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

Ans🡺 RE modue

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

Ans🡺 Python raw strings are prefixed with ‘r’ or ‘R’. Prefix a string with ‘R’ or ‘r’ and it will be treated as a raw string. r. Python raw strings are useful for writing regular expressions

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

Ans🡺 re.Match

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

Ans🡺 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🡪

the group() match object method to grab the matching text from just one group.

The first set of parentheses in a regex string will be group 1.

The second set of parentheses in a regex string 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.

Passing 0 or nothing to the group() method will return the entire 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🡪 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. The findall() method returns a list of strings or a list of tuples of strings.

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 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?

Ans🡪 (vertical bar **|** )- The vertical bar symbol acts as an OR operator and matches the values to the left and right of the vertical bar.

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

Ans🡪The chars are necessary for regex.

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

Each of them are quantifiers, the star quantifier(\*) means that the preceding expression can match zero or more times it is like {0,} while the plus quantifier(+) indicate that the preceding expression MUST match at least one time or multiple times and it is the same as {1,}

a\* ---> a{0,} ---> Match a or aa or aaaaa or an empty string

a+ ---> a{1,} ---> Match a or aa or aaaa but not a string empty

+ means **one or more** of the previous atom. ({1,})

\* means **zero or more**. This can match nothing, in addition to the characters specified in your square-bracket expression. ({0,})

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

{n} The simplest quantifier is a number in curly braces: {n}.

**The exact count: {4}**

\d{4} denotes exactly 4 digits, the same as \d\d\d\d.

The example below looks for a 4-digit number:

print( "I'm 1234 years old".match(/\d{4}/) ); // "1234"

**The range: {4,5}, match 4-5 times**

To find numbers from 4 to 5 digits we can put the limits into curly braces: \d{4,5}

alert( "I'm not 12, but 1234 years old".match(/\d{4,5}/) ); // "1234"

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

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

Ans🡺

\D A non-digit [^\d]

\w A word character [a-zA-Z0-9\_]

\s A whitespace character, including line break [ \t\r\n\f\x0B]

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

Ans🡪

It is the difference between greedy and non-greedy quantifiers.

Consider the input 101000000000100.

Using 1.\*1, \* is greedy - it will match all the way to the end, and then backtrack until it can match 1, leaving you with 1010000000001.  
.\*? is non-greedy. \* will match nothing, but then will try to match extra characters until it matches 1, eventually matching 101.

All quantifiers have a non-greedy mode: .\*?, .+?, .{2,6}?, and even .??.

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

Ans🡪 '[a-z,0-9]'

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

Ans🡪 Can be done using regular expression case-insensitive by using the (?i:) pattern modifier.

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

Ans🡪 By passing re.DOTALL as the second argument to re.compile(), you can make the dot character match all characters, including the newline character.

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🡪

re.VERBOSE : This flag allows you to write regular expressions that look nicer and are more readable by allowing you to visually separate logical sections of the pattern and add comments.

It’s passed as an argument to re.compile() i.e re.compile(Regular Expression, re.VERBOSE). re.compile() returns a RegexObject which is then matched with the given string.

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🡪 ^((?:\d{1,3},(?:\d{3},)\*\d{3})|(?:\d{1,3}))$

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)

Ans🡪 re.compile(r'[A-Z][a-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.'

Ans🡪 re.compile(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats |baseballs)\.', re.IGNORECASE)