A **regular expression** (regex) defines or describes a search pattern.
Literal characters always match themselves. For example, the search pattern ‘dark’ matches the word ‘dark’ in the line ‘It was a dark and stormy night’.
Metacharacters have special meanings. To match a metacharacter, put a backslash before it in the regular expression.

Regular Expression	**Description**	**Input String**	**Match**
‘dark’	Matches ‘dark’	‘It was a dark and stormy night’	‘dark’
‘a’	Matches the first ‘a’	‘It was a dark and stormy night’	The ‘a’ in ‘was’.
‘[ae]’	Matches either ‘a’ or ‘e’. Note use of brackets as metacharacters.	‘easy’	‘e’
‘gr[ae]y’	Could match either ‘gray’ or ‘grey’	‘grey’	grey’
‘[a-z]’	Matches any lowercase character	‘grey’	‘g’
‘[a-zA-Z]’	Matches any alphabetical character	‘Jack and Jill’	The first ‘J’
‘[0-9]’ ‘\d’	Matches any digit. ‘\d’ pattern uses the backslash metcharacter	‘1776’	‘1’
‘[a-zA-Z]+’	Matches a word. + previous expression can be matched one or more times	‘Jack and Jill’	‘Jack’
‘a*bc’	* previous expression can be matched 0 or more times	Matches ‘bc’ or ‘abc’ or ‘aaaaabc’	
‘[a-zA-Z\s]+’	\s matches any whitespace character	‘The Fifth Season (Orbit)’	‘The Fifth Season ‘
‘[\w\s]+\(Orbit\)’	(is a metacharacter so use a backslash to search it. \w matches any alphanumeric character	‘The Fifth Season (Orbit)’	‘(Orbit)’
‘([a-zA-Z]+) and ([a-zA-Z]+)’	Grouping allows you to pull out data of interest. You group by surrounding subpatterns with parentheses	‘Jack and Jill’	‘Jack’ ‘Jill’

Grouping allows you to pull out data of interest. The following example pulls out one word but grouping easily allows multiple words to be matched and returned.

Code	**Code with Grouping**	**Output**
import re line = ‘It was a dark and stormy night’ p = re.compile(‘dark’) m = p.match(line) print (line[m.start():m.end()])	import re line = ‘It was a dark and stormy night’ p = re.compile(‘(dark)’) m = p.match(line) print (m.group(1))	‘dark’

https://github.com/JudythG/RegEx-Demo