2.3.6.2 String Formatting Operations

String and Unicode objects have one unique built-in operation: the `%` operator (modulo). This is also known as the string *formatting* or *interpolation* operator. Given *format* `%` *values* (where *format* is a string or Unicode object), `%` conversion specifications in *format* are replaced with zero or more elements of *values*. The effect is similar to the using `sprintf()` in the C language. If *format* is a Unicode object, or if any of the objects being converted using the `%s` conversion are Unicode objects, the result will also be a Unicode object.

If *format* requires a single argument, *values* may be a single non-tuple object.[2.8] Otherwise, *values* must be a tuple with exactly the number of items specified by the format string, or a single mapping object (for example, a dictionary).

A conversion specifier contains two or more characters and has the following components, which must occur in this order:

1. The "`%`" character, which marks the start of the specifier.
2. Mapping key (optional), consisting of a parenthesised sequence of characters (for example, `(somename)`).
3. Conversion flags (optional), which affect the result of some conversion types.
4. Minimum field width (optional). If specified as an "`*`" (asterisk), the actual width is read from the next element of the tuple in *values*, and the object to convert comes after the minimum field width and optional precision.
5. Precision (optional), given as a "`.`" (dot) followed by the precision. If specified as "`*`" (an asterisk), the actual width is read from the next element of the tuple in *values*, and the value to convert comes after the precision.
6. Length modifier (optional).
7. Conversion type.

When the right argument is a dictionary (or other mapping type), then the formats in the string *must* include a parenthesised mapping key into that dictionary inserted immediately after the "`%`" character. The mapping key selects the value to be formatted from the mapping. For example:

```
>>> print '%(language)s has %(#)03d quote types.' % \
          {'language': "Python", "#": 2}
Python has 002 quote types.
```

In this case no `*` specifiers may occur in a format (since they require a sequential parameter list).

The conversion flag characters are:

Flag	Meaning
`#`	The value conversion will use the ``alternate form'' (where defined below).
`0`	The conversion will be zero padded for numeric values.
`-`	The converted value is left adjusted (overrides the "`0`" conversion if both are given).
	(a space) A blank should be left before a positive number (or empty string) produced by a signed conversion.
`+`	A sign character ("`+`" or "`-`") will precede the conversion (overrides a "space" flag).

A length modifier (`h`, `l`, or `L`) may be present, but is ignored as it is not necessary for Python.

The conversion types are:

Conversion	Meaning	Notes
`d`	Signed integer decimal.	
`i`	Signed integer decimal.	
`o`	Unsigned octal.	(1)
`u`	Unsigned decimal.	
`x`	Unsigned hexadecimal (lowercase).	(2)
`X`	Unsigned hexadecimal (uppercase).	(2)
`e`	Floating point exponential format (lowercase).	
`E`	Floating point exponential format (uppercase).	
`f`	Floating point decimal format.	
`F`	Floating point decimal format.	
`g`	Same as "`e`" if exponent is greater than -4 or less than precision, "`f`" otherwise.	
`G`	Same as "`E`" if exponent is greater than -4 or less than precision, "`F`" otherwise.	
`c`	Single character (accepts integer or single character string).	
`r`	String (converts any python object using `repr()`).	(3)
`s`	String (converts any python object using `str()`).	(4)
`%`	No argument is converted, results in a "`%`" character in the result.	

Notes:

(1)
The alternate form causes a leading zero ("`0`") to be inserted between left-hand padding and the formatting of the number if the leading character of the result is not already a zero.

(2)
The alternate form causes a leading `'0x'` or `'0X'` (depending on whether the "`x`" or "`X`" format was used) to be inserted between left-hand padding and the formatting of the number if the leading character of the result is not already a zero.

(3)
The `%r` conversion was added in Python 2.0.

(4)
If the object or format provided is a `unicode` string, the resulting string will also be `unicode`.

Since Python strings have an explicit length, `%s` conversions do not assume that `'\0'` is the end of the string.

For safety reasons, floating point precisions are clipped to 50; `%f` conversions for numbers whose absolute value is over 1e25 are replaced by `%g` conversions.[2.9] All other errors raise exceptions.

Additional string operations are defined in standard modules `string` and `re`.

Footnotes

... object.[2.8]
To format only a tuple you should therefore provide a singleton tuple whose only element is the tuple to be formatted.

... conversions.[2.9]

These numbers are fairly arbitrary. They are intended to avoid printing endless strings of meaningless digits without hampering correct use and without having to know the exact precision of floating point values on a particular machine.