inline — code inlined in a LaTeX document*

Jérôme LAURENS†

Released ?

Abstract

Usually, documentation is put inside the code, inline allows to work the other way round by putting code inside the documentation. This is particularly interesting when different code files share some logic and should be documented all at once.
The file `inline-manual` gives different examples. Here is the implementation of the package.
This LaTeX package requires LuaTeX and may use syntax coloring based on pygment.

1 Package dependencies

luacode, verbatim, datetime2, xcolor, fancyvrb and dependencies of these packages.

2 Similar technologies

The docstrip utility offers similar features, it is somehow more powerful than inline at the cost of more technicality and less practicality,

The ydoc.cls and skdoc.cls are full document classes with similar features but many more that are unrelated. inline focuses on code inlining and interfaces well with pygment for a smart syntax hilighting.

3 Known bugs and limitations

- inline does not play well with docstrip.

```
1 ⟨*package⟩
2 \makeatletter
```

*This file describes version ?, last revised ?.

†E-mail: jerome.laurens@u-bourgogne.fr

4 Constants

\c_NLN_comment_prop

One line comment marker per language.

```
\prop_const_from_keyval:Nn \c_NLN_comment_prop {
  tex=\c_percent_str,
  lua=--,
  python=\c_hash_str,
  c=//,
  c++=//,
  javascript=//,
}
```

(*End definition for* `\c_NLN_comment_prop`. *This variable is documented on page* ??.)

5 Global properties

\g/NLN/code/
\g/NLN/code/<name>

Tree storage for global generic code properties or named code properties. These are overriden locally in environments using key-value actions. `\l_NLN_code_name_tl` is used as ⟨*name*⟩.

```
\prop_new:c {g/NLN/code/}
```

(*End definition for* `\g/NLN/code/` *and* `\g/NLN/code/<name>`. *These variables are documented on page* ??.)

\l_NLN_code_name_tl

Locally used as ⟨*name*⟩ in `\g/NLN/code/<name>` `\g/NLN/int/<name>` and similar.

```
\tl_new:N \l_NLN_code_name_tl
```

(*End definition for* `\l_NLN_code_name_tl`. *This variable is documented on page* ??.)

5.1 Management

\NLN_item:n

`\NLN_item:n {⟨key⟩}`

```
\cs_new:Npn \NLN_item:n #1 {
  \prop_item:cn {g/NLN/code/} { #1 }
}
```

\NLN_if_in:nTF ⋆

`\NLN_if_in:nTF {⟨key⟩} {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when `\g/NLN/code/` prop's contains ⟨*key*⟩, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn \NLN_if_in:n { T, F, TF } {
  \prop_if_in:NnTF {g/NLN/code/} { #1 } {
    \prg_return_true:
  } {
    \prg_return_false:
  }
}
```

`\NLN_get:nNTF` ⋆ `\NLN_get:nNTF {⟨key⟩} ⟨tl var⟩ {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when `\g/NLN/code/` prop's ⟨`key`⟩ is retrieved in ⟨*tl var*⟩, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn \NLN_get:nN { T, F, TF } {
  \prop_get:cnNTF {g/NLN/code/} { #1 } #2 {
    \prg_return_true:
  } {
    \prg_return_false:
  }
}
```

`\NLN_item:nn` ⋆ `\NLN_item:nn {⟨name⟩} {⟨key⟩}`

⟨*name*⟩ is a code name.

```
\cs_new:Npn \NLN_item:nn #1 #2 {
  \prop_item:cn { g/NLN/code/#1 } { #2 }
}
```

`\NLN_get:nnNTF` `\NLN_get:nnNTF {⟨name⟩} {⟨key⟩} ⟨tl var⟩ {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when `g/NLN/code/{⟨meta⟩}` prop's ⟨`key`⟩ is retrieved in ⟨*tl var*⟩, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn \NLN_get:nnN { T, F, TF } {
  \prop_get:cnNTF { g/NLN/code/#1 } { #2 } #3 {
    \prg_return_true:
  } {
    \prg_return_false:
  }
}
```

`\NLN_put:nn` `\NLN_put:nn {⟨key⟩} {⟨value⟩}`
`\NLN_put:nV`
`\NLN_gput:nn` `\NLN_gput:nn {⟨key⟩} {⟨value⟩}`
`\NLN_gput:nV`

```
\cs_new:Npn \NLN_put:nn #1 #2 {
  \prop_put:cnn {g/NLN/code/} { #1 } { #2 }
}
\cs_new:Npn \NLN_gput:nn #1 #2 {
  \prop_gput:cnn {g/NLN/code/} { #1 } { #2 }
}
\cs_generate_variant:Nn \NLN_put:nn { nV }
\cs_generate_variant:Nn \NLN_gput:nn { nV }
```

\NLN_put:nnn
\NLN_put:nnV
\NLN_gput:nnn
\NLN_gput:nnV

\NLN_put:nnn {⟨*name*⟩} {⟨*key*⟩} {⟨*value*⟩}
\NLN_gput:nnn {⟨*name*⟩} {⟨*key*⟩} {⟨*value*⟩}

⟨*name*⟩ is a code name.

```
\cs_new:Npn \NLN_put:nnn #1 #2 #3 {
  \prop_put:cnn { g/NLN/code/#1 } { #2 } { #3 }
}
\cs_new:Npn \NLN_gput:nnn #1 #2 #3 {
  \prop_gput:cnn { g/NLN/code/#1 } { #2 } { #3 }
}
\cs_generate_variant:Nn \NLN_put:nnn { nnV }
\cs_generate_variant:Nn \NLN_gput:nnn { nnV }
```

5.2 Known keys and conditionals

\NLN_new_conditional:n

\NLN_new_conditional:n {⟨*key*⟩}

Create new conditionals for the given key. Does nothing out of this package..

```
\cs_new:Npn \NLN_new_conditional:n #1 {
  \exp_last_unbraced:Nx
  \prg_new_conditional:Nnn { \use:c {NLN_if_#1:} } { T, F, TF } {
    \group_begin:
    \NLN_get:nNTF { #1 } \l_tmpa_tl {
      \exp_args:NnV
      \regex_match:nnTF { ^\s*[tTyY] } \l_tmpa_tl
      { \group_end: \prg_return_true:  }
      { \group_end: \prg_return_false: }
    } { \group_end: \prg_return_false: }
  }
}
```

format/code Font/size/color specifier for inline code.

```
      \NLN_gput:nn { format/code } {
        \ttfamily
      }
```

format/name Font/size/color specifier for chunk name.

```
      \NLN_gput:nn { format/name } {
        \sffamily
        \scriptsize
        \color{gray}
      }
```

format/lineno Font/size/color specifier for line numbers.

```
      \NLN_gput:nn { format/lineno } {
        \sffamily
        \tiny
        \color{gray}
      }
```

lang the langage, defaults to **tex**

```
\NLN_gput:nn { lang } { tex }
```

lineno show line numbers, defaults to **true**

```
\NLN_gput:nn { show_lineno } { T }
```

`\NLN_if_show_lineno:TF` `\NLN_if_show_lineno:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code property **show_lineo** is truthy, ⟨*false code*⟩ otherwise.

```
\NLN_new_conditional:n { show_lineno }
```

name show chunk names, defaults to **true**

```
\NLN_gput:nn { show_name } { T }
```

`\NLN_if_show_name:TF` `\NLN_if_show_name:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code property **show_name** is truthy, ⟨*false code*⟩ otherwise.

```
\NLN_new_conditional:n { show_name }
```

only top show names only on top, defaults to **true**

```
\NLN_gput:nn { only_top } { T }
```

`\NLN_if_only_top:TF` `\NLN_if_only_top:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code property **only_top** is truthy, ⟨*false code*⟩ otherwise.

```
\NLN_new_conditional:n { only_top }
```

margin use the margin to display line numbers and chunk names, defaults to **true**

```
\NLN_gput:nn { use_margin } { T }
```

`\NLN_if_use_margin:TF` `\NLN_if_use_margin:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code property **use_margin** is truthy, ⟨*false code*⟩ otherwise.

