```
  _________
 / This is \
 \ ducksay! /
  ---------
   \
    \  __
     >(' )
       )/
      /(
     /  `----/
     \  ~=- /
   ~^~^~^~^~^~^~
```

```
 ____________________
( But which Version? )
 --------------------
  \
   \
    >()_
     (__)__ _
```

```
                   ______
                  ( v2.8 )
                   ------
            ^__^  /
    _______/(oo) /
/\/(       /(__)
   | w----||
   ||     ||
```

```
   ___________________________
  (  by Jonathan P. Spratte  )
   ---------------------------
                 /
      .-----. /
    .'_/_|_\_`.
   /<::[O]8::>>\
  _|-----------|_
 |  | -=-==== |  |
 |  | ====-=- |  |
 \  ||()|:::: |  /
  | ||()|.... | |
  | |_________| |
  | |\_______/| |
 /   \ /   \ /   \
 `---' `---' `---'
```

```
 _______________________
( Today is 2025-11-22 )
 -----------------------
  \    .\|//||\||.
   \  |/\/||/|//|/|
     /. `|/\\|/||/||
    o__,_|//|/||\||'
 --------------
/              \
|  It's always |
|    good to   |
|   keep the   |
|   overview!  |
\              /
 --------------
   \
    \ ^__^         /
      (oo)\_______/  _________
      (__)\       )=(  ____|_ \_____
          ||----w |  \ \     \_____ |
          ||     ||   ||           ||
```

Contents

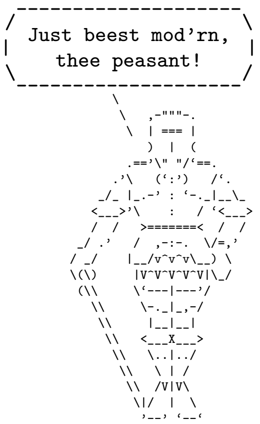
```
 ----------------------
/                      \
|     Give credit      |
| where credit is due! |
\                      /
 ----------------------
        `_`_`_`_        /
      ,(      (,  ):.  /
     ,(     ,)    ,__)
    <( (,     (,  ) o \
     (,   ,)      ,)___)
      (,    (,   ,)
       `-,_---,_-'
         ||   ||
```

1 Acknowledgements

```
 ----------------------------
/                            \
\ I was created by Plergux!* /
 ----------------------------
  \       _,_,_,_,
   \    .:(  ,)      ),
      (__,    (,     ),
     / o (  ,)     ,) )>
    (___(,      (,   ,)
          (,   ,)    ,)
           `-_,---_,-'
             ||   ||
```

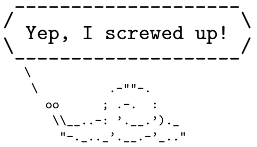
---

*https://chat.stackexchange.com/transcript/message/55986902#55986902



2 Documentation

This is ducksay! A `cowsay` for LaTeX. ducksay is part of TeXLive and MiKTeX since September 2017. If it is not part of your installation it means that your LaTeX installation is *really* out of date, you have two options: Update your installation or try to install ducksay yourself. Chances are that if you opt for the latter, the version of expl3 in your LaTeX installation is too old, too, and the l3regex module is not yet part of expl3. In that case you'll get a few undefined control sequence errors. `\usepackage{l3regex}` prior to loading ducksay might fix these issues. Additionally you'll need grabbox for version 2 of ducksay that won't be part of your LaTeX installation, too. Please note that I don't actively support out of date LaTeX installations, so if loading l3regex doesn't fix the issues and you're on an old installation, I won't provide further support.



2.1 Downward Compatibility Issues

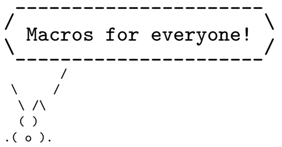
In the following list I use the term "version" to refer to package versions, the same is true if I use an abbreviation like "v2.0" (or anything that matches the regular expression `v\d+(.\d+)?`). For the code variant which can be set using the `version` option I'll use the term "variant" or specify directly that I'm referring to that option (the used font may be a hint, too).

v2.0
- Versions prior to v2.0 did use a regular expression for the option `ligatures`, see subsubsection 2.2.2 for more on this issue.
- In a document created with package versions prior to v2.0 you'll have to specify the option `version=1` with newer package versions to make those old documents behave like they used to.

v2.3
- Since v2.3 `\AddAnimal` and `\AddColoredAnimal` behave differently. You no longer have to make sure that in the first three lines every backslash which is only preceded by spaces is the bubble's tail. Instead you can specify which symbol should be the tail and how many of such symbols there are. See subsubsection 2.2.1 for more about the current behaviour.

v2.4
- The `add-think` key was deprecated in v2.3 and was removed in v2.4 since the output symbols of the bubble tail are handled differently and more efficiently now.

v2.8
- The `tail-symbol` key no longer fixes the category code of its value to 12 and you have to make sure that the correct category code is handed in. The key `gen-tail-symbol` might help. See paragraph 2.2.2.1.



2.2 Shared between versions

2.2.1 Macros

A careful reader might notice that in the below list of macros there is no `\ducksay` and no `\duckthink` contained. This is due to differences between the two usable code variants (see the `version` key in subsubsection 2.2.2 for the code variants, subsubsection 2.3.2 and subsubsection 2.4.2 for descriptions of the two macros).

`\DefaultAnimal` `\DefaultAnimal{⟨animal⟩}`

use the ⟨`animal`⟩ if none is given in the optional argument to `\ducksay` or `\duckthink`. Package default is `duck`.

`\DucksayOptions` `\DucksayOptions{⟨options⟩}`

set the defaults to the keys described in subsubsection 2.2.2, subsubsection 2.3.3 and subsubsection 2.4.3. Don’t use an ⟨`animal`⟩ here, it has no effect.

`\AddAnimal` `\AddAnimal⟨*⟩[⟨options⟩]{⟨animal⟩}⟨ascii-art⟩`

adds ⟨`animal`⟩ to the known animals. ⟨`ascii-art`⟩ is multi-line verbatim and therefore should be delimited either by matching braces or by anything that works for `\verb`. If the star is given ⟨`animal`⟩ is the new default. One space is added to the begin of ⟨`animal`⟩ (compensating the opening symbol). The symbols signalizing the speech bubble’s tail (in the `hedgehog` example below the two `s`) can be set using the `tail-symbol` option and only the first `tail-count` occurrences will be substituted (see paragraph 2.2.2.1 for more about these options). For example, `hedgehog` is added with:

```
\AddAnimal[tail-symbol=s]{hedgehog}
{  s    .\|//||\||.
    s  |/\/||/|//|/|
      /. ‘|/\\|/||/||
     o__,_|//|/||\||’}
```

It is not checked whether the animal already exists, you could therefore redefine existing animals with this macro.

`\AddColoredAnimal` `\AddColoredAnimal⟨*⟩[⟨options⟩]{⟨animal⟩}⟨ascii-art⟩`

It does the same as `\AddAnimal` but allows three different colouring syntaxes. You can use `\textcolor` in the ⟨`ascii-art`⟩ with the syntax `\textcolor{⟨color⟩}{⟨text⟩}`. Note that you can’t use braces in the arguments of `\textcolor`.

You can also use a delimited `\color` of the form `\bgroup\color{⟨color⟩}⟨text⟩ \egroup`, a space after that `\egroup` will be considered a space in the output, so you don’t have to care for correct termination of the `\egroup` (so `\bgroup\color{red}RedText \egroupOtherText` is valid syntax). You can’t nest delimited `\colors`.

Also you can use an undelimited `\color`. It affects anything until the end of the current line (or, if used inside of the ⟨`text`⟩ of a delimited `\color`, anything until the end of that delimited `\color`’s ⟨`text`⟩). The syntax would be `\color{⟨color⟩}`.

The package doesn’t load anything providing those colouring commands for you and it doesn’t provide any coloured animals. The parsing is done using regular expressions provided by LaTeX3. It is therefore slower than the normal `\AddAnimal`.

```
\AddAnimal{colored-small-duck}
{  \
    \
      \color{orange}>\color{yellow}()_
       \color{yellow}(__)\color{blue}__ _}
```

```
 ________________________
/                        \
\ Don’t I look fabulous? /
 ------------------------
  \
   \
     >()_
      (__)__ _
```

`\AnimalOptions` `\AnimalOptions⟨*⟩{⟨animal⟩}{⟨options⟩}`

With this macro you can set ⟨*animal*⟩ specific ⟨*options*⟩. If the star is given any currently set options for this ⟨*animal*⟩ are dropped and only the ones specified in ⟨*options*⟩ will be applied, else ⟨*options*⟩ will be added to the set options for this ⟨*animal*⟩. The set ⟨*options*⟩ can set the `tail-1` and `tail-2` options and therefore overwrite the effects of `\duckthink`, as `\duckthink` really is just `\ducksay` with the `think` option.

```
 --------------------
/                    \
| Options.           |
| For every occasion |
\                    /
 --------------------
  \      _//| .-~~~-.
   \ _/oo  }       }-@
    ('')_  }       |
     `--'| { }--{  }
            //_/  /_/
```

2.2.2 Options

The following options are available independent on the used code variant (the value of the `version` key). They might be used as package options – unless otherwise specified – or used in the macros `\DucksayOptions`, `\ducksay` and `\duckthink` – again unless otherwise specified. Some options might be accessible in both code variants but do slightly different things. If that's the case they will be explained in subsubsection 2.3.3 and subsubsection 2.4.3 for `version` 1 and 2, respectively.

`version=`⟨*number*⟩

With this you can choose the code variant to be used. Currently `1` and `2` are available. This can be set only during package load time. For a dedicated description of each version look into subsection 2.3 and subsection 2.4. The package author would choose `version=2`, the other version is mostly for legacy reasons. The default is `2`.

⟨*animal*⟩ One of the animals listed in subsection 2.6 or any of the ones added with `\AddAnimal`. Not useable as package option. Also don't use it in `\DucksayOptions`, it'll break the default animal selection.

`animal=`⟨*animal*⟩

Locally sets the default animal. Note that `\ducksay` and `\duckthink` do digest their options inside of a group, so it just results in a longer alternative to the use of ⟨*animal*⟩ if used in their options.

`ligatures=`⟨*token list*⟩

each token you don't want to form ligatures during `\AddAnimal` should be contained in this list. All of them get enclosed by grouping `{` and `}` so that they can't form ligatures. Giving no argument (or an empty one) might enhance compilation speed by disabling this replacement. The formation of ligatures was only observed in combination with `\usepackage[T1]{fontenc}` by the author of this package. Therefore giving the option `ligatures` without an argument might enhance the compilation speed for you without any drawbacks. Initially this is set to `` `<>,'- ``.
Note: In earlier releases this option's expected argument was a regular expression. This means that this option is not fully downward compatible with older versions. The speed gain however seems worth it (and I hope the affected documents are few).

`no-tail` Sets `tail-1` and `tail-2` to be a space.

`random=`⟨*bool*⟩

If `true` a random animal will be used instead of the default one on each usage of `\ducksay` or `\duckthink`. The initial value is false and it defaults to true.

`say` Sets `tail-1` and `tail-2` as backslashes.

`schroedinger`

Sets randomly either `animal=schroedinger-alive` or `animal=schroedinger-dead` at the time of use. Both have the same size, so this doesn't affect the needed space.

`tail-1=`⟨*`token list`*⟩

Sets the first tail symbol in the output to be ⟨*`token list`*⟩. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be `O`.

`tail-2=`⟨*`token list`*⟩

Sets every other tail symbol except the first one in the output to be ⟨*`token list`*⟩. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be `o`.

`think` Sets `tail-1=O` and `tail-2=o`.

2.2.2.1 Options for `\AddAnimal`

The options described here are only available in `\AddAnimal` and `\AddColoredAnimal`.

`tail-count=`⟨*`int`*⟩

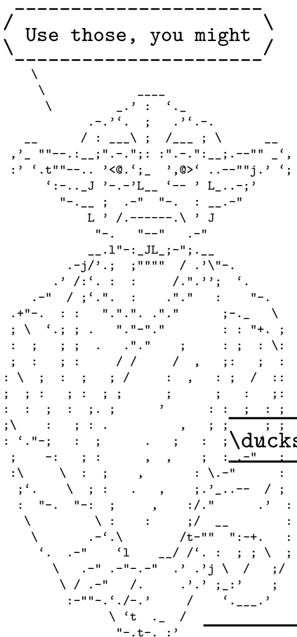
sets the number of tail symbols to be replaced in `\AddAnimal` and `\AddColoredAnimal`. Initial value is 2. If the value is negative every occurrence of `tail-symbol` will be replaced.

`tail-symbol=`⟨*`tl`*⟩

the symbol used in `\AddAnimal` and `\AddColoredAnimal` to mark the bubble's tail. In LaTeX versions prior to 2025-06-01 the argument gets `\detokenize`d, starting with the 2025-06-01 kernel release you have to make sure that whatever you give to this key has the same category codes as are produced by the `v` argument type of `ltcmd` (you can use `tail-symbol:e` to fully expand the value which might help in producing the desired token). Initially a single backslash.

`gen-tail-symbol=`⟨*`character`*⟩

this is a helper for `tail-symbol`. It sets `tail-symbol` to ⟨*`character`*⟩ which might either be the character itself or an escaped variant of it for special characters. The `tail-symbol` will then be ⟨*`character`*⟩ but with category code 12 (other). For instance you could use `gen-tail-symbol = \%` to use a percent character or `gen-tail-symbol = \\` for a backslash both with category 12 as `tail-symbol`.



