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

**Answer : import re**

**re.comile() is responsible for generating Regex objects.**

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

**Answer: raw strings is used in Regex objects so that we do not escape the \(backslash) operator from the string.**

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

**Answer: search() method search for a pattern inside the string passed into it. If it finds the given pattern it returns the index from where to where it matches the string (only first instance), in case of multiple repetition of pattern in the given string it will only return the index of first instance . And if pattern is not found it will return None**.

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

**Answer: re.match(pattern,string).group() will return the actual string that matches the pattern.**

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?

**Answer: Group 0 contains the whole pattern , Group 1 covers the first parenthesis and group 2 covers the second parenthesis.**

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?

**Answer: Using r'.()'**

7. The findall() method returns a string list or a list of string tuples. What causes it to return one of the two options?

**Answer: findall() returns the list of strings which are matched with the pattern.**

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

**Answer: | character is known as pipe. We can use it anywhere we want to match many expressions. For e.g, the regular expressions r'ineuron|AI will match either 'ineuron' or 'AI'.**

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

Answer:

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

**Answer: + means one or more no of repetition whereas \* means 0 or more number of repetition.**

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

**Answer: {4} it defines exact number of count which our expression is looking for i.e no of digits are 4.**

**{4,5} defines the range of count.**

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

**Answer : \d is for digit , \w is for word character and \s is for whitespace  character.**

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

**Answer: \D stands for non-digit, \W is for non-word character and \S is for non whitespace character.**

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

**Answer: .\*? : It is used the non greedy method to match the text.**

**.\* : it uses the greedy method and tries to match as much text as possible.**

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

**Answer : [0-9a-z]**

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

**Answer: we can pass re.IGNORECASE argument in re.complie(re.IGNORECASE).**

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

**Answer: The . character in the regular expression is called as wildcard and will match any character except newline.**

**The .\* will match everything except a newline.**

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

**For example, newRegex = re.compile('.\*', re.DOTALL)**

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

**Answer:**

**sub function replace occurange of one sub string with another. Here digits are replaced by X:**

**X drummers, X pipers, five rings, X hens**

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

**Answer: re.VERBOSE is used to ignore whitespaces and comments in a regular expression.**

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)

**Answer:**

**print('Type a number with commas')**

**sentence = input()**

**import re**

**pattern = re.compile(r'\d{1,3}(,\d{3})\*')**

**matches = pattern.match(sentence)**

**if matches.group(0) != sentence:**

**print ('Does Not Match the Regular Expression!')**

**else:**

**print(matches.group(0)+ ' matches the pattern.')**

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)

**Answer:**

**import re    #import regular expressions**

**#regular expression**

**NameSearch = re.compile(r'^[A-Z]\w+ Watanabe\n', re.VERBOSE)**

**Result = NameSearch.search("Haruto Watanabe")**

**print (Result == None)**

**if Result != None:**

**print (Result.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.'

**Answer:**

**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.'**

**but not the following:**

**'Robocop eats apples.'**

**'ALICE THROWS FOOTBALLS.'**

**'Carol eats 7 cats.''')**