```
\NLN_new_conditional:n { use_margin }
```

ignore ignore that chunk or that export, defaults to **false**

```
\NLN_gput:nn { ignore } { F }
```

`\NLN_if_ignore:TF` `\NLN_if_ignore:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code property `ignore` is truthy, ⟨*false code*⟩ otherwise.

```
\NLN_new_conditional:n { ignore }
```

`reset` reset line numbering, defaults to `false`

```
\NLN_gput:nn { reset } { F }
```

`\NLN_if_reset:TF` `\NLN_if_reset:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code property `reset` is truthy, ⟨*false code*⟩ otherwise.

```
\NLN_new_conditional:n { reset }
```

`export` whether the code should be exported, defaults to `true`

```
\NLN_gput:nn { export } { T }
```

`\NLN_if_export:TF` `\NLN_if_export:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code property `export` is truthy, ⟨*false code*⟩ otherwise.

```
\NLN_new_conditional:n { export }
```

`parskip` the parskip used to separate lines of code

```
\AddToHook { begindocument/end } {
  \prop_if_in:cnF { g/NLN/code } { parskip } {
    \exp_args:Nnx
    \NLN_gput:nn { parskip } { \the\parskip }
  }
}
```

`sep` the separation between inline code blocks and surrounding text.

```
\NLN_gput:nn { sep } { 4pt plus 2pt minus 2pt }
```

`code` the cumulated inline code

```
\NLN_gput:nn { .code } {}
```

Clean memory.

```
\cs_set_eq:NN \NLN_new_conditional:n \prg_do_nothing:
```

6 Counters

\NLN_int_new:nn

\NLN_int_new:n {⟨*name*⟩} {⟨*value*⟩}

Create an integer after ⟨*name*⟩ and set it globally to ⟨*value*⟩. ⟨*name*⟩ is a code name.

```
\cs_new:Npn \NLN_int_new:nn #1 #2 {
  \int_new:c { g/NLN/int/#1 }
  \int_gset:cn { g/NLN/int/#1 } { #2 }
}
```

\NLN_int_set:nn
\NLN_int_gset:nn

\NLN_int_set:n {⟨*name*⟩} {⟨*value*⟩}

Set the integer named after ⟨*name*⟩ to the ⟨*value*⟩. \NLN_int_gset:n makes a global change. ⟨*name*⟩ is a code name.

```
\cs_new:Npn \NLN_int_set:nn #1 #2 {
  \int_set:cn { g/NLN/int/#1 } { #2 }
}
\cs_new:Npn \NLN_int_gset:nn #1 #2 {
  \int_gset:cn { g/NLN/int/#1 } { #2 }
}
```

\NLN_int_add:nn
\NLN_int_gadd:nn

\NLN_int_add:n {⟨*name*⟩} {⟨*value*⟩}

Add the ⟨*value*⟩ to the integer named after ⟨*name*⟩. \NLN_int_gadd:n makes a global change. ⟨*name*⟩ is a code name.

```
\cs_new:Npn \NLN_int_add:nn #1 #2 {
  \int_add:cn { g/NLN/int/#1 } { #2 }
}
\cs_new:Npn \NLN_int_gadd:nn #1 #2 {
  \int_gadd:cn { g/NLN/int/#1 } { #2 }
}
```

\NLN_int_sub:nn
\NLN_int_gsub:nn

\NLN_int_sub:n {⟨*name*⟩} {⟨*value*⟩}

Substract the ⟨*value*⟩ from the integer named after ⟨*name*⟩. \NLN_int_gsub:n makes a global change. ⟨*name*⟩ is a code name.

```
\cs_new:Npn \NLN_int_sub:nn #1 #2 {
  \int_sub:cn { g/NLN/int/#1 } { #2 }
}
\cs_new:Npn \NLN_int_gsub:nn #1 #2 {
  \int_gsub:cn { g/NLN/int/#1 } { #2 }
}
```

\NLN_int_if_exist:n*TF*

\NLN_int_if_exist:nTF {⟨*name*⟩} {⟨*true code*⟩} {⟨*false code*⟩}

Execute ⟨*true code*⟩ when an integer named after ⟨*name*⟩ exist, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn \NLN_int_if_exist:n { T, F, TF } {
  \int_if_exist:cTF { g/NLN/int/#1 } {
    \prg_return_true:
  } {
    \prg_return_false:
  }
}
```

\g/NLN/int/
\g/NLN/int/<name>

Generic and named line number counter. \l_NLN_code_name_t is used as ⟨*name*⟩.

```
\NLN_int_new:nn {} { 1 }
```

(*End definition for* \g/NLN/int/ *and* \g/NLN/int/<name>*. These variables are documented on page* ??.)

\NLN_int_use:n ⋆

\NLN_int_use:n {⟨*name*⟩}

⟨*name*⟩ is a code name.

```
\cs_new:Npn \NLN_int_use:n #1 {
  \int_use:c { g/NLN/int/#1 }
}
```

7 Variables

Line number counter for the code chunks.

\g_NLN_code_int

Chunk number counter.

```
\int_new:N \g_NLN_code_int
```

(*End definition for* \g_NLN_code_int*. This variable is documented on page* ??.)

\g_NLN_code_prop

Global code property list.

```
\prop_new:N \g_NLN_code_prop
```

(*End definition for* \g_NLN_code_prop*. This variable is documented on page* ??.)

\g_NLN_export_prop

Global storage for ⟨*file name*⟩=⟨*comma separated chunk name*⟩

```
\prop_new:N \g_NLN_export_prop
```

(*End definition for* \g_NLN_export_prop*. This variable is documented on page* ??.)

\l_NLN_prop

Local scratch variable.

```
\prop_new:N \l_NLN_prop
```

(*End definition for* \l_NLN_prop*. This variable is documented on page* ??.)

\g_NLN_chunks_tl
\l_NLN_chunks_tl

The comma separated list of current chunks. If the next list of chunks is the same as the current one, then it might not display.

```
\tl_new:N \g_NLN_chunks_tl
\tl_new:N \l_NLN_chunks_tl
```

(*End definition for* \g_NLN_chunks_tl *and* \l_NLN_chunks_tl*. These variables are documented on page* ??.)

\g_NLN_vars

Tree storage for global variables.

```
\prop_new:N \g_NLN_vars
```

WHAT

(*End definition for* \g_NLN_vars*. This variable is documented on page* ??.)

\g_NLN_vars

Tree storage for global variables.

```
\tl_new:N \g_NLN_hook_tl
```

(*End definition for* \g_NLN_vars*. This variable is documented on page* ??.)

\g/NLN/Chunks/<name>

List of chunk keys for given named code.

(*End definition for* \g/NLN/Chunks/<name>*. This variable is documented on page* ??.)

7.1 Local variables

`\l_NLN_recorded_tl` Full verbatim body of the `Inline` environment.

```
\tl_new:N \l_NLN_recorded_tl
```

(*End definition for* `\l_NLN_recorded_tl`. *This variable is documented on page* ??.)

`\g_NLN_int` Global integer to store linenos locally in time.

```
\int_new:N \g_NLN_int
```

(*End definition for* `\g_NLN_int`. *This variable is documented on page* ??.)

`\l_NLN_line_tl` Token list for one line.

```
\tl_new:N \l_NLN_line_tl
```

(*End definition for* `\l_NLN_line_tl`. *This variable is documented on page* ??.)

`\l_NLN_lineno_tl` Token list for lineno display.

```
\tl_new:N \l_NLN_lineno_tl
```

(*End definition for* `\l_NLN_lineno_tl`. *This variable is documented on page* ??.)

`\l_NLN_name_tl` Token list for chunk name display.

```
\tl_new:N \l_NLN_name_tl
```

(*End definition for* `\l_NLN_name_tl`. *This variable is documented on page* ??.)

`\l_NLN_info_tl` Token list for the info of line.

```
\tl_new:N \l_NLN_info_tl
```

(*End definition for* `\l_NLN_info_tl`. *This variable is documented on page* ??.)

