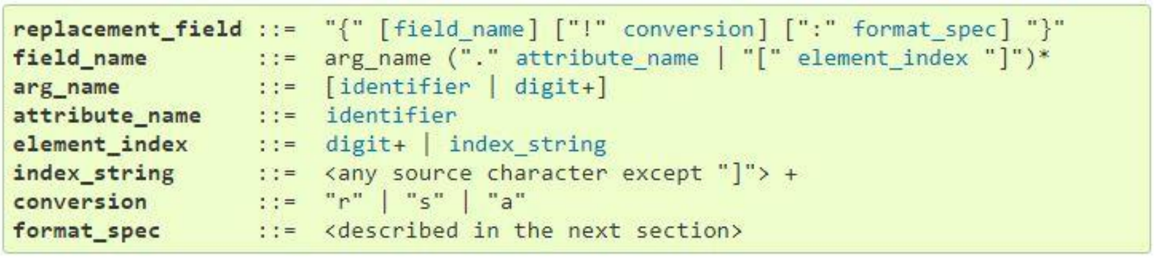
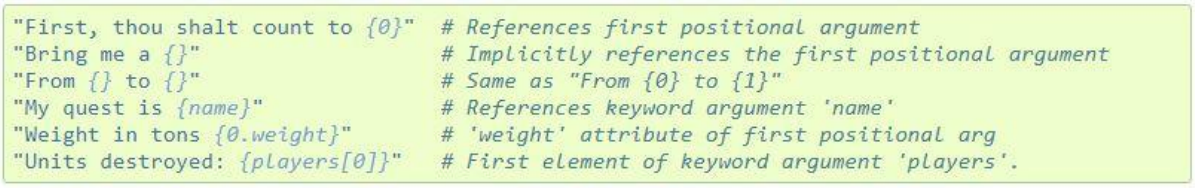
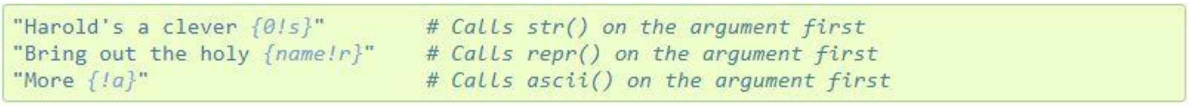


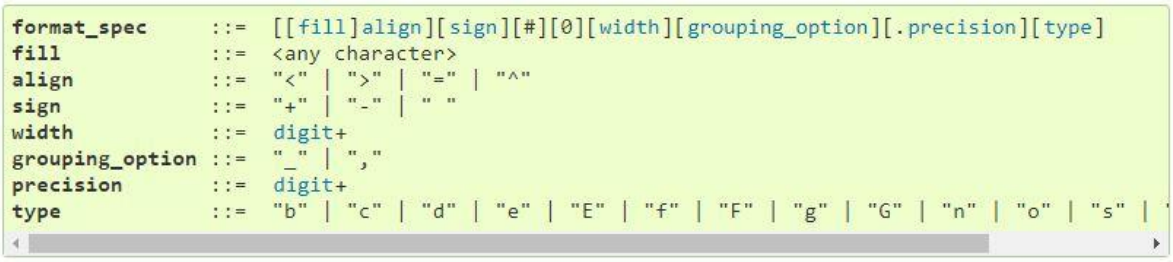
```
replacement_field ::=  "{" [field_name] ["!" conversion] [":" format_spec] "}"
field_name        ::=  arg_name ("." attribute_name | "[" element_index "]")*
arg_name          ::=  [identifier | digit+]
attribute_name    ::=  identifier
element_index     ::=  digit+ | index_string
index_string      ::=  <any source character except "]"> +
conversion        ::=  "r" | "s" | "a"
format_spec       ::=  <described in the next section>
```



