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

Ans. re.compile()

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: search () method returns the match of object and its corresponding indexes in the string

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

Ans: Call the Match object's group () method to return a string of the actual matched text.

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: Passing 0 or nothing to the group () method will return the entire matched text. The first set of parentheses in a regex string will be group 1. The second set will be group 2. By passing the *integer* 1 or 2 to the group () match object method, you can grab different parts of the matched text.

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: To fit in real parenthesis we use black slash→ re.compile(r '(\(\d\d\d\)) (\d\d\d-\d\d\d\d)')

Same is for period → \.

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: If there are no groups in regular expression then findall() method will return string list

re.compile(r'\d\d\d-\d\d\d-\d\d\d\d') # has no groups

```
['415-555-9999', '212-555-0000']
```

If there are groups in regular expression it will return list of string tuples

```
re.compile(r'(\d\d\d)-(\d\d\d)-(\d\d\d\d)') # has groups
```

```
[('415', '555', '9999'), ('212', '555', '0000')]
```

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

Ans: A|B, where *A* and *B* can be arbitrary REs, creates a regular expression that will match either *A* or *B*. When one pattern completely matches, that branch is accepted. This means that once *A* matches, *B* will not be tested further, even if it would produce a longer overall match.

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

Ans:

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

Ans: + causes the resulting RE to match 1 or more repetitions of the preceding RE. ab+ will match 'a' followed by any non-zero number of 'b's; it will not match just 'a'.

* causes the resulting RE to match 0 or more repetitions of the preceding RE, as many repetitions as are possible. ab* will match 'a', 'ab', or 'a' followed by any number of 'b's.

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

Ans: {4} specifies that exactly 4 copies of the previous RE should be matched eg. a{4}

{4,5} causes the resulting RE to match from 4 to 5 repetitions of the preceding RE

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

Ans: \d Matches any decimal digit; this is equivalent to [0-9]

\w this is equivalent to [a-z A-Z 0-9_]

\s Matches characters considered whitespace; this is equivalent to [\t\n\r\f\v].

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

Ans: \D Matches any character which is not a decimal digit. This is the opposite of \d, the equivalent of [^0-9].

\W Matches any character which is not a word character, becomes the equivalent of [^a-z A-Z 0-9_]

\S Matches any character which is not a whitespace character.

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

Ans: The dot-star (.*) to stand in for that “anything.” The dot-star uses *greedy* mode: It will always try to match as much text as possible

The dot, star, and question mark (.*?) match any and all text in a *non-greedy* fashion

In the non-greedy version of the regex, Python matches the shortest possible stringIn the greedy version, Python matches the longest possible string

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 regax case insensitive?

Ans: Passing re.I or re.IGNORECASE as the second argument to re.compile() will make the matching case insensitive.

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

Ans: The . 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 hen'**

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

Ans: The re.VERBOSE argument allows you to add whitespace and comments to the string passed to re.compile() and allows you to write regular expressions that 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)

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)

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.'