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

The feature responsible for generating Regex objects in Python is the `re` module.

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

Raw strings (prefixed with `r`) are often used in Regex objects to prevent backslashes from being treated as escape characters. This is useful because regular expressions themselves often contain backslashes for special characters, and using raw strings helps avoid conflicts.

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

The return value of the `search()` method is a match object if a match is found, or `None` if no match is found.

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

You can use the `group()` method on the match object to get the actual strings that match the pattern.

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

- Group 0 covers the entire matched string.

- Group 1 covers the first set of three digits.

- Group 2 covers the second set of three digits followed by a hyphen and the last 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 real parentheses and periods, you need to escape them using a backslash. For example, to match a literal period, you would use `\.`.

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

The `findall()` method returns a list of string tuples when the regex contains groups. If the regex doesn't contain any groups, it returns a list of strings.

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

The `|` character in a regular expression means "or". It's used to specify alternatives. For example, `a|b` matches either "a" or "b".

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

In regular expressions, the `.` character matches any single character except a newline.

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

- `+` matches one or more occurrences of the preceding character or group.

- `\*` matches zero or more occurrences of the preceding character or group.

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

- `{4}` specifies exactly four occurrences of the preceding character or group.

- `{4,5}` specifies between four and five occurrences of the preceding character or group.

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

- `\d` matches any digit (0-9).

- `\w` matches any word character (alphanumeric character plus underscore).

- `\s` matches any whitespace character (space, tab, newline, etc.).

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

- `\D` matches any non-digit character.

- `\W` matches any non-word character.

- `\S` matches any non-whitespace character.

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

- `.\*?` performs a non-greedy match, trying to match as few characters as possible.

- `.\*` performs a greedy match, trying to match as many characters as possible.

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

To match both numbers and lowercase letters, you can use the character class `[0-9a-z]`.

16. What is the procedure for making a regular expression in regex case-insensitive?

You can pass the `re.IGNORECASE` flag as the second argument to `re.compile()` to make the regular expression case-insensitive.

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

- The `.` character normally matches any character except a newline.

- If `re.DOTALL` is passed as the second argument to `re.compile()`, the `.` character will match any character, including a newline.

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

It will return `'X drummers, X pipers, five rings, X hens'`.

19. What does passing `re.VERBOSE` as the second argument to `re.compile()` allow you to do?

Passing `re.VERBOSE` as the second argument allows you to include whitespace and comments within the regex pattern for better readability. The whitespace and comments are ignored by the regex engine.

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

```plaintext

'42'

'1,234'

'6,368,745'

```

but not the following:

```plaintext

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

'1234' (which lacks commas)

Regex: `^\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:

```plaintext

'Haruto Watanabe'

'Alice Watanabe'

'RoboCop Watanabe'

```

but not the following:

```plaintext

'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)

Regex: `^[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:

```plaintext

'Alice eats apples.'

'Bob pets cats.'

'Carol throws baseballs.'

'Alice throws Apples.'

'BOB EATS CATS.'

```

but not the following:

```plaintext

'RoboCop eats apples.'

'ALICE THROWS FOOTBALLS.'

'Carol eats 7 cats.'

```

Regex: `^(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.$` (with `re.IGNORECASE` flag)