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

**ANSWER IS** "Regular Expression or re . we import re module in python.

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

**ANSWER:** they are use to avoid issues with backslashes as backslashes are used to escape special characters in string. They also make the regular expression pattern clearer and easier to understand .

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

**ANSWER:** returns a match object if the pattern is found anywhere in the string, or None if the pattern is not found.

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

**ANSWER:** To retrieve the actual strings that match the pattern from a Match object in Python's re module, one way is the group() method. If the pattern contains capturing groups, specify the group number as an argument to group(). If no argument is provided, it returns the entire match. Also use the groups() method to get all the captured groups as a tuple.

1. 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?

**ANSWERS:** Group 0: This represents the entire match. It covers the entire string that matches the pattern, including both the area code and the phone number separated by a hyphen**.**

**Group 1:** This represents the first captured group, which corresponds to the three digits representing the area code**.**

**GROUP2:** Represents the second captured group, which corresponds to the seven digits representing the phone number, separated by a hyphen.

1. 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?

**ANSWER:** In regular expressions, parentheses and intervals have special meanings:

* Parentheses ( ) are used for grouping and capturing sub patterns.
* Periods . are used as a wildcard character to match any single character except newline.

To match literal parentheses and intervals in a regular expression pattern, you need to escape them using a backslash \. This tells the regex engine to treat them as literal characters rather than special regex syntax. **So,** to match real parentheses and intervals you would include \( and \) for parentheses, and \. for a intervals.

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

**ANSWER:** If the regular expression pattern contains no capturing groups (i.e., no parentheses), findall() returns a list of strings. Each string in the list represents a complete match of the pattern in the input text.

If the regular expression pattern contains one or more capturing groups (i.e., one or more sets of parentheses), findall() returns a list of tuples. Each tuple in the list represents a complete match of the pattern, and each element of the tuple corresponds to a capturing group in the pattern.

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

**ANSWER**: This vertical bar character, "|", typically represents the **logical OR operator.** When matching objects in string one uses regular expression patterns. For example to match a cat or dog in python string one can write “dog”|’cat’. Meaning the pattern will match either cat or dog.

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

**ANSWER:** character refers to any single symbol, letter, digit, that can be matched within a string.

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

**Answer** : these operators "+" and "\*" are both quantifiers. However ,the difference is as follows:

"+" means that the preceding character or group of characters must appear one or more times in the string. matches one or more occurrences. for example, the regular expression "d+" will match one or more occurrences of the character "d" in d string. So, it would match "d", "dd", "ddd", and so on.

"\*" matches zero or more occurrences. For example "a\*" will match zero or more occurrences of the character "a" in a string. So, it would match "", "a", "aa", "aaa" .

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

ANSWERS: {4} specifies that the preceding character or group of characters must appear exactly 4 times.

{4,5} specifies that the preceding character or group of characters must appear at least 4 times and at most 5 times.

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

**ANSWERS**: \d: This shorthand represents any digit character. It is equivalent to the character range [0-9].

\w: This shorthand represents any word character. It matches any alphanumeric character or underscore ([a-zA-Z0-9\_]).

\s: This shorthand represents any whitespace character. It matches spaces, tabs, newline characters, and other whitespace characters.

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

**ANSWER:**” \D”: matches any character that is not a digit. It is equivalent to the character class [^0-9]. So \D matches any character except 0 to 9.

“\S’: matches any non-whitespace character. It is equivalent to the character class [^\t\n\r\f\v]. So \S matches any character that is not a space, tab, newline, carriage return, form feed, or vertical tab.

“\W”: This shorthand character class matches any non-word character. It is equivalent to the character class [^A-Za-z0-9\_]. So \W matches any character that is not a letter, digit, or underscore.

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

***ANSWERS*:** .\*? and .\* lies in their greediness. .\*? is non-greedy, whereas .\* is greedy.

.\*?: matches zero or more occurrences of any character (except newline characters) in a non-greedy or lazy manner. This means it matches as few characters as possible while still allowing the overall pattern to match.

.\*: This expression matches zero or more occurrences of any character (except newline characters) in a greedy manner. This means it matches as many characters as possible while still allowing the overall pattern to match.

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

**ANSWERS**: ‘[0-9a-z]’ These is the class character which can match both numbers and lowercase

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

**ANSWER:** To make a regular expression case insensitive by using the (re.IGNORECASE) method.

It is passed as the third argument to the re.search() function, which performs a case-insensitive search for the specified pattern in the given text. For example:

re. search(pattern1,text, re.I)

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

**ANSWERS:** regular expressions, the ‘.’ character normally matches any character except a newline character (\n). However, if the re.DOTALL flag is passed as the second argument in re.compile(), then the ’ .’ character will match any character including newline characters (\n). It changes the behavior of the dot to match any character, including newline characters.

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

**ANSWER**: IT returns an error as shown below.:

**NameError** Traceback (most recent call last)

**~\AppData\Local\Temp/ipykernel\_3020/1096403994.py** in <module>

**----> 1** numRegex**.**sub**('X',** **'11 drummers, 10 pipers, five rings, 4 hen')**

**NameError**: name 'numRegex' is not defined.

However, if the name could be as define above, occurrences of digits in the input string will be replaced with 'X'. because the regular expression ‘r'\d+' matches one or more digits.

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

ANSWERS: when you pass re.VERBOSE as the second argument to re.compile(), it allows to create regular expressions that are more readable and maintainable by enabling verbose mode. It ignores whitespace and comments within the regular expression pattern, allowing to format the pattern more clearly and add comments to explain its purpose. Useful where readability is important.

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'

**ANSWERS:** This is how you write the above expression:^\d{1,3}(,\d{3})\*$

‘^’ -asserts the start of the string.

\d{1,3}- matches one to three digits.

(,\d{3})\* -matches a comma followed by exactly three digits, and allows this group to repeat zero or more times.

$ -asserts the end of the string.

but not the following:

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

'1234' (which lacks commas)

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'

**ANSWERS**: ^[A-Z][a-z]\*\sWatanabe$, I explain the expression as follows:

**“^” asserts the start of the string.**

**“[A-Z]” matches a single uppercase letter (for the first name).**

**“[a-z]\*” matches zero or more lowercase letters (for the rest of the first name).**

**“\s” matches a single whitespace character.**

**“Watanabe” matches the last name 'Watanabe'.**

**“$” asserts the end of the string.**

**This expression will ensure that only one word first start with capital letter.**

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)

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.'