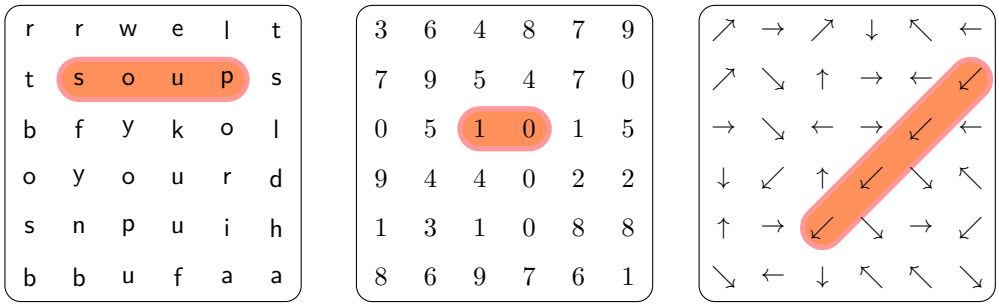
The soup package *

Thomas Simers †

Released 2017/01/15



Abstract

The goal of soup is to generate the grid of letters for a word search, puzzle sometimes called "alphabet soup" (from which this package gets its name) or "find-the-word."
In addition to supporting classic word searches, the soup can be filled with numbers or a user-defined set of glyphs.
Full functionality relies on TikZ, but limited support without TikZ is available through a package option.

*This document corresponds to soup v1.0, last revised 2017/01/15.

†E-mail: Simers.T@gmail.com

1 User Guide

The soup interface is rests primarily in two parts: The environments which determine the type of soup (alphabet, number, or homemade), and the shared macros for inserting and marking clues.

1.1 Load-Time Options

usetikz `\usepackage [⟨usetikz=false⟩] {soup}`

Usually, soup will use TikZ to draw the soup grid and provide the optional highlighting of clues.

To disable this, and use a non-TikZ fallback (the `tabular` environment), pass the option `usetikz=false` when loading soup.

highlight `\usepackage [⟨highlight=true⟩] {soup}`

As a puzzle generator, soup does not usually indicate the solution.

To have soup highlight the solutions, pass the option `highlight` (or `highlight=true`) when loading soup.

If TikZ is disabled, the solutions will be indicated with boldface letters. Note that if the the puzzle is drawn in boldface, this will hide the highlighting.

highlightcolor `\usepackage [⟨highlightcolor=color⟩] {soup}`

Specify the fill color to be used when highlighting solutions (TikZ only).

The default color is `orange`.

Color mixes are fine here, too: `green!50!white`.

linecolor `\usepackage [⟨linecolor=color⟩] {soup}`

Specify the line color to be used when highlighting solutions (TikZ only).

The default color is `red`.

Color mixes are fine here, too: `green!20!black`.

1.2 Environments

alphabetsoup
alphabetsoup*
Alphabetsoup
Alphabetsoup*

`\begin{alphabetsoup}` `[⟨width⟩]` `[⟨height⟩]` `[⟨font⟩]`
`\begin{alphabetsoup}*` `[⟨width⟩]` `[⟨height⟩]` `[⟨font⟩]`
`\begin{Alphabetsoup}` `[⟨width⟩]` `[⟨height⟩]` `[⟨font⟩]`
`\begin{Alphabetsoup}*` `[⟨width⟩]` `[⟨height⟩]` `[⟨font⟩]`

An `alphabetsoup` environment will build a grid of letters using lowercase Latin a–z, weighted for their frequence in English words. The `Alphabetsoup` environment uses uppercase A–Z. (For other alphabets, use a custom `homemadesoup`.)

A list of clues will be included after the grid. Use the starred version to omit the list. (To include the list later, use `\listofclues`.)

If the ⟨*height*⟩ is omitted, the number of rows will be the same as the number of columns.

If the ⟨*width*⟩ is omitted, it will default to 20.

Therefore, with no parameters, a 20-by-20 grid of letters will be generated.