2.3 Version 1

2.3.1 Introducktion

This version is included for legacy support (old documents should behave the same without any change to them – except the usage of `version=1` as an option, for a more or less complete list of downward compatibility related problems see subsection 2.1). For the bleeding edge version of `ducksay` skip this subsection and read subsection 2.4.

2.3.2 Macros

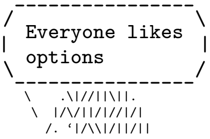
The following is the description of macros which differ in behaviour from those of version 2.

`\ducksay` `\ducksay[⟨options⟩]{⟨message⟩}`

options might include any of the options described in subsubsection 2.2.2 and subsubsection 2.3.3 if not otherwise specified. Prints an ⟨*animal*⟩ saying ⟨*message*⟩. ⟨*message*⟩ is not read in verbatim. Multi-line ⟨*message*⟩s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.

`\duckthink` `\duckthink[⟨options⟩]{⟨message⟩}`

options might include any of the options described in subsubsection 2.2.2 and subsubsection 2.3.3 if not otherwise specified. Prints an ⟨*animal*⟩ thinking ⟨*message*⟩. ⟨*message*⟩ is not read in verbatim. Multi-line ⟨*message*⟩s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.



2.3.3 Options

The following options are available to `\ducksay`, `\duckthink`, and `\DucksayOptions` and if not otherwise specified also as package options:

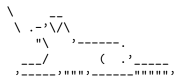
`bubble=⟨code⟩`
use ⟨*code*⟩ in a group right before the bubble (for font switches). Might be used as a package option but not all control sequences work out of the box there.

`body=⟨code⟩`
use ⟨*code*⟩ in a group right before the body (meaning the ⟨*animal*⟩). Might be used as a package option but not all control sequences work out of the box there. E.g. to right-align the ⟨*animal*⟩ to the bubble, use `body=\hfill`.

`align=⟨valign⟩`
use ⟨*valign*⟩ as the vertical alignment specifier given to the `tabular` which is around the contents of `\ducksay` and `\duckthink`.

`msg-align=⟨halign⟩`
use ⟨*halign*⟩ for alignment of the rows of multi-line ⟨*message*⟩s. It should match a `tabular` column specifier. Default is `l`. It only affects the contents of the speech bubble not the bubble.

`rel-align=⟨column⟩`
use ⟨*column*⟩ for alignment of the bubble and the body. It should match a `tabular` column specifier. Default is `l`.



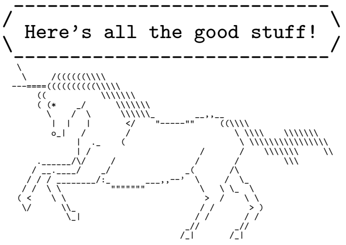
`wd=`⟨*count*⟩ in order to detect the width the ⟨*message*⟩ is expanded. This might not work out for some commands (e.g. `\url` from hyperref). If you specify the width using `wd` the ⟨*message*⟩ is not expanded and therefore the command *might* work out. ⟨*count*⟩ should be the character count.

`ht=`⟨*count*⟩ you might explicitly set the height (the row count) of the ⟨*message*⟩. This only has an effect if you also specify `wd`.

2.3.4 Defects

```
 ----------
/          \
\ Ohh, no! /
 ----------
    \
     \ (.)_(.)
    _ (   _   ) _
   / \/`-----'\/ \
 __\ ( (     ) ) /__
 )   /\ \._./ /\   (
  )_/ /|\   /|\ \_(
```

- no automatic line wrapping
- message width detection based on token count with `\edef` expansion, might fail badly



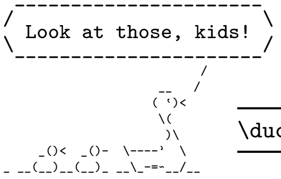
2.4 Version 2

2.4.1 Introducktion

Version 2 is the current version of `ducksay`. It features automatic line wrapping (if you specify a fixed width) and in general more options (with some nasty argument parsing).

If you're already used to version 1 you should note one important thing: You should only specify the `version` and the `ligatures` during package load time as arguments to `\usepackage`. The other keys might not work or do unintended things and only don't throw errors or warnings because of the legacy support of version 1. After the package is loaded, keys only used for version 1 will throw an error.

2.4.2 Macros



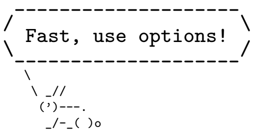
The following is the description of macros which differ in behaviour from those of version 1.

`\ducksay` `\ducksay[⟨options⟩]{⟨message⟩}`

options might include any of the options described in subsubsection 2.2.2 and subsubsection 2.4.3 if not otherwise specified. Prints an ⟨*animal*⟩ saying ⟨*message*⟩.
The ⟨*message*⟩ can be read in in four different ways. For an explanation of the ⟨*message*⟩ reading see the description of the `arg` key in subsubsection 2.4.3.
The height and width of the message is determined by measuring its dimensions and the bubble will be set accordingly. The box surrounding the message will be placed both horizontally and vertically centred inside of the bubble. The output utilizes LaTeX3's coffin mechanism described in `interface3.pdf` and the documentation of xcoffins.

`\duckthink` `\duckthink[⟨options⟩]{⟨message⟩}`

The only difference to `\ducksay` is that in `\duckthink` the ⟨*animal*⟩s think the ⟨*message*⟩ and don't say it.



2.4.3 Options

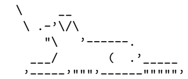
In version 2 the following options are available. Keep in mind that you shouldn't use them during package load time but in the arguments of `\ducksay`, `\duckthink` or `\DucksayOptions`.

`arg=`⟨*choice*⟩

specifies how the ⟨*message*⟩ argument of `\ducksay` and `\duckthink` should be read in. Available options are `box`, `tab` and `tab*`:

`box` the argument is read in either as a `\hbox` or a `\vbox` (the latter if a fixed width is specified with either `wd` or `wd*`). Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work (provided that you don't use `\ducksay` or `\duckthink` inside of an argument of another macro of course).

`tab` the argument is read in as the contents of a `tabular`. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will *not* work. This mode comes closest to the behaviour of version 1 of `ducksay`.



`tab*`
: the argument is read in as the contents of a `tabular`. However it is read in verbatim and uses `\scantokens` to rescan the argument. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work. You can't use `\ducksay` or `\duckthink` as an argument to another macro in this mode however.

`b`
: shortcut for `out-v=b`.

`body=`⟨*`font`*⟩
: add ⟨*`font`*⟩ to the font definitions in use to typeset the ⟨*`animal`*⟩'s body.

`body*=`⟨*`font`*⟩
: clear any definitions previously made (including the package default) and set the font definitions in use to typeset the ⟨*`animal`*⟩'s body to ⟨*`font`*⟩. The package default is `\verbatim@font`. In addition `\frenchspacing` will always be used prior to the defined ⟨*`font`*⟩.

`body-align=`⟨*`choice`*⟩
: sets the relative alignment of the ⟨*`animal`*⟩ to the ⟨*`message`*⟩. Possible choices are `l`, `c` and `r`. For `l` the ⟨*`animal`*⟩ is flushed to the left of the ⟨*`message`*⟩, for `c` it is centred and for `r` it is flushed right. More fine grained control over the alignment can be obtained with the keys `msg-to-body`, `body-to-msg`, `body-x` and `body-y`. Package default is `l`.

`body-bigger=`⟨*`count`*⟩
: vertically enlarge the body by ⟨*`count`*⟩ empty lines added to the bottom. This way top-aligning two different body types is easier (by actually bottom aligning the two):

```
             --------
            / Buuuh! \
            \--------/
            /       |
     .-.   /      \[T]/
    (o o)
    / O |
   /   /
   `~~~'
```

```
\ducksay[ghost,body-x=-7mm,b,body-mirrored]{Buuuh!}
\ducksay[crusader,body-bigger=4,b,out-h=r,no-bubble]{}
```

`body-mirrored=`⟨*`bool`*⟩
: if set true the ⟨*`animal`*⟩ will be mirrored along its vertical centre axis. Package default is `false`. If you set it `true` you'll most likely need to manually adjust the alignment of the body with one or more of the keys `body-align`, `body-to-msg`, `msg-to-body`, `body-x` and `body-y`.

`body-to-msg=`⟨*`pole`*⟩
: defines the horizontal coffin ⟨*`pole`*⟩ to be used for the placement of the ⟨*`animal`*⟩ beneath the ⟨*`message`*⟩. See `interface3.pdf` and the documentation of xcoffins for information about coffin poles.

`body-x=`⟨*`dimen`*⟩
: defines a horizontal offset of ⟨*`dimen`*⟩ length of the ⟨*`animal`*⟩ from its placement beneath the ⟨*`message`*⟩.

`body-y=`⟨*`dimen`*⟩
: defines a vertical offset of ⟨*`dimen`*⟩ length of the ⟨*`animal`*⟩ from its placement beneath the ⟨*`message`*⟩.

`bubble=`⟨*`font`*⟩
: add ⟨*`font`*⟩ to the font definitions in use to typeset the bubble. This does not affect the ⟨*`message`*⟩ only the bubble put around it.

`bubble*=`⟨*font*⟩
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the bubble to ⟨*font*⟩. This does not affect the ⟨*message*⟩ only the bubble put around it. The package default is `\verbatim@font`.

`bubble-bot-kern=`⟨*dimen*⟩
specifies a vertical offset of the placement of the lower border of the bubble from the bottom of the left and right borders.

`bubble-delim-left-1=`⟨*token list*⟩
the left delimiter used if only one line of delimiters is needed. Package default is `(`.

`bubble-delim-left-2=`⟨*token list*⟩
the upper most left delimiter used if more than one line of delimiters is needed. Package default is `/`.

`bubble-delim-left-3=`⟨*token list*⟩
the left delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.

`bubble-delim-left-4=`⟨*token list*⟩
the lower most left delimiter used if more than one line of delimiters is needed. Package default is `\`.

`bubble-delim-right-1=`⟨*token list*⟩
the right delimiter used if only one line of delimiters is needed. Package default is `)`.

`bubble-delim-right-2=`⟨*token list*⟩
the upper most right delimiter used if more than one line of delimiters is needed. Package default is `\`.

`bubble-delim-right-3=`⟨*token list*⟩
the right delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.

`bubble-delim-right-4=`⟨*token list*⟩
the lower most right delimiter used if more than one line of delimiters is needed. Package default is `/`.

`bubble-delim-top=`⟨*token list*⟩
the delimiter used to create the top and bottom border of the bubble. The package default is `{-}` (the braces are important to suppress ligatures here).

`bubble-side-kern=`⟨*dimen*⟩
specifies the kerning used to move the sideways delimiters added to fill the gap for more than two lines of bubble height. (the left one is moved to the left, the right one to the right)

`bubble-top-kern=`⟨*dimen*⟩
specifies a vertical offset of the placement of the upper border of the bubble from the top of the left and right borders.

`c` shortcut for `out-v=vc`.

`col=`⟨*column*⟩
specifies the used column specifier used for the ⟨*message*⟩ enclosing `tabular` for `arg=tab` and `arg=tab*`. Has precedence over `msg-align`. You can also use more than one column this way: `\ducksay[arg=tab,col=cc]{ You & can \\ do & it }` would be valid syntax.

`hpad=`⟨*count*⟩
Add ⟨*count*⟩ times more `bubble-delim-top` instances than necassary to the upper and lower border of the bubble. Package default is 2.

`ht=`⟨*count*⟩ specifies a minimum height (in lines) of the ⟨*message*⟩. The lines' count is that of the needed lines of the horizontal bubble delimiters. If the count of the actually needed lines is smaller than the specified ⟨*count*⟩, ⟨*count*⟩ lines will be used. Else the required lines will be used.

`ignore-body=`⟨*bool*⟩
If set `true` the ⟨*animal*⟩'s body will be added to the output but it will not contribute to the bounding box (so will not take up any space).

`msg=`⟨*font*⟩ add ⟨*font*⟩ to the font definitions in use to typeset the ⟨*message*⟩.

`msg*=`⟨*font*⟩ clear any definitions previously made (including the package default) and set the font definitions in use to typeset the ⟨*message*⟩ to ⟨*font*⟩. The package default is `\verbatim@font`.

`MSG=`⟨*font*⟩ same as `msg=`⟨*font*⟩`, bubble=`⟨*font*⟩.

`MSG*=`⟨*font*⟩ same as `msg*=`⟨*font*⟩`, bubble*=`⟨*font*⟩.

`msg-align=`⟨*choice*⟩
specifies the alignment of the ⟨*message*⟩. Possible values are `l` for flushed left, `c` for centred, `r` for flushed right and `j` for justified. If `arg=tab` or `arg=tab*` the `j` choice is only available for fixed width contents. Package default is `l`.

`msg-align-c=`⟨*token list*⟩
set the ⟨*token list*⟩ which is responsible to typeset the message centred if the option `msg-align=c` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by array is used afterwards. The package default is `\centering`. It might be useful if you want to use ragged2e's `\Centering` for example.

`msg-align-j=`⟨*token list*⟩
set the ⟨*token list*⟩ which is responsible to typeset the message justified if the option `msg-align=j` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by array is used afterwards. The package default is empty as justification is the default behaviour of contents of a `p` column and of a `\vbox`. It might be useful if you want to use ragged2e's `\justifying` for example.

`msg-align-l=`⟨*token list*⟩
set the ⟨*token list*⟩ which is responsible to typeset the message flushed left if the option `msg-align=l` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by array is used afterwards. The package default is `\raggedright`. It might be useful if you want to use ragged2e's `\RaggedRight` for example.

`msg-align-r=`⟨*token list*⟩

set the ⟨*token list*⟩ which is responsible to typeset the message flushed right if the option `msg-align=r` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by `array` is used afterwards. The package default is `\raggedleft`. It might be useful if you want to use `ragged2e`'s `\RaggedLeft` for example.

