**Regular Expressions**

Based on *An Introduction to Regular Expressions by Thomas Nield*

**Website for Testing REs:** <https://regex101.com/>

**Python Package:** re – **Example:** re.fullmatch(pattern= "[A-Z]{2}", string="TX")

We always read a Regex left to right.

######################################################################

**Special Characters:** [\^$.|?\*+()

**Literals:** Escape the special characters with \

Also, add \ to certain letters, e.g. \s matches any whitespace

######################################################################

**Character Ranges/Character Class:** e.g. to match a string containing a character of 0, 1 or 2 followed by a character of A, B or C -> use character range inside square brackets

**Example:** Regex = [012][ABC]

Input String = 1B

Match = True

**Consecutive Span of Letters/Numbers:** The above pattern can be shortened with the use of a dash -, as we can write it this way

**Example:** Regex = [0-2][A-C]

Input String = 2C

Match = True

**Qualify Multiple Ranges on a Single Character:** e.g. qualify the first character of a string to be either an uppercase letter, lowercase letter or a number

**Example:** Regex = [A-Za-z0-9][0-9]

Input String = i7

Match = True

**Negate Characters:** Match anything but the specified characters using carrot ^

**Example:** Qualify non-vowel letters

Regex = [^AEIOU]

Input String = X

Match = True

**Include Literal Dash in Character Range:** Declare it first in the range

**Example:** Regex = [-0-9][0-9]

Input String = -9

Match = True

######################################################################

**Anchors:** Qualify the start ^ or end $ of a line or string, e.g. [0-9]$ will match a number at the end of a string

^ at start plus $ at end of same pattern can force a full match – everything in between ^ and $ are the only contents allowed

**Example:** Regex = ^[0-9][0-9]$

Input String = 1432

Match = False

######################################################################

**Quantifiers:** Repeat the preceding clause of a regular expression,

e.g. instead of [A-Z][A-Z][A-Z], use [A-Z]{3}

**Example:** Regex = [A-Z]{3}

Input String = YCA

Match = True

**Example 2:** Matching a 10-digit phone number

Regex = [0-9]{3}-[0-9]{3}-[0-9]{4}

Input String = 470-127-7501

Match = True

**Min/Max Repetitions:** Minimum & maximum number of allowed repetitions

**Example:** Regex = [A-Z]{2,3}

Input String = YCA

Match = True

You can also leave the first or second arguments blank.

**0 or 1 Repetition (optional):** ? is the same as {0,1}

**Example:** Regex = [0-9]?[A-Z]{2}

Input String = 3BC

Match = True

This matches 2 uppercase alphabetic characters, preceded optionally by a number.

**1 or More Repetitions:** + is the same as {1,}

**Example:** Regex = [XYZ]+[0-9]+

Input String = XYZZZYZXXX2374676128963453452990

Match = True

This matches 1 or more upper case characters from X, Y, Z followed by 1 or more numbers

**0 or More Repetitions:** \* is the same as {0,} – makes whatever it’s quantifying completely optional but captures any number of possible repetitions

**Example:** Regex = [0-3]+[XYZ]\*

Input String = 34YYXZZ

Match = True

Matches 1 or more numbers followed by 0 or more uppercase characters from X, Y, Z.

######################################################################

**Wildcards**

**A Dot:** . is a wildcard for any character. It matches alphabetic and numeric characters as well as whitespaces, newlines, punctuation and any other symbols

**Example:** Regex = H.{3}O

Input String = HELLO

Match = True

Matches a H followed by 3 characters which can be anything and followed by an O.

**0 or More Repetitions of Any Character:** Use .\* - this will just match any text

**Example:** Regex = .\*

Input String = AsdfSJDFJSVdsfBLKJXCasdBNVJWB$TJ$@#ASDFSD@

Match = True

######################################################################

**Grouping**

Group together parts of a regular expression in parentheses – often to use a quantifier on a group, e.g. qualify uppercase letter followed by 3 numbers and repeat this whole pattern with a quantifier.

**Example:** Regex = ([A-Z][0-9]{3}-?)+

Input String = A563-X264-C578

Match = True

This matches a group of the following – A uppercase letter followed by 3 numeric digits followed by an optional dash. This group happens 0 or more times (the +).

######################################################################

**Alternation**

Use a | and this means “OR”. One of 2 or more valid patterns must match in the position, e.g. to match a 5-digit US ZIP code that ends in either “35” or “75”, we end a Regex with (35|75) – the parentheses are important in this following example.

**Example:** Regex = [0-9]{3}(35|75)

Input String = 75035

Match = True

Matches 3 numeric digits followed by either “35” or “75 – without the last pair of parentheses, we would be matching either the pattern [0-9]{3}35 OR 75

**Qualify a Set of Literal Values**

**Example:** Regex = ALPHA|BETA|GAMMA

Input String = BETA

Match = True

######################################################################

**Prefixes and Suffixes**

Use these to qualify something that precedes or follows a targeted text without capturing it.

Prefix: (?<=regex), Suffix: (?=regex)

**Example (Prefix):** Regex = (?<=[A-Z]+)[0-9]+

Input String = ALPHA12

Match = 12

This matched a pattern of 1 or more uppercase letters followed by 1 or more number, but only returns the 1 or more numbers (12). Match = False if there are no uppercase letters at the start.

**Example 2 (Suffix):** Regex = [0-9]+(?=[A-Z]+)

Input String = 12ALPHA

Match = 12

Similar to above but the letters must follow the numbers.

**Regex in Python – re Module**

Some of the common functions.

* **re.compile:** creates a regular expression object – you can then reuse this regex.
* **re.findall:** returns a list with all matches
* **re.search:** returns a Match object if there is a match anywhere (returns only the first occurrence
* **re.split:** returns list where string has been split at each match
* **re.sub:** replaces matches with another string

**Properties/Methods of Match Object**

* **.span()** – returns tuple containing start & end positions of a match
* **.string** – returns the string passed into the function
* **.group()** – returns part of the string where there’s a match

**Links/References**

<https://www.w3schools.com/python/python_regex.asp>