```
# First set a few module variables:
package = BeamerSlides # Can also be 'ProsperSlides' and 'HTMLSlides'
beamer_theme = 'hpl1' #'shadow', 'Darmstadt' are other themes
prosper_style = 'hplplainsmall' # 'default', 'hplplain' are other themes
header_footer = True # Decorations are turned on

# Add newcommands:
newcommands = r"""\newcommand{\OBS}[1]{\marginpar{\scriptsize##1}}"""

# Set institutions:
ifi = "Dept.~of Informatics, University of Oslo"
math = "Dept.~of Mathematics, University of Oslo"
simula = "Simula Research Laboratory"

# Set authors:
hpl = 'Hans Petter Langtangen'
ilmarw = 'Ilmar M. Wilbers'

slides = package(title='Using Python and Latexslides to Make Slides',
                 author_and_inst=[(hpl,simula,ifi), (ilmarw,simula,math)],
                 date='March 2008',
                 beamer_theme=beamer_theme,
                 prosper_style=prosper_style,
                 header_footer=header_footer,
                 titlepage_figure='wave-dueto-slide.ps',
                 titlepage_figure_pos='s', # Figure to the south
                 titlepage_figure_fraction_width=0.5,
                 #titlepage_left_column_width=1., # If figure to the east
                 toc_heading='List of Topics',
                 toc_figure='python1.ps',
                 toc_figure_fraction_width=1,
                 toc_left_column_width=0.5,
                 newcommands=newcommands)
```

```
# Exemplify raw LaTeX, please note that the code is for Beamer,
# so if you want to use Prosper, you should comment out this slide
# when writing to file:
first_intro = RawSlide(r"""

\begin{frame} %% plain: no header and footer

\frametitle{Do you use \LaTeX{} for writing slides?}
Continue studying these slides if your answer to at least one of
the following questions is 'yes':
\begin{enumerate}
\item Are you using prosper for writing slides?
\item Have you not yet discovered latex-beamer?
\item Would you like your slide collection to be independent of what
is the currently most popular LaTeX slide package?
\item Would you like to write less LaTeX source code when you
create presentations?
\item Would like to get more flexibility than what plain ASCII
files with LaTeX source provide?
\end{enumerate}
\end{frame}
""",
hidden=False)
```

```
# The talk can be divided into sections, subsections, etc., and
# then a toc is automatically generated, navigation utilities
# are included in the header etc.

sec_intro = Section(r’Intro to Latexslides’, short_title=’Intro’)
```

```
what_is = \
BulletSlide('What is Latexslides?',
            ['A Python module',
             'You write slides as Python code, i.e., as function calls',
             r'The function calls are translated to \LaTeX',
             'Changes are easier to perform in the Python code than '
             r'in the corresponding \LaTeX code -- that is the main '
             'purpose of Latexslides',
             'From the Python code you can automatically generate '
             r'prosper or beamer \LaTeX code and HTML',
             ],)
```

```
subsec_plaintext = SubSection(’Plain Text Slides’, ’Text’)
```

```
ex1 = \
Slide('First example: simple bullet lists',
      [TextBlock(Code("""
BulletSlide('What is Latexslides?',
['A Python module',
 'You write slides as Python code, i.e., as function calls',
 r'The function calls are translated to \LaTeX',
 'Changes are easier to perform in the Python code than '
 r'in the corresponding \LaTeX code -- that is the main '
 'purpose of Latexslides',
 'From the Python code you can automatically generate '
 r'prosper or beamer \LaTeX code and HTML'],)
"""),
                 'Here is how we wrote the previous slide:',),
       BulletBlock(heading='Explanations:',
                   bullets=[r'The first argument is the title of the slide',
                            r'Bullet lists are simply Python lists of '
                            r'(raw) strings',],),],)
```

```
structure = \
Slide(r"A general slide is defined by using \texttt{Slide}",
      [CodeBlock("""
Slide(title='title', content=[
    Text(r'some plain text'),
    BulletList([r'item1',
                r'item2',
                r'item3',
               ],),
    Code(r'some code'),],)
"""),
       BulletBlock([r'Use raw strings if the text has \LaTeX{} commands '
                    r'with backslash (always using raw strings is a good '
                    r'habit!)',
                    # If you don't like one long strintg line, split it:
                    r'The \texttt{title=} and \texttt{content=} keywords '
                    r'can be omitted if they are the first two arguments '
                    r'given to Slide or one of its subclasses.'
