Assignment -

1A. re.compile() returns regex objects

2A. Raw strings are used so that backslashes do not need to be escaped

3A. Matched objects

4A. group() returns string of matched text

5A. Group 0 is the entire match, group 1 covers the first set of parentheses, and group 2 covers the second set of parentheses.

6A. Periods and parenthesis can be escaped by \.,\(,\)

7A. If groups are not there, then list of strings is output, else list of tuples of strings will be the output

8A. | means either / or

9A. ? means match 0 or one of the preceding groups

10A. + is concatenation, * means replication. + also means one or more, * means 0 or more

11A. {3} means match 3 groups, {3,5} means match 3 to 5 groups

12A. \d,\w\s means match digit, word or string

13A. \D\W\S means don't match digit, word or string

14A. .*? means after . few characters. *.? Means few characters before the dot and single char after dot. Dot (.) performs greedy match and (.?) performs non greedy match

15A.

16A. Passing re.I or re.IGNORECASE as the second argument to re.compile() will make the matching case insensitive.

17A. 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.

18A. X drummers, X pipers, five rings, X hens

19A. The re.VERBOSE argument allows you to add whitespace and comments to the string passed to re.compile().

20A. re.compile(r'^\d{1,3}(,\d{3})*$') will create this regex, but other regex strings can produce a similar regular expression.

21A. re.compile(r'[A-Z][a-z]*\sNakamoto')

22A. re.compile(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.', re.IGNORECASE)