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

Sol: in python, the ‘re’ module provides regular expressions functionality and is responsible for generating regular expression objects

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

Sol: raw strings are often used in regular expressions to avoid unwanted escapes of special characters.

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

Sol: The search() method in a regular expression object returns a match object if it finds a match in the input string or ‘None’ if it does not find a match.

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

Sol: Once you have a match object from a regular expression search, you can use the ‘group()’ method to get the actual string that matches the pattern.

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?

Sol: group zero covers the entire match, group 1 covers the first set of three digits separated by a hyphen, and group 2 covers the second set of three 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?

Sol: In regular expression syntax, parentheses and period have special meanings and are used to specify groups and match any character, respectively. To match real parentheses and periods as literal character, you can use the backslash character ‘\’ to escape them.

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

Sol: The findall() method returns either a list of strings or a list of string tuples depending on whether the regular expression pattern contains any capturing groups. If the pattern contains a list of strings,where each string is a non-overlapping match of the pattern in the input string.

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

Sol: in standard expression | character is used to represent the logical OR operator.

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

Sol: in regular expression the dot character represents any single character, except for a newline character.

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

Sol: In regular expressions, the “+” and “\*” characters are quantifiers that indicate how many times the preceding element should occur in the string being matched.

The “+” character means “one or more occurrences of the preceding element “, while the “\*” character means “Zero or more occurrences of the preceding element”.

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

Sol: In the regular expressions, {4} and {4,5} are both quantifiers that specify the number of times a preceding element should occur in the string being matched.

{4} means exactly 4 occurrences of the preceding element.

{4,5} means between 4 and 5 occurrences of the preceding element.

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

Sol: In regular expressions, shorthand character classes are pre-defined character sets that match a specific category of characters.

\d: Matches any digit charcter(0-9)

\w: matches any word character, which includes alphanumeric characters (a-z,A-Z,0-9) and underscore(\_).

\s: matches any whitespace character, including spaces, tabs, and newline charcters.

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

Sol: \D: matches any character that is not a digit(0-9)

\W: matches any character that is not a word character, which includes any character that is not alphanumeric (a-z,A-Z,0-9) or underscore (\_).

\S: matches any character that is not a whitespace character, including any character that is not a space, tab, or newline character.

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

Sol: ‘.\*’ matches any sequence of characters except for a newline character, and ‘.\*?’ matches the same, but in the laziest way possible.

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

Sol: [0-9a-z]

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

Sol: in regular expressions, you can make a normal expression case-insensitive by using the “I” flag.

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

Sol: In regular expressions, the ‘.’ Character matches any character except a newline character (‘\n’). if the ‘re.DOTALL’ flag is passed as the second argument to ‘re.compile()’ the ‘.’n charcter will match any charcter including the newline character(‘\n’).

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

Sol: If numReg = re.compile(r'\d+'), then numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') will return the string ‘ ”X drummers, X pipers, five rings, X then” ‘.

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

Sol: passing re.VERBOSE as the 2nd argument to re.compile() allow you to write verbose regular expressions with comments and extra whitespace.

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)

Sol:^\d{1,3}(,\d{3})\*$

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)

Sol: ^[A-Z][a-z]\*\sWatanabe$

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.'

Sol: ^(Alice|Bob|Carol)\s+(eats|pets|throws)\s+(apples|cats|baseballs)\.$