Table 13.6 *Regular Expression Object Methods*

Syntax	Description
`rx.findall(s, start, end)`	Returns all nonoverlapping matches of the regex in string `s` (or in the *start*:*end* slice of `s`). If the regex has captures, each match is returned as a tuple of captures.
`rx.finditer(s, start, end)`	Returns a match object for each nonoverlapping match in string `s` (or in the *start*:*end* slice of `s`)
`rx.flags`	The flags that were set when the regex was compiled
`rx.groupindex`	A dictionary whose keys are capture group names and whose values are group numbers; empty if no names are used
`rx.match(s, start, end)`	Returns a match object if the regex matches at the start of string `s` (or at the start of the *start*:*end* slice of `s`); otherwise, returns `None`
`rx.pattern`	The string from which the regex was compiled
`rx.search(s, start, end)`	Returns a match object if the regex matches anywhere in string `s` (or in the *start*:*end* slice of `s`); otherwise, returns `None`
`rx.split(s, m)`	Returns the list of strings that results from splitting string `s` on every occurrence of the regex doing up to `m` splits (or as many as possible if no `m` is given). If the regex has captures, these are included in the list between the parts they split.
`rx.sub(x, s, m)`	Returns a copy of string `s` with every (or up to `m` if given) match replaced with `x`—this can be a string or a function; see text
`rx.subn(x, s m)`	The same as `re.sub()` except that it returns a 2-tuple of the resultant string and the number of substitutions that were made

One common task is to take an HTML text and output just the plain text that it contains. Naturally we could do this using one of Python's parsers, but a simple tool can be created using regexes. There are three tasks that need to be done: delete any tags, replace entities with the characters they represent, and insert blank lines to separate paragraphs. Here is a function (taken from the `html2text.py` program) that does the job:

```
def html2text(html_text):
    def char_from_entity(match):
        code = html.entities.name2codepoint.get(match.group(1), 0xFFFD)
        return chr(code)
```