`msg-to-body=`⟨*pole*⟩

defines the horizontal coffin ⟨*pole*⟩ to be used as the reference point for the placement of the ⟨*animal*⟩ beneath the ⟨*message*⟩. See `interface3.pdf` and the documentation of xcoffins for information about coffin poles.

`no-bubble=`⟨*bool*⟩

If `true` the ⟨*message*⟩ will not be surrounded by a bubble. Package default is of course `false`.

`none=`⟨*bool*⟩ One could say this is a special animal. If `true` no animal body will be used (resulting in just the speech bubble). Package default is of course `false`.

`out-h=`⟨*pole*⟩

defines the horizontal coffin ⟨*pole*⟩ to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See `interface3.pdf` and the documentation of xcoffins for information about coffin poles.

`out-v=`⟨*pole*⟩

defines the vertical coffin ⟨*pole*⟩ to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See `interface3.pdf` and the documentation of xcoffins for information about coffin poles.

`out-x=`⟨*dimen*⟩

specifies an additional horizontal offset of the print out of the complete result of `\ducksay` and `\duckthink`.

`out-y=`⟨*dimen*⟩

specifies an additional vertical offset of the print out of the complete result of `\ducksay` and `\duckthink`

`strip-spaces=`⟨*bool*⟩

if set `true` leading and trailing spaces are stripped from the ⟨*message*⟩ if `arg=box` is used. Initially this is set to `false`.

`t` shortcut for `out-v=t`.

`vpad=`⟨*count*⟩

add ⟨*count*⟩ to the lines used for the bubble, resulting in ⟨*count*⟩ more lines than necessary to enclose the ⟨*message*⟩ inside of the bubble.

`wd=`⟨*count*⟩ specifies the width of the ⟨*message*⟩ to be fixed to ⟨*count*⟩ times the width of an upper case M in the ⟨*message*⟩'s font declaration. A value smaller than 0 is considered deactivated, else the width is considered as fixed. For a fixed width the argument of `\ducksay` and `\duckthink` is read in as a `\vbox` for `arg=box` and the column definition uses a `p`-type column for `arg=tab` and `arg=tab*`. If both `wd` is not smaller than 0 and `wd*` is not smaller than 0pt, `wd*` will take precedence.

`wd*=`⟨*`dimen`*⟩ specifies the width of the ⟨*`message`*⟩ to be fixed to ⟨*`dimen`*⟩. A value smaller than 0pt is considered deactivated, else the width is considered as fixed. For a fixed width the argument of `\ducksay` and `\duckthink` is read in as a `\vbox` for `arg=box` and the column definition uses a `p`-type column for `arg=tab` and `arg=tab*`. If both `wd` is not smaller than 0 and `wd*` is not smaller than 0pt, `wd*` will take precedence.

`wd-eq-body=`⟨*`bool`*⟩
if this is `true`, `wd` is smaller than 0, and `wd*` is smaller than 0pt the ⟨*`message`*⟩ will be as wide as the ⟨*`animal`*⟩'s body. Note that because the ⟨*`animal`*⟩ bodies contain white space on their left end and due to the additional horizontal bubble delimiters the bubble will be wider than the ⟨*`animal`*⟩'s body. If the `none` option was also used this option has no effect.

```
 ________________
/                \
  We rely on you
\                /
 ----------------
                  /
       ___   `._ /
      /   \__/__>
     /_  \  _/
    // \ /,/
   //   \\
,_/'     `\,
```

```
 ___________________
/                   \
|     This is a     |
| bug(fix) release. |
\                   /
 -------------------
  \  o   o
   \ _\_/_
    /__!__\
  '| o : o |'
  -|o  :  o|-
  ,| o : o |,
    \__:__/
```

2.5 Dependencies

The package depends on the LaTeX kernel, for older versions of LaTeX the two packages xparse and l3keys2e and all of their dependencies are loaded. Version 2 additionally depends on array and grabbox.

2.6 Available Animals

The following animals are provided by this package. I did not create them (but altered some), they belong to their original creators.

```
 ______
( duck )
 ------
   \
    \   __
      >(' )
        )/
       /(
      /  `----/
      \  ~=- /
    ~^~^~^~^~^~^~^
```

```
 ____________
( small-duck )
 ------------
   \
    \
      >()_
       (__)__ _
```

```
 _____________
( duck-family )
 -------------
   \
    \   __
      >(' )
        )/
       /(
      /  `----/  -()_  >()_
    __\__~=-_/__ _(__)__(__)__ _
```

```
 ______________
( small-rabbit )
 --------------
   \
    \ _//
     (')---.
      _/-_( )o
```

```
 __________
( squirrel )
 ----------
  \          ,;:;;,
   \    ,   ;;;;;
     .=',   ;:;;:,
    /_', "=. ';:;:;
    @=:__,  \,;:;:'
      _(\.=  ;:;;'
     `"_(  _/="`
      `"'``
```

```
 _____
( cow )
 -----
  \  ^__^
   \ (oo)\_______
     (__)\       )\/\
         ||----w |
         ||     ||
```

```
 _____
( tux )
 -----
  \
   \  .--.
     |o_o |
     |\_/ |
    //   \ \
   (|     | )
  /'\_   _/`\
  \___)=(___/
```

```
 _________
( head-in )
 ---------
  \
   \ ^__^         /
     (oo)\_______/  _________
     (__)\       )=(  ____|_ \_____
         ||----w |  \ \     \_____ |
         ||     ||  ||           ||
```

```
 _____
( pig )
 -----
  \     _//| .-~~~-.
   \ _/oo  }       }-@
    ('')_  }       |
     `--'| { }--{  }
          //_/  /_/
```

```
 ______
( frog )
 ------
   \
    \ (.)_(.)
   _ (   _   ) _
  / \/`-----'\/ \
__\ ( (     ) ) /__
)   /\ \._./ /\   (
 )_/ /|\   /|\ \_(
```

```
 _________
( snowman )
 ---------
  \
   \_[_]_
     (")
  >-( : )-<
   (__:__)
```

```
 _________
( ladybug )
 ---------
  \  o   o
   \ _\_/_
    /__!__\
  '| o : o |'
  -|o  :  o|-
  ,| o : o |,
    \__:__/
```

```
 _______
( bunny )
 -------
  \
   \     /
     /\ /
      ( )
    .( o ).
```

```
 ________
( dragon )
 --------
      \                   / \  //\
       \    |\___/|      /   \//  \\
        \   /0  0  \__  /    //  | \ \
           /     /  \/_/    //   |  \  \
           @_^_@'/   \/_   //    |   \   \
           //_^_/     \/_ //     |    \    \
        ( //) |        \///      |     \     \
      ( / /) _|_ /   )  //       |      \     _\
    ( // /) '/,_ _ _/  ( ; -.    |    _ _\.-~        .-~~~^-.
  (( / / )) ,-{        _      `-.|.-~-.           .~         `.
 (( // / ))  '/\      /                 ~-. _ .-~      .-~^-.  \
 (( /// ))      `.   {            }                   /      \  \
  (( / ))     .----~-.\        \-'                 .~         \  `. \^-.
             ///.----..>        \             _ -~             `.  ^-`  ^-_
               ///-._ _ _ _ _ _ _}^ - - - - ~                     ~-- ,.-~
                                                                  /.-~
```

```
 ___________
( sodomized )
 -----------
  \             _
   \           (_)
     ^__^       / \
     (oo)\_____/_\ \
     (__)\       ) /
         ||----w ((
         ||     ||>>
```

```
 __________
( hedgehog )
 ----------
  \    .\|//||\||.
   \  |/\/||/|//|/|
     /. `|/\\|/||/||
    o__,_|//|/||\||'
```

```
 __________
( platypus )
 ----------
  \          _.-^~~^^`~-,,,~~''```~-''``~'``~,
   \ ______,'  -o  :. _     .      ;     ,'`,  `.
    (      -\.._,.;;'._ ,(   }    _`_-_,,    `,  `,
     ``~~~~~~~'  ((/'(((____/~~'(,(,___>       `~'
```

```
 ________
( turtle )
 --------
   \
    \  __
      /_'\ .,-;-;-,.
       \ (/_/_|_\_\_\_
       /\=<_><_><_><_>-'
      /_/'-\_\====\_\'
      ""    ""     ""
```

```
 __________
( kangaroo )
 ----------
   \
    \ _,'   ___
     <__\__/   \
        \_  /  _\
          \,\ / \\
            //   \\
          ,/'     `\_,
```

```
 _____________
( small-horse )
 -------------
   \   _,_
    \ /._ \\
     /_/ |_\\ _ __
         /  \\    \\
         \ _ -- _ /||
         | |   | | ||
         | |   | |
```

```
 _____
( dog )
 -----
   \     __
    \ .-'\/\
       "\   '______.
     ___/       (  .'_____
    ,_____,""",______"""""",
```

```
 _______
( sheep )
 -------
   \        _,_,_,_,_,
    \    .:(  ,)      ),
        (__,     (,      ),
       / o (  ,)     ,) )>
      (___(,      (,   ,)
           (,   ,)    ,)
            `-_,---_,-'
              ||   ||
```

```
 ________
( rabbit )
 --------
   \
    \   / \`\         __
     \ |  \ `\      /`/ \
       \_/`\  \-"-/` /\  \
            |       |  \  |
            (d     b)   \_/
            /       \
        ,".|.'.\_/.'.|.",
       /   /\' _|_ '/\   \
       |  /  ,_`"`_,  \  |
       | |             | |
       | \    \   /    / |
        \ \    \ /    / /
         `"`\   :   /'"`
             `""`""`
```

```
 _______
( snail )
 -------
   \
    \          .-""-.
      oo      ; .-.  :
       \\__..-:'.__.' )._
        "-._.._'.__.-'_.."
```

```
 _______
( whale )
 -------
   \                |-.
    \    .-""-._     \ \.--|
     \  /       `-..__)  ,-'
       |     .          /
        \--.__,   .__.,'
         `-.___'._\_.'
```

```
 _______
( snake )
 -------
   \
    \     /^\/^\
     \  _|__| O |
      /'     \_/ \
  \/ |_________/  \               \
   \_/ \_______    \               \
               `|   |              |\
               /   /  _---_        | |
              /   /  /  __ "-_    ," |
             |   "--"  /  "-_  "--" ,"
              "-_____-"      "-___-"
```

```
 _____
( cat )
 -----
   \
    \  _                ___       .--.
       \`.|\..----...-'`   `-._.-' .-'
       /  ' `         ,       __.-'
       )/' _/     \   `-_,   /
       `_'" `"\_  ,_.-;_.-\_ ',
           _.-'_./   {_.'   ; /
          {_.-``-'         {_/
```

```
 ____________
( sleepy-cat )
 ------------
   \
    \  |\      _,,,---,,_     _._
       /,`.-'`'    -.   )'._,'.-,)
      |,4-  ) )-,_. ,\ ( `-.-'
     '---''(_/--'  `-'\_)
```

```
 ___________________
( schroedinger-dead )
 -------------------
   \
    \ _.--"""--._
     |           |
     |    -|-    |
     |     |     |
     |           |
     |   Felix   |
   __|___________|__ _
    o    .   .    .
       ~ .  o   o
       .   ~  .
```

```
 ____________________
( schroedinger-alive )
 --------------------
   \
    \ ,_     _
      |\\__,'/
      / _  _ |    ,--.
     (  @  @ )   / ,-'
      \  _T_/-._( (
      /         `. \
     |         _  \ |
      \ \ ,  /      |
       || |-_\__   /
      ((_/`(____,-'
```

```
 __________
( crusader )
 ----------
        |
      \[T]/
```

```
 ________
( knight )
 --------
      \
       \   ,-"""-.
        \  | === |
           )  |  (
        .=='\" "/`==.
      .'\   (`:')   /`.
    _/_ |_.-' : `-._|__\_
   <___>'\    :   / `<___>
   /  /   >=======<   /  /
 _/ .'   /  ,-:-.   \/=,'
/ _/    |__/v^v^v\__) \
\(\)     |V^V^V^V^V|\_/
 (\\     \`---|---'/
   \\     \-._|_,-/
    \\     |__|__|
     \\   <___X___>
      \\   \..|../
       \\   \ | /
        \\  /V|V\
         \|/  |  \
          '__' `__`
```

```
 _______
( fairy )
 -------
   \
    \           .oOOb
      ..     .oO    O
     '::;  d        O
      ;;;;d     ..oO
 *   ::O;;;'OooO
~"\. dp'(O.o.
   \op    'oOb
           obU
          dop
         dop
          PO
         O 'b
         l  P.
         /   ;
         ,
```

```
 _______
( ghost )
 -------
   \
    \ .-.
     (o o)
     | O \
      \   \
       `~~~'
```

```
 _________
( unicorn )
 ---------
   \
    \     /((((((\\\\
  ---====((((((((((\\\\\
       ((           \\\\\\\
       ( (*    _/      \\\\\\\
         \    /  \      \\\\\\_         __,,__
          |  |   |       </    "-----""      ((\\\\
          o_|   /        /                      \ \\\\    \\\\\\\
               |  ._    (                        \ \\\\\\\\\\\\\\\\
               | /                       /       /    \\\\\\\
       .______/\/     /                 /       /         \\\
      / __.____/    _/                _(       /\
     / / / ________/:_       ___,,--'   \     /  \_
    / /  \ \          """"""""           \   \ \_  \
   ( <    \ \                             >  /    \ \
    \/     \\_                           / /       > )
             \_|                        / /       / /
                                     _//        _//
                                    /_|        /_|
```

```
 ____________
( small-yoda )
 ------------
  \
   \
   __.-._
   '-._"7'
    /'.-c
    |  /T
   _)_/LI
```

```
 ______
( r2d2 )
 ------
   \
    \ ,-----.
    ,'_/_|_\_`.
   /<<::8[0]::>\
  _|-----------|_
 |  | ====-=- |  |
 |  | -=-==== |  |
 \  | ::::|()||  /
  | | ....|()|| |
  | |_________| |
  | |\_______/| |
 /   \ /   \ /   \
 `---' `---' `---'
```

```
 _______
( vader )
 -------
   \     _.-'~~~~~~`-._
    \   /      ||      \
       /       ||       \
      |        ||        |
      | _______||_______ |
      |/ ----- \/ ----- \|
     /  (     )  (     )  \
    / \  ----- () -----  / \
   /   \      /||\      /   \
  /     \    /||||\    /     \
 /       \  /||||||\  /       \
/_        \0========0/        _\
  `--...__|`-._  _.-'|__...--'
          |    `'    |
```

```
 ___________
( yoda-head )
 -----------
  \
   \
    \
                   ____
                _.' :  `._
            .-.'`.  ;   .'`.-.
  __      / : ___\ ;  /___ ; \      __
,'_ ""--.:__;".-.";: :".-.":__;.--"" _`,
:' `.t""--.. '<@.`;_  ',@>` ..--""j.' `;
     `:-.._J '-.-'L__ `-- ' L_..-;'
       "-.__ ;  .-"  "-.  : __.-"
           L ' /.------.\ ' J
            "-.   "--"   .-"
           __.l"-:_JL_;-";.__
        .-j/'.;  ;""""  / .'\"-.
      .' /:`. :  :     /.".'';  `.
   .-"  / ;`.".  :    ."."   :    "-.
.+"-.  : :   ".".". ."."      ;-._    \
```

```
 ______
( yoda )
 ------
  \
   \
    \
                    ____
                 _.' :  `._
             .-.'`.  ;   .'`.-.
   __      / : ___\ ;  /___ ; \      __
 ,'_ ""--.:__;".-.";: :".-.":__;.--"" _`,
 :' `.t""--.. '<@.`;_  ',@>` ..--""j.' `;
      `:-.._J '-.-'L__ `-- ' L_..-;'
        "-.__ ;  .-"  "-.  : __.-"
            L ' /.------.\ ' J
             "-.   "--"   .-"
            __.l"-:_JL_;-";.__
         .-j/'.;  ;""""  / .'\"-.
       .' /:`. :  :     /.".'';  `.
    .-"  / ;`.".  :    ."."   :    "-.
 .+"-.  : :   ".".". ."."      ;-._    \
 ; \  `.; ; .   "."-"."        : : "+. ;
 :  ;   ; ;  .   ."."    ;     : ;  : \:
 ;  :   ; :     / /     /  ,   ;:   ;  :
: \  ;  :  ;   ; /     :  ,   : ;  /  ::
;  ; :   ; : ;  ;      ;      ;   :   ;:
:  :  ;  :  ;. ;      '      : :  ;  : ;
;\    :   ; : .          ,   ; ;     ; ;
: `."-;   :  ;      .   ;   :  ;    /  ;
 ;    -:   ; :      ,  ,    ;  :  .-"   :
 :\     \  :  ;    ,        : \.-"      :
  ;`.    \  ; :   .   ,    ;.'_..--  / ;
  :  "-.  "-:  ;     ,    :/."       .'  :
   \         \ :    :      ;/  __        :
    \       .-`.\        /t-""  ":-+.   :
     `.  .-"    `l     __/ /`.:   ; ; \  ;
       \   .-" .-"-.-"  .'.'j  \  /   ;/
        \ / .-"   /.      .'.' ;_:'    ;
         :-""-.`./-.'     /    `.___.'
               \ `t  ._  /
                "-.t-._:'
```

```
 ___________
( only-tail )
 -----------
  \
   \
```

```
 ____________
( only-tail3 )
 ------------
  \
   \
    \
```

