Assignment-7

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

Ans1-the feature responsible for generating regular expression (Regex) objects is the **re.compile()** function, which is part of the **re** module. The **re.compile()** function compiles a regular expression pattern into a Regex object that can be used for various operations, such as searching, matching, and replacing text.

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

Ans2- Raw strings are often used in regular expressions (Regex) because they simplify the process of writing and interpreting regular expression patterns. In regular expressions, backslashes (**\**) are commonly used as escape characters to introduce special sequences. However, Python string literals also use backslashes as escape characters, which can lead to conflicts and readability issues when writing regular expressions.

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

Ans3- The **search()** method in Python's **re** (regular expression) module returns a special match object if a match is found in the input string. If no match is found, it returns **None**

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

Ans4-To get the actual strings that match the pattern from a **Match** object in Python's **re** module, you can use the **group()** method. The **group()** method returns the matched substring(s) from the input string based on the capture groups in the regular expression 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?

Ans 5-Group 0 covers the entire matched string.

Group 1 covers the first three digits.

Group 2 covers the last seven digits (including the hyphen).

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?

Ans6- Matching a Literal Parenthesis: To match a literal parenthesis, you should escape it with a backslash \( for an opening parenthesis and \) for a closing parenthesis. For example:To match an opening parenthesis: \(

To match a closing parenthesis: \)

To match a pair of parentheses with some content in between: \(.+\)

Matching a Literal Period (Dot): To match a literal period (dot), you should escape it with a backslash \.. For example:To match a single 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?

Ans7- List of Strings (No Capturing Groups): If your regular expression pattern does not contain any capturing groups (i.e., no parentheses), findall() will return a list of strings. Each string in the list represents a complete match of the pattern in the input string.

List of String Tuples (Capturing Groups): If your regular expression pattern contains one or more capturing groups (defined using parentheses), findall() will return a list of tuples. Each tuple represents a match of the entire pattern, and each element of the tuple corresponds to a capturing group within the pattern.

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

Ans8-In regular expressions, the | character is used to denote an alternation, which acts like a logical OR operator

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

Ans9-In regular expressions (regex), the \* character (asterisk) is used as a quantifier to specify that the preceding element should be matched zero or more times. It indicates that the preceding character or group should be repeated as many times as possible, including zero times.

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

Ans10- a) \* (Asterisk): The \* quantifier means "zero or more times." It matches the preceding element (character or group) zero or more times. In other words, it allows for optional repetition.

b) + (Plus): The + quantifier means "one or more times." It matches the preceding element (character or group) at least once, and it allows for additional repetitions.

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

Ans11- a) {4}: This quantifier specifies an exact repetition count. It means that the preceding element (character or group) must be matched exactly 4 times, and no more or less.

b) {4,5}: This quantifier specifies a range of repetition counts. It means that the preceding element (character or group) must be matched at least 4 times but no more than 5 times.

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

Ans12- a) \d: This shorthand character class represents any digit from 0 to 9. It is equivalent to the character range [0-9]. It is commonly used to match numeric digits in text.

b) \w: This shorthand character class represents word characters. It matches alphanumeric characters (letters and digits) as well as underscores.

c) \s: This shorthand character class represents whitespace characters. It matches spaces, tabs, line breaks, and other whitespace characters. It is useful for matching and handling whitespace in text.

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

Ans13- \D: This shorthand character class represents any character that is not a digit. It is the negation of \d.

\W: This shorthand character class represents any character that is not a word character. It is the negation of \w.

\S: This shorthand character class represents any character that is not whitespace. It is the negation of \s.

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

Ans14- a) .\* (Greedy Match):.\* is a greedy match, which means it matches as many characters as possible while still allowing the overall pattern to match.

-It will match from the current position in the string to the end of the string, or until it satisfies the rest of the pattern, whichever comes first.

b) .\*? (Non-Greedy or Lazy Match):.\*? is a non-greedy or lazy match, which means it matches as few characters as possible while still allowing the overall pattern to match.

-It will match as few characters as needed to satisfy the rest of the pattern

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

Ans15- To match both numbers and lowercase letters using a character class in a regular expression, you can include the ranges for both numbers and lowercase letters inside square brackets.

Syntax- [0-9a-z]

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

Ans16- heres the procedure:

a)Import re module.

b)Create your regular expression pattern and compile it with the re.IGNORECASE flag.

c)Use math() ,search() , or findall() function to perform case insensitive mathing with your pattern.

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

Ans17- In a regular expression, the . (dot) character normally matches any character except a newline character (\n). This means it matches any single character, such as letters, digits, symbols, whitespace, and so on, but it does not match newline characters.

However, if you pass re.DOTALL (or re.S) as the second argument when using re.compile(), it changes the behavior of the . character to match any character, including newline characters. In other words, it makes the dot match across multiple lines.

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

Ans18-in the input string "11 drummers, 10 pipers, five rings, 4 hen," all the numeric sequences (11, 10, and 4) have been replaced with 'X', resulting in the modified string "X drummers, X pipers, five rings, X hen."

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

Ans19-When you use re.VERBOSE, whitespace (spaces and tabs) within the regular expression pattern is ignored. This means you can format your pattern across multiple lines and add indentation and comments for better readability.

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)

Ans20- import re

pattern = r'^\d{1,3}(,\d{3})\*$'

strings = ['42', '1,234', '6,368,745', '12,34,567', '1234']

for string in strings:

if re.match(pattern, string):

print(f"'{string}' is a valid match.")

else:

print(f"'{string}' is not a valid 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)

Ans21- import re

pattern = r'^[A-Z][a-zA-Z]\*\sWatanabe$'

names = [

'Haruto Watanabe',

'Alice Watanabe',

'RoboCop Watanabe',

'haruto Watanabe',

'Mr. Watanabe',

'Watanabe',

'Haruto watanabe'

]

for name in names:

if re.match(pattern, name):

print(f"'{name}' is a valid match.")

else:

print(f"'{name}' is not a valid 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.'

Ans22-

import re

pattern = r'^(?i)(Alice|Bob|Carol)\s(?:eats|pets|throws)\s(?:apples|cats|baseballs)\.$'

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 re.match(pattern, sentence):

print(f"'{sentence}' is a valid match.")

else:

print(f"'{sentence}' is not a valid match.")