1. What is the name of the feature responsible for generating Regex objects?
   1. In python feature responsible for generating Regex objects is re module
2. Why do raw strings often appear in Regex objects?
   1. Because they help to simplify the process of writing and interpreting regular expressions especially when dealing with backslashes
3. What is the return value of the search() method?
   1. Returns a match object if pattern is found in a string else it returns None.
4. From a Match item, how do you get the actual strings that match the pattern?
   1. To get the actual string that matches the pattern from a Match object in Python, you can use the group() method. By default, group(0) returns the entire matched 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?
   1. Group 0 (also known as the entire match or the default group) covers the entire matched string, which is the entire pattern matched by the regular expression.
   2. Group 1 covers the first set of digits enclosed in parentheses (\d\d\d), which corresponds to the first three digits before the hyphen.
   3. Group 2 covers the second set of digits enclosed in parentheses (\d\d\d-\d\d\d\d), which corresponds to the digits after the hyphen.
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?
   1. To match a literal left parenthesis (, you would use \(.
   2. To match a literal right parenthesis ), you would use \).
   3. 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?
   1. If the regular expression pattern contains capturing groups (expressed by enclosing parts of the pattern within parentheses ()), findall() returns a list of tuples
   2. If the regular expression pattern does not contain any capturing groups, findall() returns a list of strings.
8. In standard expressions, what does the | character mean?
   1. In regular expressions, the | character represents the logical OR operator.
9. In regular expressions, what does the character stand for?
   1. In regular expressions, the | character represents the logical OR operator.
10. In regular expressions, what is the difference between the + and \* characters?
    1. The + character specifies that the preceding element in the pattern must occur one or more times.
    2. The \* character specifies that the preceding element in the pattern can occur zero or more times
11. What is the difference between {4} and {4,5} in regular expression?
    1. {4} matches exactly 4 occurrences.
    2. {4,5} matches between 4 and 5 occurrences, inclusive.
12. What do you mean by the \d, \w, and \s shorthand character classes signify in regular expressions?
    1. \d: Represents any digit character. It is equivalent to the character range [0-9]. It matches any single digit from 0 to 9.
    2. \w: Represents any word character. It includes alphanumeric characters (letters and digits) as well as underscore (\_). It is equivalent to the character range [a-zA-Z0-9\_].
    3. \s: Represents any whitespace character. It includes spaces, tabs, and newline characters. It is equivalent to the character range [ \t\n\r\f\v].
13. What do means by \D, \W, and \S shorthand character classes signify in regular expressions?
    1. \D: Represents any character that is not a digit. It is equivalent to the negation of the character range [0-9]. It matches any single character except for digits (0-9).
    2. \W: Represents any character that is not a word character. It is equivalent to the negation of the character range [a-zA-Z0-9\_]. It matches any single character except for alphanumeric characters and underscore (\_).
    3. \S: Represents any character that is not a whitespace character. It is equivalent to the negation of the character range [ \t\n\r\f\v]. It matches any single character except for spaces, tabs, and newline characters.
14. What is the difference between .\*? and .\*?
    1. .\*?: This is a non-greedy or lazy quantifier. It matches zero or more occurrences of any character (except newline \n) as few times as possible.
    2. .\*: This is a greedy quantifier. It matches zero or more occurrences of any character (except newline \n) as many times as possible.
15. What is the syntax for matching both numbers and lowercase letters with a character class?
    1. pattern = r'[0-9a-z]'
    2. matches = re.findall(pattern, text)
       1. [ ]: Denotes a character class, which matches any single character contained within the brackets.
       2. 0-9: Specifies the range of digits from 0 to 9.
       3. a-z: Specifies the range of lowercase letters from 'a' to 'z'.
       4. So, [0-9a-z] will match any single digit or lowercase letter.
16. What is the procedure for making a normal expression in regax case insensitive?
    1. In regular expressions, to make a pattern case-insensitive, you can use the re.IGNORECASE flag (or re.I for short) when compiling the regular expression pattern using the re.compile() function
17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?
    1. In regular expressions, the . (dot) character normally matches any single character except for a newline (\n). It matches any character from the input string except for newline characters.
    2. However, when the re.DOTALL flag (or re.S) is passed as the second argument in the re.compile() function, the behavior of the dot (.) character changes. With the re.DOTALL flag enabled, the dot (.) character matches any character, including newline characters (\n).
18. If numReg = re.compile(r'\d+'), what will numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') return?
    1. 'X drummers, X pipers, five rings, X hen'
19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?
    1. Passing re.VERBOSE as the second argument to re.compile() allows you to write regular expressions in a more readable and organized manner by ignoring whitespace and adding comments.

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)

A: pattern = r'^\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)

A: pattern = r'^[A-Z][a-zA-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.'

A: pattern = r'^(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.$'