Ruby - Strings - Iteration

In our encoding tutorial, we learned about the different ways Ruby 1.8 and Ruby 1.9 (and higher versions) represent strings internally. The major difference is a wide range of encoding (non-ascii) support in the later versions. This change, however, also overhauls the way strings were iterated between the two versions.

In Ruby 1.8, there's a single `each` method (remember Enumerable?) which allowed it to iterate over lines of data. While it might seem like a logical option to have, how would one go about iterating on each byte or each character? It turns out that it was not so clean, and people had to resort to *tricks* for some of these functionalities.

With Ruby 1.9, `each` was removed from the `String` class and is no longer an Enumerable. Instead, we have more explicit choices based on what we need to iterate - bytes, chars, lines or codepoints.

- `each_byte` iterates sequentially through the individual bytes that comprise a string;
- `each_char` iterates the characters and is more efficient than `[]` operator or character indexing;
- `each_codepoint` iterates over the ordinal values of characters in the string;
- `each_line` iterates the lines.

For example:

```
> money = "¥1000"
> money.each_byte {|x| p x} # first char represented by two bytes
194
165
49
48
48
48
> money.each_char {|x| p x} # prints each character
"¥"
"1"
"0"
"0"
"0"
```

Without a doubt, Ruby 1.9 makes iteration easier to understand and implement. Hence, we'll stick with Ruby 1.9 and later versions for current and other challenges (unless otherwise stated).

Challenge: Write the method `count_multibyte_char` which takes a string as input and returns the number of multibyte characters (byte size > 1) in it.

For example:

```
> count_multibyte_char('¥1000')
1
```