⟨*font*⟩ can be optionally used to set the size of the letters in the soup (e.g., `\Large`, `\scriptsize`) or other font-related commands (e.g., `\sffamily`, `\itshape`)

numbersoup
numbersoup*

`\begin{numbersoup}` `[⟨width⟩]` `[⟨height⟩]` `{⟨max⟩}` `[⟨min⟩]` `[⟨font⟩]`
`\begin{numbersoup}*` `[⟨width⟩]` `[⟨height⟩]` `{⟨max⟩}` `[⟨min⟩]` `[⟨font⟩]`

The `numbersoup` environment follows `alphabetsoup` with two important differences:

- The grid is filled with numbers (not letters)
- Numbers are between ⟨*min*⟩ (or 0 if omitted) and ⟨*max*⟩, inclusive.

The ⟨*max*⟩ must be specified.

homemadesoup
homemadesoup*

`\begin{homemadesoup}` `[⟨width⟩]` `[⟨height⟩]` `{⟨symbols⟩}` `[⟨font⟩]`
`\begin{homemadesoup}*` `[⟨width⟩]` `[⟨height⟩]` `{⟨symbols⟩}` `[⟨font⟩]`

Instead of filling with digits or letters, the soup will be filled randomly from the user-specified comma-separated list ⟨*symbols*⟩

1.3 Macros

\hideinsoup
\hideinsoup*

`\hideinsoup` `{⟨x⟩}` `{⟨y⟩}` `{⟨dir⟩}` `{⟨seq⟩}` `[⟨clue⟩]`

Generally, an alphabetsoup will have words hidden in it. Other soups will have appropriate clues hidden (e.g., a number series).

These are put in the soup with `\hideinsoup`.

If two words overlap, and the overlapping letters (or other symbols) are different, `soup` will issue a warning, and it will display *both* letters in the grid, separated by a slash.

If highlighting is enabled, `\hideinsoup` will call `\highlightinsoup`. Use the starred version, `\hideinsoup*` to avoid this behavior.

If `soup` was loaded with `usetikz=false`, the highlighting of hidden clues will be simple boldface. The starred version will have no effect on this.

\highlightinsoup `\highlightinsoup {⟨x1⟩} {⟨y1⟩} {⟨x2⟩} {⟨y2⟩}`

Highlights the word (or sequence of symbols) between (⟨*x1*⟩,⟨*y1*⟩) and (⟨*x2*⟩,⟨*y2*⟩), where (1,1) is the top left of the soup grid, (2,1) is to the right of the top left, and (1,2) is the first symbol in the second row.

If `soup` was loaded with `usetikz=false`, this macro will have no effect.

\listofclues `\listofclues [⟨format⟩]`

Displays a list of all clues for the current puzzle.

The optional ⟨*format*⟩ should use `\theclue` where the text of the clue should appear.

Must be used after all uses of `\hideinsoup` for the current soup. If included before `\end{...soup}`, the clues will appear *before* the soup. If includes after `\end{...soup}`, then they will appear *after* the soup.

A typical use might be to display the clues as an enumerated list in columns:

```
\begin{alphabetsoup}*
    ...
\end{alphabetsoup}
\begin{multicols}{3}
    \begin{enumerate}
       \listofclues[\item \theclue]
    \end{enumerate}
\end{multicols}
```

