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

The re. compile() function returns Regex objects.

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

Raw strings are used in order that backslashes do not have to be escaped.

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

The search() method searches a string for a specified value, and returns the position of the match. The search value can be string or a regular expression. This method returns -1 if no match is found.

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

Import the regex module with import re.Create a Regex object with the re.compile() function. (Remember to use a raw string. Pass the string you want to search into the Regex object’s search() method. This returns a Match object.Call the Match object’s group() method to return a string of the actual matched text.

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?

Group 0 is the entire match, group 1 covers the first set of parentheses, and group 2 covers the second set of parentheses.

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?

Periods and parentheses can be escaped with a backslash: \., \(, and \).

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

If the regex has no groups, a list of strings is returned. If the regex has groups, a list of tuples of strings is returned.

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

The | character signifies matching "either, or" between two groups.

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

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

The + matches one or more. The \* matches zero or more.

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

{m} specifies that exactly m copies of the previous RE should be matched; fewer matches cause the entire RE not to match. So a{4} will match exactly 4 'a' characters, but not 3.

{m,n} causes the resulting RE to match from m to n repetitions of the preceding RE, attempting to match as many repetitions as possible. So, a{4,5} will match from 4 to 5 'a' characters. Omitting m specifies a lower bound of zero, and omitting n specifies an infinite upper bound. As an example, a{4,}b will match 'aaaab' or a thousand 'a' characters followed by a 'b', but not 'aaab'. The comma may not be omitted or the modifier would be confused with the previously described form.

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

\d (digit) matches any decimal digit character(same as [0-9] ).

\w matches any alphanumeric word character. Word characters are uppercase and lowercase letters, digits, and the underscore (\_) character, so \w is essentially shorthand for [a-zA-Z0-9\_].

\s matches any whitespace character:

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

\D is the opposite of \d. It matches any character that isn’t a decimal digit. \D is equivalent to [^0-9].

\W is the opposite of \w. It matches any non-word character and is equivalent to [^a-zA-Z0-9\_]

\S is the opposite of \s. It matches any character that isn’t whitespace.

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

If the RE <.\*> is matched against '<a> b <c>', it will match the entire string, and not just '<a>'. Adding ? after the qualifier makes it perform the match in non-greedy or minimal fashion; as few characters as possible will be matched. Using the RE <.\*?> will match only '<a>'.

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

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

Passing re.I or re.IGNORECASE as the second argument to re.compile() will make the matching case insensitive.

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

The . character normally matches any character except the newline character. If re.DOTALL is passed as the second argument to re.compile(), then the dot will also match newline characters.

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

'X drummers, X pipers, five rings, X hens'

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

The re.VERBOSE argument allows you to add whitespace and comments to the string passed to re.compile().

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 : re.compile(r'^\d{1,3}(,\d{3})\*$') will create this regex, but other regex strings can produce a similar regular expression.

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 : re.compile(r'[A-Z][a-z]\*\sNakamoto')

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 : re.compile(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.', re.IGNORECASE)