Preamble

Pandoc (Haskell)

This test suite requires pandoc 1.16:

```
>>> from subprocess import Popen, PIPE
>>> p = Popen(["pandoc", "-v"], stdout=PIPE)
>>> if b"pandoc 1.16" not in p.communicate()[0]:
...     raise RuntimeError("pandoc 1.16 not found")
```

Imports

```
>>> from pandoc.types import *
>>> import pandoc
```

Pandoc Test Suite

Source: Pandoc's User Guide

Paragraphs

```
>>> """
... a paragraph
...
... another paragraph
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'a'), Space(), Str(u'paragraph')]), Para([St
r(u'another'), Space(), Str(u'paragraph')])])

>>> "a paragraph  \nanother paragraph" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'a'), Space(), Str(u'paragraph'), LineBreak(
), Str(u'another'), Space(), Str(u'paragraph')])])
```

Extension: `escaped_line_breaks`

```
>>> r"""
... a paragraph\
... another paragraph
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'a'), Space(), Str(u'paragraph'), LineBreak(
), Str(u'another'), Space(), Str(u'paragraph')])])
```

Headers

Setext-style headers

```
>>> """
```

```
... A level-one header
... ==================
...
... A level-two header
... ------------------
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Header(1, (u'a-level-one-header', [], []), [Str(u'A'),
Space(), Str(u'level-one'), Space(), Str(u'header')]), Header(2, (u'a-level-
two-header', [], []), [Str(u'A'), Space(), Str(u'level-two'), Space(), Str(u
'header')])])
```

ATX-style headers

```
>>> """
... ## A level-two header
...
... ### A level-three header ###
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Header(2, (u'a-level-two-header', [], []), [Str(u'A'),
Space(), Str(u'level-two'), Space(), Str(u'header')]), Header(3, (u'a-level-
three-header', [], []), [Str(u'A'), Space(), Str(u'level-three'), Space(), S
tr(u'header')])])

>>> "# A level-one header with a [link](/url) and *emphasis*"
... # doctest: +PANDOC
Pandoc(Meta(map()), [Header(1, (u'a-level-one-header-with-a-link-and-emphasi
s', [], []), [Str(u'A'), Space(), Str(u'level-one'), Space(), Str(u'header')
, Space(), Str(u'with'), Space(), Str(u'a'), Space(), Link((u'', [], []), [S
tr(u'link')], (u'/url', u'')), Space(), Str(u'and'), Space(), Emph([Str(u'em
phasis')])])])
```

Extension: `blank_before_header`

```
>>> """
... I like several of their flavors of ice cream:
... #22, for example, and #5.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'I'), Space(), Str(u'like'), Space(), Str(u'
several'), Space(), Str(u'of'), Space(), Str(u'their'), Space(), Str(u'flavo
rs'), Space(), Str(u'of'), Space(), Str(u'ice'), Space(), Str(u'cream:'), So
ftBreak(), Str(u'#22,'), Space(), Str(u'for'), Space(), Str(u'example,'), Sp
ace(), Str(u'and'), Space(), Str(u'#5.')])])
```

Header identifiers

Extension: `header_attributes`

```
>>> """
... # My header {#foo}
...
... ## My header ##    {#foo}
...
... My other header   {#foo}
... ---------------
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Header(1, (u'foo', [], []), [Str(u'My'), Space(), Str(u
'header')]), Header(2, (u'foo', [], []), [Str(u'My'), Space(), Str(u'header'
)]), Header(2, (u'foo', [], []), [Str(u'My'), Space(), Str(u'other'), Space(
), Str(u'header')])])

>>> "# My header {-}" # doctest: +PANDOC
Pandoc(Meta(map()), [Header(1, (u'my-header', [u'unnumbered'], []), [Str(u'M
y'), Space(), Str(u'header')])])

>>> "# My header {.unnumbered}" # doctest: +PANDOC
Pandoc(Meta(map()), [Header(1, (u'my-header', [u'unnumbered'], []), [Str(u'M
y'), Space(), Str(u'header')])])
```

Extension: `auto_identifiers`

This extension does not work for the JSON output format.

Extension: `implicit_header_references`

```
>>> """
... # Header Identifiers
...
```

```
... [header identifiers](#header-identifiers),
... [header identifiers],
... [header identifiers][],
... [the section on header identifiers][header identifiers]
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Header(1, (u'header-identifiers', [], []), [Str(u'Heade
r'), Space(), Str(u'Identifiers')]), Para([Link((u'', [], []), [Str(u'header
'), Space(), Str(u'identifiers')], (u'#header-identifiers', u'')), Str(u',')
, SoftBreak(), Link((u'', [], []), [Str(u'header'), Space(), Str(u'identifie
rs')], (u'#header-identifiers', u'')), Str(u','), SoftBreak(), Link((u'', []
, []), [Str(u'header'), Space(), Str(u'identifiers')], (u'#header-identifier
s', u'')), Str(u','), SoftBreak(), Link((u'', [], []), [Str(u'the'), Space()
, Str(u'section'), Space(), Str(u'on'), Space(), Str(u'header'), Space(), St
r(u'identifiers')], (u'#header-identifiers', u''))])])
```

```
>>> """
... # Foo
...
... [foo]: bar
...
... See [foo]
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Header(1, (u'foo', [], []), [Str(u'Foo')]), Para([Str(u
'See'), Space(), Link((u'', [], []), [Str(u'foo')], (u'bar', u''))])])
```

Block quotations

```
>>> """
... > This is a block quote. This
... > paragraph has two lines.
... >
... > 1. This is a list inside a block quote.
... > 2. Second item.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BlockQuote([Para([Str(u'This'), Space(), Str(u'is'), Sp
ace(), Str(u'a'), Space(), Str(u'block'), Space(), Str(u'quote.'), Space(),
Str(u'This'), SoftBreak(), Str(u'paragraph'), Space(), Str(u'has'), Space(),
 Str(u'two'), Space(), Str(u'lines.')]), OrderedList((1, Decimal(), Period()
), [[Plain([Str(u'This'), Space(), Str(u'is'), Space(), Str(u'a'), Space(),
Str(u'list'), Space(), Str(u'inside'), Space(), Str(u'a'), Space(), Str(u'bl
ock'), Space(), Str(u'quote.')])], [Plain([Str(u'Second'), Space(), Str(u'it
em.')])]])])])

>>> """
... > This is a block quote. This
... paragraph has two lines.
...
... > 1. This is a list inside a block quote.
... 2. Second item.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BlockQuote([Para([Str(u'This'), Space(), Str(u'is'), Sp
ace(), Str(u'a'), Space(), Str(u'block'), Space(), Str(u'quote.'), Space(),
Str(u'This'), SoftBreak(), Str(u'paragraph'), Space(), Str(u'has'), Space(),
 Str(u'two'), Space(), Str(u'lines.')])]), BlockQuote([OrderedList((1, Decim
al(), Period()), [[Plain([Str(u'This'), Space(), Str(u'is'), Space(), Str(u'
a'), Space(), Str(u'list'), Space(), Str(u'inside'), Space(), Str(u'a'), Spa
ce(), Str(u'block'), Space(), Str(u'quote.')])], [Plain([Str(u'Second'), Spa
ce(), Str(u'item.')])]])])])

>>> """
... > This is a block quote.
... >
... > > A block quote within a block quote.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BlockQuote([Para([Str(u'This'), Space(), Str(u'is'), Sp
ace(), Str(u'a'), Space(), Str(u'block'), Space(), Str(u'quote.')]), BlockQu
ote([Para([Str(u'A'), Space(), Str(u'block'), Space(), Str(u'quote'), Space(
), Str(u'within'), Space(), Str(u'a'), Space(), Str(u'block'), Space(), Str(
u'quote.')])])])])

>>> ">     code" # doctest: +PANDOC
Pandoc(Meta(map()), [BlockQuote([CodeBlock((u'', [], []), u'code')])])
```