2.7 License and Bug Reports

This work may be distributed and/or modified under the conditions of the LaTeX Project Public License (LPPL), either version 1.3c of this license or (at your option) any later version. The latest version of this license is in the file: http://www.latex-project.org/lppl.txt

The package is hosted on https://github.com/Skillmon/ltx_ducksay, you might report bugs there.

```
 __________________
/ WTFPL would be a \
| better license.  |
\                  /
 ------------------
\           ,;:;;,
 \    ,     ;;;;;
   .=',     ;:;;:,
  /_', "=. ';:;:;
  @=:__,  \,;:;:'
    _(\.=  ;:;;'
   `"_(  _/="`
    `"'``
```

```
 ----------------------------
/                            \
|   Only rebel scum reads    |
|      documentation!        |
|    Join the dark side,     |
| read the implementation.   |
\                            /
 ----------------------------
        \     _.-'~~~~~~~`-._
         \   /      ||      \
            /       ||       \
           |        ||        |
           | _______||_______ |
           |/ ----- \/ ----- \|
          /  (     )  (     )  \
         / \  ----- () -----  / \
        /   \      /||\      /   \
       /     \    /||||\    /     \
      /       \  /||||||\  /       \
     /_        \O========O/        _\
       `--...__|`-._  _.-'|__...--'
               |    `'    |
```

3 Implementation

```
⟨*pkg⟩
```

3.1 Shared between versions

3.1.1 Variables

3.1.1.1 Integers

```
\int_new:N \l_ducksay_msg_width_int
\int_new:N \l_ducksay_msg_height_int
\int_new:N \l_ducksay_tail_symbol_count_int
```

3.1.1.2 Sequences

```
\seq_new:N \l_ducksay_msg_lines_seq
\seq_new:N \l_ducksay_defined_animals_seq
```

3.1.1.3 Token lists

```
\tl_new:N \l_ducksay_align_tl
\tl_new:N \l_ducksay_msg_align_tl
\tl_new:N \l_ducksay_animal_tl
\tl_new:N \l_ducksay_body_tl
\tl_new:N \l_ducksay_bubble_tl
\tl_new:N \l_ducksay_tmpa_tl
\tl_new:N \l_ducksay_tail_symbol_out_one_tl
\tl_new:N \l_ducksay_tail_symbol_out_two_tl
\tl_new:N \l_ducksay_tail_symbol_in_tl
```

3.1.1.4 Boolean

```
\bool_new:N \l_ducksay_version_one_bool
\bool_new:N \l_ducksay_version_two_bool
\bool_new:N \l_ducksay_random_animal_bool
```

3.1.1.5 Boxes

```
\box_new:N \l_ducksay_tmpa_box
```

3.1.2 Regular Expressions

Regular expressions for `\AddColoredAnimal`

```
\regex_const:Nn \c_ducksay_textcolor_regex
  { \\textcolor\{(.*?)\}\{(.*?)\} }
\regex_const:Nn \c_ducksay_color_delim_regex
  { \\bgroup\\color\{(.*?)\}(.*)\\egroup }
\regex_const:Nn \c_ducksay_color_regex
  { \\color\{(.*?)\} }
```

3.1.3 Messages

```
\msg_new:nnn { ducksay } { load-time-only }
  { The~`#1`~key~is~to~be~used~only~during~package~load~time. }
```

3.1.4 Key-value setup

```
\keys_define:nn { ducksay }
  {
    ,bubble .tl_set:N      = \l_ducksay_bubble_tl
    ,body   .tl_set:N      = \l_ducksay_body_tl
```

```
    ,align  .tl_set:N      = \l_ducksay_align_tl
    ,align  .value_required:n = true
    ,wd     .int_set:N     = \l_ducksay_msg_width_int
    ,wd     .initial:n     = -\c_max_int
    ,wd     .value_required:n = true
    ,ht     .int_set:N     = \l_ducksay_msg_height_int
    ,ht     .initial:n     = -\c_max_int
    ,ht     .value_required:n = true
    ,animal .code:n        =
      { \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } } }
    ,animal .initial:n     = duck
    ,msg-align .tl_set:N   = \l_ducksay_msg_align_tl
    ,msg-align .initial:n  = l
    ,msg-align .value_required:n = true
    ,rel-align .tl_set:N   = \l_ducksay_rel_align_tl
    ,rel-align .initial:n  = l
    ,rel-align .value_required:n = true
    ,ligatures .tl_set:N   = \l_ducksay_ligatures_tl
    ,ligatures .initial:n  = { ‘<>,’- }
    ,tail-1    .tl_set:N   = \l_ducksay_tail_symbol_out_one_tl
    ,tail-1    .initial:x  = \c_backslash_str
    ,tail-2    .tl_set:N   = \l_ducksay_tail_symbol_out_two_tl
    ,tail-2    .initial:x  = \c_backslash_str
    ,no-tail   .meta:n     = { tail-1 = { ~ }, tail-2 = { ~ } }
    ,think     .meta:n     = { tail-1 = { O }, tail-2 = { o } }
    ,random    .bool_set:N = \l_ducksay_random_animal_bool
    ,say       .code:n     =
      {
        \exp_args:Nx \DucksayOptions
          { tail-1 = { \c_backslash_str }, tail-2 = { \c_backslash_str } }
      }
    ,schroedinger .code:n  =
      {
        \int_compare:nNnTF { \int_rand:n { 2 } } = \c_one_int
          { \keys_set:nn { ducksay } { animal = schroedinger-dead } }
          { \keys_set:nn { ducksay } { animal = schroedinger-alive } }
      }
    ,schroedinger .value_forbidden:n = true
    ,version   .choice:
    ,version / 1 .code:n   =
      {
        \bool_set_false:N \l_ducksay_version_two_bool
        \bool_set_true:N  \l_ducksay_version_one_bool
      }
    ,version / 2 .code:n   =
      {
        \bool_set_false:N \l_ducksay_version_one_bool
        \bool_set_true:N  \l_ducksay_version_two_bool
      }
    ,version   .initial:n  = 2
  }
\cs_if_exist:NTF \ProcessKeyOptions
  { \ProcessKeyOptions [ ducksay ] }
  {
    \RequirePackage { l3keys2e }
    \ProcessKeysOptions { ducksay }
  }
```

Undefine the load-time-only keys

```
\keys_define:nn { ducksay }
  {
    version .code:n = \msg_error:nnn { ducksay } { load-time-only } { version }
  }
```

3.1.4.1 Keys for `\AddAnimal`

Define keys meant for `\AddAnimal` and `\AddColoredAnimal` only in their own regime:

```
\IfFormatAtLeastTF{2025-06-01}
  {
    \keys_define:nn { ducksay / add-animal }
      { tail-symbol .tl_set:N = \l_ducksay_tail_symbol_in_tl }
  }
  {
    \keys_define:nn { ducksay / add-animal }
      {
        tail-symbol .code:n =
          \tl_set:Ne \l_ducksay_tail_symbol_in_tl { \tl_to_str:n {#1} }
      }
  }
\keys_define:nn { ducksay / add-animal }
  {
     tail-symbol .initial:o = \c_backslash_str
    ,tail-count  .int_set:N = \l_ducksay_tail_symbol_count_int
    ,tail-count  .initial:n = 2
    ,gen-tail-symbol .code:n =
      \tl_set:Ne \l_ducksay_tail_symbol_in_tl
        { \char_generate:nn { `#1 } { 12 } }
  }
```

3.1.5 Functions

3.1.5.1 Generating Variants of External Functions

```
\cs_generate_variant:Nn \tl_replace_once:Nnn { NVn }
\cs_generate_variant:Nn \tl_replace_all:Nnn { NVn }
\cs_generate_variant:Nn \keys_set:nn { nx }
```

3.1.5.2 Internal

`\__ducksay_everyeof:w`

```
\cs_set_eq:NN \__ducksay_everyeof:w \tex_everyeof:D
```

(*End of definition for* `\__ducksay_everyeof:w`*.*)

`\__ducksay_scantokens:w`

```
\cs_set_eq:NN \__ducksay_scantokens:w \tex_scantokens:D
```

(*End of definition for* `\__ducksay_scantokens:w`*.*)

\ducksay_replace_verb_newline:Nn

```
\IfFormatAtLeastTF{2024-06-01}
  {
    \cs_new_protected:Npn \ducksay_replace_verb_newline:Nn #1 #2
      { \tl_replace_all:Nnn #1 \obeyedline {#2} }
  }
  {
    \cs_new_protected:Npx \ducksay_replace_verb_newline:Nn #1 #2
      { \tl_replace_all:Nnn #1 { \char_generate:nn { 13 } { 12 } } {#2} }
  }
```

(*End of definition for* `\ducksay_replace_verb_newline:Nn`.)

\ducksay_replace_verb_newline_newline:Nn

```
\IfFormatAtLeastTF{2024-06-01}
  {
    \cs_new_protected:Npn \ducksay_replace_verb_newline_newline:Nn #1 #2
      { \tl_replace_all:Nnn #1 { \obeyedline \obeyedline } {#2} }
  }
  {
    \cs_new_protected:Npx \ducksay_replace_verb_newline_newline:Nn #1 #2
      {
        \tl_replace_all:Nnn #1
          { \char_generate:nn { 13 } { 12 } \char_generate:nn { 13 } { 12 } }
          {#2}
      }
  }
```

(*End of definition for* `\ducksay_replace_verb_newline_newline:Nn`.)

\ducksay_process_verb_newline:nnn

```
\cs_new_protected:Npn \ducksay_process_verb_newline:nnn #1 #2 #3
  {
    \tl_set:Nn \ProcessedArgument { #3 }
    \ducksay_replace_verb_newline_newline:Nn \ProcessedArgument { #2 }
    \ducksay_replace_verb_newline:Nn \ProcessedArgument { #1 }
  }
```

(*End of definition for* `\ducksay_process_verb_newline:nnn`.)