                    ],),
       Text(r'The available objects on a slide are ' +
            r'\texttt{Text}, \texttt{Code} and \texttt{BulletList}',),],)
```

```
blocks = \
Slide("About the appearance of the three slide elements",
      [TextBlock('Text, code and bullets can be typeset as shadowed blocks '
                 'by using TextBlock, CodeBlock and BulletBlock instead ' +
                 'of Text, Code and BulletList',),
       BulletList(['Note that this text is not a block since we used' +
                   Code("""BulletList""") +
                   r'instead of \texttt{BulletBlock}',],),
       TextBlock(heading='Each block may have a title!',
                 text='The title is enabled by the argument' +
                 Code("heading='Each block may ...'"),),
       TextBlock(r'Note that ' + Code('Code("...")') +
                 'can be used within the other objects',),],)
```

```
classes = \
Slide('Blocks and slides',
      [TextBlock(r'If only a single block is used on a slide, ' +
                 r'subclasses of \texttt{Slide} can be used, ' +
                 r'providing a simpler syntax:'),
       BulletBlock([r'\texttt{BulletSlide}',
                    r'\texttt{TextSlide}',
                    r'\texttt{RawSlide}',],),
       TextBlock(r'These only have the following keywords:'),
       BulletBlock([Code('title'), Code('text/bullets'),
                    Code('block_heading'),],),
       TextBlock(r'\texttt{RawSlide} simply takes the raw \LaTeX{} code '
                 'for a slide such that old code can be reused (see the '
                 'code for the first slide)'),],
      dim='blocks')
```

```
dimming = \
BulletSlide('How to dim blocks and bullet points',
            [r'Want to dim the blocks (as in the previous slide)? '
             'Just add an argument' +
             Code("""dim='blocks'"""),
             r'Want bullet items to pop up one by one? '
             'Just add an argument' +
             Code("""dim='progressive'"""),
             r'Want one bullet visible and the other dimmed? Just add' +
             Code("dim='single' #dim=False (default) turns off dimming") +
             r'Note that subbullets appear at the same time:',
             ['Subbullet 1',],
             r'Want the previous effect but with all bullets appearing '
             'at the end? Just add' +
             Code("""dim='single_then_all'"""),
             r'Changing these arguments is very much easier than editing '
             r'the underlying \LaTeX{} code!',
             ],
            dim=True)
```

```
why = \
BulletSlide('Why? plain LaTeX is so easy...',
            ['Latexslides turns the talk into living data structures',
             'With the data structures, you can easily generate Prosper, '
             'Beamer, HTML or write your own format output',
             "Some of us experimented with the idea for fun, now we're "
             "regularly using it -- it's simply more convenient",
             r'Latexslides talks can make use of future fancy \LaTeX{} '
             'slide packages',
             r'You can tweak the resulting \LaTeX{} file if you want',
             'Talks are composed as lists of slide objects -- you can '
             'import slide objects from previous talks and compose new '
             'collections',
             'Figures are definitely easier with Latexslides',
             ],
            dim='single_then_all',)
```

```
# Short form is taken as the full section name
sec_fig = SubSection(’Figures’)
```

```
figures = \
Slide('Handling figures is really easy',
      [BulletBlock([r'Putting figures in \LaTeX{} slides is not that easy, '
                    'especially not if you want them to the left or right '
                    'of the figure and move them around later',
                    r'Here is what you do with Latexslides, just add' +
                    Code("""
figure='python1.ps',   # filename
figure_pos='e',        # east ('e'), west ('w'),
                       # north ('n'), south ('e')
figure_fraction_width=0.5,
left_column_width=0.6  # => 0.4 fraction width for figure
"""),],),
       Text('See next slides for an example',),],)
```

```
ex_fig = \
Slide('Slide with a figure',
      [BulletBlock(['Bullets to the west',
                    'Figure to the east',
                    r"Easy to change: \texttt{'e'} $\rightarrow$ "
                    r"\texttt{'w'}",
                    '...and then text is to the right',
                    ],),],
      figure='python1.ps',
      figure_pos='e',
