The `brandeis-problemset` Document Class

Rebecca Turner*

2019/04/15 0.5.5

Abstract

Brandeis University's computer science ("COSI") courses often assign "problem sets" which require fairly rigorous formatting. The `brandeis-problemset` document class, which extends `article`, provides a simple way to typeset these problem sets in LaTeX.

Although `brandeis-problemset` is compatible with all LaTeX flavors, XeLaTeX or LuaTeX is recommended for `fontspec` support.

NOTE The `brandeis-problemset` document class should be considered experimental; the only stable API is that of the `problem` environment.

NOTE Browse the sources, contribute, or complain at github.com/9999years/brandeis-problemset

NOTE In roughly May 2021 I will graduate from Brandeis University and this package will become unmaintained. Although I'd like the computer science department to adopt this package, I'm not sure any professors have an interest in that maintenance. If you care about the extended future of this package, send some emails!

Contents

1 Default behavior **2**
1.1 Default packages loaded 2

2 Class configuration **2**
2.1 Class options 2
2.2 Configuring `brandeis-problemset` after loading 4
2.3 Practical usage 5

3 User commands and environments **6**
3.1 General formatting commands 10

4 A class and a package **11**

5 Example **11**

6 Changelog **13**

*Brandeis University; rebeccaturner@brandeis.edu

1 Default behavior

`brandeis-problemset` provides packages and well-formatted constructs (notably the `problem` environment) for problem-set writers. `brandeis-problemset` will always render its body copy as a Times variant (`stix` for plain LaTeX or `xits` with XeLaTeX or LuaTeX) and always contains a useful header (which contains the page number, author's name, course, instructor, and assignment due date).

1.1 Default packages loaded

In general, `brandeis-problemset` will only load a feature if it's useful in more than one class; features like Gantt charts, Scheme code, assembly code, and so on, are loaded on demand.

1. `hyperref`, for a nicely-linked table of contents; `\href{url}{label}`.
2. `geometry` for page size and margins.
3. `enumitem` for better control over the margins and spacing of the `enumerate`, `itemize`, and `description` environments.
4. With any of the `listings`, `scheme`, `pseudocode`, or `assembly` options:
   (a) `listings`, for verbatim code listings (including the `assembly`, `java`, and `pseudocode` environments).
   (b) `xcolor`, for colored identifiers, strings, comments, and line numbers in listings; e.g. `\color{gray}`.
5. With the `math` option:
   (a) `amsmath` for tons of useful math commands, including `\text`, `\intertext`, and `\boxed` as well as the `bmatrix`, `multiline`, `gather`, `align`, and `alignat` environments. See "User's Guide for the amsmath Package" for a more complete listing.
   (b) `mathtools` for other useful/utilitarian commands.
   (c) With XeLaTeX or LuaTeX, `unicode-math` to allow equations to be copy-pasted.
   (d) The `stix2-otf` math fonts.
6. With the `tabu` option:
   (a) `multirow` for cells spanning multiple rows.
   (b) `booktabs` for beautiful-by-default tables and the `\cline` macro.
   (c) `tabu`, the best table package with dynamically resizable columns, easy creation of new column types, and more.

2 Class configuration

2.1 Class options

Class options are limited to configuration options which require the loading of fonts or other packages; "string" settings like the assignment's due date are configured with the `\bpsset` command.

All class options can be negated with `no`⟨*option*⟩; e.g. `scheme` can be disabled with `noscheme`. Later options override previous ones.

`antonella`

Only available in the `brandeis-problemset` document class. Use Dr. Antonella DiLillio's preferred styles (Courier for code)

`tabu`

Only available in the `brandeis-problemset` document class. Loads useful packages and defines commands for typesetting tables.

`math`

Only available in the `brandeis-problemset` document class. Loads utilities for typesetting mathematics. Because mathematics requires specialized fonts[1] as well as several other specialized packages,[2] this is disabled by default. Although I find myself needing mathematics fonts frequently, I know many students do not.