Extension: `blank_before_blockquote`

```
>>> """
... > This is a block quote.
... >> Nested.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BlockQuote([Para([Str(u'This'), Space(), Str(u'is'), Sp
ace(), Str(u'a'), Space(), Str(u'block'), Space(), Str(u'quote.'), SoftBreak
(), Str(u'>'), Space(), Str(u'Nested.')])])])
```

Verbatim (code) blocks

Indented code blocks

```
>>> """
...     if (a > 3) {
...     moveShip(5 * gravity, DOWN);
...     }
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [CodeBlock((u'', [], []), u'if (a > 3) {\nmoveShip(5 * g
ravity, DOWN);\n}')])
```

Fenced code blocks

Extension: `fenced_code_blocks`

```
>>> """
... ~~~~~~~
... if (a > 3) {
... moveShip(5 * gravity, DOWN);
... }
... ~~~~~~~
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [CodeBlock((u'', [], []), u'if (a > 3) {\nmoveShip(5 * g
ravity, DOWN);\n}')])
```

```
>>> """
... ~~~~~~~~~~~~~~~~
... ~~~~~~~~~~
... code including tildes
... ~~~~~~~~~~
... ~~~~~~~~~~~~~~~~
```

```
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [CodeBlock((u'', [], []), u'~~~~~~~~~~\ncode including t
ildes\n~~~~~~~~~~')])
```

Extension: `backtick_code_blocks`

```
>>> """
... ```````
... if (a > 3) {
... moveShip(5 * gravity, DOWN);
... }
... ```````
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [CodeBlock((u'', [], []), u'if (a > 3) {\nmoveShip(5 * g
ravity, DOWN);\n}')])
```

Extension: `fenced_code_attributes`

```
>>> """
... ~~~~ {#mycode .haskell .numberLines startFrom="100"}
... qsort []
... = []
... qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++
... qsort (filter (>= x) xs)
... ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [CodeBlock((u'mycode', [u'haskell', u'numberLines'], [(u
'startFrom', u'100')]), u'qsort []\n= []\nqsort (x:xs) = qsort (filter (< x)
 xs) ++ [x] ++\nqsort (filter (>= x) xs)')])

>>> """
... ```haskell
... qsort [] = []
... ```
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [CodeBlock((u'', [u'haskell'], []), u'qsort [] = []')])
```

```
>>> """
... ``` {.haskell}
... qsort [] = []
... ```
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [CodeBlock((u'', [u'haskell'], []), u'qsort [] = []')])
```

Line blocks

Extension: `line_blocks`

```
>>> """
... | The limerick packs laughs anatomical
... | In space that is quite economical.
... |    But the good ones I've seen
... |    So seldom are clean
... | And the clean ones so seldom are comical
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'The'), Space(), Str(u'limerick'), Space(),
Str(u'packs'), Space(), Str(u'laughs'), Space(), Str(u'anatomical'), LineBre
ak(), Str(u'In'), Space(), Str(u'space'), Space(), Str(u'that'), Space(), St
r(u'is'), Space(), Str(u'quite'), Space(), Str(u'economical.'), LineBreak(),
 Str(u'\xa0\xa0\xa0But'), Space(), Str(u'the'), Space(), Str(u'good'), Space
(), Str(u'ones'), Space(), Str(u"I've"), Space(), Str(u'seen'), LineBreak(),
 Str(u'\xa0\xa0\xa0So'), Space(), Str(u'seldom'), Space(), Str(u'are'), Spac
e(), Str(u'clean'), LineBreak(), Str(u'And'), Space(), Str(u'the'), Space(),
 Str(u'clean'), Space(), Str(u'ones'), Space(), Str(u'so'), Space(), Str(u's
eldom'), Space(), Str(u'are'), Space(), Str(u'comical')])])

>>> """
... | 200 Main St.
... | Berkeley, CA 94718
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'200'), Space(), Str(u'Main'), Space(), Str(
u'St.'), LineBreak(), Str(u'Berkeley,'), Space(), Str(u'CA'), Space(), Str(u
'94718')])])

>>> """
```

```
... | The Right Honorable Most Venerable and Righteous Samuel L.
...   Constable, Jr.
... | 200 Main St.
... | Berkeley, CA 94718
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'The'), Space(), Str(u'Right'), Space(), Str
(u'Honorable'), Space(), Str(u'Most'), Space(), Str(u'Venerable'), Space(),
Str(u'and'), Space(), Str(u'Righteous'), Space(), Str(u'Samuel'), Space(), S
tr(u'L.'), Space(), Str(u'Constable,'), Space(), Str(u'Jr.'), LineBreak(), S
tr(u'200'), Space(), Str(u'Main'), Space(), Str(u'St.'), LineBreak(), Str(u'
Berkeley,'), Space(), Str(u'CA'), Space(), Str(u'94718')])])
```

Lists

Bullet lists

```
>>> """
... * one
... * two
... * three
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'one')])], [Plain([Str(u'two')
])], [Plain([Str(u'three')])]])])

>>> """
... * one
...
... * two
...
... * three
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Para([Str(u'one')])], [Para([Str(u'two')])
], [Para([Str(u'three')])]])])


>>> """
... * here is my first
... list item.
... * and my second.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'here'), Space(), Str(u'is'),
Space(), Str(u'my'), Space(), Str(u'first'), SoftBreak(), Str(u'list'), Spac
e(), Str(u'item.')])], [Plain([Str(u'and'), Space(), Str(u'my'), Space(), St
r(u'second.')])]])])


>>> """
... * here is my first
... list item.
... * and my second.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'here'), Space(), Str(u'is'),
Space(), Str(u'my'), Space(), Str(u'first'), SoftBreak(), Str(u'list'), Spac
e(), Str(u'item.')])], [Plain([Str(u'and'), Space(), Str(u'my'), Space(), St
r(u'second.')])]])])

>>> """
...   * First paragraph.
...
...   Continued.
...
...   * Second paragraph. With a code block, which must be indented
...     eight spaces:
...
...         { code }
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'First'), Space(), Str(u'parag
raph.')])]]), Para([Str(u'Continued.')]), BulletList([[Para([Str(u'Second'),
 Space(), Str(u'paragraph.'), Space(), Str(u'With'), Space(), Str(u'a'), Spa
ce(), Str(u'code'), Space(), Str(u'block,'), Space(), Str(u'which'), Space()
, Str(u'must'), Space(), Str(u'be'), Space(), Str(u'indented'), SoftBreak(),
 Str(u'eight'), Space(), Str(u'spaces:')]), CodeBlock((u'', [], []), u'{ cod
e }')]])])

>>> """
... * fruits
...     + apples
...         - macintosh
...         - red delicious
...     + pears
...     + peaches
... * vegetables
...     + broccoli
...     + chard
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'fruits')]), BulletList([[Plai
n([Str(u'apples')]), BulletList([[Plain([Str(u'macintosh')])], [Plain([Str(u
'red'), Space(), Str(u'delicious')])]])], [Plain([Str(u'pears')])], [Plain([
Str(u'peaches')])]])], [Plain([Str(u'vegetables')]), BulletList([[Plain([Str
(u'broccoli')])], [Plain([Str(u'chard')])]])]])])

>>> """
... + A lazy, lazy, list
... item.
...
... + Another one; this looks
... bad but is legal.
...
...     Second paragraph of second
... list item.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Para([Str(u'A'), Space(), Str(u'lazy,'), S
pace(), Str(u'lazy,'), Space(), Str(u'list'), SoftBreak(), Str(u'item.')])],
 [Para([Str(u'Another'), Space(), Str(u'one;'), Space(), Str(u'this'), Space
(), Str(u'looks'), SoftBreak(), Str(u'bad'), Space(), Str(u'but'), Space(),
Str(u'is'), Space(), Str(u'legal.')]), Para([Str(u'Second'), Space(), Str(u'
paragraph'), Space(), Str(u'of'), Space(), Str(u'second'), SoftBreak(), Str(
u'list'), Space(), Str(u'item.')])]])])

>>> """
... 1.  one
... 2.  two
... 3.  three
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [OrderedList((1, Decimal(), Period()), [[Plain([Str(u'on
e')])], [Plain([Str(u'two')])], [Plain([Str(u'three')])]])])

>>> """
```

```
... 5.  one
... 7.  two
... 1.  three
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [OrderedList((5, Decimal(), Period()), [[Plain([Str(u'on
e')])], [Plain([Str(u'two')])], [Plain([Str(u'three')])]])])
```

