Q1. What is the benefit of regular expressions?

Ans : Regular expressions are useful in search and replace operations. The typical use case is to look for a sub-string that matches a pattern and replace it with something else. Most APIs using regular expressions allow you to reference capture groups from the search pattern in the replacement string.

Q2. Describe the difference between the effects of "(ab)c+" and "a(bc)+." Which of these, if any, is the unqualified pattern "abc+"?

Ans :

Q3. How much do you need to use the following sentence while using regular expressions?

import re

Ans: Regular expressions are particularly useful for defining filters. Regular expressions contain a series of characters that define a pattern of text to be matched—to make a filter more specialized, or general.

Q4. Which characters have special significance in square brackets when expressing a range, and under what circumstances?

Ans : A regular expression (shortened as regex or regexp; sometimes referred to as rational expression) is a sequence of characters that specifies a search pattern in text. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

Q5. How does compiling a regular-expression object benefit you?

Ans : compile(pattern, repl, string): We can combine a regular expression pattern into pattern objects, which can be used for pattern matching. It also helps to search a pattern again without rewriting it

Q6. What are some examples of how to use the match object returned by re.match and re.search?

Ans : There is a difference between the use of both functions. Both return the first match of a substring found in the string, but re. match() searches only from the beginning of the string and return match object if found

Q7. What is the difference between using a vertical bar (|) as an alteration and using square brackets as a character set?

Q8. In regular-expression search patterns, why is it necessary to use the raw-string indicator (r)? In   replacement strings?

Ans : The solution is to use Python’s raw string notation for regular expression patterns; backslashes are not handled in any special way in a string literal prefixed with 'r'. So r"\n" is a two-character string containing '\' and 'n', while "\n" is a one-character string containing a newline. Usually patterns will be expressed in Python code using this raw string notation.