`listings` `true`

Enables code listings and enables default styles for colored keywords, pretty line numbers, and so on. While only this option enables the `java` environment, the `scheme`, `assembly`, and `pseudocode` options also load and style the `listings` package.

`scheme`

Enable Scheme language support, in particular for the R5RS dialect. Additionally, provides the `scheme` environment.

`pseudocode`

Enables the `pseudocode` environment, notably useful for COSI 21b (data structures).

`assembly`

Enables the `assembly` environment, notably useful for COSI 131a (operating systems).

`solution`

Show the content of `solution` environments; by default, they are excluded from compilation using the `comment` package.

`gantt`

Loads the `ganttschedule` environment.

`maketitle` `true`

Redefines `\maketitle` and defines `\maketitlepage` to include information about the course, instructor, assignment, and due date. If `maketitle` is set, then `option` is automatically enabled..

`header` `true`

Adds a header to the top of every page including information about the author, assignment, due date, and instructor. If `header` is set, then `option` is automatically enabled..

[1] The STIX2 math font that this package loads weighs in at 740kb!

[2] Including `amsmath`, `mathtools`, and, with XeLaTeX or LuaTeX, `unicode-math`.

config `true`

Enables document metadata like the assignment's due date, name/number, instructor, and course through the `\bpsset` command, which supports `\maketitle`, page headers, and so on.

2.1.1 Class options or keyval options?

There's trade-offs to be made either way; class options greatly simplify the code (we don't have to worry about loading packages and defining commands in more than one place, for the most part), but are much less flexible and difficult to interface with when the package loading is hidden. At the moment, package/class options provide commands and keyval options define document-configuration data (like names, due dates, and the like; see section 2.2 for more information).

2.2 Configuring `brandeis-problemset` after loading

`\bpsset{`⟨*options*⟩`}`

Sets global `brandeis-problemset` options. Mnemonic: **B**randeis **p**roblem **s**et **set**up (this should feel familiar to users of the `listings` package).

NOTE `\bpsset` was renamed from `\problemsetsetup` in version 0.5.0. The `\problemsetsetup` command is deprecated and will be removed in a future release.

course=⟨*course name*⟩

Course name in full.

coursenumber=⟨*course number*⟩

Course name shorthand; use `21a` for "COSI 21a".

assignment=⟨*assignment name*⟩

Assignment name in full.

number=⟨*problem set number*⟩

Assignment name shorthand; use `3` for "Problem Set 3".

duedate=⟨*due date*⟩

Due date, e.g. `2018-10-18`; not parsed at all, but ISO 8601 dates are highly recommended.

instructor=⟨*course instructor*⟩

Course instructor. With the `antonella` class-option, this is automatically set to `Dr.~Antonella DiLillio`.

author=⟨*your name*⟩

Alternate interface for the `\author` command.

date=⟨*document date*⟩

Alternate interface for the `\date` command.

codefont=⟨*fontspec font name*⟩

With XeLaTeX or LuaTeX, pass the given font to `\setmonofont` and enable Unicode shortcuts for the `pseudocode` environment. (If you need to specify options to `\setmonofont`, use `\setcodefont`.)

`\setcodefont[`⟨*fontspec options*⟩`]{`⟨*fontspec font name*⟩`}`

Sets the monospaced font to ⟨*fontspec font name*⟩ and uses it for shortcuts in the `pseudocode` environment.

2.3 Practical usage

You may find it useful to define a customized document class for each course. There's no reason to install these to some system-wide directory; it makes sense for them to live in the same directory as the problem set source files. For instance, `cosi21a.cls` might read:

```
\LoadClass[antonella, pseudocode]{brandeis-problemset}

% pass all unknown options to brandeis-problemset
\DeclareOption*{\PassOptionsToClass
    {\CurrentOption}{brandeis-problemset}}
\ProcessOptions\relax

% set course/author data
\bpsset{
    coursenumber=21a,
    author=Rebecca Turner,
}

% get a prettier code font -- these can be pretty big so they're not
% loaded by default
\setcodefont[
  Extension = .otf,
  UprightFont = *-Regular,
  BoldFont = *-Bold,
]{FiraMono}
```

and then `ps1.tex` might read:

```
\documentclass{cosi21a}
% stuff specific to this assignment
\bpsset{
    number=1,
    duedate=2018-10-29,
}
\begin{document}
% etc.
\end{document}
```

Note that you could use e.g. `\documentclass[math]{cosi21a}` to add a specific per-document option. See section 5 for a more complete example.

