*Here’s the revised version in English with the same meaning but reworded and slightly expanded:*

1. *Literal Characters – Match an exact sequence of characters in the text.  
   Example: r'abc' will find the string "abc" exactly as written.*
2. *Dot ( . ) – Represents any single character except for a newline.  
   Example: r'a.c' can match "abc" or "axc" but not "ac".*
3. *Caret ( ^ ) – Ensures that the pattern appears at the beginning of a string.  
   Example: r'^abc' will match "abc" only if it appears at the start of the text.*
4. *Dollar Sign ( $ ) – Ensures that the pattern appears at the end of a string.  
   Example: r'abc$' will match "abc" only if it is at the end of the string.*
5. *Square Brackets ( [] ) – Allow matching any one character from the specified set.  
   Example: r'[aeiou]' will match any vowel letter.*
6. *Hyphen (-) in Brackets – Defines a range of characters within square brackets.  
   Example: r'[0-9]' will match any digit between 0 and 9.*
7. *Asterisk ( \* ) – Matches zero or more occurrences of the preceding character or group.  
   Example: r'abc\*' will match "ab", "abc", "abcc", and so on.*
8. *Plus ( + ) – Works like \*, but requires at least one occurrence of the preceding element.  
   Example: r'abc+' will match "abc" and "abcc" but not "ab".*
9. *Question Mark ( ? ) – Makes the preceding element optional, meaning it can appear once or not at all.  
   Example: r'abc?' will match "ab" or "abc" but not "abcc".*
10. *Parentheses ( () ) – Used to group parts of a regular expression together.  
    Example: r'(abc)+' will match "abc", "abcabc", and so on.*
11. *Pipe ( | ) – Functions as a logical "OR," allowing one of multiple options to match.  
    Example: r'abc|def' will match either "abc" or "def".*
12. *Backslash ( \ ) – Escapes special characters, making them literal.  
    Example: r'\.' will match a literal dot (.) instead of treating it as a wildcard.*