# lina Documentation

***Release 1.0.1***

**Author**

September 14, 2014

Contents

Lina is a minimal template system for Python, modelled after Google's CTemplate library. It is designed to provide fast, safe template evaluation to generate code or other text documents.:

```
enum DataTypes {
{{#types:list-separator=,NEWLINE}}  {{name}}={{value:hex}}{{/types}}
}
```

evaluated with:

```
formats = [{'name':'Vector3i', value: 0x301}, {'name':'Vector3f', 'value': 0x302}]
```

will produce:

```
enum DataTypes {
    Vector3i = 0x301,
    Vector3f = 0x302
}
```

CHAPTER 1

# Overview

The base class in Lina is `lina.Template` which must be initialized with the template contents. It can be then evaluated to a string using `lina.Template.Render()` and `lina.Template.RenderSimple()`.

Lina has two main directives, *values* and *blocks*. A value is something which is replaced directly by the provided value, while a block is used to iterate over collections. Both blocks and values can be optionally formatted using a `formatter`, which allows for example to turn a string into uppercase inside the template.

Values are escaped using double curly braces:

```
Hello {{name}}!
```

Blocks have an additional prefix before the variable, `#` for the block start and `/` for the block end:

```
{{#users}}Hello {{name}}!{{/users}}
```

This requires to pass an array of named objects:

```
template.Render ({'users':[{'name':'Alice'}, {'name':'Bob'}]})
```

In some cases, this is unnecessary complicated. Lina provides a special syntax to access the *current* element, using a single dot. The template aboves can be then simplified to:

```
{{#users}}Hello {{.}}!{{/users}}
```

and rendered with:

```
template.Render ({'users': ['Alice', 'Bob']})
```

or even simpler using `lina.Template.RenderSimple()`:

```
template.RenderSimple (users = ['Alice', 'Bob'])
```

Both self-references as well as items can also access fields of an object. Assuming the `User` class has fields `name`, `age`, the follwing template will print the user name and age:

```
{{#users}}Hello {{.name}}, you are {{.age}} years old!{{/users}}
```

The field accessor syntax works for both fields as well as associative container, that is, for Lina, the following two objects are equivalent:

```
{'name':'Alice'}
```

and:

```
class User:
    def __init__(self, name):
        self.name = name
```

For blocks, Lina provides additional modifiers to check whether the current block execution is the first, an intermediate or the last one:

```
{{#block}}{{variable}}{{#block#Separator}},{{/block#Separator}}{{/block}}
```

`#First` will be only expanded for the first iteration, `#Separator` will be expanded for every expansion which is neither first nor last and `#Last` will be expanded for the last iteration only. If there is only one element, it will considered both first and last item of the sequence.

Contents:

## 1.1 API reference

**class** lina.**CBooleanFormatter**
Bases: lina.Formatter

For booleans, write true or false to the output. Otherwise, the input is just passed through.

**Format**(*value*)

**class** lina.**DefaultFormatter**(*value*)
Bases: lina.Formatter

Emit the default if the value is None, otherwise the value itself.

**Format**(*text*)

**class** lina.**EscapeNewlineFormatter**
Bases: lina.Formatter

Escape embedded newlines.

**Format**(*text*)

**class** lina.**Formatter**(*formatterType*)
Bases: builtins.object

Base class for all formatters.

A formatter can be used to transform blocks/values during expansion.

**Format**(*text*)
Format a value or a complete block.

**IsBlockFormatter**()

**IsValueFormatter**()

**OnBlockBegin**(*isFirst*)
Called before a block is expanded.

**Parameters** **isFirst** – `True` if this is the first expansion of the block.

**Returns** String or `None`. If a string is returned, it is prepended before the current block expansion.

**OnBlockEnd**(*isLast*)
Called after a block has been expanded.

**Parameters** **isLast** – `True` if this is the last expansion of the block.

**Returns** String or `None`. If a string is returned, it is appended after the current block expansion.

**class** `lina.`**`FormatterType`**
Bases: `enum.Enum`

The formatter type, either `Block` or `Value`.

`lina.`**`GetFormatter`**(*name*, *value=None*, *position=None*)
Get a formatter.

If the formatter cannot be found, an exception is raised.