2 Implementation

2.1 Dependencies

```
\RequirePackage{xparse}
\RequirePackage{expl3}
\RequirePackage{l3keys2e}
```

2.2 Initialization and Parameter Handling

```
\ExplSyntaxOn

\msg_new:nnn{soup}{mismatch}{
    Clue~mismatch~at~#1.~Will~appear~as~#2/#3~in~the~soup.
}

\bool_new:N \g_soup_use_tikz_bool
\bool_gset_true:N \g_soup_use_tikz_bool

\bool_new:N \g_soup_highlight_bool
\bool_gset_false:N \g_soup_highlight_bool

\tl_new:N \g_soup_highlight_color
```

```
\tl_gset:Nn \g_soup_highlight_color {orange}

\tl_new:N \g_soup_line_color
\tl_gset:Nn \g_soup_line_color {red}

\keys_define:nn { soup }{
  highlightcolor .initial:n        = orange,
  highlightcolor .value_required:n = true,
  highlightcolor .code:n           = \tl_set:Nn \g_soup_highlight_color {#1},
  linecolor      .initial:n        = red,
  linecolor      .value_required:n = true,
  linecolor      .code:n           = \tl_set:Nn \g_soup_line_color {#1},
  highlight      .default:n        = true,
  highlight      .bool_set:N       = \g_soup_highlight_bool,
  usetikz        .default:n        = true,
  usetikz        .bool_set:N       = \g_soup_use_tikz_bool,
}

\ProcessKeysPackageOptions{ soup }
\IfBooleanT \g_soup_use_tikz_bool {
    \RequirePackage{tikz}
}
\clist_const:Nn \c_soup_Alphabet_clist {
    A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,
    E,T,A,O,H,N,I,S,R,D,L,U,W,M,C,G,F,Y,P,V,K,B,J,
    E,T,A,O,H,N,I,S,R,D,L,U,W,M,C,G,F,Y,P,V,K,B,
    E,T,A,O,H,N,I,S,R,D,L,U,W,M,
    E,T,A,O,H,N,I,S,
    E,T,A,O,H,
}

\clist_const:Nn \c_soup_alphabet_clist {
    a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,
    e,t,a,o,h,n,i,s,r,d,l,u,w,m,c,g,f,y,p,v,k,b,j,
    e,t,a,o,h,n,i,s,r,d,l,u,w,m,c,g,f,y,p,v,k,b,
    e,t,a,o,h,n,i,s,r,d,l,u,w,m,
    e,t,a,o,h,n,i,s,
    e,t,a,o,h,
}

\prop_new:N \g_soup_data_prop
\seq_new:N \g_soup_clue_seq
```

2.3 Internal Functions

__soup_init:oo Resets the storage in preparation for a new soup.

```
\cs_new:Nn \__soup_init:oo {
    \clist_clear_new:N \g_soup_symbol_clist
    \dim_gzero_new:N \g_soup_highlight_dim
    \dim_gzero_new:N \g_soup_spacing_dim
```

```
    \int_gzero_new:N \g_soup_columns_int
    \int_gzero_new:N \g_soup_number_max_int
    \int_gzero_new:N \g_soup_number_min_int
    \int_gzero_new:N \g_soup_number_range_int
    \int_gzero_new:N \g_soup_rows_int
    \int_gzero_new:N \g_soup_symbol_count_int
    \prop_clear_new:N \g_soup_data_prop
    \seq_clear_new:N \g_soup_clue_seq
    \seq_clear_new:N \g_soup_highlight_seq
    \int_gset:Nn \g_soup_columns_int {#1}
    \IfNoValueTF{#2} {
       \int_gset:Nn \g_soup_rows_int {\g_soup_columns_int}
    }{
       \int_gset:Nn \g_soup_rows_int {#2}
    }
    \dim_gset:Nn \g_soup_spacing_dim {\textwidth / (\g_soup_columns_int + 1)}
    \dim_gset:Nn \g_soup_highlight_dim {\g_soup_spacing_dim * 7 / 10}
    \tl_clear_new:N \g_soup_font_tl
    \tl_gset:Nn \g_soup_font_tl {\normalfont}
}
```

(*End definition for* `\__soup_init:oo`.)

`\__soup_random_int:nn` Returns a pseudo-random integer between #1 and #2.
https://en.wikipedia.org/wiki/Lehmer_random_number_generator

```
\int_gzero_new:N \g__soup_random_previous_int
\int_gzero_new:N \g__soup_random_current_int
\cs_new:Nn \__soup_random_int:nn {
    \int_compare:nNnT \g__soup_random_previous_int = 0 {
        \int_gset:Nn \g__soup_random_previous_int {\time}
    }
    % A = 16807,  Q = 127773 (M / A), R = 2836 (M % A), M = 2147483647 (2^31-1)
    \int_zero_new:N \l__hi_int
    \int_zero_new:N \l__lo_int
    \int_set:Nn \l__hi_int {\g__soup_random_previous_int / 127773}
    \int_set:Nn \l__lo_int {\int_mod:nn{\g__soup_random_previous_int}{127773}}
    \int_gset:Nn \g__soup_random_previous_int {
        16807 * \l__hi_int - 2836 * \l__lo_int
    }
    \int_compare:nNnT \g__soup_random_previous_int < 1 {
        \int_gadd:Nn \g__soup_random_previous_int {2147483647}
    }
    \int_gset:Nn \g__soup_random_current_int {
        #1 + \int_mod:nn{\g__soup_random_previous_int}{#2 - #1 + 1}
    }
}
```

(*End definition for* `\__soup_random_int:nn`.)