Extension: `fancy_lists`

```
>>> """
... #. one
... #. two
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [OrderedList((1, DefaultStyle(), DefaultDelim()), [[Plai
n([Str(u'one')])], [Plain([Str(u'two')])]])])
```

Extension: `start_num`

```
>>> """
...  9)  Ninth
...  10)  Tenth
...  11)  Eleventh
...         i. subone
...        ii. subtwo
...       iii. subthree
...  """
... # doctest: +PANDOC
Pandoc(Meta(map()), [OrderedList((9, Decimal(), OneParen()), [[Plain([Str(u'
Ninth')])], [Plain([Str(u'Tenth')])], [Plain([Str(u'Eleventh'), SoftBreak(),
 Str(u'i.'), Space(), Str(u'subone')]), OrderedList((2, LowerRoman(), Period
()), [[Plain([Str(u'subtwo')])], [Plain([Str(u'subthree')])]])]])])

>>> """
... (2) Two
... (5) Three
... 1.  Four
... *   Five
... """
... # doctest: +PANDOC
```

```
Pandoc(Meta(map()), [OrderedList((2, Decimal(), TwoParens()), [[Plain([Str(u
'Two')])], [Plain([Str(u'Three')])]]), OrderedList((1, Decimal(), Period()),
 [[Plain([Str(u'Four')])]]), BulletList([[Plain([Str(u'Five')])]])])
```

Definition Lists

Extension: **`definition_lists`**

```
>>> """
... Term 1
...
... :   Definition 1
...
... Term 2 with *inline markup*
...
... :   Definition 2
...
...         { some code, part of Definition 2 }
...
...     Third paragraph of definition 2.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [DefinitionList([([Str(u'Term'), Space(), Str(u'1')], [[
Para([Str(u'Definition'), Space(), Str(u'1')])]]), ([Str(u'Term'), Space(),
Str(u'2'), Space(), Str(u'with'), Space(), Emph([Str(u'inline'), Space(), St
r(u'markup')])], [[Para([Str(u'Definition'), Space(), Str(u'2')]), CodeBlock
((u'', [], []), u'{ some code, part of Definition 2 }'), Para([Str(u'Third')
, Space(), Str(u'paragraph'), Space(), Str(u'of'), Space(), Str(u'definition
'), Space(), Str(u'2.')])]])])])
```

```
>>> """
... Term 1
...
... :   Definition
... with lazy continuation.
...
...     Second paragraph of the definition.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [DefinitionList([([Str(u'Term'), Space(), Str(u'1')], [[
Para([Str(u'Definition'), SoftBreak(), Str(u'with'), Space(), Str(u'lazy'),
Space(), Str(u'continuation.')]), Para([Str(u'Second'), Space(), Str(u'parag
raph'), Space(), Str(u'of'), Space(), Str(u'the'), Space(), Str(u'definition
.')])]])])])

>>> """
... Term 1
...   ~ Definition 1
...
... Term 2
...   ~ Definition 2a
...   ~ Definition 2b
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [DefinitionList([([Str(u'Term'), Space(), Str(u'1')], [[
Plain([Str(u'Definition'), Space(), Str(u'1')])]]), ([Str(u'Term'), Space(),
 Str(u'2')], [[Plain([Str(u'Definition'), Space(), Str(u'2a')])], [Plain([St
r(u'Definition'), Space(), Str(u'2b')])]])])])
```

Numbered Example List

Extension: `example_lists`

```
>>> """
... (@)  My first example will be numbered (1).
... (@)  My second example will be numbered (2).
...
... Explanation of examples.
...
... (@)  My third example will be numbered (3).
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [OrderedList((1, Example(), TwoParens()), [[Plain([Str(u
'My'), Space(), Str(u'first'), Space(), Str(u'example'), Space(), Str(u'will
'), Space(), Str(u'be'), Space(), Str(u'numbered'), Space(), Str(u'(1).')])]
, [Plain([Str(u'My'), Space(), Str(u'second'), Space(), Str(u'example'), Spa
ce(), Str(u'will'), Space(), Str(u'be'), Space(), Str(u'numbered'), Space(),
 Str(u'(2).')])]]), Para([Str(u'Explanation'), Space(), Str(u'of'), Space(),
 Str(u'examples.')]), OrderedList((3, Example(), TwoParens()), [[Plain([Str(
u'My'), Space(), Str(u'third'), Space(), Str(u'example'), Space(), Str(u'wil
l'), Space(), Str(u'be'), Space(), Str(u'numbered'), Space(), Str(u'(3).')])
]])])
```

```
>>> """
... (@good)  This is a good example.
...
... As (@good) illustrates, ...
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [OrderedList((1, Example(), TwoParens()), [[Plain([Str(u
'This'), Space(), Str(u'is'), Space(), Str(u'a'), Space(), Str(u'good'), Spa
ce(), Str(u'example.')])]]), Para([Str(u'As'), Space(), Str(u'(1)'), Space()
, Str(u'illustrates,'), Space(), Str(u'...')])])
```

Compact and loose lists

```
>>> """
... +   First
... +   Second:
...     -   Fee
...     -   Fie
...     -   Foe
...
... +   Third
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'First')])], [Plain([Str(u'Sec
ond:')]), BulletList([[Plain([Str(u'Fee')])], [Plain([Str(u'Fie')])], [Plain
([Str(u'Foe')])]])], [Plain([Str(u'Third')])]])])
```