\ducksay_add_animal_inner:nnnn

```
\cs_new_protected:Npn \ducksay_add_animal_inner:nnnn #1 #2 #3 #4
  {
    \group_begin:
      \keys_set:nn { ducksay / add-animal } { #1 }
      \tl_set:Nn \l_ducksay_tmpa_tl { \ #3 }
      \int_compare:nNnTF { \l_ducksay_tail_symbol_count_int } < { \c_zero_int }
        {
          \tl_replace_once:NVn
            \l_ducksay_tmpa_tl
            \l_ducksay_tail_symbol_in_tl
            \l_ducksay_tail_symbol_out_one_tl
          \tl_replace_all:NVn
            \l_ducksay_tmpa_tl
            \l_ducksay_tail_symbol_in_tl
            \l_ducksay_tail_symbol_out_two_tl
        }
        {
          \int_compare:nNnT { \l_ducksay_tail_symbol_count_int } >
            { \c_zero_int }
            {
              \tl_replace_once:NVn
                \l_ducksay_tmpa_tl
                \l_ducksay_tail_symbol_in_tl
                \l_ducksay_tail_symbol_out_one_tl
              \int_step_inline:nnn { 2 } { \l_ducksay_tail_symbol_count_int }
                {
                  \tl_replace_once:NVn
                    \l_ducksay_tmpa_tl
                    \l_ducksay_tail_symbol_in_tl
                    \l_ducksay_tail_symbol_out_two_tl
                }
            }
        }
      \tl_map_inline:Nn \l_ducksay_ligatures_tl
        { \tl_replace_all:Nnn \l_ducksay_tmpa_tl { ##1 } { { ##1 } } }
      \ducksay_replace_verb_newline:Nn \l_ducksay_tmpa_tl
        { \tabularnewline\null }
      \exp_args:NNnV
    \group_end:
    \tl_set:cn { l_ducksay_animal_#2_tl } \l_ducksay_tmpa_tl
    \exp_args:Nnx \keys_define:nn { ducksay }
      {
        #2 .code:n =
          {
            \exp_not:n { \tl_set_eq:NN \l_ducksay_animal_tl }
            \exp_not:c { l_ducksay_animal_#2_tl }
            \exp_not:n { \exp_args:NV \DucksayOptions }
            \exp_not:c { l_ducksay_animal_#2_options_tl }
          }
      }
    \tl_if_exist:cF { l_ducksay_animal_#2_options_tl }
      { \tl_new:c { l_ducksay_animal_#2_options_tl } }
    \IfBooleanT { #4 }
      { \keys_define:nn { ducksay } { default_animal .meta:n = { #2 } } }
    \seq_if_in:NnF \l_ducksay_defined_animals_seq { #2 }
      { \seq_push:Nn \l_ducksay_defined_animals_seq { #2 } }
  }
\cs_generate_variant:Nn \ducksay_add_animal_inner:nnnn { nnVn }
```

(*End of definition for* `\ducksay_add_animal_inner:nnnn`.)

`\ducksay_default_or_random_animal:`

```
\cs_new_protected:Npn \ducksay_default_or_random_animal:
  {
    \tl_if_empty:NT \l_ducksay_animal_tl
      {
        \bool_if:NTF \l_ducksay_random_animal_bool
```

```
     {
      \keys_set:nx { ducksay }
       { \seq_rand_item:N \l_ducksay_defined_animals_seq }
     }
     { \keys_set:nn { ducksay } { default_animal } }
   }
 }
```

(*End of definition for* `\ducksay_default_or_random_animal:`*.*)

3.1.5.3 Document level

`\DefaultAnimal`

```
\NewDocumentCommand \DefaultAnimal { m }
 {
  \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } }
 }
```

(*End of definition for* `\DefaultAnimal`*. This function is documented on page 3.*)

`\DucksayOptions`

```
\NewDocumentCommand \DucksayOptions { m }
 {
  \keys_set:nn { ducksay } { #1 }
 }
```

(*End of definition for* `\DucksayOptions`*. This function is documented on page 3.*)

`\AddAnimal`

```
\NewDocumentCommand \AddAnimal { s O{} m +v }
 {
  \ducksay_add_animal_inner:nnnn { #2 } { #3 } { #4 } { #1 }
 }
```

(*End of definition for* `\AddAnimal`*. This function is documented on page 3.*)

`\AddColoredAnimal`

```
\NewDocumentCommand \AddColoredAnimal { s O{} m +v }
 {
  \tl_set:Nn \l_ducksay_tmpa_tl { #4 }
  \regex_replace_all:NnN \c_ducksay_color_delim_regex
   { \c{bgroup}\c{color}\cB\{\1\cE\}\2\c{egroup} }
   \l_ducksay_tmpa_tl
  \regex_replace_all:NnN \c_ducksay_color_regex
   { \c{color}\cB\{\1\cE\} }
   \l_ducksay_tmpa_tl
  \regex_replace_all:NnN \c_ducksay_textcolor_regex
   { \c{textcolor}\cB\{\1\cE\}\cB\{\2\cE\} }
   \l_ducksay_tmpa_tl
  \ducksay_add_animal_inner:nnVn { #2 } { #3 } \l_ducksay_tmpa_tl { #1 }
 }
```

(*End of definition for* `\AddColoredAnimal`*. This function is documented on page 3.*)

\AnimalOptions

```
\NewDocumentCommand \AnimalOptions { s m m }
  {
    \tl_if_exist:cTF { l_ducksay_animal_#2_options_tl }
      {
        \IfBooleanTF { #1 }
          { \tl_set:cn }
          { \tl_put_right:cn }
      }
      { \tl_set:cn }
    { l_ducksay_animal_#2_options_tl } { #3, }
  }
```

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

3.1.6 Load the Correct Version and the Animals

```
\bool_if:NT \l_ducksay_version_one_bool
  { \file_input:n { ducksay.code.v1.tex } }
\bool_if:NT \l_ducksay_version_two_bool
  { \file_input:n { ducksay.code.v2.tex } }
```

```
\ExplSyntaxOff
\input{ducksay.animals.tex}
```

```
⟨/pkg⟩
```

3.2 Version 1

```
⟨*code.v1⟩
\ProvidesFile{ducksay.code.v1.tex}
  [\ducksay@date\space v\ducksay@version\space ducksay code version 1]
```

3.2.1 Functions

3.2.1.1 Internal

`\ducksay_longest_line:n`

Calculate the length of the longest line

```
\cs_new:Npn \ducksay_longest_line:n #1
  {
    \int_incr:N \l_ducksay_msg_height_int
    \exp_args:NNx \tl_set:Nn \l_ducksay_tmpa_tl { #1 }
    \regex_replace_all:nnN { \s } { \c { space } } \l_ducksay_tmpa_tl
    \int_set:Nn \l_ducksay_msg_width_int
      {
        \int_max:nn
          { \l_ducksay_msg_width_int } { \tl_count:N \l_ducksay_tmpa_tl }
      }
  }
```

(*End of definition for* `\ducksay_longest_line:n`*.*)

`\ducksay_open_bubble:`

Draw the opening bracket of the bubble

```
\cs_new:Npn \ducksay_open_bubble:
  {
    \begin{tabular}{@{}l@{}}
      \null\\
      \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 } { ( }
        {
          /
          \int_step_inline:nnn
            { 3 } { \l_ducksay_msg_height_int } { \\\kern-0.2em| }
          \\\detokenize{\ }
        }
      \\[-1ex]\null
    \end{tabular}
    \begin{tabular}{@{}l@{}}
      _\\
      \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
      \mbox { - }
    \end{tabular}
  }
```

(*End of definition for* `\ducksay_open_bubble:`*.*)

`\ducksay_close_bubble:`

Draw the closing bracket of the bubble

```
\cs_new:Npn \ducksay_close_bubble:
  {
    \begin{tabular}{@{}l@{}}
      _\\
      \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
      { - }
    \end{tabular}
    \begin{tabular}{@{}r@{}}
      \null\\
      \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 }
        { ) }
        {
          \detokenize {\ }
          \int_step_inline:nnn
            { 3 } { \l_ducksay_msg_height_int } { \\|\kern-0.2em }
          \\/
        }
      \\[-1ex]\null
    \end{tabular}
  }
```

(*End of definition for* `\ducksay_close_bubble:`.)

`\ducksay_print_msg:nn` Print out the message

```
\cs_new:Npn \ducksay_print_msg:nn #1 #2
  {
    \begin{tabular}{@{} #2 @{}}
      \int_step_inline:nn { \l_ducksay_msg_width_int } { _ } \\
      #1\\[-1ex]
      \int_step_inline:nn { \l_ducksay_msg_width_int } { { - } }
    \end{tabular}
  }
\cs_generate_variant:Nn \ducksay_print_msg:nn { nV }
```

(*End of definition for* `\ducksay_print_msg:nn`.)

`\ducksay_print:nn` Print out the whole thing

```
\cs_new:Npn \ducksay_print:nn #1 #2
  {
    \int_compare:nNnTF { \l_ducksay_msg_width_int } < { 0 }
      {
        \int_zero:N \l_ducksay_msg_height_int
        \seq_set_split:Nnn \l_ducksay_msg_lines_seq { \\ } { #1 }
        \seq_map_function:NN \l_ducksay_msg_lines_seq \ducksay_longest_line:n
      }
      {
        \int_compare:nNnT { \l_ducksay_msg_height_int } < { 0 }
          {
            \regex_count:nnN { \c { \\ } } { #1 } \l_ducksay_msg_height_int
            \int_incr:N \l_ducksay_msg_height_int
          }
      }
    \group_begin:
      \frenchspacing
      \verbatim@font
      \@noligs
      \begin{tabular}[\l_ducksay_align_tl]{@{}#2@{}}
        \l_ducksay_bubble_tl
        \begin{tabular}{@{}l@{}}
          \ducksay_open_bubble:
          \ducksay_print_msg:nV { #1 } \l_ducksay_msg_align_tl
          \ducksay_close_bubble:
```

```
        \end{tabular}\\
        \l_ducksay_body_tl
        \begin{tabular}{@{}l@{}}
          \l_ducksay_animal_tl
        \end{tabular}
      \end{tabular}
    \group_end:
  }
\cs_generate_variant:Nn \ducksay_print:nn { nV }
```

(*End of definition for* `\ducksay_print:nn`.)

`\ducksay_say_and_think:nn` Reset some variables

```
\cs_new:Npn \ducksay_say_and_think:nn #1 #2
  {
    \group_begin:
      \int_set:Nn \l_ducksay_msg_width_int  { -\c_max_int }
      \int_set:Nn \l_ducksay_msg_height_int { -\c_max_int }
      \keys_set:nn { ducksay } { #1 }
      \ducksay_default_or_random_animal:
      \ducksay_print:nV { #2 } \l_ducksay_rel_align_tl
    \group_end:
  }
```

(*End of definition for* `\ducksay_say_and_think:nn`.)

3.2.1.2 Document level

`\ducksay`

```
\NewDocumentCommand \ducksay { O{} m }
  {
    \ducksay_say_and_think:nn { #1 } { #2 }
  }
```

(*End of definition for* `\ducksay`. *This function is documented on page 8.*)

`\duckthink`

```
\NewDocumentCommand \duckthink { O{} m }
  {
    \ducksay_say_and_think:nn { think, #1 } { #2 }
  }
```

(*End of definition for* `\duckthink`. *This function is documented on page 8.*)

⟨/code.v1⟩

3.3 Version 2

```
⟨*code.v2⟩
\ProvidesFile{ducksay.code.v2.tex}
  [\ducksay@date\space v\ducksay@version\space ducksay code version 2]
```

Load the additional dependencies of version 2.

```
\RequirePackage{array,grabbox}
```

3.3.1 Messages

```
\msg_new:nnn { ducksay } { justify~unavailable }
  {
    Justified~content~is~not~available~for~tabular~argument~mode~without~fixed~
    width.~`l`~column~is~used~instead.
  }
\msg_new:nnn { ducksay } { v1-key-only }
  { The~`\l_keys_key_str`~key~is~only~available~for~`version=1`. }
\msg_new:nnn { ducksay } { zero-baselineskip }
  { Current~ baselineskip~ is~ 0pt. }
```

3.3.2 Variables

3.3.2.1 Token Lists

```
\tl_new:N \l_ducksay_msg_align_vbox_tl
```

3.3.2.2 Boxes

```
\box_new:N \l_ducksay_msg_box
```

3.3.2.3 Bools

```
\bool_new:N \l_ducksay_eat_arg_box_bool
\bool_new:N \l_ducksay_eat_arg_tab_verb_bool
\bool_new:N \l_ducksay_mirrored_body_bool
\bool_new:N \l_ducksay_msg_eq_body_width_bool
```

3.3.2.4 Coffins

```
\coffin_new:N \l_ducksay_body_coffin
\coffin_new:N \l_ducksay_bubble_close_coffin
\coffin_new:N \l_ducksay_bubble_open_coffin
\coffin_new:N \l_ducksay_bubble_top_coffin
\coffin_new:N \l_ducksay_msg_coffin
```

3.3.2.5 Dimensions

```
\dim_new:N \l_ducksay_hpad_dim
\dim_new:N \l_ducksay_bubble_bottom_kern_dim
\dim_new:N \l_ducksay_bubble_top_kern_dim
\dim_new:N \l_ducksay_msg_width_dim
```

