**-:DTL Tags:-**

**# Without closing tag**

1. {% csrf\_token %}

2. {% extends %}

3. {% include %}

4. {% load %}

5. {% url %}

6. {% cycle %}

**# With closing tag:**

7. {% with %}

{% endwith %}

8. {% if (condition) %}

{% elif (condition) %}

{% else %}

{% endif %}

9. {% for x in condition %}

{% endfor %}

10. {% block content %}

{% endblock %}

11. {% comment %}

{% endcomment %}

### 12. for … empty

The for tag can take an optional {% empty %} clause whose text is displayed if the given array is empty or could not be found:

<ul>

{% for athlete in athlete\_list %}

<li>{{ athlete.name }}</li>

{% empty %}

<li>Sorry, no athletes in this list.</li>

{% endfor %}

</ul>

The above is equivalent to – but shorter, cleaner, and possibly faster than – the following:

<ul>

{% if athlete\_list %}

{% for athlete in athlete\_list %}

<li>{{ athlete.name }}</li>

{% endfor %}

{% else %}

<li>Sorry, no athletes in this list.</li>

{% endif %}

</ul>

**DTL filter**

**{{Key}} |**

### Add:---

Adds the argument to the value.

For example:

{{ value **|** add:"2" }}

If **value** is **4**, then the output will be **6**.

This filter will first try to coerce both values to integers. If this fails, it’ll attempt to add the values together anyway. This will work on some data types (strings, list, etc.) and fail on others. If it fails, the result will be an empty string.

For example, if we have:

{{ first|add:second }}

and **first** is **[1, 2, 3]** and **second** is **[4, 5, 6]**, then the output will be **[1, 2, 3, 4, 5, 6]**.

### Capfirst:----

Capitalizes the first character of the value. If the first character is not a letter, this filter has no effect.

For example:

{{ value|capfirst }}

If **value** is **"django"**, the output will be **"Django"**.

### Center

Centers the value in a field of a given width.

For example:

"{{ value|center:"15" }}"

If **value** is **"Django"**, the output will be **"     Django    "**.

### ljust

Left-aligns the value in a field of a given width.

**Argument:** field size

For example:

"{{ value|ljust:"10" }}"

If **value** is **Django**, the output will be **"Django    "**.

### rjust

Right-aligns the value in a field of a given width.

**Argument:** field size

For example:

"{{ value|rjust:"10" }}"

If **value** is **Django**, the output will be **"    Django"**.

### cut

Removes all values of arg from the given string.

For example:

{{ value|cut:" " }}

If **value** is **"String with spaces"**, the output will be **"Stringwithspaces"**.

### default

If value evaluates to **False**, uses the given default. Otherwise, uses the value.

For example:

{{ value|default:"nothing" }}

If **value** is **""** (the empty string), the output will be **nothing**.

### default\_if\_none

If (and only if) value is **None**, uses the given default. Otherwise, uses the value.

Note that if an empty string is given, the default value will *not* be used. Use the [**default**](https://docs.djangoproject.com/en/5.1/ref/templates/builtins/#std-templatefilter-default) filter if you want to fallback for empty strings.

For example:

{{ value|default\_if\_none:"nothing" }}

If **value** is **None**, the output will be **nothing**.

### dictsort

Takes a list of dictionaries and returns that list sorted by the key given in the argument.

For example:

{{ value|dictsort:"name" }}

If **value** is:

[

{"name": "zed", "age": 19},

{"name": "amy", "age": 22},

{"name": "joe", "age": 31},

]

then the output would be:

[

{"name": "amy", "age": 22},

{"name": "joe", "age": 31},

{"name": "zed", "age": 19},

]

### divisibleby

Returns **True** if the value is divisible by the argument.

For example:

{{ value|divisibleby:"3" }}

If **value** is **21**, the output would be **True**.

### join

Joins a list with a string, like Python’s **str.join(list)**

For example:

{{ value|join:" // " }}

If **value** is the list **['a', 'b', 'c']**, the output will be the string **"a // b // c"**.

### last

Returns the last item in a list.

For example:

{{ value|last }}

If **value** is the list **['a', 'b', 'c', 'd']**, the output will be the string **"d"**.

### length

Returns the length of the value. This works for both strings and lists.

For example:

{{ value|length }}

If **value** is **['a', 'b', 'c', 'd']** or **"abcd"**, the output will be **4**.

The filter returns **0** for an undefined variable.

### linenumbers

Displays text with line numbers.

For example:

{{ value|linenumbers }}

If **value** is:

one

two

three

the output will be:

1. one

2. two

3. three

### lower

Converts a string into all lowercase.

For example:

{{ value|lower }}

If **value** is **Totally LOVING this Album!**, the output will

be **totally loving this album!**.

### random

Returns a random item from the given list.