`\l_NLN_clist` The comma separated list of current chunks.

```
\clist_new:N \l_NLN_clist
```

(*End definition for* `\l_NLN_clist`. *This variable is documented on page* ??.)

`\l_NLN_in` Input file identifier

```
\ior_new:N \l_NLN_in
```

(*End definition for* `\l_NLN_in`. *This variable is documented on page* ??.)

`\l_NLN_out` Output file identifier

```
\iow_new:N \l_NLN_out
```

(*End definition for* `\l_NLN_out`. *This variable is documented on page* ??.)

8 Utilities

Utilities

\NLN_clist_map_inline:Nnn

\NLN_clist_map_inline:Nnn ⟨*clist var*⟩
{⟨ ⟩}non empty code {⟨ ⟩}empty code

Call \clist_map_inline:Nnn ⟨*clist var*⟩ {⟨*non empty code*⟩} when the list is not empty, execute metaempty code otherwise.

```
\cs_new:Npn \NLN_clist_map_inline:Nnn #1 #2 #3 {
  \clist_if_empty:NTF #1 { #3 } {
    \clist_map_inline:Nn #1 { #2 }
  }
}
```

\NLN_process_record:

Record the current line or not.

```
\cs_new:Npn \NLN_process_record: {}
```

9 Shared key-value controls

Each action is meant to store the values in a code property, for the almost eponym key.

```
\keys_define:nn { NLN } {
```

Keys are:

lineno[=true/false] to display the line numbers, or not,

```
        lineno .code:n = \NLN_put:nn { show_lineno } { #1 },
        lineno .default:n = true,
```

name[=true/false] to display the chunk names

```
        name .code:n = \NLN_put:nn { show_name } { #1 },
        name .default:n = true,
```

only top to avoid chunk names repetitions, if on the same page, two consecutive code chunks have the same chunk names, the second names are not displayed.

```
        only~top .code:n = \NLN_put:nn { only_top } { #1 },
        only~top .default:n = true,
```

ignore to ignore chunks.

```
        ignore .code:n = \NLN_put:nn { ignore } { #1 },
        ignore .default:n = true,
```

margin[=true/false] to use the magin to display line numbers, or not,

```
        margin .code:n = \NLN_put:nn { use_margin } { #1 },
        margin .default:n = true,
```

lang=⟨*language name*⟩ , where ⟨*language name*⟩ is recognized by `pygment`,

```
        lang .code:n = \NLN_put:nn { lang } { #1 },
```

code format=⟨*format*⟩ , where ⟨*format*⟩ is used to display the code (mainly font, size and color),

```
        code~format .code:n = \NLN_put:nn { format/code } { #1 },
```

lineno format=⟨*format*⟩ , where ⟨*format*⟩ is used to display the line numbers (mainly font, size and color),

```
        name~format .code:n = \NLN_put:nn { format/name } { #1 },
```

name format=⟨*format*⟩ , where ⟨*name format*⟩ is used to display the chunk names (mainly font, size and color),

```
        lineno~format .code:n = \NLN_put:nn { format/lineno } { #1 },
```

post processor the name of the pygment post processor,

```
        post~processor .code:n = \NLN_put:nn { post_processor } { #1 },
```

post processor args the arguments of the pygment post processor,

```
        post~processor~args .code:n = \NLN_put:nn { post_processor_args } { #1 },
```

sep the separation with the surrounding text,

```
        sep .code:n = \NLN_put:nn { sep } { #1 },
```

parskip the value of the `\parskip` in inline code blocks,

```
        parskip .code:n = \NLN_put:nn { parskip } { #1 },
```

test whether the chunk is a test,

```
        test .code:n = \NLN_put:nn { test } { #1 },

    unknown .code:n = \PackageWarning
      { inline }
      { Unknown~option~`\l_keys_key_str' },
}
```

10 \InlineSet

`\InlineSet` `\InlineSet {⟨key[=value] list⟩}`

To set up the package. This is executed at least once at the end of the preamble. The unique mandatory argument of `\InlineSet` is a list of ⟨*key*⟩`[=`⟨*value*⟩`]` items defined by

10.1 `NLN/set` key-value controls.

```
\keys_define:nn { } { NLN/set .inherit:n = NLN }
\keys_define:nn { NLN/set } {
```

`minted` to activate syntax coloring with `pygment`, calls `\_NLN_minted_on:` and forwards the argument as `minted` option,

```
        minted .code:n = {
          \_NLN_minted_on:
          \setkeys { minted@opt@g } { #1 }
        },
```

`minted style=`⟨***`name`***⟩ to select a predefined minted style, forwarded to `\usemintedstyle`,

```
        minted~style .code:n = {
          \RemoveFromHook { begindocument/before } [NLN/Minted]
          \AddToHook { begindocument/before } [NLN/Minted] {
            \usemintedstyle { #1 }
          }
        },
```

`only description` to typeset only the description section and ignore the implementation section.

```
        only~description .code:n = \prop_put:Nnn \l_NLN_vars
          { only_description } { #1 },

   unknown .code:n = \PackageWarning
     { NLN/set }
     { Unknown~option~‘\l_keys_key_str’ },
}
```

10.2 Implementation

```
\NewDocumentCommand \InlineSet { m } {
  \keys_set:nn { NLN/set } {#1}
  \NLN_if_use_minted:F {
    \bool_if:NT \g_NLN_minted_on_bool {
      \sys_if_shell:TF {
        \_NLN_if_pygmentize:TF {
          \bool_gset_true:N \g_NLN_use_minted_bool
        } {
          \msg_warning:nnn
            { inline }
            { :n }
            { No~"pygmentize"~found. }
        }
      } {
        \msg_warning:nnn
          { inline }
          { :n }
          { No~unrestricted~shell~escape~for~"pygmentize".}
      }
```

```
    }
  }
}
```

11 InlineSplit environment

12 Inline environment

Inline `\begin{⟨Inline⟩}{⟨key[=value] list⟩} ... \end{⟨Inline⟩}`

The ⟨*key*⟩[=⟨*value*⟩] items are defined by the

12.1 NLN/code key-value controls

```
\keys_define:nn { } { NLN/code .inherit:n = NLN }
\keys_define:nn { NLN/code } {
```

chunks=⟨*comma separated list of chunk names*⟩ When declaring an exported file, this is the list of chunks that will appear in that file. When declaring a code chunk, this the list of chunks where it will be stored. Chunks are collected unordered and ordered for comparison.

```
    chunks .clist_set:N = \l_NLN_clist,
```

reset[=<boolean string> When declaring an exported file, this is the list of chunks that will appear in that file. When declaring a code chunk, this the list of chunks where it will be stored. Chunks are collected unordered and ordered for comparison.

```
    reset .code:n = \NLN_put:nn { reset } { #1 },
    reset .default:n = true,

  unknown .code:n = \PackageWarning
    { NLN/code }
    { Unknown~option~`\l_keys_key_str' },
}
```

12.2 Implementation

\NLN_if_record:*TF* `\NLN_if_record:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when code should be recorded, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn \NLN_if_record: { T, F, TF } {
  \NLN_if_export:TF {
    \prg_return_true:
  } {
    \NLN_if_use_minted:TF {
      \prg_return_true:
    } {
      \prg_return_false:
    }
  }
}
```

