Python RegEx Cheatsheet with Examples

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. They're typically used to find a sequence of characters within a string so you can extract and manipulate them. For example, the following returns both instances of 'active':

```
import re
pattern = 'ac..ve'
test_string = 'my activestate platform account is now active'
result = re.findall(pattern, test_string)
```

RegExes are extremely useful, but the syntax can be hard to recall. With that in mind, ActiveState offers this "cheatsheet" to help point you in the right direction when building RegExes in Python.

Special characters

.	match any char except newline (eg., ac..ve)
^	match at beginning of string (eg., ^active)
$	match at end of string (eg, state$)
[3a-c]	match any char (ie., 3 or a or b or c)
[^x-z1]	match any char except x, y, z or 1
A|S	match either A or S regex
()	capture & match a group of chars (eg., (8097ba))
\	escape special characters

Quantifiers

*	match 0 or more occurrences (eg., py*)
+	match 1 or more occurrences (eg., py+)
?	match 0 or 1 occurrences (eg., py?)
{m}	match exactly m occurrences (eg., py{3})
{m,n}	match from m to n occurrences (eg., py{1,3})
{,n}	match from 0 to n occurrences (eg., py{,3})
{m,}	match m to infinite occurrences (eg., py{3,})
{m,n}	match m to n occurrences, but as few as possible (eg., py{1,3}?)

Special sequences

\A	match occurrence only at start of string
\Z	match occurrence only at end of string
\b	match empty string at word boundary (e.g., between \w and \W)
\B	match empty string not at word boundary
\d	match a digit
\D	match a non-digit
\s	match any whitespace char: [\t\n\r\f\v]
\S	match any non-whitespace char
\w	match any alphanumeric: [0-9a-zA-Z_]
\W	match any non-alphanumeric
\g<id>	matches a previously captured group
(?:A)	match expression represented by A (non-capture group)
A(?=B)	match expression A only if followed by B
A(?!B)	match expression A only if not followed by B
(?<=B)A	match expression A only if it follows B
(?<!B)A	match expression A only if not preceded by B
(?aiLmsux)	where a, i, L, m, s, u, and x are flags:

a =	match ASCII only
i =	make matches ignore case
L =	make matches locale dependent
m =	multi-line makes ^ and $ match at the beginning/end of each line, respectively
s =	makes '.' match any char including newline
u =	match unicode only
x =	verbose increases legibility by allowing comments & ignoring most whitespace

re Module Functions

Besides enabling the above functionality, the 're' module also features a number of popular functions:

re.findall(A, B)	match all occurrences of expression A in string B
re.search(A, B)	match the first occurrence of expression A in string B
re.split(A, B)	split string B into a list using the delimiter A
re.sub(A, B, C)	replace A with B in string C

Need more help?
Read the documentation here:
https://docs.python.org/3/library/re.html