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

Ans1

The re module is responsible for generating Regex objects. The re.compile() function is used to compile a regular expression pattern into a Regex object, which can then be used to search, match, and manipulate text.

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

Ans2

Raw strings are often used in Regex objects because they allow you to include special characters in a regular expression without having to escape them the backslash character is used as an escape character include a literal backslash in a regular expression pattern, you need to escape it by using two backslashes.

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

Ans3

The search() method in regular expression module (re) searches a string for the presence of a pattern specified in a regular expression and returns a match object if the pattern is found. If the pattern is not found, search() returns None.

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

Ans4

Match the pattern in a Match object in Python, you can use the group() method of the Match object.

The group() method returns the actual string that matched the pattern. If the regular expression pattern contains capturing groups, you can pass an integer argument to group() to retrieve the string that matched a specific capturing group.

5. In the regex which created from the ‘(\d\d\d)-(\d\d\d-\d\d\d\d)’, what does group zero cover? Group 2? Group 1?

Ans5

The regular expression (\d\d\d)-(\d\d\d-\d\d\d\d) contains two capturing groups enclosed in parentheses.

Group 0 (or group() without any arguments) covers the entire match of the regular expression pattern, which means the string that matches the entire expression. In this case, group 0 includes the entire phone number in the format of ###-###-####.

Group 1 covers the first capturing group, which is (\d\d\d) and matches three digits in a row. In this case, group 1 would contain the area code of the phone number, which is the first three digits in the format of ###-.

Group 2 covers the second capturing group, which is (\d\d\d-\d\d\d\d) and matches a phone number in the format of ###-####. In this case, group 2 would contain the local part of the phone number, which is the last seven digits in the format of -####.

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

In regular expression syntax, parentheses and periods have special meanings and are used to define grouping and matching patterns, respectively. To match literal parentheses or periods in a regular expression, escape them using a backslash (\).

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

The findall() method in Python's re module returns a list of all non-overlapping matches of a regular expression in a given string. The format of the returned list depends on whether the regular expression has capturing groups or not.

If the regular expression does not have any capturing groups, findall() returns a list of strings, where each string is a match found in the input string.

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

Ans8

In standard regular expressions, the | character, also known as the vertical bar or pipe symbol, is used to denote alternation. It is a logical operator that allows you to match one pattern or another.

The syntax for alternation using the | symbol is as follows:

pattern1|pattern2

This will match either pattern1 or pattern2, depending on what occurs in the input string. For example, the regular expression cat|dog will match either the word "cat" or the word "dog" in the input string.

import re

text = "I have a cat and a dog"

pattern = r"cat|dog"

matches = re.findall(pattern, text)

print(matches)

# Output: ['cat', 'dog']

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

Ans9

In regular expressions, the dot (.) character is a metacharacter that represents any single character, except for a newline character (\n).

The dot character is often used to match a single character that can be any letter, digit, or symbol. For example, the regular expression pattern b.t will match any three-character sequence that starts with "b", ends with "t", and has any character in between.

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

Ans10

In regular expressions, the + and \* characters are quantifiers that specify how many times the preceding character or group should be matched.

The + character means "one or more occurrences", while the \* character means "zero or more occurrences".

Here's how each of these quantifiers work:

+ : Matches the preceding character or group one or more times. For example, the regular expression pattern ab+ will match the letter "a" followed by one or more occurrences of the letter "b". This will match strings such as "ab", "abb", "abbb", "abbbb", etc. but will not match "a" or "ac".

\* : Matches the preceding character or group zero or more times. For example, the regular expression pattern a\*b\* will match any string that starts with zero or more occurrences of the letter "a", followed by zero or more occurrences of the letter "b". This will match strings such as "ab", "aabb", "bb", "aabbb", "bbbbb", "a", "", etc.

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

Ans11

In regular expressions, the notation {n} represents matching exactly n occurrences of the preceding character or group, while the notation {m,n} represents matching at least m and at most n occurrences of the preceding character or group.

Therefore, {4} means "match exactly four occurrences of the preceding character or group," while {4,5} means "match at least four and at most five occurrences of the preceding character or group."

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

Ans12

d - Matches any digit character. This is equivalent to the character class [0-9].

\w - Matches any word character. This includes letters, digits, and underscore (\_).

