| **Sequence** | **Matches** |
| --- | --- |
| . | Any character except newline |
| [xyz] | Any character listed between the brackets (x, y, and z in this example) |
| [a-z] | Any character between a and z, inclusive |
| [^xyz] | The opposite of [xyz] |
| \d | ASCII digits (0 through 9, inclusive) |
| \D | Anything except ASCII digits |
| \s | ASCII spaces (space, tab, newline, carriage return, form feed) |
| \S | Anything except ASCII spaces |
| \w | The same as [0-9A-Za-z\_] |
| \W | Anything except the characters matched by \w |
| [[:alnum:]] | The same as [0-9A-Za-z] |
| [[:alpha:]] | The same as [A-Za-z] |
| [[:ascii:]] | Any ASCII character |
| [[:blank:]] | ASCII tab or space |
| [[:cntrl:]] | ASCII/Unicode control characters |
| [[:digit:]] | The same as [0-9] |
| [[:graph:]] | All "graphical" (printable) ASCII characters |
| [[:lower:]] | The same as [a-z] |
| [[:print:]] | The same as [[:graph:]] |
| [[:punct:]] | The same as [!-/:-@[-`{-~] |
| [[:space:]] | The same as [\t\n\v\f\r ] |
| [[:upper:]] | The same as [A-Z] |
| [[:word:]] | The same as \w |
| [[:xdigit:]] | The same as [0-9A-Fa-f] |
| \pN | Unicode character class by using single-letter class names ("N" in this example) |
| \p{Greek} | Unicode character class by unicode name ("Greek" in this example) |
| \PN | The opposite of \pN |
| \P{Greek} | The opposite of \p{Greek} |
| xy | x followed immediately by y |
| x\|y | either x or y, preferring x |
| x\* | zero or more x, preferring more |
| x\*? | zero or more x, preferring fewer |
| x+ | one or more x, preferring more |
| x+? | one or more x, preferring fewer |
| x? | zero or one x, preferring one |
| x?? | zero or one x, preferring zero |
| x{n,m} | between n and m repetitions of x, preferring more |
| x{n,m}? | between n and m repetitions of x, preferring fewer |
| x{n,} | at least n repetitions of x, preferring more |
| x{n,}? | at least n repetitions of x, preferring fewer |
| x{n} | exactly n repetitions of x |
| (x) | unnamed capture group for sub-pattern x |
| (?P<name>x) | named capture group, named name, for sub-pattern x |
| (?:x) | non-capturing sub-pattern x |
| \\* | Literal \* for any punctuation character \* |
| \Q...\E | Literal ... for any text ... as long as it does not include literally \E |

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

import re

txt = "abaababa"

pattern=re.compile(r'\Baba')

matches=pattern.finditer(txt)

for match in matches:

print(match)

Anchor Char – used with the strings \b string=’ha haa ha’ use /b to find all ha’s

Text

Description automatically generated

Text

Description automatically generated

[-.+]matches any of dot, plus, dash here no need for backslash for dot as its mentioned in the char set,

Text

Description automatically generated

First digit is 8 or 9 and followed by 00 and the remaining number continues

Text

Description automatically generated upper case or lower case letters

Graphical user interface, text, application

Description automatically generatednot an upper or lower case letters

import re

txt = "abaababa"

pattern=re.compile(r'[^b]at') first letter should not be b.followed letter is at

matches=pattern.finditer(txt)

for match in matches:

print(match)

Quantifiers:

Graphical user interface

Description automatically generated

Text

Description automatically generated

Text

Description automatically generatedquestion marks makes the match optional

Maybe either a dot or empty space can be matched

Matches:

MR. or MR<space>