Ending a List

```
>>> """
... -   item one
... -   item two
...
...     { my code block }
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'item'), Space(), Str(u'one')]
)], [Para([Str(u'item'), Space(), Str(u'two')]), Para([Str(u'{'), Space(), S
tr(u'my'), Space(), Str(u'code'), Space(), Str(u'block'), Space(), Str(u'}')
])]])])
```

```
>>> """
... -   item one
... -   item two
...
... <!-- end of list -->
...
...     { my code block }
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BulletList([[Plain([Str(u'item'), Space(), Str(u'one')]
)], [Plain([Str(u'item'), Space(), Str(u'two')])]]), RawBlock(Format(u'html'
), u'<!-- end of list -->'), CodeBlock((u'', [], []), u'{ my code block }')]
)
```

```
>>> """
... 1.  one
... 2.  two
... 3.  three
...
... <!-- -->
...
... 1.  uno
... 2.  dos
... 3.  tres
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [OrderedList((1, Decimal(), Period()), [[Plain([Str(u'on
e')])], [Plain([Str(u'two')])], [Plain([Str(u'three')])]]), RawBlock(Format(
u'html'), u'<!-- -->'), OrderedList((1, Decimal(), Period()), [[Plain([Str(u
'uno')])], [Plain([Str(u'dos')])], [Plain([Str(u'tres')])]])])
```

Horizontal Rules

```
>>> """
... *  *  *  *
...
... ---------------
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [HorizontalRule(), HorizontalRule()])
```

Tables

Extension `table_captions`

Extension `simple_tables`

```
>>> """
...   Right     Left     Center     Default
... -------     ------ ----------   -------
...      12     12        12            12
...     123     123       123          123
...       1     1          1             1
...
... Table:  Demonstration of simple table syntax.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Table([Str(u'Demonstration'), Space(), Str(u'of'), Spac
e(), Str(u'simple'), Space(), Str(u'table'), Space(), Str(u'syntax.')], [Ali
gnRight(), AlignLeft(), AlignCenter(), AlignDefault()], [0.0, 0.0, 0.0, 0.0]
, [[Plain([Str(u'Right')])], [Plain([Str(u'Left')])], [Plain([Str(u'Center')
])], [Plain([Str(u'Default')])]], [[[Plain([Str(u'12')])], [Plain([Str(u'12'
)])], [Plain([Str(u'12')])], [Plain([Str(u'12')])]], [[Plain([Str(u'123')])]
, [Plain([Str(u'123')])], [Plain([Str(u'123')])], [Plain([Str(u'123')])]], [
[Plain([Str(u'1')])], [Plain([Str(u'1')])], [Plain([Str(u'1')])], [Plain([St
r(u'1')])]]])])

>>> """
... -------     ------ ----------   -------
...      12     12        12            12
...     123     123       123          123
...       1     1          1             1
... -------     ------ ----------   -------
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Table([], [AlignRight(), AlignLeft(), AlignCenter(), Al
ignRight()], [0.0, 0.0, 0.0, 0.0], [[], [], [], []], [[[Plain([Str(u'12')])]
, [Plain([Str(u'12')])], [Plain([Str(u'12')])], [Plain([Str(u'12')])]], [[Pl
ain([Str(u'123')])], [Plain([Str(u'123')])], [Plain([Str(u'123')])], [Plain(
[Str(u'123')])]], [[Plain([Str(u'1')])], [Plain([Str(u'1')])], [Plain([Str(u
'1')])], [Plain([Str(u'1')])]]])])
```

Extension `multiline_tables`

```
>>> """
... -------------------------------------------------------------
...  Centered   Default           Right Left
...   Header    Aligned         Aligned Aligned
... ----------- ------- --------------- -------------------------
...    First    row                12.0 Example of a row that
...                                     spans multiple lines.
...
...   Second    row                 5.0 Here's another one. Note
...                                     the blank line between
...                                     rows.
... -------------------------------------------------------------
...
... Table: Here's the caption. It, too, may span
... multiple lines.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Table([Str(u"Here's"), Space(), Str(u'the'), Space(), S
tr(u'caption.'), Space(), Str(u'It,'), Space(), Str(u'too,'), Space(), Str(u
'may'), Space(), Str(u'span'), SoftBreak(), Str(u'multiple'), Space(), Str(u
'lines.')], [AlignCenter(), AlignDefault(), AlignRight(), AlignLeft()], [0.1
6666666666666666, 0.1111111111111111, 0.2222222222222222, 0.3472222222222222
], [[Plain([Str(u'Centered'), SoftBreak(), Str(u'Header')])], [Plain([Str(u'
Default'), SoftBreak(), Str(u'Aligned')])], [Plain([Str(u'Right'), SoftBreak
(), Str(u'Aligned')])], [Plain([Str(u'Left'), SoftBreak(), Str(u'Aligned')])
]], [[[Plain([Str(u'First')])], [Plain([Str(u'row')])], [Plain([Str(u'12.0')
])], [Plain([Str(u'Example'), Space(), Str(u'of'), Space(), Str(u'a'), Space
(), Str(u'row'), Space(), Str(u'that'), SoftBreak(), Str(u'spans'), Space(),
 Str(u'multiple'), Space(), Str(u'lines.')])]], [[Plain([Str(u'Second')])],
[Plain([Str(u'row')])], [Plain([Str(u'5.0')])], [Plain([Str(u"Here's"), Spac
e(), Str(u'another'), Space(), Str(u'one.'), Space(), Str(u'Note'), SoftBrea
k(), Str(u'the'), Space(), Str(u'blank'), Space(), Str(u'line'), Space(), St
r(u'between'), SoftBreak(), Str(u'rows.')])]]])])

>>> """
... ----------- ------- --------------- -------------------------
...    First    row                12.0 Example of a row that
...                                     spans multiple lines.
...
...   Second    row                 5.0 Here's another one. Note
...                                    the blank line between
...                                    rows.
... ----------- ------- --------------- -------------------------
...
... : Here's a multiline table without headers.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Table([Str(u"Here's"), Space(), Str(u'a'), Space(), Str
(u'multiline'), Space(), Str(u'table'), Space(), Str(u'without'), Space(), S
tr(u'headers.')], [AlignCenter(), AlignLeft(), AlignRight(), AlignLeft()], [
0.16666666666666666, 0.1111111111111111, 0.2222222222222222, 0.3472222222222
222], [[], [], [], []], [[[Plain([Str(u'First')])], [Plain([Str(u'row')])],
[Plain([Str(u'12.0')])], [Plain([Str(u'Example'), Space(), Str(u'of'), Space
(), Str(u'a'), Space(), Str(u'row'), Space(), Str(u'that'), SoftBreak(), Str
(u'spans'), Space(), Str(u'multiple'), Space(), Str(u'lines.')])]], [[Plain(
[Str(u'Second')])], [Plain([Str(u'row')])], [Plain([Str(u'5.0')])], [Plain([
Str(u"Here's"), Space(), Str(u'another'), Space(), Str(u'one.'), Space(), St
r(u'Note'), SoftBreak(), Str(u'the'), Space(), Str(u'blank'), Space(), Str(u
'line'), Space(), Str(u'between'), SoftBreak(), Str(u'rows.')])]]])])
```

Extension: grid_tables

```
... +---------------+---------------+--------------------+
... | Fruit         | Price         | Advantages         |
... +===============+===============+====================+
... | Bananas       | $1.34         | - built-in wrapper |
... |               |               | - bright color     |
... +---------------+---------------+--------------------+
... | Oranges       | $2.10         | - cures scurvy     |
... |               |               | - tasty            |
... +---------------+---------------+--------------------+
... """
... # doctest: +PANDOC
```

