Option Flags

Regular expression literals may include an optional modifier to control various aspects of matching. The modifiers are specified as an optional flag. You can provide multiple modifiers using exclusive OR (|).

|  |  |
| --- | --- |
| **Modifier** | **Description** |
| **re.I** | Performs case-insensitive matching. |
| **re.L** | Interprets words according to the current locale. This interpretation affects the alphabetic group (\w and \W), as well as word boundary behavior (\b and \B). |
| **re.X** | Permits “cuter” regular expression syntax. It ignores whitespace (except inside a set [] or when escaped by a backslash) and treats unescaped # as a comment marker |
| **re.U** | Interprets letters according to the Unicode character set. This flag affects the behavior of \w, \W, \b, \B |
| **re.S** | Makes a period (dot) match any character, including a newline. |
| **re.M** | Makes $ match the end of a line (not just the end of the string) and makes ^ match the start of any line (not just the start of the string). |

Regular-Expression Patterns:

In python all characters match themselves, except for control characters, **(+ ? . \* ^ $ ( ) [ ] { } | \)**. A control character can be escaped by preceding it with a backslash.

Given below is the table for those patterns:

|  |  |
| --- | --- |
| **Patterns** | **Description** |
| **^** | Matches beginning of line. |
| **.** | Matches any single character except a newline. Using the m option allows it to match the newline as well. |
| **$** | Matches end of line. |
| **\G** | Matches the point where the last match finished. |
| **[…]** | Matches any single character in brackets. |
| **[^…]** | Matches any single character not in brackets. |
| **re\*** | Matches 0 or more occurrences of the preceding expression. |
| **re?** | Matches 0 or 1 occurrence of preceding expression. |
| **re+** | Matches 1 or more occurrence of the preceding expression. |
| **\bre{ n,}** | Matches n or more occurrences of the preceding expression. |
| **\b** | Matches word boundaries when outside brackets. Matches backspace (0x08) when inside  brackets. |
| **re{ n}** | Matches exactly n number of occurrences of preceding expression. |
| **re{ n,m}** | Matches at least n and at most m occurrences of preceding expression. |
| **(re)** | Groups regular expressions and remembers matched text. |
| **a| b** | Matches either a or b. |
| **\s** | Matches whitespace. Equivalent to [\t\n\r\f] |
| **\S** | Matches non-whitespace. |
| **\d** | Matches digits. Equivalent to [0-9]. |
| **\D** | Matches non-digits. |
| **\Z** | Matches end of the string. If a newline exists, it matches just before the newline. |
| **\z** | Matches end of the string. |
| **\A** | Matches beginning of string. |
| **\w** | Matches word characters. |
| **\W** | Matches non-word characters. |
| **(?#…)** | Comments. |
| **(?> re)** | Matches independent pattern without backtracking. |
| **(?! re)** | Specifies position using pattern negation. Does not have a range. |
| **(?= re)** | Specifies position using a pattern. Doesn’t have a range. |
| **(?imx)** | Temporarily toggles on i, m, or x options within a regular expression. If in parentheses, only that area is affected. |
| **(?: re)** | Groups regular expressions without remembering matched text. |
| **(?-imx: re)** | Temporarily toggles off i, m, or x options within parentheses. |
| **(?imx: re)** | Temporarily toggles on i, m, or x options within parentheses. |
| **(?-imx)** | Temporarily toggles off i, m, or x options within a regular expression. If in parentheses, only that area is affected. |
| **\B** | Matches non-word boundaries. |
| **\n, \t, etc.** | Matches newlines, carriage returns, tabs, etc. |
| **\1…\9** | Matches nth grouped subexpression. |
| **\10** | Matches nth grouped subexpression if it matched already. Otherwise refers to the octal representation of a character code. |

Regular Expression Examples:

**1)**Literal characters:

|  |  |
| --- | --- |
| **Example** | **Description** |
| **python** | Match “python”. |

**2)** Character classes:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **[Pp]ython** | Match “Python” or “python” |
| **[aeiou]** | Match any one lowercase vowel |
| **rub[ye]** | Match “ruby” or “rube” |
| **[0-9]** | Match any digit; same as [0123456789] |
| **[a-z]** | Match any lowercase ASCII letter |
| **[A-Z]** | Match any uppercase ASCII letter |
| **[a-zA-Z0-9]** | Match any of the above |
| **[^aeiou]** | Match anything other than a lowercase vowel |
| **[^0-9]** | Match anything other than a digit |

**3)** Special Character Classes:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **.** | Match any character except newline |
| **\D** | Match a nondigit: [^0-9] |
| **\d** | Match a digit: [0-9] |
| **\s** | Match a whitespace character: [ \t\r\n\f] |
| **\S** | Match nonwhitespace: [^ \t\r\n\f] |
| **\w** | Match a single word character: [A-Za-z0-9\_] |
| **\W** | Match a nonword character: [^A-Za-z0-9\_] |

**4)** Repetition Cases:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **ruby?** | Match “rub” or “ruby”: the y is optional |
| **ruby\*** | Match “rub” plus 0 or more ys |
| **ruby+** | Match “rub” plus 1 or more ys |
| **\d{3}** | Match exactly 3 digits |
| **\d{3,}** | Match 3 or more digits |
| **\d{3,5}** | Match 3, 4, or 5 digits |

**5)**Nongreedy repetition:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **<.\*>** | Greedy repetition: matches “<python>perl>” |
| **<.\*?>** | Nongreedy: matches “<python>” in “<python>perl>” |

**6)** Grouping with Parentheses:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **\D\d+** | No group: + repeats \d |
| **(\D\d)+** | Grouped: + repeats \D\d pair |
| **([Pp]ython(, )?)+** | Match “Python”, “Python, python, python”, etc. |

**7)** Backreferences:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **([Pp])ython&\1ails** | Match python&pails or Python&Pails |
| **([‘”])[^\1]\*\1** | Single or double-quoted string. \1 matches whatever the 1st group matched. \2 matches whatever the 2nd group matched, etc. |
|  |  |

**8)** Special Syntax with Parentheses:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **R(?#comment)** | Matches “R”. All the rest is a comment |
| **R(?i:uby)** | Case-insensitive while matching “uby” |
| **R(?i)uby** | Case-insensitive while matching “uby” |
| **rub(?:y|le))** | Group only without creating \1 backreference |

**9)**Anchors:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **^Python** | Match “Python” at the start of a string or internal line. |
| **\APython** | Match “Python” at the start of a string. |
| **Python\Z** | Match “Python” at the end of a string. |
| **\bPython\b** | Match “Python” at a word boundary. |
| **\brub\B** | \B is nonword boundary: match “rub” in “rube” and “ruby” but not alone. |
| **Python(?=!)** | Match “Python”, if followed by an exclamation point. |
| **Python(?!!)** | Match “Python”, if not followed by an exclamation point. |
| **Python$** | Match “Python” at the end of a string or line. |

**10)** Alternatives:

|  |  |
| --- | --- |
| **Examples** | **Description** |
| **python|perl** | Match “python” or “perl” |
| **rub(y|le))** | Match “ruby” or “ruble” |
| **Python(!+|\?)** | “Python” followed by one or more ! or one ? |