Assignment-7

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

Ans: In Python, the feature responsible for generating Regex objects is the **re** module.

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

Ans: Raw strings are often used in Regex objects because regular expressions use a lot of backslashes and other special characters, which have special meanings in Python strings.

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

Ans: The ‘search()’ method of a compiled regular expression object in Python returns a ‘Match’ object if it finds a match for the pattern within the searched string, and ‘None’ if no match is found.

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

Ans: To get the actual strings that match the pattern from a ‘Match’ object in Python, you can use the ‘group()’ method. This method returns the string that was matched by the regular expression.

1. 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: In the regular expression ‘r'(\d\d\d)-(\d\d\d-\d\d\d\d)'’, which matches phone numbers in the format ‘xxx-xxx-xxxx’, group zero covers the entire matched string, group 1 covers the three digits before the dash, and group 2 covers the seven digits after the dash.

1. 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: In regular expressions, parentheses and periods have special meanings as metacharacters, so if you want to match them as literal characters, you need to escape them with a backslash character ‘\’.

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

Ans: The ‘findall()’ method in Python's ‘re’ module returns a list of all non-overlapping matches of a regular expression in a given string. Whether it returns a list of strings or a list of tuples depends on whether the regular expression contains capturing groups or not.

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

Ans: In regular expressions, the ‘|’ character (vertical bar) is known as the "pipe" or "alternation" operator, and it allows you to specify multiple alternatives for a pattern to match.

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

Ans: In regular expressions, the ‘.’ character (dot) is known as a metacharacter that matches any single character except newline characters. It can be used to match any character in a string, which makes it a powerful tool for pattern matching.

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

Ans: In regular expressions, both the + and \* characters are quantifiers that allow you to specify how many times a preceding character or group of characters can appear in a string.

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

Ans: {4} is an exact quantifier that specifies an exact number of occurrences, while {4,5} is a range quantifier that specifies a range of possible occurrences.

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

Ans: The \d, \w, and \s shorthand character classes have the following meanings:

* \d matches any digit character (0-9).
* \w matches any word character, which includes letters, digits, and underscores (\_).
* \s matches any whitespace character, which includes spaces, tabs, and line breaks.

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

Ans:In regular expressions, the uppercase versions of the shorthand character classes \d, \w, and \s are used to match any character that is not in the corresponding character class. Here is what they signify:

* \D matches any non-digit character.
* \W matches any non-word character.
* \S matches any non-whitespace character.

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

* Ans: .\* is a greedy expression that matches as many characters as possible while still allowing the overall pattern to match. This means that it will match all the characters from the start of the string until the last possible occurrence of the pattern.
* .\*? is a non-greedy (or lazy) expression that matches as few characters as possible while still allowing the overall pattern to match. This means that it will match the fewest possible characters that still allow the pattern to match.

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 to use the pattern "[0-9a-z]". This allows you to match any single character that is either a digit or a lowercase letter.

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

Ans: To make a regular expression case-insensitive in regex, you can use the "i" flag at the end of the expression. This flag tells the regular expression engine to ignore case when matching characters.

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

Ans: In regular expressions, the "." (dot) character normally matches any character except a newline character "\n".

However, if ‘re.DOTALL’ is passed as the second argument to the ‘re.compile()’ function or used as a flag in a regex pattern, then the dot character will match any character, including newline characters.

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

Ans: The ‘numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen')’ function call will replace all occurrences of one or more digits in the input string with the character 'X'. So the expected 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: Passing ‘re.VERBOSE’ as the second argument to ‘re.compile()’ allows you to create a regex pattern that is more readable and easier to understand by adding comments and whitespace within the pattern.

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

^[0-9]{1,3}(,[0-9]{3})\*$

This pattern will match any number with commas for every three digits, such as '42', '1,234', and '6,368,745', but not '12,34,567' (which has only two digits between the commas) or '1234' (which lacks commas).

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: To match the full name of someone whose last name is Watanabe, with the first name always being one word that begins with a capital letter, you can use the following regular expression:

^[A-Z][a-z]\*\sWatanabe$

This pattern will match any full name with the last name "Watanabe" and a first name that starts with a capital letter, such as 'Haruto Watanabe', 'Alice Watanabe', and 'RoboCop Watanabe', but not '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), or 'Haruto watanabe' (where Watanabe is 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.'

Ans: To match 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 either apples, cats, or baseballs; and the sentence ends with a period, you can use the following regular expression:

^(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.$

This pattern will match any sentence that follows the given structure, such as 'Alice eats apples.', 'Bob pets cats.', and 'Carol throws baseballs.', as well as variations in capitalization like 'Alice throws Apples.' and 'BOB EATS CATS.', but not sentences that do not follow the structure or have any other words or characters, such as 'RoboCop eats apples.', 'ALICE THROWS FOOTBALLS.', and 'Carol eats 7 cats.'