Klb biology book 1 pdf download

I'm not robot
reCAPTCHA

Continue

Regular expressions are a special sequence of characters that can help you easily check if a line fits a pattern. Python has added a re module with version 1.5 that provides regular Perl-style expression templates. The re module gives Python all the regular expression functionality. The compilation function generates a normal expression object based on the pattern line and additional flag settings. The object has a number of methods for regularly mapping and replacing the expression. The re module also provides features that accurately match the functionality of these methods, using the template line as the first argument. This section focuses on the regular expression processing functions commonly used in Python, and if you don't know about regular expressions, check out our regular expressions - a tutorial. Re.match re.match tries to match the pattern from the beginning of the line, and if the starting position match is not successful, the match will not return. Syntax function: re.match (pattern, string, flags)0) Feature parameter description: Argument describes a line that will correspond to the usual line of expression of the template. Flag flag bits that control how regular expressions match, such as insiness of cases, multi-layered alignment, and so on. See also: Regular Expression Modifier - Extra Flag Match Successful Re.match returns the appropriate object, otherwise not? We can use a group (num) or a group to compare the functions of objects to compare expressions. The match object method describes the entire expression's string by grouping (num-0), which can enter more than one group at a time, in which case a metagroup containing values corresponding to those groups is returned. Groups () returns a metagroup containing all group lines from 1 to the group number contained in . Import reprint ('www', 'www.runoob.com'span.span)) print ('re.match','www.runoob.com')) above the instance launches the results of the output as follows: (0, 3) No import re-line with Cats smarter than dog matchObj. re.match (r'.) are (.?) .', line, re. m're. I) if matchObj: print (matchObj.group (), matchObj.group ()) print (matchObj.group (1): , matchObj.group (1) print (matchObj.group (2), matchObj.group (2)) still: print (No Match!!) The above example performs as follows: matchObj.group (): Cats are smarter than dogs matchObj.group (1): Cats matchObj.group (2): smarter re.search method re.search scans the entire line and returns the first successful match. string, Flags,0) Feature description: The argument describes a line that would match the usual line of expression pattern. that controls how regular expressions match, such as insensitivity multi-layered compliance, and so on. See also: Regular Expression Modifier - Extra Flag Match Successful Re-Search Method returns the appropriate object, otherwise no one returns. We can use a group (num) or a group to compare the functions of objects to compare expressions. The match object method describes the line of the entire expression that connects by group (num-0), group () () Groups () returns a metagroup containing all group lines from 1 to the group number contained in . Re.search ('www', 'www.runoob.com'.span()) print ('re.search', 'www.runoob.com'span)) the above copy launches output results as follows: (0, 3) (11, 14) import re line Cats smarter than dogs searchj's re.search (r'M'M'? I) if searchObj: print (searchObj.group(), searchObj.group ()) print (searchObj.group (1), searchObj.group (1)print (2), searchObj.group (2)) still: print (nothing found!!) than the dogs searchObj.group (1): Cats searchObj.group (2): The difference between smarter re.match and re.search re.match only corresponds to the top of the line, if the line starts not to match the usual expression, the match fails, the feature returns No. , and the feature returns No, and re-search will not be found. Imports re-line with Cats smarter than dogs matchObj s re.match (r'dogs', line, re. m're. I) if matchObj: seal (matchobj.group), matchObj.group () still: seal (No match!!) matchObj s re.search (r'dogs', line, re. M're. I) if matchObj: print (search -- matchObj.group (), matchObj.group ()) another: print (No match!!) The above example works as follows: No match!! MatchObj.group: Dogs remove and replace the Re-Python module to provide re.sub to replace matches in rows. Syntax: re.sub (pattern, repl, string, count-0, flags-0) Argument: pattern: pattern line in positive. repl : Replaces a line that can also be a feature. Line : The original line that needs to be found to replace. Count: The maximum number of times a template is replaced after a pattern is matched, and the default 0 indicates that all matches will be replaced. Flags: The corresponding pattern used during the compilation is in numerical form. The first three are necessary and the last two are optional. import re-phone No 2004-959-559 这是个电话号码 num - re.sub (r'... $, phone) print (电话号码:, num) num and re.sub (r'D', , phone) print (电话号码: , num) 以上实例执行结果如下: 电话号码: 2004-959-559 电话号码: 20049595559 repl 参数是个函数 以下实例电话号码将字符串的匹配的数字乘于 2: Import re def double (appropriate): Int (matched.group ('value')) return str (value No. 2) s 'A23G4HFD567' print (re.sub,?? The output of the A46G8HFD1134 compilation function is a regular expression and generates a normal expression object (pattern) for use as a match and a search function. : re.compile (pattern, flags)) pattern: regular expression of flags in the form of a line is optional, specifying appropriate patterns, such as ignoring the case, multi-layered patterns, etc., specific parameters: re. L means a special set of s characters s, s, s, s, s зависит от текущей среды re. M многослойный режим re. S является ' . и любой символ, включая разрывы линии ('. </value> </value> Again. U is a special set of characters .w,. W, .b, .B. , .d, .D. , s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, s, , , ,, , , , , ,, ,, X In order to increase readability, ignore the gaps and comments after the 'zgt'gt'gt;gt; s compile.' 2, 10) (m _sre) (m) SRE_Match object at 0x10a42aac0((m) m.group (0) '12' (0) m.start (0) 3 m.gt; m.end (0) 5 m.span (0) (3, 5) above, returning the object of the match when the match is successful, where: the group. ((group1, ...)) The method is used to produce one or more lines that correspond to the group, and if you want to get the entire suitable sub-family chain, you can use the group method () or group (0) directly; The end method (group) is used to get the final position of the subsection line, the line across the line (the last pick-up symbol index) and the default of the argument 0; range (group) returns (group). Look at another example: 're-re.compile (r'z)' pattern, re. I) >>> m s pattern.match ('Hello World Wide Web') >>> print <_sre. SRE_Match object at q 0x10bea83e8 (m) m.gt; m.group (0) 'Hello World' (0) m.span (0) (0, 11) m.span (1) (0, 5) (0, 5) ('Hello', Hello 'World' (3) File, line 1, IndexError: No such findall group finds all subsections connected by the usual zlt'z1'gt;expressions in the line and returns the list, if a match is not found, a blank list is returned. Note: Match and search match once findall coincides with all. Syntax format: re.findall (pattern, string, flags-0) or pattern.findall (string, pos, endpos)) options: pattern matching pattern. A line that will match. Pos is an additional setting that determines the starting position of the line, which by default is up to 0. An additional Endpos option that determines the final position of the line, which is by default by the length of the line. Find all the numbers in line: import re result1 s re.findall (r'd', 'runoob 123 Google 456') pattern re.compile (r'd') result2 s pattern.findall ('runoob 123 google 456') result3 s pattern.findall ('run'88o123gogogle456', 0, 10) Seal (result1) seal (result2) print (result3) Exit: '123', '456'. . . . . re.finditer (pattern, string, flags-0) argument: The argument describes a line that will match the pattern-consistent line of expression. Flag flag bits that control how regular expressions match, such as insiness of cases, multi-layered alignment, and so on. See also: Regular expression modifier - zlt; module/gt; zlt;gt; _sre. SRE_Match> </_sre. SRE_Match> . . import reit s re.finditer (r sd, 12a32bc43jf3) to match in it: print (match.group()Exit: 12 32 43 3 re.split method returns the list after splitting the line according to the corresponding subnet line, which is used as follows: re.split, string, string, maxsplit-0, flags'0) Argument: Argument describes a line that should be matched with the usual line of expression. Maxsplit is separated once, maxsplit is separated once, by default 0, without limiting the number of times. Flag flag bits that control how regular expressions match, such as insiness of cases, multi-layered alignment, and so on. See also: Regular Expression Modifier - Additional Flag , Re.gt.split (W, 'runoob, runoob, runoob.) 'runoob', 'runoob', 'runoob', "' >>>'re.split', 'runoob, runoob, runoob.') '', '', 'runoob', ', 'runoob', '', 'runoob', '.', ''>>> >>>'''''''