is the basic string type in python, which represents a sequence of Unicode characters. The ‘str’ type provides a wide range of methods and functionality for working with text data.

2. ‘bytes’ type: this type represents a sequence of bytes, which can be used to represent binary data or text data encoded in a specific character encoding. Unlike the str type, which is immutable, bytes objects can be modified.

3. ‘byte array’ type: this type is like the ‘bytes’ type, but it is mutable, which means that the contents of the object can be modified after it is created. The ‘byte array’ type is often used in situations where binary data needs to be manipulated.

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

***Ans***:

the string forms in Python 3.X vary in terms of the operations they support and the contexts in which they are used. The str type is typically used for text data, while the bytes and ‘byte array’ types are used for binary data or text data encoded in a specific character encoding.

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

***Ans***:

In Python 3.X, you can include non-ASCII Unicode characters in a string by using Unicode escape sequences or by using the actual Unicode character directly in the string.

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

***Ans***:

1. Text-mode files: Text-mode files are files that are opened with the 't' mode flag or without any mode flag specified. When working with text-mode files, the contents of the file are treated as a sequence of Unicode characters. the newline character (\n) is automatically converted to the appropriate platform-specific newline character sequence.
2. Binary-mode files: Binary-mode files are files that are opened with the 'b' mode flag. When working with binary-mode files, the contents of the file are treated as a sequence of bytes. In binary-mode, newline characters are not automatically converted, and the file is read and written as a sequence of bytes.

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

***Ans***: In Python 3.X, you can interpret a Unicode text file that is encoded in a different encoding than your platform's default by using the open () function with the encoding parameter.

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

***Ans***: In Python 3.X, the best way to create a Unicode text file in a particular encoding format is to use the open() function with the encoding parameter when writing to the file.

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

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