1. All the regular expression work is done using the package called ‘re’.
2. The usage of a raw string in regular expression is for a reason, and the reason is if we use raw string we don’t need to use escape character for every back slash we use in the regular expression.
3. The return value of a search() function is re.Match Object, it has two usable properties one is span, and second is match, if no match found this function returns None
4. From a match item which is a re.Match object we can extract the matched string by using a property called ‘match’
5. r'(\d\d\d)-(\d\d\d-\d\d\d\d)' for this regular expression, group zero will have three digits, group one convers again three digits and group two covers four digits.

import re

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

print(regx.findall('923-123-1234'))

The output of the programm will be [('923', '123-1234')], so there will be two groups, first have three digits and the second will have three digits – four digits

6. To match the literals '(' or ')', we can use \( or \), or enclose them inside a character class: [(], [)].

7. The output of the findall() method depends upon the way you have compiled your regular expression, if you have use () in your regular expression that the response will be list of string tuple, otherwise the response will be list of string.

8. A|B, where A and B can be arbitrary REs, creates a regular expression that will match either A or B.

9. Unable to understand the quesion clearly.

10. There is a small difference between \* and + in regular expression, ab\* will match ‘a’, ‘ab’, and any number of ‘b’ after ‘a’ while ab+ will not match alone ‘a’, it will only match a non zero character followed by ‘a’.

11. {4} specifies that exactly 4 character should be matched, while {4, 5} means anything between 4 to 5 characters should be matched.

12. They are as following

\d means it matchs a decimal digit, which is equivalent to [0-9]

\s means it matchs a whitespace character, which is equivalent to [ \t\n\r\f\v]

\w means it matches a unicod word characters, which is equivalent to [a-zA-Z0-9\_]

13. They are as following

\D means it does not match a decimal digit, which is equivalent to [^0-9]

\S means it does not match a whitespace character, which is equivalent to [^ \t\n\r\f\v]

\W means it does not matche a unicod word characters, which is equivalent to [^a-zA-Z0-9\_]

14. There is no difference between ‘\*?’ and ‘+?’.

15. The syntax for matching both number and lower case character is [0-9a-z].

16. We can provide an argument re.IGNORECASE in order to make the regular expression case insensitive.

17. (Dot.) In the default mode, this matches any character except a newline. If the DOTALL flag has been specified, this matches any character including a newline.

18. If we compile numReg = re.compile(r'/d+')

and then run the command numReg.sub('X', '11 drummers, 10 pipers, five rings, 4 hen')

it will return 11 drummers, 10 pipers, five rings, 4 hen'

19. re.VERBOSE 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.

20. The regular expression that will work is regex = re.compile(r',\d\d\d,')

21. The regular expression is re.compile(r'[A-Z][a-z]{2,25}\s[W][a-z]{7}')

22. The regular expression is re.compile(r'(alice|bob|carol)\s(eats|pets|throws)\s(apples|cats|baseballs)[.]', re.IGNORECASE)