```
\cs_set:Npn \NLN_process_record: {
  \tl_put_right:Nx \l_NLN_recorded_tl { \the\verbatim@line \iow_newline: }
  \group_begin:
  \tl_set:Nx \l_tmpa_tl { \the\verbatim@line }
  \exp_args:Nx \directlua {NLN.records.append([===[\l_tmpa_tl]===])}
  \group_end:
}
\DeclareDocumentEnvironment { Inline } { m } {
  \directlua{NLN:start_recording()}
  \clist_clear:N \l_NLN_clist
  \keys_set:nn { NLN/code } { #1 }
  \clist_map_inline:Nn \l_NLN_clist {
    \NLN_int_if_exist:nF { ##1 } {
      \NLN_int_new:nn { ##1 } { 1 }
      \seq_new:c { g/NLN/chunks/##1 }
    }
  }
  \NLN_if_reset:T {
    \NLN_clist_map_inline:Nnn \l_NLN_clist {
      \NLN_int_gset:nn { ##1 } 1
    } {
      \NLN_int_gset:nn { } 1
    }
  }
  \tl_clear:N \l_NLN_code_name_tl
  \clist_map_inline:Nn \l_NLN_clist {
    \prop_concat:ccc
      {g/NLN/code/}
      { g/NLN/code/##1 }
      {g/NLN/code/}
    \tl_set:Nn \l_NLN_code_name_tl { ##1 }
    \clist_map_break:
  }
  \int_gset:Nn \g_NLN_int
    { \NLN_int_use:n { \l_NLN_code_name_tl } }
  \tl_clear:N \l_NLN_info_tl
  \tl_clear:N \l_NLN_name_tl
  \tl_clear:N \l_NLN_recorded_tl
  \tl_clear:N \l_NLN_chunks_tl
  \cs_set:Npn \verbatim@processline {
    \NLN_process_record:
  }
  \NLN_if_show_code:TF {
    \exp_args:NNx
    \skip_set:Nn \parskip { \NLN_item:n { parskip } }
    \clist_if_empty:NTF \l_NLN_clist {
      \tl_gclear:N \g_NLN_chunks_tl
    } {
      \clist_set_eq:NN \l_tmpa_clist \l_NLN_clist
      \clist_sort:Nn \l_tmpa_clist {
        \str_compare:nNnTF { ##1 } > { ##2 } {
          \sort_return_swapped:
        } {
          \sort_return_same:
```

```
      }
    }
    \tl_set:Nx \l_tmpa_tl { \clist_use:Nn \l_tmpa_clist , }
    \NLN_if_show_name:T {
      \NLN_if_use_margin:T {
        \NLN_if_only_top:T {
          \tl_if_eq:NNT \l_tmpa_tl \g_NLN_chunks_tl {
            \tl_gset_eq:NN \g_NLN_chunks_tl \l_tmpa_tl
            \tl_clear:N \l_tmpa_tl
          }
        }
        \tl_if_empty:NF \l_tmpa_tl {
          \tl_set:Nx \l_NLN_chunks_tl {
            \clist_use:Nn \l_NLN_clist ,
          }
          \tl_set:Nn \l_NLN_name_tl {
            {
              \NLN_item:n { format/name }
              \l_NLN_chunks_tl :
              \hspace*{1ex}
            }
          }
        }
      }
      \tl_if_empty:NF \l_tmpa_tl {
        \tl_gset_eq:NN \g_NLN_chunks_tl \l_tmpa_tl
      }
    }
  }
  \if_mode_vertical:
  \else:
  \par
  \fi:
  \vspace{ \NLN_item:n { sep } }
  \noindent
  \frenchspacing
  \@vobeyspaces
  \normalfont\ttfamily
  \NLN_item:n { format/code }
  \hyphenchar\font\m@ne
  \@noligs
  \NLN_if_record:F {
    \cs_set_eq:NN \NLN_process_record: \prg_do_nothing:
  }
  \NLN_if_use_minted:F {
    \NLN_if_show_lineno:T {
      \NLN_if_use_margin:TF {
        \tl_set:Nn \l_NLN_info_tl {
          \hbox_overlap_left:n {
            {
              \l_NLN_name_tl
              \NLN_item:n { format/name }
              \NLN_item:n { format/lineno }
              \int_use:N \g_NLN_int
```

```
                \int_gincr:N \g_NLN_int
              }
              \hspace*{1ex}
            }
          }
        } {
          \tl_set:Nn \l_NLN_info_tl {
            {
              \NLN_item:n { format/name }
              \NLN_item:n { format/lineno }
              \hspace*{3ex}
              \hbox_overlap_left:n {
                \int_use:N \g_NLN_int
                \int_gincr:N \g_NLN_int
              }
            }
            \hspace*{1ex}
          }
        }
      }
      \cs_set:Npn \verbatim@processline {
        \NLN_process_record:
        \hspace*{\dimexpr \linewidth-\columnwidth}%
        \hbox_to_wd:nn { \columnwidth } {
          \l_NLN_info_tl
          \the\verbatim@line
          \color{lightgray}\dotfill
        }
        \tl_clear:N \l_NLN_name_tl
        \par\noindent
      }
    }
  } {
    \@bsphack
  }
  \group_begin:
  \g_NLN_hook_tl
  \let \do \@makeother
  \dospecials \catcode `\^^M \active
  \verbatim@start
} {
  \int_gsub:Nn \g_NLN_int {
    \NLN_int_use:n { \l_NLN_code_name_tl }
  }
  \int_compare:nNnT { \g_NLN_int } > { 0 } {
    \NLN_clist_map_inline:Nnn \l_NLN_clist {
      \NLN_int_gadd:nn { ##1 } { \g_NLN_int }
    } {
      \NLN_int_gadd:nn { } { \g_NLN_int }
    }
    \int_gincr:N \g_NLN_code_int
    \tl_set:Nx \l_tmpb_tl { \int_use:N \g_NLN_code_int }
    \clist_map_inline:Nn \l_NLN_clist {
      \seq_gput_right:cV { g/NLN/chunks/##1 } \l_tmpb_tl
```

```
    }
    \prop_gput:NVV \g_NLN_code_prop \l_tmpb_tl \l_NLN_recorded_tl
  }
  \group_end:
  \NLN_if_show_code:T {
  }
  \NLN_if_show_code:TF {
    \NLN_if_use_minted:TF {
      \tl_if_empty:NF \l_NLN_recorded_tl {
        \exp_args:Nnx \setkeys { FV } {
          firstnumber=\NLN_int_use:n { \l_NLN_code_name_tl },
        }
        \iow_open:Nn \minted@code { \jobname.pyg }
        \exp_args:NNV \iow_now:Nn \minted@code \l_NLN_recorded_tl
        \iow_close:N \minted@code
        \vspace* { \dimexpr -\topsep-\parskip }
        \tl_if_empty:NF \l_NLN_info_tl {
          \tl_use:N \l_NLN_info_tl
          \skip_vertical:n { \dimexpr -\topsep-\parskip-\baselineskip }
          \par\noindent
        }
        \exp_args:Nnx \minted@pygmentize { \jobname.pyg } { \NLN_item:n { lang } }
        %\DeleteFile { \jobname.pyg }
        \skip_vertical:n { -\topsep-\partopsep }
      }
    } {
      \exp_args:Nx \skip_vertical:n { \NLN_item:n { sep } }
      \noindent
    }
  } {
    \@esphack
  }
}
```

NLN `\begin{`⟨*NLN*⟩`} ... \end{`⟨*NLN*⟩`}`

Private environment.

```
\newenvironment{NLN}{
  \def \verbatim@processline {
    \group_begin:
    \NLN_processline_code_append:
    \group_end:
  }
%  \NLN_if_show_code:T {
%    \NLN_if_use_minted:TF {
%      \Needspace* { 2\baselineskip }
%    } {
%      \frenchspacing\@vobeyspaces
%    }
%  }
} {
  \NLN_get:nNTF { lang } \l_tmpa_tl {
    \tl_if_empty:NT \l_tmpa_tl {
      \clist_map_inline:Nn \l_NLN_clist {
```

```
        \NLN_get:nnNT { ##1 } { lang } \l_tmpa_tl {
          \tl_if_empty:NF \l_tmpa_tl {
            \clist_map_break:
          }
        }
      }
      \tl_if_empty:NT \l_tmpa_tl {
        \tl_set:Nn \l_tmpa_tl { tex }
      }
    }
  } {
    \tl_set:Nn \l_tmpa_tl { tex }
  }
  \clist_map_inline:Nn \l_NLN_clist {
    \NLN_gput:nnV { ##1 } { lang } \l_tmpa_tl
  }
}
```

NLN.M

\begin{⟨*NLN.M*⟩} ... \end{⟨*NLN.N*⟩}

Private environment when `minted`.

