1. What is the name of the feature responsible for generating Regex objects?

**Ans;-** The feature responsible for generating regex objects in python is ‘re’ module. The ‘re’ module provides support for regular expressions and includes a ‘compile()’ function that can be used to generate regex objects. This objects can then be used to search, match and manipulate texts according to the regular expressions patterns.

2. Why do raw strings often appear in Regex objects?

**Ans;-** Raw strings are often used in regular expressions in python because they allow the use of backslashes (\) without any special treatment of backslash character by python’s string literal parser. this is important in python because backslashes are frequently used as escape characters to represent special characters or character sequence.

3. What is the return value of the search() method?

**Ans;-** The ‘search()’ method in python is used to search a string for a specified pattern and returns a match object if there is a match. If there is not match, then it returns ‘None’.

4. From a Match item, how do you get the actual strings that match the pattern?

**Ans;-**  In python, you can get the actual strings that match the pattern from a match object using the ‘group()’ method. The ‘grop()’ method returns a substring of the input string that matched the pattern. By default, ‘group()’ returns an entire match or we can pass an argument to ‘group()’ to specify a particular capturing group within the pattern to return.

5. In the regex which created from the r'(\d\d\d)-(\d\d\d-\d\d\d\d)', what does group zero cover? Group 2? Group 1?

**Ans:-** In python, ‘group(0)’ always return the entire match. Therefore, in this pattern, ‘group(0)’ will return the entire string that matches the pattern. There are two capturing groups in this pattern;-

GROUP(1) ‘(\d\d\d)’ :- which matches three consecutive digits.

GROUP(2) ’(\d\d\d-\d\d\d\d)':- which matches a group of three digits followed by a hyphen and then a group of four digits.

‘group(0)’ always return the entire match. Therefore, in this pattern, ‘group(0)’ will return the entire string that matches the pattern.

‘group(1)’ will return the first capturing group, which matches three consecutive digits.

‘group(2)’ will return the second capturing group, which matches a group of three digits followed by an hyphen and then a group of four digits.

6. In standard expression syntax, parentheses and intervals have distinct meanings. How can you tell a regex that you want it to fit real parentheses and periods?

**Ans:-** To match real parentheses and periods in a regular expression in python, you need to escape them with a backslash(‘\’), which tells python that you want to match the actual characters rather than using the special meaning of the parentheses or periods.

7. The findall() method returns a string list or a list of string tuples. What causes it to return one of the two options?

**Ans:-**  The ‘findall()’ function is a method from the python ‘re’(regular expression) module, and it returns a list of non-overlapping matches of a pattern in a string. The ‘findall()’ method returns a list of strings when the regular expression pattern being searched for contains no capturing groups, there are no parentheses in the pattern. In this case, the method returns a list of all the non-overlapping matches of the pattern in the input string.

8. In standard expressions, what does the | character mean?

**Ans:-** In python’s regular expressions, The ‘|’ character is known as ‘pipe’ or ‘alternation’ operator. It allows you to match one pattern or another pattern. The ‘|’ operator is used within a regular expression pattern to specify multiple alternatives that can match at a given position in the input string. It functions as a logical ‘OR’ operator, allowing you to match one pattern or another pattern. The pipe character is placed between the patterns that you want to match, enclosed in parentheses if necessary.

9. In regular expressions, what does the character stand for?

**Ans:-** In python’s regular expressions, The ‘ . ‘ character is known as the ‘dot’ or ‘period’ operator. It is a metacharacter that matches any single character except for a new line character.

10.In regular expressions, what is the difference between the + and \* characters?

**Ans:-**  In python regular expressions, The “+” and “\*” characters are quantifiers that specifies how many times a preceding character or group of characters should appear in the matching string.

The “+” character matches one or more occurrences of the preceding character or group of characters.

The “\*” matches zero or more occurrences of the preceding character or group of characters.