3 User commands and environments

`brandeis-problemset` provides a number of commands for typesetting problems.

`\begin{problem}[`⟨*options*⟩`]...\end{problem}`

Defines a problem. A problem is set 1 inch from the left margin (although this amount may be customized by modifying the `\problemindent` length) and begins a new page.

NOTE The `problem` and `subproblem` environments are typeset using `\section` and `\subsection` respectively, and as such can be customized with styling packages like `titlesec`; note, however, that they ignore and hide the usual section counters.

⟨*options*⟩ may include:

`title`=⟨*problem title*⟩

Displayed after "Problem" and the problem's number.

`number`=⟨*problem number*⟩

If given, the problem-number counter will not advance. The number must be robust, because it goes inside a `\section`.

`pagebreak`=⟨`true`|`false`⟩ `true`

Add a pagebreak before the problem?

`label`=⟨*problem label*⟩

Adds a custom label to the problem with `\label` that can be used with `\ref`. I recommend prefixing your problem labels with `p:` as in `p:big-o-proofs`.

`toc`=⟨`true`|`false`⟩ `true`

Include this problem in the table of contents?

`part`=⟨*part name*⟩

Indicates that this problem starts a new "part" of the assignment; actually calls `\part` under the hood.

`partlabel`=⟨*part label*⟩

Adds a custom label to this part in the same fashion as the `label` key.

Vertical material is allowed in a `problem`.

`\begin{subproblem}[`⟨*options*⟩`]...\end{subproblem}`

Defines a subproblem. A subproblem is set 1 inch from the left margin (although this amount may be customized by modifying the `\subproblemindent` length) and begins a new page. Valid ⟨*options*⟩ are identical to the `problem` environment's, with the following exceptions:

`pagebreak` `false`

False by default.

`part`

partlabel

Not available in the `subproblem` environment.

EXAMPLE Note that although these examples are too short to display them, vertical material — including listings — is allowed in the `problem` and `subproblem` environments.

Problem 1

This is a problem in an assignment.

Some solution here...

Problem 1.1

This is a subproblem.

```
\begin{problem}
  This is a problem in an
     assignment.
\end{problem}

Some solution here\dots

\begin{subproblem}
  This is a subproblem.
\end{subproblem}
```

`\begin{solution}...\end{solution}`

Defines a solution for a problem; a solution prints in blue and is excluded from the compiled document entirely unless the `solutions` package option is given.

In this way, the same `.tex` file can serve as both a postable assignment prompt and an answer key.

NOTE The style of solutions is customizable by redefining `\solutionstyle`; it's defined to `\color{blue}` by default.

EXAMPLE The default solution style is shown below; note, however, that this document is compiled with the `solution` option passed to `brandeis-problemset`; without it, the typeset solution is entirely blank.

Some solution here...

```
\begin{solution}
  Some solution here\dots
\end{solution}
```

`\Th[`⟨*column spec*⟩`]{`⟨*header text*⟩`}`

Typesets a table header in bold-face. ⟨*column spec*⟩ defaults to `l`. Useful for when a column is wrapped in a math environment; if you have a column `>{\ttfamily}l`, using `\Th` will not print the header in `\ttfamily`.

Server	**IP**
juice	1.1.1.1
dogs	2.2.2.2

```
\begin{tabular}{ll}
\Th{Server} & \Th{\textsc{ip}} \\
juice & 1.1.1.1 \\
dogs & 2.2.2.2 \\
\end{tabular}
```

`\begin{pseudocode}[`⟨*keywords*⟩`]...\end{pseudocode}`

Prints pseudocode.[3]

[3] Designed for COSI 21a as taught by Dr. Antonella DiLillo

Several "shortcuts," which replace a source-code sequence like `->` with a symbol like →, are shown in table 1.