```
\newenvironment{NLN_M}{
  \setkeys { FV } { firstnumber=last, }
  \clist_if_empty:NTF \l_NLN_clist {
    \exp_args:Nnx \setkeys { FV } {
      firstnumber=\NLN_int_use:n { },
  } } {
    \clist_map_inline:Nn \l_NLN_clist {
      \exp_args:Nnx \setkeys { FV } {
        firstnumber=\NLN_int_use:n { ##1 },
      }
      \clist_map_break:
  } }
  \iow_open:Nn \minted@code { \jobname.pyg }
  \tl_set:Nn \l_NLN_line_tl {
    \tl_set:Nx \l_tmpa_tl { \the\verbatim@line }
    \exp_args:NNV \iow_now:Nn \minted@code \l_tmpa_tl
  }
} {
  \NLN_if_show_code:T {
    \NLN_if_use_minted:TF {
      \iow_close:N \minted@code
      \vspace* { \dimexpr -\topsep-\parskip }
      \tl_if_empty:NF \l_NLN_info_tl {
        \tl_use:N \l_NLN_info_tl
        \vspace* { \dimexpr -\topsep-\parskip-\baselineskip }
        \par\noindent
      }
      \exp_args:NV \minted@pygmentize \l_tmpa_tl
      \DeleteFile { \jobname.pyg }
      \vspace* { \dimexpr -\topsep -\partopsep }
    } {
      \@esphack
    }
```

```
  }
}
```

NLN.P `\begin{⟨NLN.P⟩} ... \end{⟨NLN.P⟩}`

Private pseudo environment. This is just a practical way of declaring balanced actions.

```
\newenvironment{NLN_P}{
  \if_mode_vertical:
    \noindent
  \else
    \vspace*{ \topsep }
    \par\noindent
  \fi
  \NLN_gset_chunks:
  \tl_if_empty:NTF \g_NLN_chunks_tl {
    \NLN_if_show_lineno:TF {
      \NLN_if_use_margin:TF {
```

No chunk name, line numbers in the margin

```
        \tl_set:Nn \l_NLN_info_tl {
          \hbox_overlap_left:n {
            \NLN_item:n { format/code }
            {
              \NLN_item:n { format/name }
              \NLN_item:n { format/lineno }
              \clist_if_empty:NTF \l_NLN_clist {
                \NLN_int_use:n { }
              } {
                \clist_map_inline:Nn \l_NLN_clist {
                  \NLN_int_use:n { ##1 }
                  \clist_map_break:
                }
              }
            }
            \hspace*{1ex}
          }
        }
      } {
```

No chunk name, line numbers not in the margin

```
        \tl_set:Nn \l_NLN_info_tl {
          {
            \NLN_item:n { format/code }
            {
              \NLN_item:n { format/name }
              \NLN_item:n { format/lineno }
              \hspace*{3ex}
              \hbox_overlap_left:n {
                \clist_if_empty:NTF \l_NLN_clist {
                  \NLN_int_use:n { }
                } {
                  \clist_map_inline:Nn \l_NLN_clist {
                    \NLN_int_use:n { ##1 }
                    \clist_map_break:
```

```
                  }
                }
              }
              \hspace*{1ex}
            }
          }
        }
      }
    } {
```

No chunk name, no line numbers

```
      \tl_clear:N \l_NLN_info_tl
    }
  } {
    \NLN_if_show_lineno:TF {
```

Chunk names, line numbers, in the margin

```
      \tl_set:Nn \l_NLN_info_tl {
        \hbox_overlap_left:n {
          \NLN_item:n { format/code }
          {
            \NLN_item:n { format/name }
            \g_NLN_chunks_tl :
            \hspace*{1ex}
            \NLN_item:n { format/lineno }
            \clist_map_inline:Nn \l_NLN_clist {
              \NLN_int_use:n { ####1 }
              \clist_map_break:
            }
          }
          \hspace*{1ex}
        }
        \tl_set:Nn \l_NLN_info_tl {
          \hbox_overlap_left:n {
            \NLN_item:n { format/code }
            {
              \NLN_item:n { format/name }
              \NLN_item:n { format/lineno }
              \clist_map_inline:Nn \l_NLN_clist {
                \NLN_int_use:n { ####1 }
                \clist_map_break:
              }
            }
            \hspace*{1ex}
          }
        }
      }
    } {
```

Chunk names, no line numbers, in the margin

```
      \tl_set:Nn \l_NLN_info_tl {
        \hbox_overlap_left:n {
          \NLN_item:n { format/code }
          {
            \NLN_item:n { format/name }
```

```
            \g_NLN_chunks_tl :
          }
          \hspace*{1ex}
        }
        \tl_clear:N \l_NLN_info_tl
      }
    }
  }
  \NLN_if_use_minted:F {
    \tl_set:Nn \l_NLN_line_tl {
      \noindent
      \hbox_to_wd:nn { \textwidth } {
        \tl_use:N \l_NLN_info_tl
        \NLN_item:n { format/code }
        \the\verbatim@line
        \hfill
      }
      \par
    }
    \@bsphack
  }
} {
  \vspace*{ \topsep }
  \par
  \@esphack
}
```

13 \InlineExport

\InlineExport `\InlineExport{⟨key[=value] list⟩}`

The ⟨*key*⟩`[=`⟨*value*⟩`]` items are defined by

13.1 `NLN/export` key-value controls

```
\keys_define:nn { } { NLN/export .inherit:n = NLN/code }
\keys_define:nn { NLN/export } {
```

file the output file name

```
      file .tl_set:N = \l_NLN_tl,
      file .value_required:n = true,
```

preamble the added preamble.

```
      preamble .code:n = \prop_put:Nnn \l_NLN_vars { preamble } { #1 },
```

raw to remove any additional material,

```
      raw .code:n = \prop_put:Nnn \l_NLN_vars { raw } { #1 },

  unknown .code:n = \PackageWarning
    { NLN/export }
    { Unknown~option~`\l_keys_key_str' },
}
```

13.2 Implementation

```
\DeclareDocumentCommand \InlineExport { m } {
  \group_begin:
  \clist_clear:N \l_NLN_clist
  \prop_clear:c {g/NLN/code/}
  \prop_put:cnn {g/NLN/code/} { lang } { tex }
  \keys_set:nn { NLN/export } { #1 }
  \prop_gput:NVV \g_NLN_export_prop \l_NLN_tl \l_NLN_clist
  \prop_gput:cnV { g/NLN/export/\l_NLN_tl } { chunks } \l_NLN_clist
  \prop_gput:cnx { g/NLN/export/\l_NLN_tl } { preamble }
    { \prop_item:Nn \l_NLN_vars { preamble } }
  \bool_set:Nx \l_tmpa_bool { \prop_item:Nn \l_NLN_vars { raw } }
  \prop_gput:cnV { g/NLN/export/\l_NLN_tl } { preamble } \l_tmpa_bool
  \NLN_get:nNT { lang } \l_tmpa_tl {
    \clist_map_inline:Nn \l_NLN_clist {
      \prop_gconcat:ccc
        { g/NLN/code/##1 }
        { g/NLN/code/##1 }
        {g/NLN/code/}
    }
  }
  \group_end:
}
```

Files are created at the end of the typesetting process.

