Q1. What is the benefit of regular expressions?

Ans: Regular Expressions, also known as **regex** or **regexp**, are used to match strings of text such as particular characters, words, or patterns of characters. It means that we can match and extract any string pattern from the text with the help of regular expressions. it helps the programmers to Write less and cleaner code. it also avoids multiple use of **if/else** statements.

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

Ans: Both **(ab)c+** and **a(bc)+** are valid patterns. the difference between both these patterns is in **(ab)c+ ab** is group whereas in **a(bc)+ bc** is a group.

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

import re

Ans: import re statement always has to be imported before using regular expressions.

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

Ans: The Characters **.,*,?,**^,or,**()**, have a special signiface when used with square brackets. They need not be be explicitly escaped by \ as in case of pattern texts in a raw string.

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

Ans: 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 rewritting it.

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

Ans: The **re.search()** and **re.match()** both are functions of re module in python. These functions are very efficient and fast for searching in strings. The function searches for some substring in a string and returns a match object if found, else it returns none.

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. But if a match of substring is found somewhere in the middle of the string, it returns none.

While **re.search()** searches for the whole string even if the string contains multi-lines and tries to find a match of the substring in all the lines of string

[1]:

```
import re
Substring ='string'
String1 ='We are learning regex with geeksforgeeks regex is very useful for string matching. It is fast too.'
String2 ='string We are learning regex with geeksforgeeks regex is very useful for string matching. It is fast too.'
print(re.search(Substring, String1, re.IGNORECASE))
print(re.match(Substring, String1, re.IGNORECASE))
print(re.search(Substring, String2, re.IGNORECASE))
print(re.match(Substring, String2, re.IGNORECASE))
```

```
<_sre.SRE_Match object; span=(66, 72), match='string'>
None
<_sre.SRE_Match object; span=(0, 6), match='string'>
<_sre.SRE_Match object; span=(0, 6), match='string'>
```

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

Ans: When | us used then patterns searches for **or** option. i.e <**pattern_1**>|<**pattern_2**> means it searches as <**pattern_1**>**or**<<**pattern_2**> in the searched string. the first occurance of matched string will be returned as the Match Object. Using Character set in square Brackets searches for all the character set in the square bracket and if match is found, it returns it.

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

Ans: Raw Strings are used in the regular-expression search patterns, so that blackslashes donot have to be escaped.