PYTHON ASSIGNMENT 7

1. **What is the name of the feature responsible for generating Regex objects?**
   The re.compile() function returns Regex object. Python’s re.compile() method is used to compile a regular expression pattern provided as a string into a regex pattern object (re.Pattern). Later we can use this pattern object to search for a match inside different target strings using regex methods such as a re.match() or re.search().

2. **Why do raw strings often appear in Regex objects?**
   Raw strings are particularly useful when working with regular expressions, as they allow you to specify patterns that may contain backslashes without having to escape them. They are also useful when working with file paths, as they allow you to specify paths that contain backslashes without having to escape them. Raw strings can also be useful when working with strings that contain characters that are difficult to type or read, such as newline characters or tabs. In general, raw strings are useful anytime you need to specify a string that contains characters that have special meaning in Python, such as backslashes, and you want to specify those characters literally without having to escape them.

3. **What is the return value of the search() method?**
   The search() function searches the string for a match, and returns a Match object if there is a match.

4. **From a Match item, how do you get the actual strings that match the pattern?**
   Pass the string you want to search into the Regex object's search() method. This returns a Match object. Call the Match object's group() method to return a string of the actual matched text.

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?**
   Group 0 is the entire match, group 1 covers the first set of parentheses, and group 2 covers the second set of parentheses.

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?**
   Periods and parentheses can be escaped with a backslash: \., \(, and \).

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

   If the pattern has one capturing group, the findall() function returns a list of strings that match the group. If the pattern has multiple capturing groups, the findall() function returns the tuples of strings that match the groups.

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

   The | character signifies matching "either, or" between two groups.

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

   Character: All characters, except those having special meaning in regex, matches themselves. E.g., the regex x matches substring "x"; regex 9 matches "9"; regex = matches "="; and regex @ matches "@". Special Regex Characters: These characters have special meaning in regex (to be discussed below): ., +, *, ?, ^, $, (,), [,], {, }, |, \.

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

    Occurrence Indicators (or Repetition Operators):

    - +: one or more (1+), e.g., [0-9]+ matches one or more digits such as '123', '000'.
    - *: zero or more (0+), e.g., [0-9]* matches zero or more digits. It accepts all those in [0-9]+ plus the empty string.

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

    In python, the operators mean:

    - {m,n}: m to n (both inclusive)
    - {m}: exactly m times

    So, the {4} matches exactly three instances of the preceding group. The {4,5} matches between four and five instances.

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

    Metacharacters: matches a character

    - \d, \D: ANY ONE digit/non-digit character. Digits are [0-9]
    - \w, \W: ANY ONE word/non-word character. For ASCII, word characters are [a-zA-Z0-9_]
    - \s, \S: ANY ONE space/non-space character. For ASCII, whitespace characters are [\n\r\t\f]

13. **What do means by \D, \W, and \S shorthand character classes signify in regular expressions?**
    - \d, \D: ANY ONE digit/non-digit character. Digits are [0-9]
    - \w, \W: ANY ONE word/non-word character. For ASCII, word characters are [a-zA-Z0-9_]
    - \s, \S: ANY ONE space/non-space character. For ASCII, whitespace characters are [\n\r\t\f]

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

    The.* performs a greedy match, and the .*? performs a non-greedy match.

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

    Either [0-9a-z] or [a-z0-9]

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

    re. IGNORECASE : This flag allows for case-insensitive matching of the Regular Expression with the given string i.e. expressions like [A-Z] will match lowercase letters, too. Generally, It's passed as an optional argument to re. compile().

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

    The . character normally matches any character except the newline character. If re.DOTALL is passed as the second argument to re.compile(), then the dot will also match newline characters.

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

    'X drummers, X pipers, five rings, X hens'

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

    re. VERBOSE : This flag allows you to write regular expressions that look nicer and are more readable by allowing you to visually separate logical sections of the pattern and add comments.

    The re.VERBOSE argument allows you to add whitespace and comments to the string passed to re.compile()

20. **How would you write a regex that match a number with comma for every three digits? It must match the given following:**

    a. **'42'**
    b. **'1,234'**
    c. **'6,368,745'**
    d. **but not the following:**
    e. **'12,34,567' (which has only two digits between the commas)**

f. '1234' (which lacks commas)

e.compile(r'^\d{1,3}(,{3})*$') will create this regex, but other regex strings can produce a similar regular expression.

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:**

    a. **'Haruto Watanabe'**
    b. **'Alice Watanabe'**
    c. **'RoboCop Watanabe'**
    d. **but not the following:**
    e. **'haruto Watanabe' (where the first name is not capitalized)**
    f. **'Mr. Watanabe' (where the preceding word has a nonletter character)**
    g. **'Watanabe' (which has no first name)**
    h. **'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:**

    a. **'Alice eats apples.'**
    b. **'Bob pets cats.'**
    c. **'Carol throws baseballs.'**
    d. **'Alice throws Apples.'**
    e. **'BOB EATS CATS.'**
    f. **but not the following:**
    g. **'RoboCop eats apples.'**
    h. **'ALICE THROWS FOOTBALLS.'**

    re.compile(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.',
    re.IGNORECASE)'Carol eats 7 cats.'