```
\AddToHook { enddocument / end } {
  \group_begin:
  \prop_map_inline:Nn \g_NLN_export_prop {
    \iow_open:Nn \l_NLN_out { #1 }
    \iow_term:x { Exporting~chunks~#2~to~#1 }
    \prop_get:cnNF { g/NLN/export/#1 } { raw } \l_tmpa_bool {
      \bool_set_false:N \l_tmpa_bool
    }
    \bool_if:NF \l_tmpa_bool {
      \prop_get:cnNT { g/NLN/export/#1 } { preamble } \l_tmpa_tl {
        \prop_get:cnNF { g/NLN/export/#1 } { lang } \l_tmpa_str {
          \str_set:Nn \l_tmpa_str { tex }
        }
        \prop_get:NVNTF \c_NLN_comment_prop \l_tmpa_str \l_tmpa_str {
          \tl_set:Nn \l_tmpb_tl {
            \l_tmpa_str\l_tmpa_str\space\space
          }
        } {
          \tl_clear:N \l_tmpb_tl
        }
        \tl_put_right:Nx \l_tmpb_tl {
          This~is~file~`#1'~
          generated~from~`\c_sys_jobname_str.tex'~on~\DTMnow.
        }
        \iow_now:Nx \l_NLN_out { \l_tmpb_tl }
        \iow_now:Nx \l_NLN_out { \l_tmpa_tl }
      }
    }
    \clist_map_inline:nn { #2 } {
```

```
      \NLN_get:nnNT { ##1 } { .code } \l_tmpa_tl {
        \exp_args:NNV \iow_now:Nn \l_NLN_out \l_tmpa_tl
      }
    }
    \iow_close:N \l_NLN_out
  }
  \group_end:
}
```

14 Management

\g_NLN_in_impl_bool Whether we are currently in the implementation section.

```
\bool_new:N \g_NLN_in_impl_bool
```

(*End definition for* `\g_NLN_in_impl_bool`. *This variable is documented on page* ??.)

\NLN_if_show_code:*TF* `\NLN_if_show_code:TF {`⟨*true code*⟩`} {`⟨*false code*⟩`}`

Execute ⟨*true code*⟩ when code should be printed, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn \NLN_if_show_code: { T, F, TF } {
  \bool_if:nTF {
    \g_NLN_in_impl_bool && !\g_NLN_with_impl_bool
  } {
    \prg_return_false:
  } {
    \prg_return_true:
  }
}
```

\g_NLN_with_impl_bool

```
\bool_new:N \g_NLN_with_impl_bool
```

(*End definition for* `\g_NLN_with_impl_bool`. *This variable is documented on page* ??.)

15 All purpose messaging

16 `minted` and `pygment`

\g_NLN_minted_on_bool Whether minted is available, initially set to `false`.

```
\bool_new:N \g_NLN_minted_on_bool
```

(*End definition for* `\g_NLN_minted_on_bool`. *This variable is documented on page* ??.)

\g_NLN_use_minted_bool Whether minted is used, initially set to `false`.

```
\bool_new:N \g_NLN_use_minted_bool
```

(*End definition for* `\g_NLN_use_minted_bool`. *This variable is documented on page* ??.)

`\NLN_if_use_minted:TF` `\NLN_if_use_minted:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when using `minted`, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn \NLN_if_use_minted: { T, F, TF } {
  \bool_if:NTF \g_NLN_use_minted_bool
    { \prg_return_true:  }
    { \prg_return_false: }
}
```

`\_NLN_if_pygmentize:TF` `\NLN_if_pygmentize:TF {⟨true code⟩} {⟨false code⟩}`

Execute ⟨*true code*⟩ when `pygmentize` is available, ⟨*false code*⟩ otherwise.

```
\prg_new_conditional:Nnn\_NLN_if_pygmentize: { T, F, TF } {
  \group_begin:
  \sys_get_shell:nnN {which~pygmentize} {} \l_tmpa_tl
  \tl_if_empty:NTF \l_tmpa_tl {
    \tl_set:Nn \l_tmpa_tl { \prg_return_false: }
  } {
    \tl_set:Nn \l_tmpa_tl { \prg_return_true: }
  }
  \exp_last_unbraced:NV
  \group_end: \l_tmpa_tl
}
```

`\_NLN_minted_on:` `\_NLN_minted_on:`

Private function. During the preamble, loads `minted`, sets `\g_NLN_minted_on_bool` to `true` and prepares `pygment` processing.

```
\cs_set:Npn \_NLN_minted_on: {
  \directlua{NLN.make_directory("_pygmented")}
  \bool_gset_true:N \g_NLN_minted_on_bool
  \RequirePackage{minted}
  \setkeys{ minted@opt@g } { linenos=false }
  \minted@def@opt{post~processor}
  \minted@def@opt{post~processor~args}
  \pretocmd\minted@inputpyg{
    \NLN@postprocesspyg {\minted@outputdir\minted@infile}
  }{}{\fail}
```

In the execution context of `\minted@inputpyg`,

`#1` is the name of the python script, e.g., "`process.py`"

`#2` is the input ".pygtex" file "`\minted@outputdir\minted@infile`"

`#3` are more args passed to the python script, possibly empty

```
  \newcommand{\NLN@postprocesspyg}[1]{%
    \group_begin:
    \tl_set:Nx \l_tmpa_tl {\NLN_item:n { post_processor } }
    \tl_if_empty:NF \l_tmpa_tl {
```

Execute ‘python3 <script.py> <file.pygtex> <more_args>‘

```
      \tl_set:Nx \l_tmpb_tl {\NLN_item:n { post_processor_args } }
      \exp_args:Nx
      \sys_shell_now:n {
        python3\space
        \l_tmpa_tl\space
        ##1\space
        \l_tmpb_tl
      }
    }
    \group_end:
  }
}
%\AddToHook { begindocument / end } {
%  \cs_set_eq:NN \_NLN_minted_on: \prg_do_nothing:
%}
```

Utilities to setup `pygment` post processing. The `pygment` post processor marks some code with `\InlineEmph`.

```
\ProvideDocumentCommand{\InlineEmph}{m}{\textcolor{red}{#1}}
```

`\InlineStorePreamble` `\InlineStorePreamble {`⟨*variable*⟩`} {`⟨*file name*⟩`}`

Store the content of ⟨*file name*⟩ into the variable ⟨*variable*⟩.

17 Separators

`\InlineImplementation` `\InlineImplementation`

Start an implementation part where all the sectioning commands do nothing.

`\InlineFinale` `\InlineFinale`

Stop an implementation part.

18 Finale

```
\DeclareDocumentCommand \InlineStorePreamble { m m } {
  \group_begin:
  \msg_info:nnn
    { inline }
    { :n }
    { Reading~preamble~from~file~"#2". }
  \tl_clear:N \g_tmpa_tl
  \tl_clear:N \g_tmpb_tl
  \ior_open:Nn \l_NLN_in { #2 }
  \bool_until_do:nn { \ior_if_eof_p:N \l_NLN_in } {
    \ior_str_get:NN \l_NLN_in \l_tmpa_tl
    \tl_if_empty:NTF \l_tmpa_tl {
      \tl_put_right:Nn \g_tmpb_tl { \iow_newline: }
    } {
```

```
      \tl_put_right:Nx \g_tmpa_tl { \g_tmpb_tl }
      \tl_set:Nn \g_tmpb_tl { \iow_newline: }
      \tl_put_right:NV \g_tmpa_tl \l_tmpa_tl
    }
  }
  \ior_close:N \l_NLN_in
  \exp_args:NNNx
  \group_end:
  \tl_set:Nn #1 { \tl_to_str:N \g_tmpa_tl }
}
\newcounter{NLN@impl@page}
\DeclareDocumentCommand \InlineImplementation {} {
  \bool_if:NF \g_NLN_with_impl_bool {
    \clearpage
    \bool_gset_true:N \g_NLN_in_impl_bool
    \let\NLN@old@part\part
    \DeclareDocumentCommand\part{som}{}
    \let\NLN@old@section\section
    \DeclareDocumentCommand\section{som}{}
    \let\NLN@old@subsection\subsection
    \DeclareDocumentCommand\subsection{som}{}
    \let\NLN@old@subsubsection\subsubsection
    \DeclareDocumentCommand\subsubsection{som}{}
    \let\NLN@old@paragraph\paragraph
    \DeclareDocumentCommand\paragraph{som}{}
    \let\NLN@old@subparagraph\subparagraph
    \DeclareDocumentCommand\subparagraph{som}{}
    \cs_if_exist:NT \refsection{ \refsection }
    \setcounter{ NLN@impl@page }{ \value{page} }
  }
}
\DeclareDocumentCommand\InlineFinale {} {
  \bool_if:NF \g_NLN_with_impl_bool {
    \clearpage
    \bool_gset_false:N \g_NLN_in_impl_bool
    \let\part\NLN@old@part
    \let\section\NLN@old@section
    \let\subsection\NLN@old@subsection
    \let\subsubsection\NLN@old@subsubsection
    \let\paragraph\NLN@old@paragraph
    \let\subparagraph\NLN@old@subparagraph
    \setcounter { page } { \value{ NLN@impl@page } }
  }
}
\cs_set_eq:NN \NLN_line_number: \prg_do_nothing:
```

19 Finale

