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

**Ans.** The function re.compile() returns regex objects.

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

**Ans.** Raw strings are often used in Regex objects because regular expressions use special characters such as "", "^", "$", ".", "\*", "+", "?" which have special meaning in Python strings. Using raw strings (prefixing the string literal with "r") prevents these characters from being interpreted as escape sequences and instead treats them as literal characters in the regular expression pattern. This makes it easier to specify the desired pattern.

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

**Ans.**  The ‘search()’ method of a compiled regular expression(regex) object returns ‘Match Object’ if a match is found anywhere in the string. If there is no match, it returns ‘None’.

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

**Ans.** From a Match item, you get the actual strings that match the pattern using ‘group()’ method.

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 the regex r‘(\d\d\d)-(\d\d\d-\d\d\d\d)’ :

Group zero covers the entire match of the pattern, which is ‘(\d\d\d)-(\d\d\d-\d\d\d\d)’.

Group 1 covers the first capturing group, which is ‘(\d\d\d)’.

Group 2 covers the second capturing group, which is ‘(\d\d\d-\d\d\d\d)’.

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.** In regular expression syntax, parentheses and periods have special meaning, but you can tell the regex to match actual parentheses and periods by escaping them with a backslash **‘\’**.

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()’ method of a compiled regular expression (regex) object returns either a list of strings or a list of tuples, depending on whether the pattern includes capturing groups or not.

If the pattern does not include capturing groups, then ‘findall()’ returns a list of strings, where each string is the match of the pattern found in the searched text.

If the pattern includes capturing groups, then ‘findall()’ returns a list of tuples, where each tuple contains the text that matches the capturing groups in the pattern.

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

**Ans.** In regular expression syntax, the | character is used as the "OR" operator. It allows you to match one of several alternative patterns. The | operator separates two or more alternative patterns and matches the first one that successfully matches the searched text.

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

**Ans.** **Which expression? didn’t mention in the question.**

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

**Ans.** In regular expression syntax, the + and \* characters are both used to match repeating sequences of characters, but they have different meanings.

The + character matches one or more consecutive repetitions of the preceding character or group.

The \* character matches zero or more consecutive repetitions of the preceding character or group.

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

**Ans.** The {4} matches exactly three instances of the preceding group. The {4,5} matches between four and five instances.

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

**Ans.** In regular expression syntax, the shorthand character classes \d, \w, and \s are used to match specific sets of characters.

The ‘\d’ shorthand character class matches any digit character.

The ‘\w’ shorthand character class matches any word character (letters, digits, or underscores).

The ‘\s’ shorthand character class matches any whitespace character (space, tab, newline, etc.).

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

**Ans.** In regular expression syntax, the shorthand character classes \D, \W, and \S are used to match specific sets of characters.

The ‘\D’ shorthand character class matches any non-digit character.

The ‘\W’ shorthand character class matches any non-word character (non-letters, non-digits, and non-underscores).

The ‘\S’ shorthand character class matches any non-whitespace character (non-space, non-tab, non-newline, etc.).

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

**Ans.** The difference between ‘.\*?’ and ‘.\*’ in regular expression syntax is the behavior of matching the shortest string possible.

The ‘.\*’ operator matches any character (represented by ‘.’) zero or more times (represented by ‘\*’). It matches as many characters as possible in the string.

The ‘.\*?’ operator also matches any character zero or more times, but it matches as few characters as possible. This is known as a non-greedy match.

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

**Ans.** To match both numbers and lowercase letters using a character class in a regular expression, you can use the shorthand character class \w. The \w character class matches any alphanumeric character (letters and digits).

For example:

import re

text = "abc123"

pattern = re.compile(r"\w+")

match = pattern.search(text)

print(match.group())

In this example, the pattern r"\w+" matches the text "abc123". The \w+ matches one or more alphanumeric characters, resulting in the whole string being matched.

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

**Ans.** To make a regular expression case insensitive, you can add the "i" flag at the end of the expression. For example, if your regular expression is `/Hello/`, to make it case insensitive, you can change it to `/Hello/i`. This tells the regex engine to match both "Hello" and "hello".

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

**Ans.** The `.` character in a regular expression normally matches any character except a newline (`\n`).

When `re.DOTALL` is passed as the second argument in `re.compile()`, the `.` character will match any character, including newline characters. So in this case, `.` will match a newline `\n` as well.

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

**Ans.** The expression `numReg = re.compile(r'\d+')` creates a regular expression pattern object that matches one or more consecutive digits.

The `.sub()` method is used to substitute all occurrences of the pattern in the input string with a new string.

So, `numReg.sub('X', '11 drummers, 10 pipers, five rings, 4 hen')` will replace all occurrences of one or more consecutive digits in the input string with the character 'X', and return the following 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?

**Ans.** Passing `re.VERBOSE` as the second argument to `re.compile()` allows to write regular expressions in a more readable way, by adding comments and white space to the expression. The regular expression compiler will ignore these extra characters and parse the expression as if they were not there. This makes it easier to understand and maintain complex regular expressions.

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.** The following regex will match a number with a comma for every three digits:

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

Explanation:

- `^` is the start of string anchor

- `\d{1,3}` matches 1 to 3 digits

- `(,\d{3})\*` matches 0 or more sequences of a comma followed by 3 digits

- `$` is the end of string anchor, which ensures the whole string consists of the pattern defined before it.

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.** The regex to match the full name of someone whose last name is Watanabe would be:

**`^[A-Z][a-z]+ Watanabe$`**

Explanation:

- `^` - matches the start of the string

- `[A-Z]` - matches a single uppercase letter

- `[a-z]+` - matches one or more lowercase letters

- `Watanabe` - matches the exact string "Watanabe"

- `$` - matches the end of the string

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 case-insensitive regex that matches the sentence pattern you described:

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

Explanation:

- `^` matches the start of the string.

- `(Alice|Bob|Carol)` matches one of the three words (Alice, Bob, or Carol). The parentheses create a capturing group.

- `\s` matches a whitespace character.

- `(eats|pets|throws)` matches one of the three words (eats, pets, or throws).

- `\s` matches a whitespace character.

- `(apples|cats|baseballs)` matches one of the three words (apples, cats, or baseballs).

- `\.$` matches a period at the end of the string (`\.` matches a literal period, `$` matches the end of the string).