`\__soup_draw_nodes:` Must be used inside a `tikzpicture` environment.

For every node pushed, now draw a node using either the previously set value or one now generated by the `getrand` macro.

```
\cs_new:Nn \__soup_draw_nodes: {
    \int_step_variable:nnnNn {1} {1} {\g_soup_columns_int} \l_tmpb_int {
        \int_step_variable:nnnNn {1} {1} {\g_soup_rows_int} \l_tmpc_int {
            \exp_args:Nnx
            \prop_get:NnNTF \g_soup_data_prop {
                (\l_tmpb_int,\l_tmpc_int)
            } \l_tmpa_tl {
                \node
                    at (\l_tmpb_int,\l_tmpc_int)
                    {\l_tmpa_tl};
            }{
                \node
                    at (\l_tmpb_int,\l_tmpc_int)
                    {\__soup_show_random_symbol:};
            }
        }
    }
}
```

(*End definition for* `\__soup_draw_nodes:`.)

`\__soup_draw_highlights:` Must be used inside a `tikzpicture` environment.

For every previously stored highlight coords, now draw the lines.

```
\cs_new:Nn \__soup_draw_highlights: {
    \seq_map_inline:Nn \g_soup_highlight_seq {
        \draw[
            double=\g_soup_highlight_color,
            double~distance=\g_soup_highlight_dim,
            line~width=2pt,
            color=\g_soup_line_color,
            opacity=0.4,
            line~cap=round
        ] ##1;
    }
}
```

(*End definition for* `\__soup_draw_highlights:`.)

`\__soup_draw_soup_tikz:` Do the actual work of drawing the soup

```
\cs_new:Nn \__soup_draw_soup_tikz: {

    \tikzset{
        every~node/.style={
            font=\g_soup_font_tl,
        },
    }
    \begin{tikzpicture}[
```

```
        x=\g_soup_spacing_dim,
        y=-\g_soup_spacing_dim,
    ]
        \draw[rounded~corners=6pt, use~as~bounding~box]
            (0.5,0)
            ++(0,0.5) rectangle +(\g_soup_columns_int, \g_soup_rows_int);
        \__soup_draw_highlights:
        \__soup_draw_nodes:
    \end{tikzpicture}
}
```

(*End definition for* `\__soup_draw_soup_tikz:`.)

`\__soup_draw_soup_tabular:` Do the actual work of drawing the soup (as a table)

```
\cs_new:Nn \__soup_draw_soup_tabular: {
    \dim_zero_new:N \l_soup_lineheight_dim
    \dim_set:Nn \l_soup_lineheight_dim {\g_soup_spacing_dim - \baselineskip}

    \vspace{0.25\g_soup_spacing_dim}\par
    \noindent\fbox{\parbox[c][
        \g_soup_rows_int\g_soup_spacing_dim
    ][c]{\g_soup_columns_int\g_soup_spacing_dim}{
    \begin{tabular*}{
        \g_soup_columns_int\g_soup_spacing_dim
    }{
        @{\extracolsep{\fill}}
        *{\g_soup_columns_int}{c}
    }
        \int_step_inline:nnnn {1} {1} {\g_soup_rows_int } {
            \int_gset:Nn \g_tmpa_int {##1}
            \int_step_variable:nnnNn {1} {1} {\g_soup_columns_int} \l_tmpb_int {
                \exp_args:Nnx
                \prop_get:NnNTF \g_soup_data_prop {
                    (\l_tmpb_int,\the\g_tmpa_int)
                } \l_tmpa_tl {
                    \g_soup_font_tl
                    \IfBooleanTF{\g_soup_highlight_bool}{
                        {\bfseries\l_tmpa_tl}
                    }{
                        \l_tmpa_tl
                    }
                }{
                    \g_soup_font_tl\__soup_show_random_symbol:
                }
                \int_compare:nNnT \l_tmpb_int < \g_soup_columns_int {
                    &
                }
            }
            \int_compare:nNnTF \g_tmpa_int < \g_soup_rows_int {
                \\[\l_soup_lineheight_dim]
```

```
            }{
            }
        }
    \end{tabular*}
    }
    }
}
```