```
\AddToHook { cmd/FancyVerbFormatLine/before } {
  \NLN_line_number:
}
\AddToHook { shipout/before } {
  \tl_gclear:N \g_NLN_chunks_tl
}
```

```
\InlineSet {}
% ========================================================
% Auxiliary:
%   finding the widest string in a comma
%   separated list of strings delimited by parenthesis
% ========================================================

% arguments:
% #1) text: a comma separeted list of strings
% #2) formatter: a macro to format each string
% #3) dimension: will hold the result

\cs_new:Npn \NLNWidest (#1) #2 #3 {
  \group_begin:
  \dim_set:Nn #3 { 0pt }
  \clist_map_inline:nn { #1 } {
    \hbox_set:Nn \l_tmpa_box { #2{##1} }
    \dim_set:Nn \l_tmpa_dim { \dim_eval:n { \box_wd:N \l_tmpa_box } }
    \dim_compare:nNnT { #3 } < { \l_tmpa_dim } {
      \dim_set_eq:NN #3 \l_tmpa_dim
    }
  }
  \exp_args:NNNV
  \group_end:
  \dim_set_eq:NN #3 #3
}

\ExplSyntaxOff
```

20 pygmentex implementation

```
% ========================================================
% fancyvrb new commands to append to a file
% ========================================================

% See http://tex.stackexchange.com/questions/47462/inputenc-error-with-unicode-chars-and-verbatim

\ExplSyntaxOn

\seq_new:N \l_NLN_records_seq

\long\def\unexpanded@write#1#2{\write#1{\unexpanded{#2}}}

\def\VerbatimOutAppend{\FV@Environment{}{VerbatimOutAppend}}

\def\FVB@VerbatimOutAppend#1{%
  \@bsphack
  \begingroup
    \seq_clear:N \l_NLN_records_seq
    \FV@UseKeyValues
    \FV@DefineWhiteSpace
    \def\FV@Space{\space}%
```

```
    \FV@DefineTabOut
    \def\FV@ProcessLine{%##1
%       \seq_put_right:Nn \l_NLN_records_seq { ##1 }%
      \immediate\unexpanded@write#1%{##1}
    }%
    \let\FV@FontScanPrep\relax
    \let\@noligs\relax
    \FV@Scan
}

\def\FVE@VerbatimOutAppend{
  \seq_use:Nn \l_NLN_records_seq /
  \endgroup
  \@esphack
}

\DefineVerbatimEnvironment{VerbatimOutAppend}{VerbatimOutAppend}{}

% ========================================================
% Main options
% ========================================================

\newif\ifNLN@left
\newif\ifNLN@right

% some settings used by fancyvrb:
% * for line numbering:
%     numbers, numbersep, firstnumber, stepnumber, numberblanklines
% * for selection of lines to print:
%     firstline, lastline,

\pgfkeys{%
  /NLN/.cd,
  %
  boxing~method/.code  = \NLN_put:nn {boxin_method} { #1 },
  inline~method/.code  = \NLN_put:nn {inline_method} { #1 },
  %
  lang/.code           = \NLN_put:nn {lang} { #1 },
  sty/.code            = \NLN_put:nn {sty} { #1 },
  escapeinside/.code   = \NLN_put:nn {escapeinside} { #1 },
  texcomments/.code    = \NLN_put:nn {texcomments} { #1 },% boolean
  mathescape/.code     = \NLN_put:nn {mathescape} { #1 },% boolean
  %
  label/.code          = \NLN_put:nn {label} { #1 },
  caption/.code        = \NLN_put:nn {caption} { #1 },
  %
  gobble/.code         = \NLN_put:nn {gobble} { #1 },
  tabsize/.code        = \NLN_put:nn {tabsize} { #1 },
  %
  linenos/.code        = \NLN_put:nn {linenos} { #1 },% boolean
  linenostart/.code    = \NLN_put:nn {linenostart} { #1 },
  linenostep/.code     = \NLN_put:nn {linenostep} { #1 },
  linenosep/.code      = \NLN_put:nn {linenosep} { #1 },
  %
```

```
  colback/.code        = \NLN_put:nn {colback} { #1 },
  font/.code           = \NLN_put:nn {font} { #1 },
  %
  texcomments/.default = true,
  mathescape/.default  = true,
  linenos/.default     = true,
}

\pgfqkeys{/NLN}{
  boxing~method = mdframed,
  inline~method = efbox,
  sty           = default,
  linenos       = false,
  linenosep     = 2pt,
  font          = \ttfamily,
  tabsize       = 0,
}

% ============================================================
% pygmented commands and environments
% ============================================================

\newwrite\NLN@outfile

\newcount\NLN@counter

\newcommand\NLN@process@options[1]{%
  \pgfkeys{%
    /pgf/key~filters/defined/.install~key~filter,%
    /pgf/key~filter~handlers/append~filtered~to/.install~key~filter~handler=\NLNRemainingGlo
  }%
  \def\NLNRemainingGlobalOptions{}%
  \pgfkeysalsofilteredfrom{\NLN@global@options}%
  \pgfkeysalso{%
    /pgf/key~filter~handlers/append~filtered~to/.install~key~filter~handler=\NLNRemainingUse
  }%
  \def\NLNRemainingUserOptions{}%
  \pgfqkeysfiltered{/NLN}{#1}%
  % %%%%%%% DEBUGING
  % \typeout{}%
  % \typeout{\string\NLN@global@options:}\typeout{\meaning\NLN@global@options}%
  % \typeout{\string\NLNRemainingGlobalOptions:}\typeout{\meaning\NLNRemainingGlobalOptions}
  % \typeout{\string\NLNRemainingUserOptions:}\typeout{\meaning\NLNRemainingUserOptions}%
  %
  \fvset{gobble=0,tabsize=0}%
}

\newcommand\NLN@process@more@options[1]{%
  \pgfkeysalso{%
    /pgf/key~filters/false/.install~key~filter,%
    /pgf/key~filter~handlers/append~filtered~to/.install~key~filter~handler=\NLNRemainingOpt
  }%
  \def\NLNRemainingOptions{}%
  \pgfkeysalsofilteredfrom{\NLNRemainingGlobalOptions}%
```

```
  \cs_if_exist:cT {NLN@#1@more@options} {
    \exp_args:Nx
    \pgfkeysalsofilteredfrom { \use:c{NLN@#1@more@options}, }
  }
  \pgfkeysalsofilteredfrom{\NLNRemainingUserOptions}%
  % %%%%%%% DEBUGING
  % \typeout{}%
  % \typeout{\string\NLNRemainingOptions:}%
  % \typeout{\meaning\NLNRemainingOptions}%
}

\newcommand\inputpygmented[2][]{%
  \begingroup
    \NLN@process@options{#1}%
    \immediate\write\NLN@outfile{<@@NLN@input@\the\NLN@counter}%
    \immediate\write\NLN@outfile{\exp_args:NV\detokenize\NLN@global@options,\detokenize{#1}}%
    \immediate\write\NLN@outfile{#2}%
    \immediate\write\NLN@outfile{>@@NLN@input@\the\NLN@counter}%
    %
    \csname NLN@snippet@\the\NLN@counter\endcsname
    \global\advance\NLN@counter by 1\relax
  \endgroup
}