3.3.3 Options

```
\keys_define:nn { ducksay }
  {
    ,arg .choice:
    ,arg / box  .code:n = \bool_set_true:N  \l_ducksay_eat_arg_box_bool
    ,arg / tab  .code:n =
      {
        \bool_set_false:N \l_ducksay_eat_arg_box_bool
        \bool_set_false:N \l_ducksay_eat_arg_tab_verb_bool
      }
    ,arg / tab* .code:n =
      {
        \bool_set_false:N \l_ducksay_eat_arg_box_bool
        \bool_set_true:N  \l_ducksay_eat_arg_tab_verb_bool
      }
    ,arg .initial:n = tab
    ,wd* .dim_set:N = \l_ducksay_msg_width_dim
    ,wd* .initial:n = -\c_max_dim
    ,wd* .value_required:n = true
    ,wd-eq-body    .bool_set:N = \l_ducksay_msg_eq_body_width_bool
    ,none          .bool_set:N = \l_ducksay_no_body_bool
    ,no-bubble     .bool_set:N = \l_ducksay_no_bubble_bool
    ,body-mirrored .bool_set:N = \l_ducksay_mirrored_body_bool
    ,ignore-body   .bool_set:N = \l_ducksay_ignored_body_bool
    ,body-x      .dim_set:N = \l_ducksay_body_x_offset_dim
    ,body-x      .value_required:n = true
    ,body-y      .dim_set:N = \l_ducksay_body_y_offset_dim
    ,body-y      .value_required:n = true
    ,body-to-msg .tl_set:N  = \l_ducksay_body_to_msg_align_body_tl
    ,msg-to-body .tl_set:N  = \l_ducksay_body_to_msg_align_msg_tl
    ,body-align .choice:
    ,body-align / l .meta:n = { body-to-msg = l , msg-to-body = l }
    ,body-align / c .meta:n = { body-to-msg = hc , msg-to-body = hc }
    ,body-align / r .meta:n = { body-to-msg = r , msg-to-body = r }
    ,body-align  .initial:n = l
    ,body-bigger .int_set:N = \l_ducksay_body_bigger_int
    ,body-bigger .initial:n = \c_zero_int
    ,msg-align   .choice:
    ,msg-align  / l .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { l } }
    ,msg-align  / c .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { c } }
    ,msg-align  / r .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { r } }
    ,msg-align  / j .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { j } }
    ,msg-align-l .tl_set:N  = \l_ducksay_msg_align_l_tl
    ,msg-align-l .initial:n = \raggedright
    ,msg-align-c .tl_set:N  = \l_ducksay_msg_align_c_tl
    ,msg-align-c .initial:n = \centering
    ,msg-align-r .tl_set:N  = \l_ducksay_msg_align_r_tl
    ,msg-align-r .initial:n = \raggedleft
    ,msg-align-j .tl_set:N  = \l_ducksay_msg_align_j_tl
    ,msg-align-j .initial:n = {}
    ,out-h   .tl_set:N  = \l_ducksay_output_h_pole_tl
    ,out-h   .initial:n = l
    ,out-v   .tl_set:N  = \l_ducksay_output_v_pole_tl
    ,out-v   .initial:n = vc
    ,out-x   .dim_set:N = \l_ducksay_output_x_offset_dim
    ,out-x   .value_required:n = true
    ,out-y   .dim_set:N = \l_ducksay_output_y_offset_dim
    ,out-y   .value_required:n = true
    ,t       .meta:n    = { out-v = t }
    ,c       .meta:n    = { out-v = vc }
    ,b       .meta:n    = { out-v = b }
    ,body*   .tl_set:N  = \l_ducksay_body_fount_tl
    ,msg*    .tl_set:N  = \l_ducksay_msg_fount_tl
    ,bubble* .tl_set:N  = \l_ducksay_bubble_fount_tl
    ,body*   .initial:n = \verbatim@font
    ,msg*    .initial:n = \verbatim@font
    ,bubble* .initial:n = \verbatim@font
    ,body    .code:n    = \tl_put_right:Nn \l_ducksay_body_fount_tl   { #1 }
    ,msg     .code:n    = \tl_put_right:Nn \l_ducksay_msg_fount_tl    { #1 }
    ,bubble  .code:n    = \tl_put_right:Nn \l_ducksay_bubble_fount_tl { #1 }
    ,MSG     .meta:n    = { msg  = #1 , bubble  = #1 }
    ,MSG*    .meta:n    = { msg* = #1 , bubble* = #1 }
    ,hpad    .int_set:N = \l_ducksay_hpad_int
    ,hpad    .initial:n = 2
    ,hpad    .value_required:n = true
    ,vpad    .int_set:N = \l_ducksay_vpad_int
    ,vpad    .value_required:n = true
    ,col     .tl_set:N  = \l_ducksay_msg_tabular_column_tl
    ,bubble-top-kern  .tl_set:N  = \l_ducksay_bubble_top_kern_tl
    ,bubble-top-kern  .initial:n = { -.5ex }
    ,bubble-top-kern  .value_required:n = true
    ,bubble-bot-kern  .tl_set:N  = \l_ducksay_bubble_bottom_kern_tl
    ,bubble-bot-kern  .initial:n = { .2ex }
    ,bubble-bot-kern  .value_required:n = true
    ,bubble-side-kern .tl_set:N  = \l_ducksay_bubble_side_kern_tl
    ,bubble-side-kern .initial:n = { .2em }
    ,bubble-side-kern .value_required:n = true
    ,bubble-delim-top     .tl_set:N  = \l_ducksay_bubble_delim_top_tl
    ,bubble-delim-left-1  .tl_set:N  = \l_ducksay_bubble_delim_left_a_tl
    ,bubble-delim-left-2  .tl_set:N  = \l_ducksay_bubble_delim_left_b_tl
    ,bubble-delim-left-3  .tl_set:N  = \l_ducksay_bubble_delim_left_c_tl
    ,bubble-delim-left-4  .tl_set:N  = \l_ducksay_bubble_delim_left_d_tl
    ,bubble-delim-right-1 .tl_set:N  = \l_ducksay_bubble_delim_right_a_tl
    ,bubble-delim-right-2 .tl_set:N  = \l_ducksay_bubble_delim_right_b_tl
    ,bubble-delim-right-3 .tl_set:N  = \l_ducksay_bubble_delim_right_c_tl
    ,bubble-delim-right-4 .tl_set:N  = \l_ducksay_bubble_delim_right_d_tl
    ,bubble-delim-top     .initial:n = { { - } }
    ,bubble-delim-left-1  .initial:n = (
    ,bubble-delim-left-2  .initial:n = /
    ,bubble-delim-left-3  .initial:n = |
    ,bubble-delim-left-4  .initial:n = \c_backslash_str
    ,bubble-delim-right-1 .initial:n = )
    ,bubble-delim-right-2 .initial:n = \c_backslash_str
    ,bubble-delim-right-3 .initial:n = |
    ,bubble-delim-right-4 .initial:n = /
    ,strip-spaces .bool_set:N  = \l_ducksay_msg_strip_spaces_bool
  }
```

Redefine keys only intended for version 1 to throw an error:

```
\clist_map_inline:nn
  { align, rel-align }
  {
    \keys_define:nn { ducksay }
      { #1 .code:n = \msg_error:nn { ducksay } { v1-key-only } }
  }
```

3.3.4 Functions

3.3.4.1 Internal

luate_message_alignment_fixed_width_common:

```
\cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_common:
  {
    \str_case:Vn \l_ducksay_msg_align_tl
      {
        { l } { \exp_not:N \l_ducksay_msg_align_l_tl }
        { c } { \exp_not:N \l_ducksay_msg_align_c_tl }
        { r } { \exp_not:N \l_ducksay_msg_align_r_tl }
        { j } { \exp_not:N \l_ducksay_msg_align_j_tl }
      }
  }
```

(*End of definition for* `\ducksay_evaluate_message_alignment_fixed_width_common:`.)

uate_message_alignment_fixed_width_tabular:

```
\cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_tabular:
  {
    \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
      {
        \tl_set:Nx \l_ducksay_msg_tabular_column_tl
          {
            >
            {
              \ducksay_evaluate_message_alignment_fixed_width_common:
              \exp_not:N \arraybackslash
            }
            p { \exp_not:N \l_ducksay_msg_width_dim }
          }
      }
  }
```

(*End of definition for* `\ducksay_evaluate_message_alignment_fixed_width_tabular:`.)

valuate_message_alignment_fixed_width_vbox:

```
\cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_vbox:
  {
    \tl_set:Nx \l_ducksay_msg_align_vbox_tl
      { \ducksay_evaluate_message_alignment_fixed_width_common: }
  }
```

(*End of definition for* `\ducksay_evaluate_message_alignment_fixed_width_vbox:`.)

\ducksay_calculate_msg_width_from_int:

```
\cs_new:Npn \ducksay_calculate_msg_width_from_int:
  {
    \hbox_set:Nn \l_ducksay_tmpa_box { { \l_ducksay_msg_fount_tl M } }
    \dim_set:Nn \l_ducksay_msg_width_dim
      { \l_ducksay_msg_width_int \box_wd:N \l_ducksay_tmpa_box }
  }
```

(*End of definition for* `\ducksay_calculate_msg_width_from_int:`.)

\ducksay_msg_tabular_begin:

```
\cs_new:Npn \ducksay_msg_tabular_begin:
  {
    \ducksay_msg_tabular_begin_inner:V \l_ducksay_msg_tabular_column_tl
  }
\cs_new:Npn \ducksay_msg_tabular_begin_inner:n #1
  {
    \begin { tabular } { @{} #1 @{} }
  }
\cs_generate_variant:Nn \ducksay_msg_tabular_begin_inner:n { V }
```

(*End of definition for* \ducksay_msg_tabular_begin:.)

\ducksay_msg_tabular_end:

```
\cs_new:Npn \ducksay_msg_tabular_end:
  {
    \end { tabular }
  }
```

(*End of definition for* \ducksay_msg_tabular_end:.)

\ducksay_width_case_none_int_dim:nnn

```
\cs_new:Npn \ducksay_width_case_none_int_dim:nnn #1 #2 #3
  {
    \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
      {
        \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
          { #1 }
          { #2 }
      }
      { #3 }
  }
```

(*End of definition for* \ducksay_width_case_none_int_dim:nnn.)

\ducksay_digest_options:n

```
\cs_new:Npn \ducksay_digest_options:n #1
  {
    \group_begin:
    \keys_set:nn { ducksay } { #1 }
    \ducksay_default_or_random_animal:
    \bool_if:NF \l_ducksay_no_body_bool
      {
        \hcoffin_set:Nn \l_ducksay_body_coffin
          {
            \frenchspacing
            \l_ducksay_body_fount_tl
            \begin{tabular} { @{} l @{} }
              \l_ducksay_animal_tl
              \ducksay_make_body_bigger:
              \relax
            \end{tabular}
          }
        \bool_if:NT \l_ducksay_msg_eq_body_width_bool
```

```
     {
      \bool_lazy_and:nnT
       { \int_compare_p:nNn \l_ducksay_msg_width_int < \c_zero_int }
       { \dim_compare_p:nNn \l_ducksay_msg_width_dim < \c_zero_dim }
       {
        \dim_set:Nn \l_ducksay_msg_width_dim
         { \coffin_wd:N \l_ducksay_body_coffin }
       }
     }
   }
  \bool_if:NTF \l_ducksay_eat_arg_box_bool
   {
    \ducksay_width_case_none_int_dim:nnn
     { \ducksay_eat_argument_hbox:w }
     {
      \ducksay_calculate_msg_width_from_int:
      \ducksay_eat_argument_vbox:w
     }
     { \ducksay_eat_argument_vbox:w }
   }
   {
    \ducksay_width_case_none_int_dim:nnn
     {
      \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
       {
        \str_case:Vn \l_ducksay_msg_align_tl
         {
          { l } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l } }
          { c } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { c } }
          { r } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { r } }
          { j }
           {
            \msg_error:nn { ducksay } { justify~unavailable }
            \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l }
           }
         }
       }
     }
     {
      \ducksay_calculate_msg_width_from_int:
      \ducksay_evaluate_message_alignment_fixed_width_tabular:
     }
     { \ducksay_evaluate_message_alignment_fixed_width_tabular: }
    \ducksay_eat_argument_tabular:w
   }
 }
```

(*End of definition for* `\ducksay_digest_options:n`*.*)

\ducksay_set_bubble_top_kern:

```
\cs_new:Npn \ducksay_set_bubble_top_kern:
 {
  \group_begin:
  \l_ducksay_bubble_fount_tl
```

```
    \exp_args:NNNx
    \group_end:
    \dim_set:Nn \l_ducksay_bubble_top_kern_dim
      { \dim_eval:n { \l_ducksay_bubble_top_kern_tl } }
  }
```

(*End of definition for* `\ducksay_set_bubble_top_kern:`.)

`\ducksay_set_bubble_bottom_kern:`

```
\cs_new:Npn \ducksay_set_bubble_bottom_kern:
  {
    \group_begin:
    \l_ducksay_bubble_fount_tl
    \exp_args:NNNx
    \group_end:
    \dim_set:Nn \l_ducksay_bubble_bottom_kern_dim
      { \dim_eval:n { \l_ducksay_bubble_bottom_kern_tl } }
  }
```

(*End of definition for* `\ducksay_set_bubble_bottom_kern:`.)

`\ducksay_make_body_bigger:`

```
\cs_new:Npn \ducksay_make_body_bigger:
  { \prg_replicate:nn \l_ducksay_body_bigger_int \\ }
```

(*End of definition for* `\ducksay_make_body_bigger:`.)

`\ducksay_baselineskip:` This is an overly cautious way to get the current baselineskip. Inside of `tabular` the baselineskip is 0pt, so we fall back to `\normalbaselineskip`, or issue an error and fall back to some arbitrary value not producing an error if that one is also 0pt.

```
\cs_new_protected_nopar:Npn \ducksay_baselineskip:
  {
    \the\dimexpr
      \ifdim \baselineskip = \c_zero_dim
        \ifdim \normalbaselineskip = \c_zero_dim
          \msg_expandable_error:nn { ducksay } { zero-baselineskip } { 12pt }
          12pt
        \else
          \normalbaselineskip
        \fi
      \else
        \baselineskip
      \fi
    \relax
  }
```

(*End of definition for* `\ducksay_baselineskip:`.)

