1. What is the name of the feature responsible for generating Regex objects?
   Solution 1: We use re. compile() function to return Regex objects.

2. Why do raw strings often appear in Regex objects?
   Solution 2: raw string notation (r"text") keeps regular expressions meaningful and confusion-free. Without it, every backslash ('\') in a regular expression would have to be prefixed with another one to escape it.

3. What is the return value of the search() method?
   Solution 3: The search() method returns Match objects.

```
import re
txt = "The rain in Spain"
x = re.search("\s", txt)
print("The first white-space character is located in position:", x.start())
```

   o/p: The first white-space character is located in position: 3

4. From a Match item, how do you get the actual strings that match the pattern?
   Solution 4: The group() method returns strings of the matched text.
   - Import the regex module with import re.
   - Create a Regex object with the re.
   - Pass the string you want to search into the Regex object's search() method.
   - Call the Match object's group() method to return a string of the actual matched text.

   For example:

```
import re
#Search for an upper case "S" character in the beginning of a word, and print the word:
txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.group())
```

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?
   Solution 5: Group 0 is the entire match, group 1 covers the first set of parentheses, and group 2 covers the second set of parentheses.

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?
   Solution 6: Periods and parentheses can be escaped with a backslash: \., \(, and \).

7. The findall() method returns a string list or a list of string tuples. What causes it to return one of the two options?
   Solution 7: If the regex has no groups, a list of strings is returned. If the regex has groups, a list of tuples of strings is returned.

8. In standard expressions, what does the | character mean?
   Solution 8: It signifies matching “either, or” between two groups.

```
import re
txt = "The rain in Spain falls mainly in the plain!"
```

```
#Check if the string contains either "falls" or "stays":
x = re.findall("falls|stays", txt)

print(x)
if x:
  print("Yes, there is at least one match!")
else:
  print("No match")
```

9. In regular expressions, what does the character stand for?
   Solution 9: Characters in regular expression and their significance is as follows:

[]	A set of characters	
\	Signals a special sequence (can also be used to escape special characters)	
.	Any character (except newline character)	
^	Starts with	e.g "^hello"
$	Ends with	e.g. "planet$"
*	Zero or more occurrences	
+	One or more occurrences	
?	Zero or one occurrences	
{}	Exactly the specified number of occurrences	
|	Either or	
()	Capture and group	

10. In regular expressions, what is the difference between the + and * characters?
    Solution 10: We use + for **one or more** occurrences and * for **zero or more** occurrences.
    For example:

```
import re
txt = "hello planet cbo"

#Search for a sequence that starts with "he", followed by 1 or more  (any) characters, and an "o":
x = re.findall("he.+o", txt)
print(x)

txt2 = "heo planet"        #this will print an empty list when used with +
#Search for a sequence that starts with "he", followed by 0 or more  (any) characters, and an "o":
y = re.findall("he.*o", txt2)
print(y)
o/p:  ['hello planet cbo']
['heo']
```

11. What is the difference between {4} and {4,5} in regular expression?
    Solution 11: The {4} matches exactly four instances of the preceding group. The {4,5} matches between four and five instances.
    For example, a{4} will match exactly four 'a' characters, but not five.

a{4,5} will match from 4 to 5 'a' characters. Omitting m specifies a lower bound of zero, and omitting n specifies an infinite upper bound. As an example, a{4,}b will match 'aaaab' or a thousand 'a' characters followed by a 'b', but not 'aaab'. The comma may not be omitted or the modifier would be confused with the previously described form.

12. What do you mean by the \d, \w, and \s shorthand character classes signify in regular expressions?
    Solution 12: The \d, \w, and \s shorthand character classes match a single digit, word, or space character, respectively.

13. What do means by \D, \W, and \S shorthand character classes signify in regular expressions?
    Solution 13: The \D, \W, and \S shorthand character classes match a single character that is not a digit, word, or not space character, respectively.

14. What is the difference between .*? and .*?
    Solution 14: The .* performs a greedy match, and the .*? performs a non-greedy match.

15. What is the syntax for matching both numbers and lowercase letters with a character class?
    Solution 15: The character class [a-z0-9] or [0-9a-z] will match all lowercase letters and numbers.

16. What is the procedure for making a normal expression in regax case insensitive?
    Solution 16: To make a normal expression in regax case insensitive, we use- IGNORECASE. re.IGNORECASE : This flag allows for case-insensitive matching of the Regular Expression with the given string i.e. expressions like [A-Z] will match lowercase letters, too. Generally, It's passed as an optional argument to re.compile().

17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?
    Solution 17: The . (period) special character matches any character except \n (newline). If re. DOTALL is passed as the second argument to re. compile(), then the dot will also match newline characters. For example:

```
import re
match = re.search(r'.+', 'Hi,\nGuys')
print(match)
```

    o/p: <re.Match object; span=(0, 3), match='Hi,'>

```
import re
match = re.search(r'.+', 'Hi,\nGuys',re.DOTALL)
print(match)
```

    o/p: <re.Match object; span=(0, 8), match='Hi,\nGuys'>

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

19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?
Solution 19: We use VERBOSE to ignore whitespace and comments inside the regular expression string. This flag allows us to write regular expressions that look nicer and are more readable by allowing us to visually separate logical sections of the pattern and add comments. Whitespace within the pattern is ignored, except when in a character class, or when preceded by an unescaped backslash, or within tokens.

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)
Solution 20:
numCommas = re.compile(r'(^\d{1,3})(,\d{3})*$')numCommas.search('12,34,567').group()

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 non-letter character)
'Watanabe' (which has no first name)
'Haruto watanabe' (where Watanabe is not capitalized)
Solution 21:
fullName = re.compile(r'[A-Z]\w [A-Z]\w')
mo = fullName.findall('Haruto Watanabe', Alice Watanabe, RoboCop Watanabe, haruto Watanabe, Mr. Watanabe ,Watanabe, Haruto watanabe ') mo.group()

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.'

Solution 22:

```
senRegex = re.compile(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs).',
re.I|re.DOTALL)

senRegex.findall('''Alice eats apples.'
'Bob pets cats.'
'Carol throws baseballs.'
'Alice throws Apples.'
'BOB EATS CATS.'
'Robocop eats apples.'
'ALICE THROWS FOOTBALLS.'
'Carol eats 7 cats.''')
```

o/p: [('Alice', 'eats', 'apples'), ('Bob', 'pets', 'cats'), ('Carol', 'throws', 'baseballs'), ('Alice', 'throws', 'Apples'), ('BOB', 'EATS', 'CATS')]