11. What is the difference between {4} and {4,5} in regular expression?

**Ans:-** In python regular expressions, {4} and {4,5} are quantifiers that specify the exact number of occurrences of the preceding character or a group of characters in a matching string.

The {4} quantifier matches exactly 4 occurrences of the preceding character or group of characters.

The {4,5} quantifier matches at least 4 and at most 5 occurrences of the preceding character or group of characters.

12. What do you mean by the \d, \w, and \s shorthand character classes signify in regular expressions?

**Ans:-** In python regular expressions, The shorthand character classes ‘\d’, ‘\w’ and ‘\s’ have special meanings and are used to match certain types of characters.

Here what each of these shorthand characters classes signifies:-

1. ‘\d’ :- This matches any digit character(0-9)
2. ‘\w’ :- This matches any alphanumeric character, including letters, digits and underscores.
3. ‘\s’ :- This matches any whitespace characters, including space, tabs and line breaks.

13. What do means by \D, \W, and \S shorthand character classes signify in regular expressions?

**Ans:-** In python regular expressions, The uppercase versions of the shorthand character classes ‘\D’, ‘\W’ and ‘\S’ have the opposite meaning as there lowercase counterparts. They are used to match any character that is not a digit, alphanumeric, or whitespace characters.

1. ‘\D’ :- This matches any character that is not a digit (0-9)
2. ‘\W’ ;- This matches any character that is not an alphanumeric character including spaces, punctuation and special characters.
3. ‘\S’ ;- This matches any character that is not a whitespace character, including letter, digits, punctuation and special characters.

14. What is the difference between .\*? and .\*?

**Ans:-**  In python regular expressions, ‘.\*?’ and ‘.\*’ both used to match any character zero or more times. However, there is a difference between both of these two:-

‘.\*?’ is a non-greedy match that matches the shortest possible string that satisfies the pattern.

On the other hand, ‘.\*’ is a greedy match that matches the longest possible string that satisfies the pattern.

15. What is the syntax for matching both numbers and lowercase letters with a character class?

**Ans:-** To match both numbers and lowercase letters with a character class in python, you can use the square brackets notation to define a character class that includes all the characters you want to match. The syntax for matching both numbers and lowercase letters with a character class in python is ;- **[0-9a-z]**

This character class matches any character that is either a digit(0-9) or a lowercase letters (a-z).

16. What is the procedure for making a normal expression in regax case insensitive?

**Ans:-** To make a regular expression case-sensitive in python, we can use the ‘IGNORECASE’ flag. Here’s an example:-

import re

pattern = "hello"

text = "Hello, World!"

matches = re.findall(pattern, text, flags=re.IGNORECASE)

print(matches) # Output: ['Hello']

17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?

**Ans:-**  In python’s ‘re’ module, The dot(‘ . ‘) character normally matches any character except for a newline character.

However, if ‘re.DOTALL’ is passed as the second argument to the ‘re.compile()’ function, then the dot character will match any character, including newline characters.

18. If numReg = re.compile(r'\d+'), what will numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') return?

**Ans:-**  If numReg = re.compile(r'\d+')’ and we call ‘numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen')’ in python, the output will be:-

'X drummers, X pipers, five rings, X hen'

19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?

**Ans:-**  In python’s ‘re’ module, passing ‘re.VERBOSE’ as the second argument to the ‘re.compile()’ function allow you to create a regular expression pattern that includes whitespace and comments for readability, without affecting the actual pattern being matched.

20. How would you write a regex that match a number with comma for every three digits? It must match the given following:

'42'

'1,234'

'6,368,745'

but not the following:

'12,34,567' (which has only two digits between the commas)

'1234' (which lacks commas)

**Ans;-** We can use the following regular expressions to match a number with a comma for every three digits :-

import re

regex = r'^\d{1,3}(,\d{3})\*$'

print(re.match(regex, '42')) # Match

print(re.match(regex, '1,234')) # Match

print(re.match(regex, '6,368,745')) # Match

print(re.match(regex, '12,34,567')) # No match

print(re.match(regex, '1234')) # No match

21. How would you write a regex that matches the full name of someone whose last name is Watanabe? You can assume that the first name that comes before it will always be one word that begins with a capital letter. The regex must match the following:

'Haruto Watanabe'

'Alice Watanabe'

'RoboCop Watanabe'

but not the following:

'haruto Watanabe' (where the first name is not capitalized)

'Mr. Watanabe' (where the preceding word has a nonletter character)

'Watanabe' (which has no first name)

'Haruto watanabe' (where Watanabe is not capitalized)

**Ans:-** Here’s a regular expression in python that matches full name with the last name ‘Watanabe’, where the first name is the capitalized word :-

import re

regex = r"^[A-Z][a-z]\*\sWatanabe$"

test\_cases = ['Haruto Watanabe', 'Alice Watanabe', 'RoboCop Watanabe', 'haruto Watanabe', 'Mr. Watanabe', 'Watanabe', 'Haruto watanabe']

for test\_case in test\_cases:

if re.match(regex, test\_case):

print(f"{test\_case} matches the pattern")

else:

print(f"{test\_case} does not match the pattern")

Explanation:-

1. ‘^’ :- matches the start of the string.
2. ‘[A-Z]’ :- matches a single uppercase letters( the first letter of the first name)
3. ‘[a-z]\*’ :- matches zero or more lowercase letters (the remaining letters of the first name, if any)
4. ‘\s’ :- matches a single whitespace character( the space between the first and the last names)
5. ‘Watanabe’ :- matches the string “Watanabe”.
6. ‘$’ :- matches the end of the string.

The ‘re.match()’ function is used to check if the regex matches the entire string from start to end. If you want to match substrings within a larger text, we can use the ‘re.search()’ function instead.

22. How would you write a regex that matches a sentence where the first word is either Alice, Bob, or Carol; the second word is either eats, pets, or throws; the third word is apples, cats, or baseballs; and the sentence ends with a period? This regex should be case-insensitive. It must match the following:

'Alice eats apples.'

'Bob pets cats.'

'Carol throws baseballs.'

'Alice throws Apples.'

'BOB EATS CATS.'

but not the following:

'RoboCop eats apples.'

'ALICE THROWS FOOTBALLS.'

'Carol eats 7 cats.'

**Ans:-** Here’s a regular expression in python that matches a sentence that starts with one of three names and follows with one of three verbs, and one of three objects, ending with a period.

import re

regex = r"^(Alice|Bob|Carol)\s+(eats|pets|throws)\s+(apples|cats|baseballs)\.$"

test\_cases = ['Alice eats apples.', 'Bob pets cats.', 'Carol throws baseballs.', 'Alice throws Apples.', 'BOB EATS CATS.', 'RoboCop eats apples.', 'ALICE THROWS FOOTBALLS.', 'Carol eats 7 cats.']

for test\_case in test\_cases:

if re.match(regex, test\_case, re.IGNORECASE):

print(f"{test\_case} matches the pattern")

else:

print(f"{test\_case} does not match the pattern")

Explanation;-

1. ‘^’ :- matches the start of the string.
2. ‘(Alice|Bob|Carol)’ :- matches one of the three names (case insensitive due to the ‘re.IGNORECASE’ flag)
3. ‘\s+’ :- matches one or more whitespace characters(to separate the words)
4. ‘(eats|pets|throws)’ ;- matches one of the three verbs( case insensitive due to the ‘re.IGNORECASE’ flag)
5. ‘\s+’ :- matches one or more whitespace characters( to separate the words)
6. ‘(apples|cats|baseballs)’ :- matches one of the three objects (case insensitive due to the ‘re.IGNORECASE’ flag)
7. ‘\.’ :- matches a period( escaped with a backslash to indicate a literal match)
8. ‘$’ :- matches the end of the string.

The ‘re.match()’ function is used to check if the regex matches the entire string from start to end. If you want to match substrings within a larger text, then we can use the ‘re.search()’ function instead.