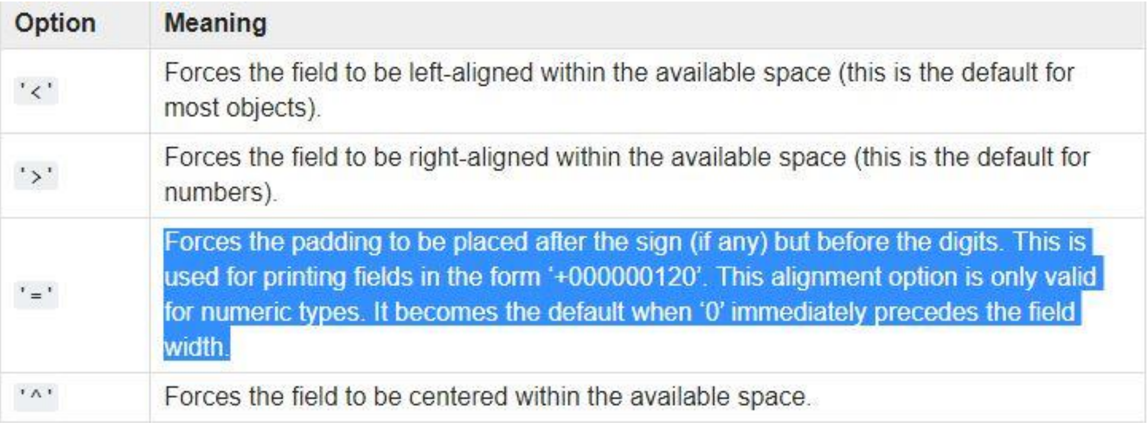
```
"First, thou shalt count to {0}"  # References first positional argument
"Bring me a {}"                   # Implicitly references the first positional argument
"From {} to {}"                   # Same as "From {0} to {1}"
"My quest is {name}"              # References keyword argument 'name'
"Weight in tons {0.weight}"       # 'weight' attribute of first positional arg
"Units destroyed: {players[0]}"   # First element of keyword argument 'players'.
```



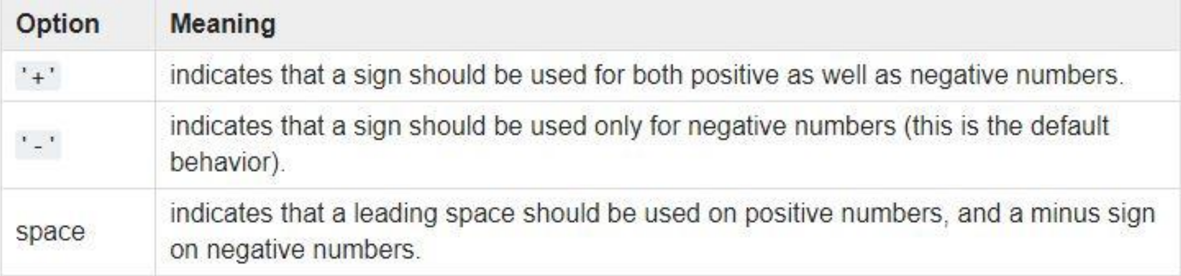
```
"Harold's a clever {0!s}"        # Calls str() on the argument first
"Bring out the holy {name!r}"    # Calls repr() on the argument first
"More {!a}"                      # Calls ascii() on the argument first
```