Extension: pipe_tables

```
>>> """
... | Right | Left | Default | Center |
... |------:|:-----|---------|:------:|
... |   12  |  12  |    12   |    12  |
... |  123  |  123 |   123   |   123  |
... |    1  |    1 |     1   |     1  |
...
...   : Demonstration of pipe table syntax.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Table([Str(u'Demonstration'), Space(), Str(u'of'), Spac
e(), Str(u'pipe'), Space(), Str(u'table'), Space(), Str(u'syntax.')], [Align
Right(), AlignLeft(), AlignDefault(), AlignCenter()], [0.0, 0.0, 0.0, 0.0],
[[Plain([Str(u'Right')])], [Plain([Str(u'Left')])], [Plain([Str(u'Default')]
)], [Plain([Str(u'Center')])]], [[[Plain([Str(u'12')])], [Plain([Str(u'12')]
)], [Plain([Str(u'12')])], [Plain([Str(u'12')])]], [[Plain([Str(u'123')])],
[Plain([Str(u'123')])], [Plain([Str(u'123')])], [Plain([Str(u'123')])]], [[P
lain([Str(u'1')])], [Plain([Str(u'1')])], [Plain([Str(u'1')])], [Plain([Str(
u'1')])]]])])

>>> """
... fruit| price
... -----|-----:
... apple|2.05
... pear|1.37
... orange|3.09
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Table([], [AlignDefault(), AlignRight()], [0.0, 0.0], [
[Plain([Str(u'fruit')])], [Plain([Str(u'price')])]], [[[Plain([Str(u'apple')
])], [Plain([Str(u'2.05')])]], [[Plain([Str(u'pear')])], [Plain([Str(u'1.37'
)])]], [[Plain([Str(u'orange')])], [Plain([Str(u'3.09')])]]])])

>>> """
... | One | Two   |
... |-----+-------|
... | my  | table |
... | is  | nice  |
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Table([], [AlignDefault(), AlignDefault()], [0.0, 0.0],
 [[Plain([Str(u'One')])], [Plain([Str(u'Two')])]], [[[Plain([Str(u'my')])],
[Plain([Str(u'table')])]], [[Plain([Str(u'is')])], [Plain([Str(u'nice')])]]]
)])
```

Metadata blocks

Extension: `pandoc_title_block`

```
>>> """\
... % title
... % author(s) (separated by semicolons)
... % date
... """
... # doctest: +PANDOC
Pandoc(Meta(map([(u'date', MetaInlines([Str(u'date')])), (u'title', MetaInli
nes([Str(u'title')])), (u'author', MetaList([MetaInlines([Str(u'author(s)'),
 Space(), Str(u'(separated'), Space(), Str(u'by'), Space(), Str(u'semicolons
)')])]))])), [])

>>> """\
... %
... % Author
... """
... # doctest: +PANDOC
Pandoc(Meta(map([(u'author', MetaList([MetaInlines([Str(u'Author')])]))])),
[])

>>> """\
... % My title
... %
... % June 15, 2006
... """
... # doctest: +PANDOC
Pandoc(Meta(map([(u'date', MetaInlines([Str(u'June'), Space(), Str(u'15,'),
Space(), Str(u'2006')])), (u'title', MetaInlines([Str(u'My'), Space(), Str(u
'title')]))])), [])

>>> """\
... % Title
... % Author One; Author Two
... """
... # doctest: +PANDOC
Pandoc(Meta(map([(u'title', MetaInlines([Str(u'Title')])), (u'author', MetaL
ist([MetaInlines([Str(u'Author'), Space(), Str(u'One')]), MetaInlines([Str(u
'Author'), Space(), Str(u'Two')])]))])), [])
```

Pandoc does not conform to its documentation when title blocks use multiple lines with leading space. The corresponding examples have been removed.

Extension: `yaml_metadata_block`

The order of key-value pairs in maps are the same than in the json representation, but this initial order is not specified by pandoc. Hence, the following test is too strict and may fail.

```
>>> """
... ---
... title:  'This is the title: it contains a colon'
... author:
... - name: Author One
...   affiliation: University of Somewhere
... - name: Author Two
...   affiliation: University of Nowhere
... tags: [nothing, nothingness]
... abstract: |
...   This is the abstract.
...
...   It consists of two paragraphs.
... ...
... """
... # doctest: +PANDOC
Pandoc(Meta(map([(u'abstract', MetaBlocks([Para([Str(u'This'), Space(), Str(
u'is'), Space(), Str(u'the'), Space(), Str(u'abstract.')]), Para([Str(u'It')
, Space(), Str(u'consists'), Space(), Str(u'of'), Space(), Str(u'two'), Spac
e(), Str(u'paragraphs.')])])), (u'title', MetaInlines([Str(u'This'), Space()
, Str(u'is'), Space(), Str(u'the'), Space(), Str(u'title:'), Space(), Str(u'
it'), Space(), Str(u'contains'), Space(), Str(u'a'), Space(), Str(u'colon')]
)), (u'tags', MetaList([MetaInlines([Str(u'nothing')]), MetaInlines([Str(u'n
othingness')])])), (u'author', MetaList([MetaMap(map([(u'affiliation', MetaI
nlines([Str(u'University'), Space(), Str(u'of'), Space(), Str(u'Somewhere')]
)), (u'name', MetaInlines([Str(u'Author'), Space(), Str(u'One')]))])), MetaM
ap(map([(u'affiliation', MetaInlines([Str(u'University'), Space(), Str(u'of'
), Space(), Str(u'Nowhere')])), (u'name', MetaInlines([Str(u'Author'), Space
(), Str(u'Two')]))]))]))])), [])
```

Backslash escapes

Extension: `all_symbols_escapable`

```
>>> """
... *\*hello\**
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Emph([Str(u'*hello*')])])])
```

Smart punctuation

Extension

Not tested, disabled by default.

Inline Formatting

Emphasis

```
>>> """
... This text is _emphasized with underscores_, and this
... is *emphasized with asterisks*.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'This'), Space(), Str(u'text'), Space(), Str
(u'is'), Space(), Emph([Str(u'emphasized'), Space(), Str(u'with'), Space(),
Str(u'underscores')]), Str(u','), Space(), Str(u'and'), Space(), Str(u'this'
), SoftBreak(), Str(u'is'), Space(), Emph([Str(u'emphasized'), Space(), Str(
u'with'), Space(), Str(u'asterisks')]), Str(u'.')])])
```

```
>>> """
... This is **strong emphasis** and __with underscores__.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'This'), Space(), Str(u'is'), Space(), Stron
g([Str(u'strong'), Space(), Str(u'emphasis')]), Space(), Str(u'and'), Space(
), Strong([Str(u'with'), Space(), Str(u'underscores')]), Str(u'.')])])
```

```
>>> """
... This is * not emphasized *, and \*neither is this\*.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'This'), Space(), Str(u'is'), Space(), Str(u
'*'), Space(), Str(u'not'), Space(), Str(u'emphasized'), Space(), Str(u'*,')
, Space(), Str(u'and'), Space(), Str(u'*neither'), Space(), Str(u'is'), Spac
e(), Str(u'this*.')])])
```

Extension: `intraword_underscores`

```
>>> "feas*ible*, not feas*able*." # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'feas'), Emph([Str(u'ible')]), Str(u','), Sp
ace(), Str(u'not'), Space(), Str(u'feas'), Emph([Str(u'able')]), Str(u'.')])
])
```