      figure_fraction_width=0.5,
      left_column_width=0.6  # 0.4 left for figure
      )
```

```
ex_fig2 = \
Slide('Slide with a figure',
      [BulletBlock(['Bullets to the east',
                    'Figure to the west',
                    r"Easy to change: \texttt{'w'} $\rightarrow$ "
                    r"\texttt{'n'}",
                    '...and then text below the figure',
                    ],),],
      figure='python1.ps',
      figure_pos='w',
      figure_fraction_width=0.5,
      left_column_width=0.6  # 0.4 left for figure
      )
```

```
ex_fig3 = \
Slide('Slide with a figure',
      [BulletBlock(['Bullets to the south',
                    'Figure to the north',
                    r"Easy to change: \texttt{'n'} $\rightarrow$ "
                    r"\texttt{'s'}",
                    '...and then text is above the figure',
                    ],),],
      figure='python1.ps',
      figure_pos='n',
      figure_fraction_width=0.5,
      left_column_width=0.6  # 0.4 left for figure
      )
```

```
multiple_figs = \
Slide('Several figures in one slide',
      [BulletList(['You simply provide a tuple (or list) of figure '
                   'file names and a tuple of fraction widths',
                  r'Example:' + Code("""
figure=('python2.ps','python3.ps'),
figure_fraction_width=(0.45,0.55),
figure_pos='n',
"""),],),],
      figure=('python2.ps','python3.ps'),
      figure_fraction_width=(0.45,0.55),
      figure_pos='n',)
```

```
subsec_code = SubSection('Computer Code', 'Code')
```

```
code_obj = \
BulletSlide('Code objects take care of verbatim text',
            [r'Want to include computer code or some '
             r'other verbatim text?\\ ' +
             Code('''
bullets=[r'Here is an example:' +
Code("""
def mypyfunc(somearg):
    for i in somearg:
        p = process(i)
        if p in mylist:
            return p
        else:
            return None
""")
'''),
             r'Code objects are wrapped in fancyvrb verbatim environments'
             ,],)
```

```
code_obj_result = \
BulletSlide('Result of using Code objects',
            block_heading='Here is the result of the constructions on the '
                          'previous slide:',
            bullets=[r'Here is an example:' +
                     Code("""
def mypyfunc(somearg):
    for i in somearg:
        p = process(i)
        if p in mylist:
            return p
        else:
            return None
"""),],)
```

```
sec_specials = Section('More information', 'More')
```

```
sections = \
BulletSlide('Adding sections and subsections',
            ['Adding a section is just like adding a slide: ' +
            Code(r"""sec = Section('Long title', 'Short title')""") +
            'The short title is optional, and will be used if there is not '
            'enough room for the long title',
            'SubSection works the same way, but a Section needs to be '
            'defined prior to a SubSection',
            'Slide objects are automatically a part of the current section '
            'or subsection',
            'If no sections are defined, all slides will be part of the '
            'main talk',],)
```

```
navigation = \
BulletSlide('You can turn off the header and footer',
            bullets=[r'Want to navigate in your talk? Click in the header!',
                     r'Sometimes the navigation header and the '
                     'author/title in the footer is disturbing',
                     r'Turn header/footer decoration off for all slides:' +
                     Code("""
header_footer = False
"""),],)
```

```
ex_header = \
TextSlide('The look of the file header',
          Code(r'''
from latexslides import *

# First set some module variables:
package = BeamerSlides
theme = 'hpl1'
header_footer = True

# Add newcommands:
newcommands = r"""
\newcommand{\OBS}[1]{\marginpar{\scriptsize##1}}
"""
'''),)
```

```
prosper = \
BulletSlide('Can I change from Beamer to Prosper or HTML?',
            bullets=['Of course, this is trivial:' +
                     Code("""
#package = BeamerSlides
package = ProsperSlides
package = HTMLSlides
"""),
                     'Prosper is fine (best?) for handouts',
                     'Handouts for Beamer are made setting the keyword ' +
                     Code('handout=True') + 'for colour prints and ' +
                     Code('colour=False') + 'for b/w handouts',],)
```

```
ex_titlepage = \
TextSlide('The titlepage',
          Code(r'''
ifi = "Dept.~of Informatics, University of Oslo"
math = "Dept.~of Mathematics, University of Oslo"