**class** `lina.`**`HexFormatter`**
Bases: `lina.Formatter`

Write an integer as a hex literal (0x133F).

**`Format`**(*value*)

**class** `lina.`**`IncludeHandler`**
Bases: `builtins.object`

Base interface for include handlers.

**`Get`**(*name*)

**class** `lina.`**`IndentFormatter`**(*depth*)
Bases: `lina.Formatter`

Indent a block using tabs.

**`Format`**(*block*)

**`OnBlockBegin`**(*isFirst*)

**exception** `lina.`**`InvalidBlock`**(*message*, *position*)
Bases: `lina.TemplateException`

**exception** `lina.`**`InvalidFormatter`**(*message*, *position*)
Bases: `lina.TemplateException`

**exception** `lina.`**`InvalidToken`**(*message*, *position*)
Bases: `lina.TemplateException`

**exception** `lina.`**`InvalidWhitespaceToken`**(*message*, *position*)
Bases: `lina.TemplateException`

**class** `lina.`**`ListSeparatorFormatter`**(*value*)
Bases: `lina.Formatter`

Separate block entries.

This formatter will insert a value between block expansions.

**`OnBlockEnd`**(*isLast*)

**class** `lina.`**`PrefixFormatter`**(*prefix*)
Bases: `lina.Formatter`

Add a prefix to a value.

**`Format`**(*text*)

**class** lina.**SuffixFormatter**(*suffix*)
Bases: lina.Formatter

Add a suffix to a value.

**Format**(*text*)

**class** lina.**Template**(*template*, *includeHandler=None*)
Bases: builtins.object

The main template class.

**Render**(*context*)
Render the template using the provided context.

**RenderSimple**(***items*)
Simple rendering function.

This is just a convenience function which creates the context from the passed items and forwards them to Template.Render().

**exception** lina.**TemplateException**(*message*, *position*)
Bases: builtins.Exception

**GetPosition**()
Get the position where the exception occured.

**Returns** An object with two fields, line and column.

**class** lina.**TemplateRepository**(*templateDirectory*, *suffix=''*)
Bases: lina.IncludeHandler

A file template repository.

This template repository will load files from a specified folder.

**Get**(*name*)

**class** lina.**TextStream**(*text*)
Bases: builtins.object

A read-only text stream.

The text stream is used for input only and keeps track of the current read pointer position in terms of line/column numbers.

**Get**()
Get a character.

If the end of the stream has been reached, None is returned.

**GetOffset**()

**GetPosition**()

**IsAtEnd**()
Check if the end of the stream has been reached.

**Peek**()
Peek at the next character in the stream if possible. Returns None if the end of the stream has been reached.

**Reset**()

**Skip**(*length*)
Skip a number of characters starting from the current position.

**Substring**(*start*, *end*)

**Unget**()
Move one character back in the input stream.

**class** lina.**Token**(*name*, *start*, *end*, *position*)
Bases: builtins.object

Represents a single token.

Each token may contain an optional list of flags, separated by colons. The grammar implemented here is:

```
[prefix]?[^:}]+(:[^:})+, for example:
{{#Foo}} -> name = Foo, prefix = #
{{Bar:width=8}} -> name = Bar, prefix = None,
                                        flags = {width:8}
```

**EvaluateWhiteSpaceToken**(*position*)

**GetEnd**()

**GetFormatters**()

**GetName**()

**GetPosition**()

**GetStart**()

**IsBlockClose**()

**IsBlockStart**()

**IsIncludeToken**()

**IsSelfReference**()

**IsValue**()

**IsWhiteSpaceToken**()

**class** lina.**UppercaseFormatter**
Bases: lina.Formatter

Format a value as uppercase.

**Format**(*text*)

**class** lina.**WidthFormatter**(*width*)
Bases: lina.Formatter

Align the value to a particular width.

Negative values align to the left (i.e., the padding is added on the left: `'  42'`), positive values to the right (`'42 '`).

**Format**(*text*)

**class** lina.**WrapStringFormatter**
Bases: lina.Formatter

Wrap strings with quotation marks.

**Format**(*text*)

CHAPTER 2

# Indices and tables

- *genindex*
- *modindex*
- *search*

## l