Strikeout

Extension: `strikeout`

```
>>> "This ~~is deleted text.~~" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'This'), Space(), Strikeout([Str(u'is'), Spa
ce(), Str(u'deleted'), Space(), Str(u'text.')])])])
```

Superscripts and Subscripts

Extension: `superscript`, `subscript`

```
>>> "H~2~O is a liquid.  2^10^ is 1024." # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'H'), Subscript([Str(u'2')]), Str(u'O'), Spa
ce(), Str(u'is'), Space(), Str(u'a'), Space(), Str(u'liquid.'), Space(), Str
(u'2'), Superscript([Str(u'10')]), Space(), Str(u'is'), Space(), Str(u'1024.
')])])
```

Verbatim

```
>>> "What is the difference between `>>=` and `>>`?" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'What'), Space(), Str(u'is'), Space(), Str(u
'the'), Space(), Str(u'difference'), Space(), Str(u'between'), Space(), Code
((u'', [], []), u'>>='), Space(), Str(u'and'), Space(), Code((u'', [], []),
u'>>'), Str(u'?')])])

>>> "Here is a literal backtick `` ` ``." # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'Here'), Space(), Str(u'is'), Space(), Str(u
'a'), Space(), Str(u'literal'), Space(), Str(u'backtick'), Space(), Code((u'
', [], []), u'`'), Str(u'.')])])

>>> "This is a backslash followed by an asterisk: `\*`." # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'This'), Space(), Str(u'is'), Space(), Str(u
'a'), Space(), Str(u'backslash'), Space(), Str(u'followed'), Space(), Str(u'
by'), Space(), Str(u'an'), Space(), Str(u'asterisk:'), Space(), Code((u'', [
], []), u'\\*'), Str(u'.')])])

>>> "`<$>`{.haskell}" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Code((u'', [u'haskell'], []), u'<$>')])])
```

Extension: `inline_code_attributes`

```
>>> "`<$>`{.haskell}" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Code((u'', [u'haskell'], []), u'<$>')])])
```

Small Caps

```
>>> "<span style='font-variant:small-caps;'>Small caps</span>"
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([SmallCaps([Str(u'Small'), Space(), Str(u'caps')])
])])
```

Math

Extension: `tex_math_dollars`

```
>>> "$a=1$" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Math(InlineMath(), u'a=1')])])

>>> "$$\int_0^1 f(x)\, dx$$" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Math(DisplayMath(), u'\\int_0^1 f(x)\\, dx')])])
```

Raw HTML

Extension: `raw_html`

```
>>> "<html></html>" # doctest: +PANDOC
Pandoc(Meta(map()), [RawBlock(Format(u'html'), u'<html>'), RawBlock(Format(u
'html'), u'</html>')])
```

Extension: `markdown_in_html_blocks`

```
>>> """
... <table>
... <tr>
... <td>*one*</td>
... <td>[a link](http://google.com)</td>
... </tr>
... </table>
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [RawBlock(Format(u'html'), u'<table>'), RawBlock(Format(
u'html'), u'<tr>'), RawBlock(Format(u'html'), u'<td>'), Plain([Emph([Str(u'o
ne')])]), RawBlock(Format(u'html'), u'</td>'), RawBlock(Format(u'html'), u'<
td>'), Plain([Link((u'', [], []), [Str(u'a'), Space(), Str(u'link')], (u'htt
p://google.com', u''))]), RawBlock(Format(u'html'), u'</td>'), RawBlock(Form
at(u'html'), u'</tr>'), RawBlock(Format(u'html'), u'</table>')])
```

Extension: `native_divs`

```
>>> "<div></div>" # doctest: +PANDOC
Pandoc(Meta(map()), [Div((u'', [], []), [])])
```

Extension: `native_spans`

```
>>> "<span></span>" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Span((u'', [], []), [])])])
```

Raw TeX

Extension: `raw_tex`

```
>>> "This result was proved in \cite{jones.1967}."
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'This'), Space(), Str(u'result'), Space(), S
tr(u'was'), Space(), Str(u'proved'), Space(), Str(u'in'), Space(), RawInline
(Format(u'tex'), u'\\cite{jones.1967}'), Str(u'.')])])

>>> r"""
... \begin{tabular}{|l|l|}\hline
... Age & Frequency \\ \hline
... 18--25  & 15 \\
... 26--35  & 33 \\
... 36--45  & 22 \\ \hline
... \end{tabular}
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [RawBlock(Format(u'latex'), u'\\begin{tabular}{|l|l|}\\h
line\nAge & Frequency \\\\ \\hline\n18--25  & 15 \\\\\n26--35  & 33 \\\\\n36
--45  & 22 \\\\ \\hline\n\\end{tabular}')])
```

LaTeX macros

Extension: `latex_macros`

```
>>> r"""
... \newcommand{\tuple}[1]{\langle #1 \rangle}
...
... $\tuple{a, b, c}$
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Math(InlineMath(), u'{\\langle a, b, c \\rangle}'
)])])
```

Links

Automatic Links

```
>>> "<http://google.com>" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Link((u'', [], []), [Str(u'http://google.com')],
(u'http://google.com', u''))])])

>>> "<sam@green.eggs.ham>" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Link((u'', [], []), [Str(u'sam@green.eggs.ham')],
 (u'mailto:sam@green.eggs.ham', u''))])])
```

Inline links

```
>>> """
... This is an [inline link](/url), and here's [one with
... a title](http://fsf.org "click here for a good time!
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'This'), Space(), Str(u'is'), Space(), Str(u
'an'), Space(), Link((u'', [], []), [Str(u'inline'), Space(), Str(u'link')],
 (u'/url', u'')), Str(u','), Space(), Str(u'and'), Space(), Str(u"here's"),
Space(), Str(u'[one'), Space(), Str(u'with'), SoftBreak(), Str(u'a'), Space(
), Str(u'title](http://fsf.org'), Space(), Str(u'"click'), Space(), Str(u'he
re'), Space(), Str(u'for'), Space(), Str(u'a'), Space(), Str(u'good'), Space
(), Str(u'time!')])])

>>> "[Write me!](mailto:sam@green.eggs.ham)" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Link((u'', [], []), [Str(u'Write'), Space(), Str(
u'me!')], (u'mailto:sam@green.eggs.ham', u''))])])
```

