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

Answer:

To generate an object of Regex type, we need to use **re.compile()** function. The output can be used for other functions like match(), search(), etc. Also, it is convenient that the result can be used multiple times in same program.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

Raw strings allow us to specify the patterns containing backslashes without having to escape them.

For example,

str1 = r“C:\parent\to\child\new directory”

print(str1)

**>>> C:\parent\to\child\new directory**

Without using the raw string, the output would have been different like below because of the escape characters,

**>>> C:\parent o\child**

**ew directory**

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

The search() method looks for the first match of the pattern in the given string and returns the corresponding value as **match object**

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

The Match objects has **group()** method that returns the matched pattern from the given string.

--------------------------------------------------------------------------------------------------------------------------------------

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 covers the entire matched string whereas Group 1 and Group 2 covers the first and second set of the parenthesis respectively.

For example,

import re

phoneNumRegex1 = re.compile(r'(\d\d\d)-(\d\d\d-\d\d\d\d)')

mo1 = phoneNumRegex1.search('My number is 415-555-4242.')

print('Group 0: ' + mo1.group(0))

print('Group 1: ' + mo1.group(1))

print('Group 2: ' + mo1.group(2))

**>>> Group 0: 415-555-4242**

**Group 1: 415**

**Group 2: 555-4242**

--------------------------------------------------------------------------------------------------------------------------------------

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:

By using escape characters for the intervals and open and close parenthesis - **\.**, **\(**, and **\)**.

--------------------------------------------------------------------------------------------------------------------------------------

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:

It depends on the number of groups the regex contains, if there are no groups then a list of string is returned whereas if there groups a list of string tuples is returned.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

The | character can be used as **“OR”**- to either match the left pattern or the right pattern written on the either side of the | character.

For example, **pattern1|pattern2 will match either pattern1 or pattern2.**

--------------------------------------------------------------------------------------------------------------------------------------

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

**Answer: SORRY, THE CHARACTER IS NOT MENTIONED IN THE QUESTION.**

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

+ matched one or more characters whereas \* matches zero or more characters.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

{4} – matches 3 instances.

{4,5} – matches between 4 and 4 instances.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

\d – matches one decimal digit from [0-9]

\w – matches word

\s – matches white space

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

\D – matches any character that is not decimal digit [^0-9]. It is the opposite of \d.

\W – matches any character that is not a word character [^a-zA-Z0-9\_]. It is opposite of \w.

\S – matches any character that is not white space. It is the opposite of \s.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

.\* is a **greedy** quantifier that matches as much characters as possible.

.\*? is a **non-greedy** quantifier that matches few characters as possible.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

[a-z0-9] or [0-9a-z] -> both can be used.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

The functions re.I or re.IGNORECASE can be used to make the regex case insensitive. Using this flag in regex, [A-Z] can also matches alphabets in lower case.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

Normally the . character matches all characters except new line whereas using re.DOTALL as a second argument it can match the new line character as well.

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

**X drummers, X pipers, five rings, X hen**

--------------------------------------------------------------------------------------------------------------------------------------

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

Answer:

Passing re.VERBOSE allows us to make the regular expression look nicer and more readable and add comments if necessary.

--------------------------------------------------------------------------------------------------------------------------------------

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:

**re.compile(r”^\d{1,3}(,{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)

Answer:

**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.'

Answer:

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

**--------------------------------------------------------------------------------------------------------------------------------------**