(*End definition for* `\__soup_draw_soup_tabular:`.)

`\__soup_show_random_symbol:` Called for every coordinate not defined by calls to `\hideinsoup`, this generates a random symbol—either a number from the `\g_soup_number_range_int` (if nonzero) or from the list of symbols in `\g_soup_symbol_clist` set by `homemadesoup`, `alphabetsoup`, and `Alphabetsoup`.

```
\cs_new:Nn \__soup_show_random_symbol: {
    \int_compare:nNnTF \g_soup_symbol_count_int = 0 {
        \__soup_random_int:nn {\g_soup_number_min_int}{\g_soup_number_max_int}
        \the\g__soup_random_current_int
    }{
        \__soup_random_int:nn {1}{\g_soup_symbol_count_int}
        \clist_item:Nn \g_soup_symbol_clist {\g__soup_random_current_int}
    }
}
```

(*End definition for* `\__soup_show_random_symbol:`.)

2.4 User Document Functions

`\listofclues` Display the list of clues. THe optional argument will be expanded with `\theclue` as each clue. The default is defined as `\theclue\par`.

```
\NewDocumentCommand \listofclues { +o } {
    \tl_clear_new:N \theclue
    \IfNoValueTF{#1}{
        \tl_set:Nn \l_tmpa_tl {\theclue\par}
    }{
        \tl_set:Nn \l_tmpa_tl {#1}
    }
    \seq_map_variable:NNn \g_soup_clue_seq \theclue {
        \l_tmpa_tl
    }
}
```

(*End definition for* `\listofclues`. *This function is documented on page* 4.)

`\highlightinsoup` Given the coordinates of a word (expressed as `{x1}{y1}{x2}{y2}`), this will mark the word (or other sequence).

This is automatically called for every clue hidden via `\hideinsoup`.

This does nothing unless `highlight=true` was passed to the package.

```
\NewDocumentCommand \highlightinsoup { m m m m }{
```

```
  \bool_if:NT \g_soup_highlight_bool {
    \seq_gput_left:Nx \g_soup_highlight_seq {(#1, #2) -- (#3, #4)}
  }
}
```

(*End definition for* `\highlightinsoup`. *This function is documented on page* 4.)

`\hideinsoup` `\hideinsoup*`

Given a starting coordinate, a direction, a comma-separated list of symbols, and an optional clue, set the appropriate coordinates to these symbols.

`{`⟨*x1*⟩`}`, `{`⟨*y1*⟩`}`, `{`⟨*direction*⟩`}`, `{`⟨*word*⟩`}`, `[`⟨*clue*⟩`]`

The starred version will disable highlighting (if enabled) to allow setting parts of the soup that are outside actual answers.