Reference links

```
>>> """
... [my label 1], [my label 2], [my label 3], [my label 4].
...
... [my label 1]: /foo/bar.html  "My title, optional"
... [my label 2]: /foo
... [my label 3]: http://fsf.org (The free software foundation)
... [my label 4]: /bar#special  'A title in single quotes'
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Link((u'', [], []), [Str(u'my'), Space(), Str(u'l
abel'), Space(), Str(u'1')], (u'/foo/bar.html', u'My title, optional')), Str
(u','), Space(), Link((u'', [], []), [Str(u'my'), Space(), Str(u'label'), Sp
ace(), Str(u'2')], (u'/foo', u'')), Str(u','), Space(), Link((u'', [], []),
[Str(u'my'), Space(), Str(u'label'), Space(), Str(u'3')], (u'http://fsf.org'
, u'The free software foundation')), Str(u','), Space(), Link((u'', [], []),
 [Str(u'my'), Space(), Str(u'label'), Space(), Str(u'4')], (u'/bar#special',
 u'A title in single quotes')), Str(u'.')])])

>>> """
... [my label 5].
...
... [my label 5]: <http://foo.bar.baz>
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Link((u'', [], []), [Str(u'my'), Space(), Str(u'l
abel'), Space(), Str(u'5')], (u'http://foo.bar.baz', u'')), Str(u'.')])])

>>> """
... [my label 3].
...
... [my label 3]: http://fsf.org "The free software foundation"
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Link((u'', [], []), [Str(u'my'), Space(), Str(u'l
abel'), Space(), Str(u'3')], (u'http://fsf.org', u'The free software foundat
ion')), Str(u'.')])])

>>> """
... Here is [my link][FOO]
...
... [Foo]: /bar/baz
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'Here'), Space(), Str(u'is'), Space(), Link(
(u'', [], []), [Str(u'my'), Space(), Str(u'link')], (u'/bar/baz', u''))])])

>>> """
... See [my website][].
...
... [my website]: http://foo.bar.baz
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'See'), Space(), Link((u'', [], []), [Str(u'
my'), Space(), Str(u'website')], (u'http://foo.bar.baz', u'')), Str(u'.')])]
)

>>> """
... > My block [quote].
... >
... > [quote]: /foo
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [BlockQuote([Para([Str(u'My'), Space(), Str(u'block'), S
pace(), Link((u'', [], []), [Str(u'quote')], (u'/foo', u'')), Str(u'.')])])]
)
```

Extension: `shortcut_reference_links`

```
>>> """
... See [my website].
...
... [my website]: http://foo.bar.baz
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'See'), Space(), Link((u'', [], []), [Str(u'
my'), Space(), Str(u'website')], (u'http://foo.bar.baz', u'')), Str(u'.')])]
)
```

Internal links

```
>>> "See the [Introduction](#introduction)." # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'See'), Space(), Str(u'the'), Space(), Link(
(u'', [], []), [Str(u'Introduction')], (u'#introduction', u'')), Str(u'.')])
])


>>> """
... See the [Introduction].
...
... [Introduction]: #introduction
... """
... # doctest: +PANDOC
```

```
Pandoc(Meta(map()), [Para([Str(u'See'), Space(), Str(u'the'), Space(), Link(
(u'', [], []), [Str(u'Introduction')], (u'#introduction', u'')), Str(u'.')])
])
```

Images

```
>>> """
... ![la lune](lalune.jpg "Voyage to the moon")
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Image((u'', [], []), [Str(u'la'), Space(), Str(u'
lune')], (u'lalune.jpg', u'fig:Voyage to the moon'))])])

>>> """
... ![movie reel]
...
... [movie reel]: movie.gif
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Image((u'', [], []), [Str(u'movie'), Space(), Str
(u'reel')], (u'movie.gif', u'fig:'))])])
```

Extension: `implicit_figures`

```
>>> """
... ![This is the caption](/url/of/image.png)
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Image((u'', [], []), [Str(u'This'), Space(), Str(
u'is'), Space(), Str(u'the'), Space(), Str(u'caption')], (u'/url/of/image.pn
g', u'fig:'))])])
```

```
ISSUE HERE.
```

```
>>> r"""
... ![This image won't be a figure](/url/of/image.png)\
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Image((u'', [], []), [Str(u'This'), Space(), Str(
u'image'), Space(), Str(u"won't"), Space(), Str(u'be'), Space(), Str(u'a'),
Space(), Str(u'figure')], (u'/url/of/image.png', u'')), Str(u'\xa0')])])
```

Extension: link_attributes

```
>>> """
... An inline ![image](foo.jpg){#id .class width=30 height=20px}
... and a reference ![image][ref] with attributes.
...
... [ref]: foo.jpg "optional title" {#id .class key=val key2="val 2"}
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'An'), Space(), Str(u'inline'), Space(), Ima
ge((u'id', [u'class'], [(u'width', u'30'), (u'height', u'20px')]), [Str(u'im
age')], (u'foo.jpg', u'')), SoftBreak(), Str(u'and'), Space(), Str(u'a'), Sp
ace(), Str(u'reference'), Space(), Image((u'id', [u'class'], [(u'key', u'val
'), (u'key2', u'val 2')]), [Str(u'image')], (u'foo.jpg', u'optional title'))
, Space(), Str(u'with'), Space(), Str(u'attributes.')])])

>>> "![](file.jpg){ width=50% }" # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Image((u'', [], [(u'width', u'50%')]), [], (u'fil
e.jpg', u'fig:'))])])
```

Footnotes

Extension: `footnotes`

Pandoc's Markdown allows footnotes, using the following syntax:

```
>>> """
... Here is a footnote reference,[^1] and another.[^longnote]
...
... [^1]: Here is the footnote.
...
... [^longnote]: Here's one with multiple blocks.
...
...     Subsequent paragraphs are indented to show that they
... belong to the previous footnote.
...
...         { some.code }
...
...     The whole paragraph can be indented, or just the first
...     line.  In this way, multi-paragraph footnotes work like
...     multi-paragraph list items.
```

```
...
... This paragraph won't be part of the note, because it
... isn't indented.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'Here'), Space(), Str(u'is'), Space(), Str(u
'a'), Space(), Str(u'footnote'), Space(), Str(u'reference,'), Note([Para([St
r(u'Here'), Space(), Str(u'is'), Space(), Str(u'the'), Space(), Str(u'footno
te.')])]), Space(), Str(u'and'), Space(), Str(u'another.'), Note([Para([Str(
u"Here's"), Space(), Str(u'one'), Space(), Str(u'with'), Space(), Str(u'mult
iple'), Space(), Str(u'blocks.')]), Para([Str(u'Subsequent'), Space(), Str(u
'paragraphs'), Space(), Str(u'are'), Space(), Str(u'indented'), Space(), Str
(u'to'), Space(), Str(u'show'), Space(), Str(u'that'), Space(), Str(u'they')
, SoftBreak(), Str(u'belong'), Space(), Str(u'to'), Space(), Str(u'the'), Sp
ace(), Str(u'previous'), Space(), Str(u'footnote.')]), CodeBlock((u'', [], [
]), u'{ some.code }'), Para([Str(u'The'), Space(), Str(u'whole'), Space(), S
tr(u'paragraph'), Space(), Str(u'can'), Space(), Str(u'be'), Space(), Str(u'
indented,'), Space(), Str(u'or'), Space(), Str(u'just'), Space(), Str(u'the'
), Space(), Str(u'first'), SoftBreak(), Str(u'line.'), Space(), Str(u'In'),
Space(), Str(u'this'), Space(), Str(u'way,'), Space(), Str(u'multi-paragraph
'), Space(), Str(u'footnotes'), Space(), Str(u'work'), Space(), Str(u'like')
, SoftBreak(), Str(u'multi-paragraph'), Space(), Str(u'list'), Space(), Str(
u'items.')])])]), Para([Str(u'This'), Space(), Str(u'paragraph'), Space(), S
tr(u"won't"), Space(), Str(u'be'), Space(), Str(u'part'), Space(), Str(u'of'
), Space(), Str(u'the'), Space(), Str(u'note,'), Space(), Str(u'because'), S
pace(), Str(u'it'), SoftBreak(), Str(u"isn't"), Space(), Str(u'indented.')])
])
```

