1. What is the name of the feature responsible for generating Regex objects?  
Ans. The re.compile() function returns Regex object.

2. Why do raw strings often appear in Regex objects?  
Ans. Raw strings are used so that backslashes do not have to be escaped.

3. What is the return value of the search() method?  
Ans. The search() method returns the index (position) of the first match.

4. From a Match item, how do you get the actual strings that match the pattern?  
Ans. Match objects have a group() method that will return the actual matched text from the searched 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?  
Ans. 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?  
Ans. 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?  
Ans. findall() method is used for getting all the non-overlapping matches of the pattern in the string of data as return, in the form of the list of strings.

8. In standard expressions, what does the | character mean?  
Ans. the | character, also known as a pipe or alternation operator, is used to specify a pattern that matches either the left-hand side or the right-hand side of the operator.

9. In regular expressions, what does the character stand for?  
Ans. Regular expressions are combinations of special character operators, which are symbols that control the search, that we can use to construct search strings for advanced find and/or replace searches.

10.In regular expressions, what is the difference between the + and \* characters?  
Ans. In regular expressions, the + and \* characters are quantifiers that control how many times the preceding character or group of characters can appear in the input string.

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

12. What do you mean by the \d, \w, and \s shorthand character classes signify in regular expressions?  
Ans. 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?  
Ans. The \D, \W, and \S shorthand character classes match a single character that is not a digit, word, or space character, respectively.

14. What is the difference between .\*? and .\*?  
Ans. The main difference between them is that .\* is greedy and will match as many characters as possible, while .\*? is non-greedy and will match as few characters as possible.

15. What is the syntax for matching both numbers and lowercase letters with a character class?  
Ans. Either [0-9a-z] or [a-z0-9].

16. What is the procedure for making a normal expression in regex case insensitive?  
Ans. use the re.IGNORECASE flag to make a regular expression case insensitive in Python.

17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?  
Ans. character normally matches any character except the newline character. If re. DOTALL is passed as the second argument to re. compile() , then the dot will also match newline characters.

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

19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?  
Ans. The re. VERBOSE. The VERBOSE flag of the regex package allows the user to write regular expressions that can look nicer and are more readable.

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**.

import re  
pattern = re.compile(r'^[1-9]\d{0,2}(,\d{3})\*$')  
numbers = ['42', '1,234', '6,368,745', '12,34,567', '1234']  
for number in numbers:

if pattern.match(number):

print(number, "matches")

else:

print(number, "does not match")

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.

import re

pattern = re.compile(r'^[A-Z][a-z]\*\sWatanabe$')

names = ['Haruto Watanabe', 'Alice Watanabe', 'RoboCop Watanabe', 'haruto Watanabe', 'Mr. Watanabe', 'Watanabe', 'Haruto watanabe']

for name in names:

if pattern.match(name):

print(name, "matches")

else:

print(name, "does not match")

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.

import re

pattern = re.compile(r'^(Alice|Bob|Carol)\s+(eats|pets|throws)\s+(apples|cats|baseballs)\.$', re.IGNORECASE)

sentences = ['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.']

for sentence in sentences:

if pattern.match(sentence):

print(sentence, "matches")

else:

print(sentence, "does not match")