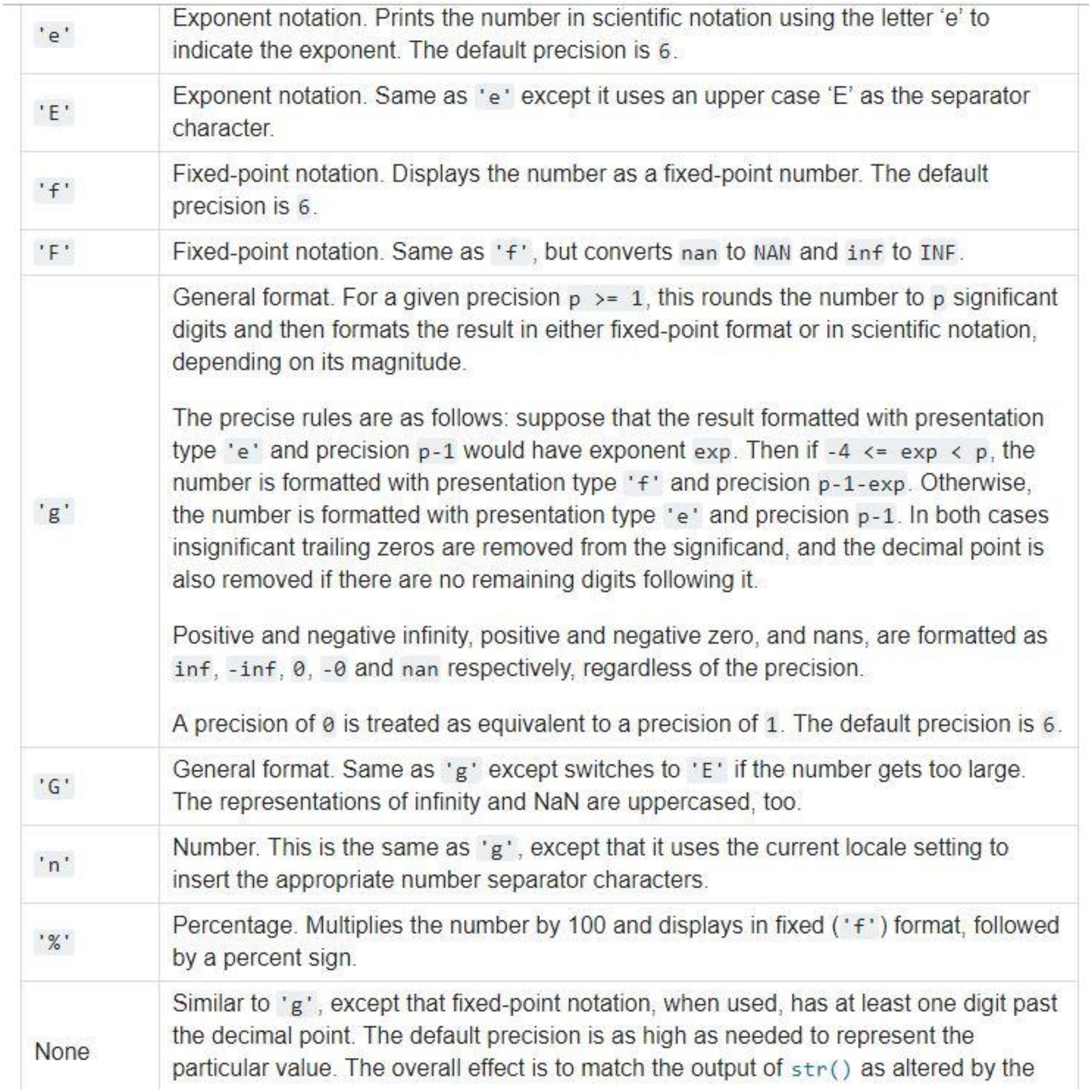
```
format_spec     ::=  [[fill]align][sign][#][0][width][grouping_option][.precision][type]
fill            ::=  <any character>
align           ::=  "<" | ">" | "=" | "^"
sign            ::=  "+" | "-" | " "
width           ::=  digit+
grouping_option ::=  "_" | ","
precision       ::=  digit+
type            ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" |
```



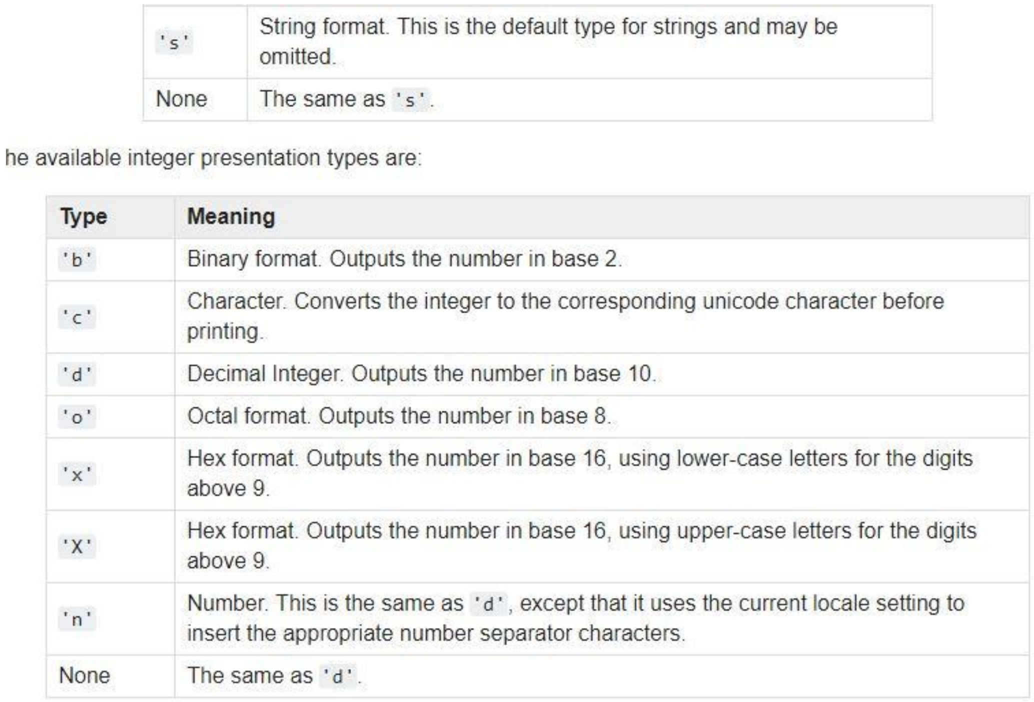
Option	Meaning
`'<'`	Forces the field to be left-aligned within the available space (this is the default for most objects).
`'>'`	Forces the field to be right-aligned within the available space (this is the default for numbers).
`'='`	Forces the padding to be placed after the sign (if any) but before the digits. This is used for printing fields in the form '+000000120'. This alignment option is only valid for numeric types. It becomes the default when '0' immediately precedes the field width.
`'^'`	Forces the field to be centered within the available space.



