Format () function in Python

The format () function is built into Python to use to format an input value into a specific format.

The format () function is built into Python to use to format an input value into a specific format.

Syntax of format () function in Python:

```
format(value[, format_spec])
```

Parameters of format function ()

The format () function has 2 parameters:

1. `value` : value to be formatted.
2. `format_spec` : the format you want for the *value.*

Format sets:

Type
Describe

Align the result = Set a positive sign (+) or negative (-) in the left margin d Integer format c Unicode character corresponding b Binary format o Definition octal format x Hexadecimal format (lower case) X Hexadecimal format (uppercase) n Decimal format e Hat notation. (lowercase e) E Hat notation. (uppercase E) f Real number floating point format (Default: 6) F Same as 'f', but displays uppercase letters, for example 'inf' is 'INF' and 'nan' is 'NAN' g General format. Round numbers to p significant digits. (Default: 6) G Same as 'g', switch to 'E' if the number is too large. % Percentage.
The order format is as follows:

```
[[fill]align][sign][#][0][width][,][.precision][type]
```

Inside:

1. *fill:* is any character
2. *align:* alignment, "" | ">" | "=" | "^"
3. *sign:* sign, "+" | "-" | ""
4. *width:* string length, is an integer
5. *precision:* precision, is an integer
6. *type:* format type, "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"

Value returned from format ()

The format () method returns the formatted result of a given value determined by a specific format.

Example 1: Number format with format ()

```
 # d, f và b là lo?i ??nh d?ng # s? nguyên print(format(123, "d")) # s? th?
p phân # vi?t b?i TipsMake.com print(format(123.4567898, "f")) # nh?
 phân print(format(12, "b"))
```

Run the program, the result is:

```
 123 123.456790 1100
```

Example 2: Number format with alignment

```
 # s? nguyên print(format(1234, "*>+7,d")) # s? th?
p phân print(format(123.4567, "^-09.3f"))
```

Output returns:

```
 *+1,234 0123.4570
```

Here, when formatting the integer 1234, we specify the format including `*+7,d` . Why is that? Details are as follows:

1. `*` Character fill, fill in the blank after formatting.
2. `>` Right alignment option, so the output string is on the right.
3. `+` Option to force the output to be signed (with the sign on the left).
4. `7` Optionally specify the length of the result, forcing the return number to have a minimum length of 7 characters, other spaces will be filled with fill characters.
5. `,` Thousands operators, commas will be placed between thousands of digits and the remaining digits.
6. `d` option to specify the result is an integer.

When formatting floating point numbers 123.4567, we specify the format including `^-09.3f` , details are as follows:

1. `^` option to align the center position, so the output sequence will be in the middle of the specified space.
2. `-` Option to force the output to display a negative sign.
3. `0` Characters added to the spaces after formatting.
4. `9` Optionally specify the length of the result, forcing the return number to have a minimum length of 9 characters (including decimal points, commas, and a negative sign).
5. `.3` precision operator, determine the precision, round to the third digit after the decimal point.
6. `f` option to specify the result to return is a floating-point number.

Example 3: Use format () by overwriting __format __ ()

```
 # ph??ng th?c __format __ () tùy ch?
nh class Person: def __format__(self, format): if(format == 'age'): return '23' r
```

Run the program, the result is:

```
23
```

Here, we override the *__format__* () method of the Person class, and the *format () method* inside of running `Person().__format__("age")` to return `23`.

See also: Built-in Python functions

You finished reading the article "**Format () function in Python**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