`\ducksay_measure_msg:`

```
\cs_new_protected_nopar:Npn \ducksay_measure_msg:
  {
    \hbox_set:Nn \l_ducksay_tmpa_box
      { \l_ducksay_bubble_fount_tl \l_ducksay_bubble_delim_top_tl }
    \int_set:Nn \l_ducksay_msg_width_int
      {
        \fp_eval:n
          {
            ceil
              ( \box_wd:N \l_ducksay_msg_box / \box_wd:N \l_ducksay_tmpa_box )
          }
      }
    \group_begin:
    \l_ducksay_bubble_fount_tl
    \exp_args:NNNx
    \group_end:
    \int_set:Nn \l_ducksay_msg_height_int
      {
        \int_max:nn
          {
            \fp_eval:n
              {
                ceil
                  (
                    (
                      \box_ht:N \l_ducksay_msg_box
                      + \box_dp:N \l_ducksay_msg_box
                    )
                    / ( \arraystretch * \ducksay_baselineskip: )
                  )
              }
            + \l_ducksay_vpad_int
          }
          { \l_ducksay_msg_height_int }
      }
  }
```

(*End of definition for* `\ducksay_measure_msg:`.)

`\ducksay_set_bubble_coffins:`

```
\cs_new_protected_nopar:Npn \ducksay_set_bubble_coffins:
  {
    \hcoffin_set:Nn \l_ducksay_bubble_open_coffin
      {
        \l_ducksay_bubble_fount_tl
        \begin{tabular}{@{}l@{}}
          \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
            {
              \l_ducksay_bubble_delim_left_a_tl
            }
            {
              \l_ducksay_bubble_delim_left_b_tl\\
              \int_step_inline:nnn
                { 3 } { \l_ducksay_msg_height_int }
                {
                  \kern-\l_ducksay_bubble_side_kern_tl
                  \l_ducksay_bubble_delim_left_c_tl
                  \\
                }
              \l_ducksay_bubble_delim_left_d_tl
            }
        \end{tabular}
      }
    \hcoffin_set:Nn \l_ducksay_bubble_close_coffin
      {
        \l_ducksay_bubble_fount_tl
        \begin{tabular}{@{}r@{}}
          \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
            {
              \l_ducksay_bubble_delim_right_a_tl
            }
            {
              \l_ducksay_bubble_delim_right_b_tl \\
              \int_step_inline:nnn
                { 3 } { \l_ducksay_msg_height_int }
                {
                  \l_ducksay_bubble_delim_right_c_tl
                  \kern-\l_ducksay_bubble_side_kern_tl
                  \\
                }
              \l_ducksay_bubble_delim_right_d_tl
            }
        \end{tabular}
      }
    \hcoffin_set:Nn \l_ducksay_bubble_top_coffin
      {
        \l_ducksay_bubble_fount_tl
        \int_step_inline:nn
          { \l_ducksay_msg_width_int + \l_ducksay_hpad_int }
          { \l_ducksay_bubble_delim_top_tl }
      }
  }
```

(*End of definition for* `\ducksay_set_bubble_coffins:`.)

`\ducksay_join_bubble_to_msg_coffin:`

```
\cs_new_protected_nopar:Npn \ducksay_join_bubble_to_msg_coffin:
  {
    \dim_set:Nn \l_ducksay_hpad_dim
      {
        (
          \coffin_wd:N \l_ducksay_bubble_top_coffin
          - \coffin_wd:N \l_ducksay_msg_coffin
        ) / 2
      }
    \coffin_join:NnnNnnnn
      \l_ducksay_msg_coffin         { l } { vc }
      \l_ducksay_bubble_open_coffin { r } { vc }
      { - \l_ducksay_hpad_dim } { \c_zero_dim }
    \coffin_join:NnnNnnnn
      \l_ducksay_msg_coffin          { r } { vc }
      \l_ducksay_bubble_close_coffin { l } { vc }
      { \l_ducksay_hpad_dim } { \c_zero_dim }
    \coffin_join:NnnNnnnn
```

```
      \l_ducksay_msg_coffin        { hc } { t }
      \l_ducksay_bubble_top_coffin { hc } { b }
      { \c_zero_dim } { \l_ducksay_bubble_top_kern_dim }
    \coffin_join:NnnNnnnn
      \l_ducksay_msg_coffin        { hc } { b }
      \l_ducksay_bubble_top_coffin { hc } { t }
      { \c_zero_dim } { \l_ducksay_bubble_bottom_kern_dim }
  }
```

(*End of definition for* `\ducksay_join_bubble_to_msg_coffin:`.)

`\ducksay_shipout:`

```
\cs_new_protected:Npn \ducksay_shipout:
  {
    \hcoffin_set:Nn \l_ducksay_msg_coffin { \box_use:N \l_ducksay_msg_box }
    \bool_if:NF \l_ducksay_no_bubble_bool
      {
        \ducksay_measure_msg:
        \ducksay_set_bubble_coffins:
        \ducksay_set_bubble_top_kern:
        \ducksay_set_bubble_bottom_kern:
        \ducksay_join_bubble_to_msg_coffin:
      }
    \bool_if:NF \l_ducksay_no_body_bool
      {
        \bool_if:NT \l_ducksay_mirrored_body_bool
          {
            \coffin_scale:Nnn \l_ducksay_body_coffin
              { -\c_one_int } { \c_one_int }
            \str_case:Vn \l_ducksay_body_to_msg_align_body_tl
              {
                { l } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { r } }
                { r } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { l } }
              }
          }
        \bool_if:NTF \l_ducksay_ignored_body_bool
          { \coffin_attach:NVnNVnnn }
          { \coffin_join:NVnNVnnn   }
          \l_ducksay_msg_coffin  \l_ducksay_body_to_msg_align_msg_tl  { b }
          \l_ducksay_body_coffin \l_ducksay_body_to_msg_align_body_tl { t }
          { \l_ducksay_body_x_offset_dim } { \l_ducksay_body_y_offset_dim }
      }
    \coffin_typeset:NVVnn \l_ducksay_msg_coffin
      \l_ducksay_output_h_pole_tl \l_ducksay_output_v_pole_tl
      { \l_ducksay_output_x_offset_dim } { \l_ducksay_output_y_offset_dim }
    \group_end:
  }
```

(*End of definition for* `\ducksay_shipout:`.)

3.3.4.1.1 Message Reading Functions Version 2 has different ways of reading the message argument of `\ducksay` and `\duckthink`. They all should allow almost arbitrary content and the height and width are set based on the dimensions.

\ducksay_eat_argument_tabular:w

```
\cs_new:Npn \ducksay_eat_argument_tabular:w
  {
    \bool_if:NTF \l_ducksay_eat_arg_tab_verb_bool
      { \ducksay_eat_argument_tabular_verb:w }
      { \ducksay_eat_argument_tabular_normal:w }
  }
```

(*End of definition for* `\ducksay_eat_argument_tabular:w`.)

\ducksay_eat_argument_tabular_inner:w

```
\cs_new:Npn \ducksay_eat_argument_tabular_inner:w #1
  {
    \hbox_set:Nn \l_ducksay_msg_box
      {
        \l_ducksay_msg_fount_tl
        \ducksay_msg_tabular_begin:
          #1
        \ducksay_msg_tabular_end:
      }
    \ducksay_shipout:
  }
```

(*End of definition for* `\ducksay_eat_argument_tabular_inner:w`.)

\ducksay_eat_argument_tabular_verb:w

```
\NewDocumentCommand \ducksay_eat_argument_tabular_verb:w
  { >{ \ducksay_process_verb_newline:nnn { ~ } { ~ \par } } +v }
  {
    \ducksay_eat_argument_tabular_inner:w
      {
        \group_begin:
          \__ducksay_everyeof:w { \exp_not:N }
          \exp_after:wN
        \group_end:
        \__ducksay_scantokens:w { #1 }
      }
  }
```

(*End of definition for* `\ducksay_eat_argument_tabular_verb:w`.)

\ducksay_eat_argument_tabular_normal:w

```
\NewDocumentCommand \ducksay_eat_argument_tabular_normal:w { +m }
  { \ducksay_eat_argument_tabular_inner:w { #1 } }
```

(*End of definition for* `\ducksay_eat_argument_tabular_normal:w`.)

\ducksay_eat_argument_hbox:w

```
\cs_new_protected_nopar:Npn \ducksay_eat_argument_hbox:w
  {
    \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
      { \@grabbox }
      { \@grabbox* }
      {} \l_ducksay_msg_box \l_ducksay_msg_fount_tl \hbox {} \ducksay_shipout:
  }
```

(*End of definition for* `\ducksay_eat_argument_hbox:w`.)

`\ducksay_eat_argument_vbox:w`

```
\cs_new_protected_nopar:Npn \ducksay_eat_argument_vbox:w
  {
    \ducksay_evaluate_message_alignment_fixed_width_vbox:
    \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
      { \@grabbox }
      { \@grabbox* }
      {
        \hsize \l_ducksay_msg_width_dim
        \linewidth \hsize
        \l_ducksay_msg_align_vbox_tl
        \@afterindentfalse
        \@afterheading
      }
      \l_ducksay_msg_box \l_ducksay_msg_fount_tl \vbox {} \ducksay_shipout:
  }
```

(*End of definition for* `\ducksay_eat_argument_vbox:w`.)

3.3.4.1.2 Generating Variants of External Functions

```
\cs_generate_variant:Nn \coffin_join:NnnNnnnn { NVnNVnnn }
\cs_generate_variant:Nn \coffin_attach:NnnNnnnn { NVnNVnnn }
\cs_generate_variant:Nn \coffin_typeset:Nnnnn { NVVnn }
\cs_generate_variant:Nn \str_case:nn { Vn }
```

3.3.4.2 Document level

`\ducksay`

```
\NewDocumentCommand \ducksay { O{} }
  {
    \ducksay_digest_options:n { #1 }
  }
```

(*End of definition for* `\ducksay`. *This function is documented on page 8.*)

`\duckthink`

```
\NewDocumentCommand \duckthink { O{} }
  {
    \ducksay_digest_options:n { think, #1 }
  }
```

(*End of definition for* `\duckthink`. *This function is documented on page 8.*)