For example:

{{ value|random }}

If **value** is the list **['a', 'b', 'c', 'd']**, the output could be **"b"**.

### slice

Returns a slice of the list.

Uses the same syntax as Python’s list slicing. See the [Python documentation](https://docs.python.org/3/tutorial/introduction.html#lists) for an introduction.

Example:

**{{ some\_list|slice:":2" }}**

If **some\_list** is **['a', 'b', 'c']**, the output will be **['a', 'b']**.

### slugify

Converts to ASCII. Converts spaces to hyphens. Removes characters that aren’t alphanumerics, underscores, or hyphens. Converts to lowercase. Also strips leading and trailing whitespace.

For example:

{{ value|slugify }}

If **value** is **"Joel is a slug"**, the output will be **"joel-is-a-slug"**.

### title

Converts a string into titlecase by making words start with an uppercase character and the remaining characters lowercase. This tag makes no effort to keep “trivial words” in lowercase.

For example:

{{ value|title }}

If **value** is **"my FIRST post"**, the output will be **"My First Post"**.

### truncatechars

Truncates a string if it is longer than the specified number of characters. Truncated strings will end with a translatable ellipsis character (”…”).

**Argument:** Number of characters to truncate to

For example:

{{ value|truncatechars:7 }}

If **value** is **"Joel is a slug"**, the output will be **"Joel i…"**.

### truncatewords

Truncates a string after a certain number of words.

**Argument:** Number of words to truncate after

For example:

{{ value|truncatewords:2 }}

If **value** is **"Joel is a slug"**, the output will be **"Joel is …"**.

Newlines within the string will be removed.

### upper

Converts a string into all uppercase.

For example:

{{ value|upper }}

If **value** is **"Joel is a slug"**, the output will be **"JOEL IS A SLUG"**.

### wordcount

Returns the number of words.

For example:

{{ value|wordcount }}

If **value** is **"Joel is a slug"**, the output will be **4**.

The for loop sets a number of variables available within the loop:

| **Variable** | **Description** |
| --- | --- |
| **forloop.counter** | The current iteration of the loop (1-indexed) |
| **forloop.counter0** | The current iteration of the loop (0-indexed) |
| **forloop.revcounter** | The number of iterations from the end of the loop (1-indexed) |
| **forloop.revcounter0** | The number of iterations from the end of the loop (0-indexed) |
| **forloop.first** | True if this is the first time through the loop |
| **forloop.last** | True if this is the last time through the loop |
| **forloop.parentloop** | For nested loops, this is the loop surrounding the current one |

### floatformat

When used without an argument, rounds a floating-point number to one decimal place – but only if there’s a decimal part to be displayed. For example:

| **value** | **Template** | **Output** |
| --- | --- | --- |
| **34.23234** | **{{ value|floatformat }}** | **34.2** |
| **34.00000** | **{{ value|floatformat }}** | **34** |
| **34.26000** | **{{ value|floatformat }}** | **34.3** |

If used with a numeric integer argument, **floatformat** rounds a number to that many decimal places. For example:

| **value** | **Template** | **Output** |
| --- | --- | --- |
| **34.23234** | **{{ value|floatformat:3 }}** | **34.232** |
| **34.00000** | **{{ value|floatformat:3 }}** | **34.000** |
| **34.26000** | **{{ value|floatformat:3 }}** | **34.260** |

Particularly useful is passing 0 (zero) as the argument which will round the float to the nearest integer.

| **value** | **Template** | **Output** |
| --- | --- | --- |
| **34.23234** | **{{ value|floatformat:"0" }}** | **34** |
| **34.00000** | **{{ value|floatformat:"0" }}** | **34** |
| **39.56000** | **{{ value|floatformat:"0" }}** | **40** |

If the argument passed to **floatformat** is negative, it will round a number to that many decimal places – but only if there’s a decimal part to be displayed. For example:

| **value** | **Template** | **Output** |
| --- | --- | --- |
| **34.23234** | **{{ value|floatformat:"-3" }}** | **34.232** |
| **34.00000** | **{{ value|floatformat:"-3" }}** | **34** |
| **34.26000** | **{{ value|floatformat:"-3" }}** | **34.260** |

If the argument passed to **floatformat** has the **g** suffix, it will force grouping by the [**THOUSAND\_SEPARATOR**](https://docs.djangoproject.com/en/5.1/ref/settings/#std-setting-THOUSAND_SEPARATOR) for the active locale. For example, when the active locale is **en** (English):

| **value** | **Template** | **Output** |
| --- | --- | --- |
| **34232.34** | **{{ value|floatformat:"2g" }}** | **34,232.34** |
| **34232.06** | **{{ value|floatformat:"g" }}** | **34,232.1** |
| **34232.00** | **{{ value|floatformat:"-3g" }}** | **34,232** |