**ASSIGNMENT 7**

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

Ans: re.compile() function create the Regex objects.

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

Ans: Raw strings are used in Regex objects because with raw string backslashes will not be escaped, which makes the string simpler to execute.

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

Ans: Return value of the search() method is re.Match object.

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

Ans: By using the .group() method with Match item, we can get the string that matches the pattern. For example, x.group() will print the string containing the match.

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 will cover the whole match of the string; group 1 will cover the first set of parentheses I.e.(\d\d\d) and group 2 will cover the second set of parentheses I.e.(\d\d\d-\d\d\d\d).

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 we want to fit real parenthesis and periods in our regex then we can use the backslash character(\) to escape the real meaning of the parentheses or periods in the expression syntax.

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: Generally, findall() method returns a string list, if there is only one pattern to be found. But if we have two or more parenthesis with different patterns in our regex find syntax then the output will be in list of string tuples.

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

Ans: | character mean ‘bitwise OR’ in standard expressions.

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

Ans: In regular expressions, | character stand for either or. For example, if we want to find a or b in a string, we can use a|b syntax.

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

Ans: + character is used to check if there is at least one character present between two words, whereas, \* character is used to check if there is no character or many characters present in the findable string. For example: string= “I am cat”

x=re.findall(“a.+m”, string)

y=re.findall(“a.\*m”, string)

print(y)#output will be ‘am’.

print(x) #output will be empty string.

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

Ans: {4} will check the exactly 4 characters between two findable characters, whereas, {4,5} will check 4 to 5 characters.

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

Ans: \d character will check if string contains digits (0-9), \w character is used to check word characters (a-Z), digits (0-9) and underscore character and, \s is used to check white space in the string.

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

Ans: \D character is used to find non- digit numbers in a string, \W is used to find non word characters and \S character is used to find non-white space characters in the string.

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

Ans: .\* Is known as greedy repetition because it tries to matches as many characters between the findable characters in the string, whereas, .\*? Is known as reluctant repetition because it stops whenever a pattern is formed by taking minimum characters of the string.

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

Ans: The syntax for matching both numbers and lowercase letters with a character class is either [a-z0-9] or [0-9a-z]

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

Ans: To make an expression case insensitive we can use re.IGNORECASE OR re.I as an argument in the method. For example: stri=”I am Me.”

X=re.findall(“me”,stri,re.IGNORECASE) #output=['Me’]

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 between the findable words except new line. If we pass re.DOTALL as a second argument then the expression will also match the new line in the string.

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

Ans: sub() function will substitute digits with ‘X’ in the string.

Output will be: '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: By passing re.VERBOSE as the 2nd argument in re.compile() will allow to add comments and white space characters in the 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)

Ans: y=re.search(r'^\d{1,3}(,\d{3})\*$',txt)

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: z=re.compile(r'^[A-Z][a-z]{2}.\*Watanabe$')

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