

```
print "Phone Num : ", num
```

When the above code is executed, it produces the following result:

```
Phone Num : 2004-959-559
```

```
Phone Num : 2004959559
```

## Regular-Expression Modifiers: Option Flags

Regular expression literals may include an optional modifier to control various aspects of matching. The modifiers are specified as an optional flag. You can provide multiple modifiers using exclusive OR (`|`), as shown previously and may be represented by one of these:

Modifier	Description
re.I	Performs case-insensitive matching.
re.L	Interprets words according to the current locale. This interpretation affects the alphabetic group ( <code>\w</code> and <code>\W</code> ), as well as word boundary behavior ( <code>\b</code> and <code>\B</code> ).
re.M	Makes <code>\$</code> match the end of a line (not just the end of the string) and makes <code>^</code> match the start of any line (not just the start of the string).
re.S	Makes a period (dot) match any character, including a newline.
re.U	Interprets letters according to the Unicode character set. This flag affects the behavior of <code>\w</code> , <code>\W</code> , <code>\b</code> , <code>\B</code> .
re.X	Permits "cuter" regular expression syntax. It ignores whitespace (except inside a set <code>[]</code> or when escaped by a backslash) and treats unescaped <code>#</code> as a comment marker.

## Regular-Expression Patterns

Except for control characters, `(+ ? . * ^ $ ( ) [ ] { } | \)`, all characters match themselves. You can escape a control character by preceding it with a backslash.

Following table lists the regular expression syntax that is available in Python:

279