If a clue is specified, insert it into the `\listofclues`

```
\NewDocumentCommand \hideinsoup { smmmmo } {
    \int_zero_new:N \l__soup_dx_int
    \int_zero_new:N \l__soup_dy_int

    \str_case:nn {#4} {
        {left}{
            \int_set:Nn \l__soup_dx_int {-1}
            \int_set:Nn \l__soup_dy_int { 0}
        }
        {right}{
            \int_set:Nn \l__soup_dx_int { 1}
            \int_set:Nn \l__soup_dy_int { 0}
        }
        {up}{
            \int_set:Nn \l__soup_dx_int { 0}
            \int_set:Nn \l__soup_dy_int {-1}
        }
        {upleft}{
            \int_set:Nn \l__soup_dx_int {-1}
            \int_set:Nn \l__soup_dy_int {-1}
        }
        {upright}{
            \int_set:Nn \l__soup_dx_int { 1}
            \int_set:Nn \l__soup_dy_int {-1}
        }
        {down}{
            \int_set:Nn \l__soup_dx_int { 0}
            \int_set:Nn \l__soup_dy_int { 1}
        }
        {downleft}{
            \int_set:Nn \l__soup_dx_int {-1}
            \int_set:Nn \l__soup_dy_int { 1}
        }
        {downright}{
            \int_set:Nn \l__soup_dx_int { 1}
            \int_set:Nn \l__soup_dy_int { 1}
```

```
        }
    }

    \clist_set:Nn \l__soup_clue_clist {#5}
    \int_zero_new:N \l__soup_clue_count_int
    \int_set:Nn \l__soup_clue_count_int {\clist_count:N \l__soup_clue_clist}

    \int_zero_new:N \l__soup_cx_int
    \int_zero_new:N \l__soup_cy_int
    \tl_clear_new:N \l__soup_ci_tl
    \tl_clear_new:N \l__soup_ch_tl
    \tl_clear_new:N \l__soup_nn_tl

    \int_step_variable:nnnNn {1} {1} {\l__soup_clue_count_int} \l__soup_ci_tl {
        \int_set:Nn \l__soup_cx_int
            {#2 + \l__soup_dx_int * (\l__soup_ci_tl - 1)}

        \int_set:Nn \l__soup_cy_int
            {#3 + \l__soup_dy_int * (\l__soup_ci_tl - 1)}

        \exp_args:Nnx
        \tl_set:Nn \l__soup_ch_tl
            {\clist_item:Nn \l__soup_clue_clist {\l__soup_ci_tl}}

        \exp_args:Nnx
        \tl_set:Nn \l__soup_nn_tl
            {(\the\l__soup_cx_int,\the\l__soup_cy_int)}

        \exp_args:Nnx
        \tl_set:Nn \l__soup_cv_tl
            {\exp_args:Nno \prop_item:Nn \g_soup_data_prop \l__soup_nn_tl}

        \str_if_empty:NTF \l__soup_cv_tl {
            \exp_args:Nnx \prop_gput:Noo \g_soup_data_prop {
                \l__soup_nn_tl
            } {\l__soup_ch_tl}
        }{
            \str_if_eq:NNF \l__soup_cv_tl \l__soup_ch_tl {
                \msg_warning:nnxxx{soup}{mismatch}{
                    \l__soup_nn_tl
                }{\l__soup_cv_tl}{\l__soup_ch_tl}

                \tl_put_left:Nx \l__soup_ch_tl
                    {\l__soup_cv_tl/}

                \exp_args:Nnx
                \prop_gput:Noo \g_soup_data_prop {\l__soup_nn_tl}
                    {\l__soup_ch_tl}
            }
        }
```

```
    }

    \IfBooleanF{#1}{
        \exp_args:Nnx
        \int_set:Nn \l__soup_cx_int
            {#2 + \l__soup_dx_int * (\l__soup_clue_count_int - 1)}

        \exp_args:Nnx
        \int_set:Nn \l__soup_cy_int
            {#3 + \l__soup_dy_int * (\l__soup_clue_count_int - 1)}

        \exp_args:Nnx
        \tl_set:Nn \l__soup_nn_tl
            {(\the\l__soup_cx_int,\the\l__soup_cy_int)}

        \exp_args:Nnx
        \seq_gput_left:Nx \g_soup_highlight_seq
            {(#2, #3) -- \l__soup_nn_tl}
    }
    \IfNoValueF{#6}{
        \seq_gput_left:No \g_soup_clue_seq {#6}
    }
}
```

(*End definition for* `\hideinsoup` *and* `\hideinsoup*`. *These functions are documented on page* 3.)