⟨/code.v2⟩

3.4 Definition of the Animals

```
⟨*animals⟩
\ProvidesFile{ducksay.animals.tex}
  [\ducksay@date\space v\ducksay@version\space ducksay animals]
%^^A some of the below are from http://ascii.co.uk/art/
\AddAnimal{duck}%>>=
{  \
    \   __
      >(' )
        )/
       /(
      /  `----/
      \  ~=- /
    ~^~^~^~^~^~^~^}%=<<
\AddAnimal{small-duck}%>>=
{  \
    \
      >()_
       (__)__ _}%=<<
\AddAnimal{duck-family}%>>=
{  \
    \   __
      >(' )
        )/
       /(
      /  `----/  -()_  >()_
    __\__~=-_/__ _(__)__(__)__ _}%=<<
\AddAnimal{cow}%>>=
{  \  ^__^
    \ (oo)\_______
      (__)\       )\/\
          ||----w |
          ||      ||}%=<<
\AddAnimal{head-in}%>>=
{  \
    \ ^__^         /
      (oo)\_______/  ________
      (__)\       )=(  ___|_ \____
          ||----w |  \ \    \____ |
          ||      ||   ||         ||}%=<<
\AddAnimal{sodomized}%>>=
{  \             _
    \           (_)
      ^__^       / \
      (oo)\_____/_\ \
      (__)\       ) /
          ||----w ((
          ||      ||>>}%=<<
\AddAnimal{tux}%>>=
{  \
    \  .--.
      |o_o |
      |\_/ |
     //   \ \
    (|     | )
   /'\_   _/`\
   \___)=(___/}%=<<
\AddAnimal{pig}%>>=
+  \     _//| .-~~~~-.
    \ _/oo  }        }-@
     ('')_  }        |
      `--'| { }--{  }
           //_/  /_/+%=<<
\AddAnimal{frog}%>>=
{   \
     \ (.)_(.)
    _ (   _   ) _
   / \/`-----'\/ \
 __\ ( (     ) ) /__
 )   /\ \._./ /\   (
  )_/ /|\   /|\ \_(}%=<<
\AddAnimal{snowman}%>>=
{  \
    \_[_]_
      (")
   >-( : )-<
    (__:__)}%=<<
\AddAnimal[tail-symbol=s]{hedgehog}%>>=
{  s    .\|//||\||.
    s  |/\/||/|//|/|
      /. `|/\\|/||/||
     o__,_|//|/||\||'}%=<<
\AddAnimal{kangaroo}%>>=
{  \
    \ _,'    ___
     <__\__/   \
        \_  /  _\
          \,\ / \\
            //   \\
          ,/'     `\_,}%=<<
%^^A http://chris.com/ascii/index.php?art=animals/rabbits
\AddAnimal[tail-symbol=s,tail-count=3]{rabbit}%>>=
{  s
    s   / \`\         __
     s |  \ `\      /`/ \
       \_/`\  \-"-/` /\  \
            |       |  \  |
            (d     b)   \_/
            /       \
        ,".|.'.\_/.'.|.",
       /   /\' _|_ '/\   \
       |  /  '-`"`-'  \  |
       | |             | |
       | \    \   /    / |
        \ \    \ /    / /
         `"`\   :   /'"`
             `""`""`}%=<<
```

```
\AddAnimal{bunny}%>>=
{  \
    \     /
      /\ /
       ( )
     .( o ).}%=<<
\AddAnimal{small-rabbit}%>>=
{  \
    \ _//
     (')---.
      _/-_( )o}%=<<
\AddAnimal[tail-symbol=s,tail-count=3]{dragon}%>>=
{     s                   / \  //\
       s    |\___/|      /   \//  \\
        s   /0  0  \__  /    //  | \ \
           /     /  \/_/    //   |  \  \
           @_^_@'/   \/_   //    |   \   \
           //_^_/     \/_ //     |    \    \
        ( //) |        \///      |     \     \
      ( / /) _|_ /   )  //       |      \     _\
    ( // /) '/,_ _ _/  ( ; -.    |    _ _\.-~        .-~~~^-.
  (( / / )) ,-{        _      `-.|.-~-.           .~         `.
 (( // / ))  '/\      /                 ~-. _ .-~      .-~^-.  \
 (( /// ))      `.   {            }                   /      \  \
  (( / ))     .----~-.\        \-'                 .~         \  `. \^-.
             ///.----..>        \             _ -~             `.  ^-`  ^-_
               ///-._ _ _ _ _ _ _}^ - - - - ~                     ~-- ,.-~
                                                                  /.-~}%=<<
%^^A http://www.ascii-art.de/ascii/def/dogs.txt
\AddAnimal{dog}%>>=
{  \     __
    \ .-'\/\
       "\   '------.
     ___/       (  .'_____
    '-----'"""'------"""""'}%=<<
%^^A http://ascii.co.uk/art/squirrel
\AddAnimal{squirrel}%>>=
{  \           ,;:;;,
    \    ,    ;;;;;
      .=',    ;:;;:,
     /_', "=. ';:;:;
     @=:__,  \,;:;:'
       _(\.=  ;:;;'
      `"_(  _/="`
       `"'``}%=<<
\AddAnimal{snail}%>>=
{  \
    \          .-""-.
      oo      ; .-.  :
       \\__..-: '.__.')._
        "-._.._'.__.-'_.."}%=<<
%^^A http://www.ascii-art.de/ascii/uvw/unicorn.txt
\AddAnimal{unicorn}%>>=
{  \
```

```
    \     /((((((\\\\
  ---====((((((((((\\\\\
       ((           \\\\\\\
       ( (*    _/      \\\\\\\
         \    /  \      \\\\\\_       __,,__
          |  |   |      </    "-----""     ((\\\\
          o_|   /        /                    \ \\\\    \\\\\\\
               |  ._    (                      \ \\\\\\\\\\\\\\\\
               | /                      /      /    \\\\\\\     \\
       .______/\/     /                /      /         \\\
      / __.____/    _/              _(       /\
     / / / ________/:_       ___,,--'  \    /  \_
    / /  \ \          """""""           \  \ \_  \
   ( <    \ \                           >  /    \ \
    \/      \\_                        / /       > )
             \_|                      / /       / /
                                    _//       _//
                                   /_|       /_|}%=<<
%^^A https://asciiart.website//index.php?art=animals/other%20(water)
\AddAnimal[tail-count=3,tail-symbol=s]{whale}%>>=
{  s                |-.
    s    .-""-._     \ \.--|
     s  /       `-..__)  ,-'
       |     .          /
        \--.__,   .__.,'
         `-.___'._\_.'}%=<<
%^^A from http://www.ascii-art.de/ascii/s/starwars.txt :
\AddAnimal[tail-count=3]{yoda}%>>=
{   \
     \             ____
      \         _.' :  `._
            .-.'`.  ;   .'`.-.
   __      / : ___\ ;  /___ ; \      __
 ,'_ ""--.:__;".-.";: :".-.":__;.--"" _`,
 :' `.t""--.. '<@.`;_  ',@>` ..--""j.' `;
      `:-.._J '-.-'L__ `-- ' L_..-;'
        "-.__ ;  .-"  "-.  : __.-"
            L ' /.------.\ ' J
             "-.   "--"   .-"
            __.l"-:_JL_;-";.__
         .-j/'.;  ;""""  / .'\"-.
       .' /:`. :  :     /.".'';  `.
    .-"  / ;`.".  :    ."."   :    "-.
 .+"-.  : :   ".".". ."."      ;-._   \
 ; \  `.; ; .   "."-"."        : : "+. ;
 :  ;   ; ;  .   ."."    ;      : ; : \:
 ;  :   ; :    / /     /  ,    ;:   ;  :
: \  ;  :  ;   ; /     :  ,   : ;  /  ::
;  ; :   ; :  ; ;      ;      ;   :   ;:
:  :  ;  :  ;. ;      '      : :  ;  : ;
;\    :   ; : .          ,   ; ;     ; ;
: `."-;   :  ;      .   ;   :  ;    /  ;
 ;    -:    ; :      ,  ,    ;  : .-"  :
 :\      \  :  ;    ,       : \.-"      :
  ;‘.    \  ; :   .   ,    ;.’_..--  / ;
  :  "-.  "-:  ;     ,    :/."      .’  :
   \         \ :    :     ;/  __        :
    \       .-‘.\        /t-""  ":-+.   :
     ‘.  .-"    ‘l    __/ /‘. :  ; ; \  ;
       \   .-" .-"-.-"  .’ .’j \  /   ;/
        \ / .-"   /.     .’.’ ;_:’    ;
         :-""-.‘./-.’     /    ‘.___.’
               \ ‘t  ._  /
                "-.t-._:’}%=<<
\AddAnimal[tail-count=3]{yoda-head}%>>=
{   \
     \             ____
      \         _.’ :  ‘._
            .-.’‘.  ;   .’‘.-.
   __      / : ___\ ;  /___ ; \      __
 ,’_ ""--.:__;".-.";: :".-.":__;.--"" _‘,
 :’ ‘.t""--.. ’<@.‘;_  ’,@>‘ ..--""j.’ ‘;
      ‘:-.._J ’-.-’L__ ‘-- ’ L_..-;’
        "-.__ ;  .-"  "-.  : __.-"
            L ’ /.------.\ ’ J
             "-.   "--"   .-"
            __.l"-:_JL_;-";.__
         .-j/’.;  ;""""  / .’\"-.
       .’ /:‘. :  :     /.".’’;  ‘.
    .-"  / ;‘.".  :    ."."    :    "-.
 .+"-.  : :   ".".". ."."      ;-._   \}%=<<
%^^A from https://www.ascii-code.com/ascii-art/movies/star-wars.php
\AddAnimal{small-yoda}%>>=
{  \
    \
    __.-._
    ’-._"7’
     /’.-c
     |  /T
    _)_/LI}%=<<
\AddAnimal{r2d2}%>>=
{  \
    \ ,-----.
    ,’_/_|_\_‘.
   /<<::8[O]::>\
  _|-----------|_
 |  | ====-=- |  |
 |  | -=-==== |  |
 \  | ::::|()||  /
  | | ....|()|| |
  | |_________| |
  | |\_______/| |
 /   \ /   \ /   \
 ‘___’ ‘___’ ‘___’}%=<<
\AddAnimal{vader}%>>=
{  \     _.-’~~~~~~‘-._
    \   /      ||      \
       /       ||       \
```

```
      |        ||        |
      | _______||_______ |
      |/ ----- \/ ----- \|
     /  (     )  (     )  \
    / \  ----- () -----  / \
   /   \      /||\      /   \
  /     \    /||||\    /     \
 /       \  /||||||\  /       \
/_        \O========O/        _\
  `--...__|`-._  _.-'|__...--'
          |    `'    |}%=<<
\AddAnimal[tail-symbol=|,tail-count=1]{crusader}%>>=
{ |
\[T]/}
\csname bool_if:cT\endcsname {l_ducksay_version_one_bool}
  {\AnimalOptions{crusader}{tail-1=|,rel-align=c}}
\csname bool_if:cT\endcsname {l_ducksay_version_two_bool}
  {\AnimalOptions{crusader}{tail-1=|,body-align=c}}%=<<
%^^A http://ascii.co.uk/art/knights
\AddAnimal[tail-count=3]{knight}%>>=
{     \
       \   ,-"""-.
        \  | === |
           )  |  (
        .=='\" "/`==.
      .'\   (`:')   /`.
    _/_ |_.-' : `-._|__\_
   <___>'\    :   / `<___>
   /  /   >=======<  /  /
 _/ .'   /  ,-:-.  \/=,'
/ _/    |__/v^v^v\__) \
\(\)     |V^V^V^V^V|\_/
 (\\     \`---|---'/
   \\     \-._|_,-/
    \\     |__|__|
     \\   <___X___>
      \\   \..|../
       \\   \ | /
        \\  /V|V\
         \|/  |  \
          '--' `--`}%=<<
%^^A https://www.asciiart.eu/mythology/ghosts
\AddAnimal{ghost}%>>=
{  \
    \ .-.
     (o o)
     | O \
      \   \
       `~~~'}%=<<
%^^Ahttps://asciiart.website/index.php?art=creatures/fairies
\AddAnimal{fairy}%>>=
{  \
    \          .oOOb
      ..    .oO    O
```

```
     '::;  d       O
      ;;;;d    ..oO
 *   ::O;;;'OooO
~"\. dp'(O.o.
   \op    'oOb
           obU
          dop
         dop
          PO
         O 'b
         l  P.
         /   ;
         '}%=<<
\AddAnimal[tail-symbol=s]{only-tail}%>>=
{  s
    s}%=<<
\AddAnimal[tail-symbol=s,tail-count=3]{only-tail3}%>>=
{  s
    s
     s}%=<<
%^^A head taken from https://www.asciiart.eu/animals/reptiles/snakes
\AddAnimal[tail-symbol=s,tail-count=3]{snake}% >>=
{  s
    s     /^\/^\
     s  _|__| O |
      /'     \_/ \
  \/ |_________/  \               \
   \_/ \_______    \               \
               `|   |              |\
               /   /  _---_        | |
              /   /  /  __ "-_    ," |
             |   "--"  /  "-_ "--"  ,"
              "-_____-"      "-___-"}% =<<
%^^A http://www.ascii-art.de/ascii/c/cat.txt
\AddAnimal{cat}% >>=
+  \
    \  _                ___       .--.
       \`.|\..----...-'`   `-._.-' .-'
       /  ' `         ,       __.-'
       )/' _/     \   `-_,   /
       `-'" `"\_  ,_.-;_.-\_ ',
           _.-'_./   {_.'   ; /
          {_.-``-'         {_/+% =<<
%^^A https://www.asciiart.eu/animals/cats
\AddAnimal{sleepy-cat}% >>=
{  \
    \  |\      _,,,---,,_     _._
       /,`.-'`'    -.   )'._,'.-,)
      |,4-  ) )-,_. ,\ ( `-.-'
     '---''(_/--'  `-'\_)}% =<<
\AddAnimal{schroedinger-dead}% >>=
{  \
    \ _.--"""--._
     |           |
     |    -|-    |
     |     |     |
     |           |
     |   Felix   |
   __|___________|__ _
    o    .   .    .
       ~ .  o   o
       .   ~  .}% =<<
%^^A https://www.asciiart.eu/animals/cats
\AddAnimal{schroedinger-alive}% >>=
{  \
    \ ,_     _
      |\\__,'/
      / _  _ |    ,--.
     (  @  @ )   / ,-'
      \  _T_/-._( (
      /         `. \
     |         _  \ |
      \ \ ,  /      |
       || |-_\__   /
      ((_/`(____,-'}% =<<
%^^A provided by Plergux
%^^A (https://chat.stackexchange.com/transcript/message/55986902#55986902)
\AddAnimal{sheep}% >>=
{  \        _,_,_,_,_,
    \    .:(  ,)      ),
        (__,    (,     ),
       / o (  ,)     ,) )>
      (___(,      (,   ,)
           (,   ,)    ,)
            `-_,---_,-'
              ||   ||}% =<<
%^^A based on joe schmuck (http://www.ascii-art.de/ascii/pqr/platypus.txt)
\AddAnimal[tail-symbol=s]{platypus}% >>=
|  s          _.-^~~^^`~-,,,~~''```~-''``~'``~,
    s ______,'  -o  :. _    .      ;     ,'`,  `.
     (      -\.._,.;;'._,(   }    _`_-_,,    `, `,
      ``~~~~~~'   ((/'(((____/~~'(,(,___>      `~'|% =<<
\AddAnimal[tail-symbol=s]{small-horse}% >>=
{  s   _,_
    s /._ \\
     /_/ |_\\ _ __
         /  \\    \\
         \ _ __ _ /||
         | |   | | ||
         | |   | |}% =<<
\AddAnimal{ladybug}% >>=
{  \  o   o
    \ _\_/_
     /__!__\
   '| o : o |'
   -|o  :  o|-
   ,| o : o |,
     \__:__/}% =<<
```

```
%^^A based on art by Joan Stark (jgs)
%^^A (https://www.asciiart.eu/animals/reptiles/turtles)
\AddAnimal[tail-symbol=s]{turtle}
{  s
    s  __
      /_'\ .,-;-;-,.
       \ (/_/_|_\_\_\_
       /\=<_><_><_><_>-'
      /_/'-\_\====\_\'
      ""     ""      ""}
⟨/animals⟩
```

```
 ----------------------------
/                            \
  Who's gonna use it anyway?
\                            /
 ----------------------------
   O
    o   __
      >(' )
        )/
       /(
      /  `----/
      \  ~=- /
    ~^~^~^~^~^~^~^
```

```
 -----------------------------------------
/                                         \
|                Hosted at                |
| https://github.com/Skillmon/ltx_ducksay |
|                 it is.                  |
\                                         /
 -----------------------------------------
                                   \
                                    \
                                    __.-._
                                    '-._"7'
                                     /'.-c
                                     |  /T
                                    _)_/LI
```