Cheatography

python string formatting Cheat Sheet

by mutanclan (mutanclan) via cheatography.com/79625/cs/19406/

Field definitions

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	`Format Specification Mini-Language`

field_name

The *replacement_field* can start with a *field_name* to specify the object whose value is to be formatted and inserted.
The *field_name* begins with an *arg_name*. The *arg_name* can be followed by any number of index or attribute expressions.

arg_name

An *arg_name* is either a number or a keyword. If it's a number it refers to a positional argument. If it's a keyword, it refers to a named keyword argument. If the numerical *arg_names* in a format string are 0,1,2 in sequence, the can be omitted (They are automatically inserted).

attribute_name

An expression of the form `'.name'` selects the named attribute using **getattr()**

element_index

An expression of the form `'[index]'` does an index lookup using **__getitem__()**.
For example:
List index: `[0]`
Dictionary: `[name]`

conversion

!s	calls **str()**
!r	calls **repr()**
!a	calls **ascii()**

The *conversion* field forces a type conversion **before** formatting, so not by the **__format__()** method of the value itself.

String presentation types

`s`	String format. This is the default for strings
None	The same as `s`

Integer presentation types

`b`	Binary format. Outputs the number in base 2
`c`	Character. Converts the integer to unicode
`d`	Decimal integer. Outputs number in base 10
`o`	Octal format. Outputs number in base 8
`x`	Hex format. Outputs number in base 16 using lowercase letters
`X`	Hex format. Outputs number in base 16 using uppercase letters
`n`	Number. Same as `d` but uses current locale setting for the separator
None	Same as `d`

Format Specification Mini-Language

format_spec	`[[fill]align][sign][#][0][width][grouping_option][.precision][type]`
fill	`<any character>`
align	`"<" | ">" | "=" | "^"`
sign	`"+" | "-" | " "`
width	`digit+`

Cheatography

python string formatting Cheat Sheet

by mutanclan (mutanclan) via cheatography.com/79625/cs/19406/

Format Specification Mini-Language (cont)

grouping_option	`"_" | ","`
precision	`digit+`
type	`"b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"`

fill, sign and align

`<`	Force left-alignment within available space
`>`	Force rigth-alignment within available space
`=`	Only valid for numeric types. Forces the padding to be placed after the *sign* but before the digits
`^`	Forces the field to be centered within available space
`+`	Use a *sign* for both positiv and negative numbers
`-`	Use *sign* only for negative numbers
`space`	Use a leading space for positiv numbers and a minus sign for negative numbers
`#`	Causes the alternate form to be used for the conversion. binary: `0b`, octal: `0o` and hex: `0x`. For floats, complex and Decimal types that causes to contain a trailing decimal-point even if no digits follow it
`,`	Use `,` for thousands separator
`_`	Use _ for thousands separator

If an *align* value is specified it can be preceded by a *fill* character, that can be any character (default is space)

The *sign* option is only valid on numeric types

width and precision

width is a decimal integer defining the minimum field width. A leading `0` enables sign-aware zero-padding for numeric types.

precision is a decimal number indicating how many digits should be displayed after the decimal point. For non-number types it indicates the maximum field-size. Not allowed for integer values

Floating point and decimal presentation types

`e`	Exponent notation using the letter **e** to indicate the exponent. Default *precision* is `6`
`E`	Exponent notation. Same as `e` but with uppercase **E**
`f`	Fixed-point notation. Default *precision* is `6`
`F`	Fixed-point notation. Same as `f` but converts `nan` to `NAN` and `inf` to `INF`
`g`	General format. If *precision* is `p>=1` this rounds the number to `p` significant digits. Output format is either fixed-point or in scientific notation, depending on the magnitude
`G`	General format. Same as `g` but switches to `E` if the number gets too large.
`n`	Number. Same as `g` but use current locale for the number separator character
`%`	Percentage. Multiplies number by 100 and displays it in `f` format followed by a percentage sign
None	Same as `g` but fixed-point notation has at least one digit past the decimal point