\s - Matches any whitespace character, including space, tab, and newline.

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

Ans13

the \D, \W, and \S shorthand character classes are used to match specific sets of characters that are not matched by their lowercase counterparts (\d, \w, and \s).

\D - Matches any non-digit character. This is equivalent to the character class [^0-9].

\W - Matches any non-word character. This includes any character that is not a letter, digit, or underscore (\_).

\S - Matches any non-whitespace character.

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

Ans14

There is no difference between the two regular expressions .\*? and .\*?, as they are identical.

So .\*? means "match any character zero or more times, non-greedily", and .\*? means the same thing. The ? character applies to the preceding expression, which in both cases is .\*, so it has the same effect in both cases.

Therefore, both .\*? and .\*? will match the same set of strings, which is any string that contains zero or more characters (except newline), while matching as few characters as possible.

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, you can use the following syntax

This will match any character that is a number (0-9) or a lowercase letter (a-z). The square brackets indicate a character class, and the hyphen (-) inside the brackets specifies a range of characters to match. In this case, the range is from 0 to 9 and from a to z.

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

Ans16

To make a regular expression case-insensitive in most programming languages and text editors, you can use a flag called "i". This flag tells the regular expression engine to ignore the case of the characters in the pattern.

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

Ans17

the re.DOTALL flag as the second argument to the re.compile() the dot character will match any character, including newline characters. This flag causes the dot to behave like the [\s\S] character class, which matches any whitespace character or any non-whitespace character, effectively matching any character.

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

Ans18

The numRegex pattern r'\d+' matches one or more digits in a row call numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen'), it will replace any sequence of digits with the string "X".

Output is

X drummers, X pipers, five rings, X hen

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

Ans18

Passing re.VERBOSE as the second argument to re.compile() in Python allows you to write regular expressions in a more human-readable and organized way by ignoring whitespace and adding comments.

in a regular expression, whitespace characters (spaces, tabs, and newlines) are significant and can affect the pattern matching. This can make regular expressions hard to read and understand, especially for complex patterns.

The re.VERBOSE flag tells to ignore whitespace characters in the regular expression pattern and to allow you to add comments using the # symbol. This makes the regular expression easier to read and understand, and can also help you spot errors more easily.

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

To match a number with commas for every three digits, you can use the following regular expression:

Here's how this regular expression works:

^ matches the start of the string.

\d{1,3} matches between 1 and 3 digits.

(,\d{3})\* matches zero or more occurrences of a comma followed by exactly 3 digits.

$ matches the end of the string.

This regular expression will match any number with commas every three digits, including numbers with just one digit, like "42", and numbers with multiple commas, like "1,234" and "6,368,745". However, it will not match numbers with less than three digits between the commas, like "12,34,567", or numbers without commas, like "1234".

'42' # matches

'1,234' # matches

'6,368,745' # matches

'12,34,567' # does not match

'1234' # does not 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

'Haruto Watanabe' # matches

'Alice Watanabe' # matches

'RoboCop Watanabe' # matches

'haruto Watanabe' # does not match (first name not capitalized)

'Mr. Watanabe' # does not match (preceding word has a non-letter character)

'Watanabe' # does not match (no first name)

'Haruto watanabe' # does not match (last name 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.’

Ans22

import re

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

sentence1 = 'Alice eats apples.'

sentence2 = 'Bob pets cats.'

sentence3 = 'Carol throws baseballs.'

sentence4 = 'Alice throws Apples.'

sentence5 = 'BOB EATS CATS.'

sentence6 = 'RoboCop eats apples.'

sentence7 = 'ALICE THROWS FOOTBALLS.'

sentence8 = 'Carol eats 7 cats.'

regex = re.compile(pattern, re.IGNORECASE)

match1 = regex.match(sentence1)

match2 = regex.match(sentence2)

match3 = regex.match(sentence3)

match4 = regex.match(sentence4)

match5 = regex.match(sentence5)

match6 = regex.match(sentence6)

match7 = regex.match(sentence7)

match8 = regex.match(sentence8)

print(bool(match1)) # True

print(bool(match2)) # True

print(bool(match3)) # True

print(bool(match4)) # True

print(bool(match5)) # True

print(bool(match6)) # False

print(bool(match7)) # False

print(bool(match8)) # False