\NewDocumentEnvironment{pygmented}{+O{}m}{%
  \directlua{NLN:start_recording()}
  \NLN@process@options{#1}%
  \immediate\write\NLN@outfile{<@@NLN@display@\the\NLN@counter}%
  \immediate\write\NLN@outfile{
    \exp_args:NV\detokenize\NLN@global@options,\detokenize{#1}
  }%
  \VerbatimEnvironment
  \begin{VerbatimOutAppend}{\NLN@outfile}%
}{%
  \end{VerbatimOutAppend}%
  \immediate\write\NLN@outfile{>@@NLN@display@\the\NLN@counter}%
  \csname NLN@snippet@\the\NLN@counter\endcsname
  \global\advance\NLN@counter by 1\relax
}

\newcommand\pyginline[2][]{%
  \begingroup
    \typeout{DEBUG1}
   \prop_set_eq:Nc \l_NLN_prop {g/NLN/prop}
   \cs_set:Npn \NLN_put:nn #1 #2 {
     \prop_put:Nnn \l_NLN_prop { #1 } { #2 }
   }
    \typeout{DEBUG2}
    \NLN@process@options{#1}%
    \typeout{DEBUG3}
    \directlua{NLN:clear_options()}
    \typeout{DEBUG4}
    \prop_map_inline:Nn \l_NLN_prop {
      \typeout{DEBUG5/#1/#2/}
```

```
      \directlua{NLN:add_option([===[#1]===], [===[#2]===])}
    }
    \DefineShortVerb{#2}%
    \SaveVerb
      [aftersave={%
       \UndefineShortVerb{#2}%
       \directlua{NLN:process_inline([===[\FV@SV@NLN]===])}
       \endgroup
      }]%
      {NLN}#2%
}

\cs_generate_variant:Nn \exp_last_unbraced:NnNo { NxNo }

\newcommand\NLN@snippet@inlined[1]{%
  \group_begin:
  \typeout{DEBUG~PY~STYLE:<\NLN@opt@style>}
  \use_c:n { PYstyledefault }
  \tl_if_empty:NF \NLN@opt@style {
    \use_c:n { PYstyle\NLN@opt@style }
  }
  \cs_if_exist:cTF {PY} {PYOK} {PYKO}
  \NLN@opt@font
  \NLN@process@more@options{ \NLN_item:n { inline_method} }%
  \exp_last_unbraced:NxNo
  \use:c { \NLN_item:n { inline_method } } [ \NLNRemainingOptions ]{#1}%
  \group_end:
}

% ERROR: JL undefined \NLN@alllinenos

\ProvideDocumentCommand\captionof{mm}{}
\def\NLN@alllinenos{(0)}
\prg_new_conditional:Nnn \NLN_yorn:n { T, F, TF } {
  \group_begin:
  \prop_get:cnNT {g/NLN/code/} { #1 } \l_tmpa_tl {
    \exp_args:NnV
    \regex_match:nnT {^[tTyY]} \l_tmpa_tl {
      \group_end:
      \prg_return_true:
    }
  } {
  \group_end:
  \prg_return_false:
}
\newenvironment{NLN@snippet@framed}{%
  \group_begin:
  \NLN@leftmargin\z@
  \NLN_yorn:nT {linenos} {
    \expandafter \NLNWidest\NLN@alllinenos{\FormatLineNumber}{\NLN@leftmargin}%
    \exp_args:NNx
    \advance\NLN@leftmargin { \NLN_item:n {linenosep} }
  }
  %
```

```
  \tl_clear:N \l_NLN_tl
  \NLN_get:nNTF {label} \l_tmpa_tl {
    \tl_set:N \l_NLN_tl {%
      \captionof{pygcode}{\label{\NLN_item:n {label}} \NLN_item:n {caption}}%
      % \nopagebreak
      \vskip -0.7\baselineskip
    }%
  } {
    \NLN_get:nNT {caption} \l_tmpa_tl {
      \tl_set:N \l_NLN_tl {%
        \captionof {pygcode} {\l_tmpa_tl}%
        % \nopagebreak
        \vskip -0.7\baselineskip
      }%
    \fi
  }
  \l_NLN_tl
  %
  \exp_args:Nx \tl_if_empty:nF { \NLN_item:n {boxing_method} } {
    \exp_args:Nx
    \NLN@process@more@options { \NLN_item:n {boxing_method} }%
    \exp_last_unbraced:NxNo
    \begin { \NLN_item:n {boxing_method} } [ \NLNRemainingOptions ]
  }
  \csname PYstyle\NLN@opt@style\endcsname
  \NLN@opt@font
  \noindent
} {
  \exp_args:Nx \tl_if_empty:nF { \NLN_item:n {boxing_method} } {
    \exp_args:Nx
    \end { \NLN_item:n {boxing_method} }
  }
  \group_end:
}


\newcommand\NLN@inlined[1]{%
  \exp_last_unbraced:NNV
  \efbox[\NLNRemainingOptions]{#1}%
}

\def\FormatLineNumber#1{{\rmfamily\tiny#1}}


\newdimen\NLN@leftmargin
\newdimen\NLN@linenosep

\def\NLN@lineno@do#1{%
  \NLN@linenosep 0pt%
  \use:c { NLN@ \NLN_item:n {boxing_method} @margin }
  \exp_args:NNx
  \advance \NLN@linenosep { \NLN_item:n {linenosep} }
  \hbox_overlap_left:n {%
    \FormatLineNumber{#1}%
```

```
    \hspace*{\NLN@linenosep}}%
}

\newcommand\NLN@tcbox@more@options{%
  nobeforeafter,%
  tcbox~raise~base,%
  left=0mm,%
  right=0mm,%
  top=0mm,%
  bottom=0mm,%
  boxsep=2pt,%
  arc=1pt,%
  boxrule=0pt,%
  \NLN_if_in:nNT {colback} {
    colback=\NLN_item:n {colback}
  }
}

\newcommand\NLN@efbox@more@options{%
  \NLN_if_in:nNT {colback} {
    backgroundcolor=\NLN_item:n {colback}
  }
}

\newcommand\NLN@mdframed@more@options{%
  leftmargin=\NLN@leftmargin,%
  frametitlerule=true,%
  \NLN_if_in:nNT {colback} {
    backgroundcolor=\NLN_item:n {colback}
  }
}

\newcommand\NLN@tcolorbox@more@options{%
  grow~to~left~by=-\NLN@leftmargin,%
  \NLN_if_in:nNT {colback} {
    colback=\NLN_item:n {colback}
  }
}

\newcommand\NLN@boite@more@options{%
  leftmargin=\NLN@leftmargin,%
  \ifcsname NLN@opt@colback\endcsname
    colback=\NLN@opt@colback,%
  \fi
}

\newcommand\NLN@mdframed@margin{%
  \advance \NLN@linenosep \mdflength{outerlinewidth}%
  \advance \NLN@linenosep \mdflength{middlelinewidth}%
  \advance \NLN@linenosep \mdflength{innerlinewidth}%
  \advance \NLN@linenosep \mdflength{innerleftmargin}%
}

\newcommand\NLN@tcolorbox@margin{%
```

```
  \advance \NLN@linenosep \kvtcb@left@rule
  \advance \NLN@linenosep \kvtcb@leftupper
  \advance \NLN@linenosep \kvtcb@boxsep
}

\newcommand\NLN@boite@margin{%
  \advance \NLN@linenosep \boite@leftrule
  \advance \NLN@linenosep \boite@boxsep
}

\def\NLN@global@options{}

\newcommand\setpygmented[1]{%
  \def\NLN@global@options{/NLN/.cd,#1}%
}


% ============================================================
% final actions
% ============================================================

\AtEndOfPackage{%
  \IfFileExists{\jobname.pygmented}{%
    \input{\jobname.pygmented}%
  }{%
    \PackageWarning{inline}{File `\jobname.pygmented' not found.}%
  }%
  \immediate\openout\NLN@outfile\jobname.snippets%
}

\AtEndDocument{%
  \closeout\NLN@outfile%
}
\ExplSyntaxOff
```

⟨/package⟩