2.5 Environments

alphabetsoup
alphabetsoup*

A soup environment where unspecified coordinates are fill with a–z
For something else, see the `homemadesoup` environment.

```
\NewDocumentEnvironment{alphabetsoup}{ sO{15}oo }
{
    \par\noindent
    \__soup_init:oo {#2}{#3}
    \IfBooleanTF{#1}{
        \def\showlist{}
    }{
        \def\showlist{\par\vspace*{1em}\listofclues}
    }
    \IfNoValueF{#4}{
        \tl_gset:Nn \g_soup_font_tl {#4}
    }

    \clist_gset_eq:NN \g_soup_symbol_clist
        \c_soup_alphabet_clist

    \int_gset:Nn \g_soup_symbol_count_int
        {\clist_count:N \g_soup_symbol_clist}
}{
```

```
    \IfBooleanTF \g_soup_use_tikz_bool {
        \__soup_draw_soup_tikz:
    }{
        \__soup_draw_soup_tabular:
    }
    \showlist
}
```

(*End definition for* `alphabetsoup` *and* `alphabetsoup*`. *These functions are documented on page 3.*)

Alphabetsoup
Alphabetsoup*

A soup environment where unspecified coordinates are A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

For something else, see the `homemadesoup` environment.

```
\NewDocumentEnvironment{Alphabetsoup}{ sO{15}oo }
{
    \par\noindent
    \__soup_init:oo {#2}{#3}
    \IfBooleanTF{#1}{
        \def\showlist{}
    }{
        \def\showlist{\par\vspace*{1em}\listofclues}
    }
    \IfNoValueF{#4}{
        \tl_gset:Nn \g_soup_font_tl {#4}
    }

    \clist_gset_eq:NN \g_soup_symbol_clist
        \c_soup_Alphabet_clist

    \int_gset:Nn \g_soup_symbol_count_int
        {\clist_count:N \g_soup_symbol_clist}
}{
    \IfBooleanTF \g_soup_use_tikz_bool {
        \__soup_draw_soup_tikz:
    }{
        \__soup_draw_soup_tabular:
    }
    \showlist
}
```

(*End definition for* `Alphabetsoup` *and* `Alphabetsoup*`. *These functions are documented on page 3.*)

homemadesoup
homemadesoup*

The `homemadesoup` environment builds a soup from the user-supplied comma-separated list of symbols.

```
\NewDocumentEnvironment{homemadesoup}{ sO{15}omo }
{
    \par\noindent
    \__soup_init:oo {#2}{#3}
    \IfBooleanTF{#1}{
        \def\showlist{}
```

```
    }{
        \def\showlist{\par\vspace*{1em}\listofclues}
    }
    \IfNoValueF{#5}{
        \tl_gset:Nn \g_soup_font_tl {#5}
    }

    \clist_gset:Nn \g_soup_symbol_clist
        {#4}

    \int_gset:Nn \g_soup_symbol_count_int
        {\clist_count:N \g_soup_symbol_clist}
}
{
    \IfBooleanTF \g_soup_use_tikz_bool {
        \__soup_draw_soup_tikz:
    }{
        \__soup_draw_soup_tabular:
    }
    \showlist
}
```

(*End definition for* `homemadesoup` *and* `homemadesoup*`. *These functions are documented on page 3.*)

`numbersoup` `numbersoup*` Sets up a soup with all unspecified coordinates displaying numbers.

```
\NewDocumentEnvironment{numbersoup}{ sO{15}omO{0}o }
{
    \par\noindent
    \__soup_init:oo{#2}{#3}
    \IfBooleanTF{#1}{
        \def\showlist{}
    }{
        \def\showlist{\par\vspace*{1em}\listofclues}
    }
    \IfNoValueF{#6}{
        \tl_gset:Nn \g_soup_font_tl {#6}
    }

    \int_gset:Nn \g_soup_number_max_int
        {#4}

    \int_gset:Nn \g_soup_number_min_int
        {#5}

    \int_gset:Nn \g_soup_number_range_int
        {\g_soup_number_max_int - \g_soup_number_min_int}
}
{
    \IfBooleanTF \g_soup_use_tikz_bool {
        \__soup_draw_soup_tikz:
```

```
    }{
        \__soup_draw_soup_tabular:
    }
    \showlist
}
```

(*End definition for* `numbersoup` *and* `numbersoup*`. *These functions are documented on page 3.*)

```
\ExplSyntaxOff
```

Change History

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

U