Option	Meaning
`'+'`	indicates that a sign should be used for both positive as well as negative numbers.
`'-'`	indicates that a sign should be used only for negative numbers (this is the default behavior).
space	indicates that a leading space should be used on positive numbers, and a minus sign on negative numbers.



`'e'`	Exponent notation. Prints the number in scientific notation using the letter 'e' to indicate the exponent. The default precision is `6`.
`'E'`	Exponent notation. Same as `'e'` except it uses an upper case 'E' as the separator character.
`'f'`	Fixed-point notation. Displays the number as a fixed-point number. The default precision is `6`.
`'F'`	Fixed-point notation. Same as `'f'`, but converts `nan` to `NAN` and `inf` to `INF`.
`'g'`	General format. For a given precision `p >= 1`, this rounds the number to `p` significant digits and then formats the result in either fixed-point format or in scientific notation, depending on its magnitude. The precise rules are as follows: suppose that the result formatted with presentation type `'e'` and precision `p-1` would have exponent `exp`. Then if `-4 <= exp < p`, the number is formatted with presentation type `'f'` and precision `p-1-exp`. Otherwise, the number is formatted with presentation type `'e'` and precision `p-1`. In both cases insignificant trailing zeros are removed from the significand, and the decimal point is also removed if there are no remaining digits following it. Positive and negative infinity, positive and negative zero, and nans, are formatted as `inf`, `-inf`, `0`, `-0` and `nan` respectively, regardless of the precision. A precision of `0` is treated as equivalent to a precision of `1`. The default precision is `6`.
`'G'`	General format. Same as `'g'` except switches to `'E'` if the number gets too large. The representations of infinity and NaN are uppercased, too.
`'n'`	Number. This is the same as `'g'`, except that it uses the current locale setting to insert the appropriate number separator characters.
`'%'`	Percentage. Multiplies the number by 100 and displays in fixed (`'f'`) format, followed by a percent sign.
None	Similar to `'g'`, except that fixed-point notation, when used, has at least one digit past the decimal point. The default precision is as high as needed to represent the particular value. The overall effect is to match the output of `str()` as altered by the



`'s'`	String format. This is the default type for strings and may be omitted.
None	The same as `'s'`.

he available integer presentation types are:

Type	Meaning
`'b'`	Binary format. Outputs the number in base 2.
`'c'`	Character. Converts the integer to the corresponding unicode character before printing.
`'d'`	Decimal Integer. Outputs the number in base 10.
`'o'`	Octal format. Outputs the number in base 8.
`'x'`	Hex format. Outputs the number in base 16, using lower-case letters for the digits above 9.
`'X'`	Hex format. Outputs the number in base 16, using upper-case letters for the digits above 9.
`'n'`	Number. This is the same as `'d'`, except that it uses the current locale setting to insert the appropriate number separator characters.
None	The same as `'d'`.