Extension: `inline_notes`

```
>>> """
... Here is an inline note.^[Inlines notes are easier to write, since
... you don't have to pick an identifier and move down to type the
... note.]
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'Here'), Space(), Str(u'is'), Space(), Str(u
'an'), Space(), Str(u'inline'), Space(), Str(u'note.'), Note([Para([Str(u'In
lines'), Space(), Str(u'notes'), Space(), Str(u'are'), Space(), Str(u'easier
'), Space(), Str(u'to'), Space(), Str(u'write,'), Space(), Str(u'since'), So
ftBreak(), Str(u'you'), Space(), Str(u"don't"), Space(), Str(u'have'), Space
(), Str(u'to'), Space(), Str(u'pick'), Space(), Str(u'an'), Space(), Str(u'i
dentifier'), Space(), Str(u'and'), Space(), Str(u'move'), Space(), Str(u'dow
n'), Space(), Str(u'to'), Space(), Str(u'type'), Space(), Str(u'the'), SoftB
reak(), Str(u'note.')])])])])])
```

Citations

Extension: `citations`

```
>>> """
... ---
... references:
... - type: article-journal
...   id: WatsonCrick1953
...   author:
...   - family: Watson
...     given: J. D.
...   - family: Crick
...     given: F. H. C.
...   issued:
...     date-parts:
...     - - 1953
...       - 4
...       - 25
...   title: 'Molecular structure of nucleic acids: a structure for deoxyribose
...     nucleic acid'
...   title-short: Molecular structure of nucleic acids
...   container-title: Nature
...   volume: 171
...   issue: 4356
...   page: 737-738
...   DOI: 10.1038/171737a0
...   URL: http://www.nature.com/nature/journal/v171/n4356/abs/171737a0.html
...   language: en-GB
... ---
...
... [@WatsonCrick1953]
... """
... # doctest: +PANDOC
Pandoc(Meta(map([(u'references', MetaList([MetaMap(map([(u'DOI', MetaInlines
([Str(u'10.1038/171737a0')])), (u'language', MetaInlines([Str(u'en-GB')])),
(u'author', MetaList([MetaMap(map([(u'given', MetaInlines([Str(u'J.'), Space
(), Str(u'D.')])), (u'family', MetaInlines([Str(u'Watson')]))])), MetaMap(ma
p([(u'given', MetaInlines([Str(u'F.'), Space(), Str(u'H.'), Space(), Str(u'C
.')])), (u'family', MetaInlines([Str(u'Crick')]))]))])), (u'URL', MetaInline
s([Str(u'http://www.nature.com/nature/journal/v171/n4356/abs/171737a0.html')
])), (u'issued', MetaMap(map([(u'date-parts', MetaList([MetaList([MetaString
(u'1953'), MetaString(u'4'), MetaString(u'25')])]))]))), (u'title', MetaInli
nes([Str(u'Molecular'), Space(), Str(u'structure'), Space(), Str(u'of'), Spa
ce(), Str(u'nucleic'), Space(), Str(u'acids:'), Space(), Str(u'a'), Space(),
 Str(u'structure'), Space(), Str(u'for'), Space(), Str(u'deoxyribose'), Spac
e(), Str(u'nucleic'), Space(), Str(u'acid')])), (u'id', MetaInlines([Str(u'W
atsonCrick1953')])), (u'volume', MetaString(u'171')), (u'issue', MetaString(
u'4356')), (u'container-title', MetaInlines([Str(u'Nature')])), (u'title-sho
rt', MetaInlines([Str(u'Molecular'), Space(), Str(u'structure'), Space(), St
r(u'of'), Space(), Str(u'nucleic'), Space(), Str(u'acids')])), (u'type', Met
aInlines([Str(u'article-journal')])), (u'page', MetaInlines([Str(u'737-738')
]))]))]))])), [Para([Cite([Citation(u'WatsonCrick1953', [], [], NormalCitati
on(), 0, 0)], [Str(u'[@WatsonCrick1953]')])])])

>>> """
... Blah blah [see @doe99, pp. 33-35; also @smith04, chap. 1].
...
... Blah blah [@doe99, pp. 33-35, 38-39 and *passim*].
...
... Blah blah [@smith04; @doe99].
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'Blah'), Space(), Str(u'blah'), Space(), Cit
e([Citation(u'doe99', [Str(u'see')], [Str(u','), Space(), Str(u'pp.'), Space
(), Str(u'33-35')], NormalCitation(), 0, 0), Citation(u'smith04', [Str(u'als
o')], [Str(u','), Space(), Str(u'chap.'), Space(), Str(u'1')], NormalCitatio
n(), 0, 0)], [Str(u'[see'), Space(), Str(u'@doe99,'), Space(), Str(u'pp.'),
Space(), Str(u'33-35;'), Space(), Str(u'also'), Space(), Str(u'@smith04,'),
Space(), Str(u'chap.'), Space(), Str(u'1]')]), Str(u'.')]), Para([Str(u'Blah
'), Space(), Str(u'blah'), Space(), Cite([Citation(u'doe99', [], [Str(u','),
 Space(), Str(u'pp.'), Space(), Str(u'33-35,'), Space(), Str(u'38-39'), Spac
e(), Str(u'and'), Space(), Emph([Str(u'passim')])], NormalCitation(), 0, 0)]
, [Str(u'[@doe99,'), Space(), Str(u'pp.'), Space(), Str(u'33-35,'), Space(),
 Str(u'38-39'), Space(), Str(u'and'), Space(), Str(u'*passim*]')]), Str(u'.'
)]), Para([Str(u'Blah'), Space(), Str(u'blah'), Space(), Cite([Citation(u'sm
ith04', [], [], NormalCitation(), 0, 0), Citation(u'doe99', [], [], NormalCi
tation(), 0, 0)], [Str(u'[@smith04;'), Space(), Str(u'@doe99]')]), Str(u'.')
])])
```

```
>>> "Smith says blah [-@smith04]." # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Str(u'Smith'), Space(), Str(u'says'), Space(), St
r(u'blah'), Space(), Cite([Citation(u'smith04', [], [], SuppressAuthor(), 0,
 0)], [Str(u'[-@smith04]')]), Str(u'.')])])

>>> """
... @smith04 says blah.
...
... @smith04 [p. 33] says blah.
... """
... # doctest: +PANDOC
Pandoc(Meta(map()), [Para([Cite([Citation(u'smith04', [], [], AuthorInText()
, 0, 0)], [Str(u'@smith04')]), Space(), Str(u'says'), Space(), Str(u'blah.')
]), Para([Cite([Citation(u'smith04', [], [Str(u'p.'), Space(), Str(u'33')],
AuthorInText(), 0, 0)], [Str(u'@smith04'), Space(), Str(u'[p.'), Space(), St
r(u'33]')]), Space(), Str(u'says'), Space(), Str(u'blah.')])])

>>> """
... ---
... nocite: |
...   @item1, @item2
... ---
...
... @item3
... """
... # doctest: +PANDOC
Pandoc(Meta(map([(u'nocite', MetaBlocks([Para([Cite([Citation(u'item1', [],
[], AuthorInText(), 0, 0)], [Str(u'@item1')]), Str(u','), Space(), Cite([Cit
ation(u'item2', [], [], AuthorInText(), 0, 0)], [Str(u'@item2')])])]))])), [
Para([Cite([Citation(u'item3', [], [], AuthorInText(), 0, 0)], [Str(u'@item3
')])])])
```