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

:- The ‘re’ module is responsible for generating Regex objects.

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

:- Raw strings are created by prefixing a string with the letter 'r', which tells Python to treat backslashes as literal characters instead of escape characters.

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

:- The search() method in Python's regular expression module re returns a Match object if the regular expression pattern is found in the string being searched, otherwise it returns None.

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

:- using match.group() which returns match pattern from string.

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?

:- The regular expression r'(\d\d\d)-(\d\d\d-\d\d\d\d)' has two groups, defined by the parentheses. Group 0 represents the entire match, while group 1 and group 2 represent the portions of the match captured by the two sets of parentheses.

In this particular regular expression, group 0 would cover the entire match of a string that matches the pattern, which would be a sequence of three digits followed by a hyphen, followed by a sequence of three digits, another hyphen, and then a sequence of four digits.

Group 1 would cover the first set of three digits, and group 2 would cover the second set of three digits followed by the 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?

:- To match a literal left parenthesis (, we can use the escape character \(. Similarly, to match a literal right parenthesis ),we would use the escape character \).

To match a literal period ., we would use the escape character \..

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

:- A capture group is a part of the regular expression enclosed in parentheses (). It is used to group a part of the regular expression and capture the text matched by that part. If the regular expression contains one or more capture groups, findall() will return a list of tuples, where each tuple represents a match, and the items in the tuple correspond to the captured groups.

If the regular expression does not contain any capture groups, findall() will return a list of strings, where each string represents a match of the entire regular expression pattern.

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

:- | character is used to denote alternation, which allows you to match either one pattern or another.

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

:- In regular expressions, the dot character . is a special character that matches any single character except for a newline character \n.

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

:- The + character matches one or more occurrences of the preceding pattern

The \* character matches zero or more occurrences of the preceding pattern.

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

:- In regular expressions, the {} characters are used as a quantifier to specify how many times the preceding pattern should match.

{4} specifies that the preceding pattern should match exactly four times.

{4,5} specifies that the preceding pattern should match at least four times and at most five times.

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

:- \d: Matches any digit character. It is equivalent to the character class [0-9].

\w: Matches any alphanumeric character (i.e., letters and digits) as well as underscores. It is equivalent to the character class [a-zA-Z0-9\_].

\s: Matches any whitespace character, including spaces, tabs, and newlines. It is equivalent to the character class [\t\n\f\r\p].

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

:- \D: Matches any character that is not a digit. This is equivalent to the character class [^0-9].

\W: Matches any non-alphanumeric character. This includes spaces and punctuation. This is equivalent to the character class [^a-zA-Z0-9\_].

\S: Matches any non-whitespace character. This includes any character that is not a space, tab, or newline. This is equivalent to the character class [^ \t\n\r].

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

:- In regular expressions, .\* and .\*? are both quantifiers that match zero or more occurrences of any character (except newline)

The .\* quantifier is greedy, meaning it will match as many characters as possible while still allowing the overall pattern to match.

The .\*? quantifier is non-greedy, meaning it will match as few characters as possible while still allowing the overall pattern to match.

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

:- To match both numbers and lowercase letters using a character class in regular expressions we can use [0-9a-z]

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

:- We can use the re.IGNORECASE flag in Python

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

:- In a regular expression, the . (dot) character normally matches any character except for a newline (\n).

However, if the re.DOTALL flag is passed as the second argument to the re.compile() function in Python, then the . character will match any character, including a newline.

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

:- will replace all occurrences of one or more digits with the letter "X" in the input string.

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

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

:- It allows for more readable and organized regular expressions by allowing you to add comments and whitespace within the expression.

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)

:- ((?:(?<=\s)|(?<=[\'"])|(?<=^))\d{1,3}(?:,\d{3})\*(?:(?=\s)|(?=[\'"])|(?=$)))

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)

:- [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.'

:-/ (Alice|Bob|Carol)\s+(eats|pets|throws)\s+(apples|cats|baseballs)\./i