simula = "Simula Research Laboratory"

hpl = 'Hans Petter Langtangen'
ilmarw = 'Ilmar M. Wilbers'

slides = \
package(title='Using Python and Latexslides to Make Slides',
        author_and_inst=[(hpl, simula, ifi),
                         (ilmarw, simula, math)],
        date='March 2008',
        titlepage_figure='wave-dueto-slide.ps',
        # Figure to the south of the title:
        titlepage_figure_pos='s',
        titlepage_figure_fraction_width=0.5,
        # Used if titlepage_figure_pos is 'e':
        #titlepage_left_column_width=1.0,
        toc_heading='List of Topics',
        toc_figure='clipart/python1.ps',
        toc_figure_fraction_width=1,
        toc_left_column_width=0.5,
        newcommands=newcommands)
'''),)
```

```
emacs = \
BulletSlide('Emacs commands',
            ['The authors have found the following Emacs shortcuts '
             'very helpful:',
             ['Alt + up-arrow:' +
             Code(R"""
(global-set-key [ (meta up)] " = Slide('',
content=[BulletBlock(bullets=[
    '','")
"""),],
             ['Alt + down-arrow:' +
             Code(r"""
(global-set-key [ (meta down)] "
]),  # end bullets and BulletBlock
],   # end contents
)")
"""),],
            r'These should be included in the \texttt{.emacs} file '
            'in your home directory',
            'This example is for the opening and closing of a BulletBlock, '
            'but illustrate how Emacs shortcuts can be used',],)
```

```
slide_obj1 = \
BulletSlide('Slide Objects 1 ',
            ['You may save each slide in a Slide object (recommended!!)' +
             Code(r"""
slides = BeamerSlides(...)
motivation2 = \
Slide('Motivation Cont.',
      [BulletBlock([...], ...), ...]
   """),
             'A list of all slide objects in a file can be generated ' +
             'with the following executable:' +
             Code(r"""
extract_slidenames mytalk.py
"""),
             'The generated list can be included at the bottom ' +
             'of your file',],)
```

```
slide_obj2 = \
BulletSlide('Slide Objects 2 ',
            ['Talks can be composed of lists of slide objects' +
            Code("""
slides = BeamerSlides(...)
collection = [header, title, sec1, intro1, intro2, sec2,
              plainloop, parallel1, summary]
# Can make some slides invisible:
for s in parallel1, references:
    s.hidden = True
# Or perhaps more elegant:
collection = [header, title, sec1, intro1, intro2, sec2,
              plainloop, parallel1.hide, summary]

for s in collection:
    slides.add_slide(s)
# Write slides to file:
f = open('exampletalk.tex', 'w')
f.write(slides.get_latex())

# Or the simplest, which will output the
# necessary latex commands as well:
slides.write(filename)
"""),
            Text('In this way you can reuse old slides in new contexts '
                 'without cut and paste, i.e., you can have a single '
                 'source for each slide (important for large slide '
                 'collections!)'),],)
```

```
compiling = \
BulletSlide('How to Compile the Talk',
            ['You write your talk as Python code in a plain text file, '
             r'say \emp{mytalk.py}',
             r'The next step is to generate \LaTeX{} code:' +
             Code(r"""
unix> python mytalk.py
"""),
             'The latex commands you need to run will be the output ' +
             'if the \emp{write} function is used',],)
```

```
maths = \
Slide('How to write mathematics',
      [TextBlock('Use triple-quoted raw strings and just write the plain '
                 '\LaTeX{} code'),
       TextBlock(heading='Latexslide source:',
                 text=Code(r'''TextBlock(heading='Latexslide source:',
          text=r"""Here is an equation
\[ ax^2 + bx + c = 0\]
that is easy to solve.
""")
''')),
      TextBlock(heading='Result:',
                text=r"""Here is an equation
\[ ax^2 + bx + c = 0\]
that is easy to solve.
"""),],)
```

```
learning = \
Slide('Learning Latexslides',
      [BulletBlock(['Have a look at the source code for this presentation, '
                    r'it can be found in the file \emp{exampletalk.py}. '
                    'Going through the presentation and the source code '
                    'simultaneously should get you started.'],),
       BulletBlock(['When running' +
                    Code(r"""
unix> latexslides mytalk.py
""") +
                    'the file \emp{mytalk.py} is created. This file '
                    'contains the basics and will help you get started '
                    'on a new talk.',],),],)
```