These shortcuts display in `\pseudocodesymbolfont` (default: `\ttfamily`), which may be redefined if you prefer something else. The easiest way to change `\pseudocodesymbolfont` is with `\setcodefont`. If you use the `antonella` option with XeLaTeX or LuaTeX, `brandeis-problemset` will load `lm-math` and display the symbols seen in table 1, which look significantly better with Courier than `stix`' symbols.

The following words are treated as keywords in pseudocode, and will be bolded as appropriate: `Input`, `Output`, `Complexity`, `while`, `do`, `return`, `for`, `to`, `if`, `then`, `else`, `True`, `False`, `None`, `and`, `or`, `nil`, and `len`.

Table 1: Shortcuts provided by the `pseudocode` environment

Input	Command	Display	Codepoint
`<-`	`\pseudocodeleftarrow`	$\leftarrow$	U+2190
`->`	`\pseudocoderightarrow`	$\rightarrow$	U+2192
`(/)`	`\pseudocodeemptyset`	$\emptyset$	U+2205
`inf`	`\pseudocodeinfty`	∞	U+221E
`!=`	`\pseudocodene`	$\neq$	U+2260
`>=`	`\pseudocodege`	$\geq$	U+2265
`<=`	`\pseudocodele`	$\leq$	U+2264

EXAMPLE Note how the option `[Bar]` argument makes `Bar` appear bold like any other keyword in the typeset listing.

```
Bar(a, n)
    Input:  two integers, a and n
    Output: a^n
    k ← n  # k is a counter
    b ← ∞
    c ← a
    while k ≥ 0 do
        if k mod 2 = 0 then
            k ← k / 2
            c ← c * c
        else
            k ← k - 1
            b ← b * c
    return b
```

```
\begin{pseudocode}[Bar]
Bar(a, n)
    Input:  two integers, a and n
    Output: a^n
    k <- n  # k is a counter
    b <- inf
    c <- a
    while k >= 0 do
        if k mod 2 = 0 then
            k <- k / 2
            c <- c * c
        else
            k <- k - 1
            b <- b * c
    return b
\end{pseudocode}
```

`\begin{assembly}[`⟨*listings options*⟩`]...\end{assembly}`

Typesets assembly code.[4] Several considerations are taken into account; most notably, line numbers are printed as `x + n`, where n starts at 0 and counts by 4; the line number actually indicates the instruction's location in memory as an offset from the program start. Additionally, all valid instructions are treated as keywords and styled appropriately.

⟨*listings options*⟩ is passed directly to the `listings` package.

[4] Designed for COSI 131a as taught by Dr. Liuba Shrira

```
 x + 0  LOAD  R4, $200       ; sum addr
 x + 4  LOAD  R1, =0         ; sum
 x + 8  LOAD  R2, =0         ; i
x + 12  LOAD  R3, =0         ; j
x + 16  BR    OUTER          ; we know i < 10
x + 20  INNER:
x + 24  ADD   R1, R3         ; sum += j
x + 28  INC   R3             ; j++
x + 32  OUTER:
x + 36  BLT   R3, R2, INNER  ; while j < i goto inner
x + 40  INC   R2             ; i++
x + 44  LOAD  R3, =0         ; j = 0
x + 48  BLT   R2, =10, OUTER ; while i < 10 goto outer
x + 52  STORE R1, @R4        ; store sum
x + 56  HALT
```

```
\begin{assembly}
LOAD  R4, $200       ; sum addr
LOAD  R1, =0         ; sum
LOAD  R2, =0         ; i
LOAD  R3, =0         ; j
BR    OUTER          ; we know i < 10
INNER:
ADD   R1, R3         ; sum += j
INC   R3             ; j++
OUTER:
BLT   R3, R2, INNER  ; while j < i goto inner
INC   R2             ; i++
LOAD  R3, =0         ; j = 0
BLT   R2, =10, OUTER ; while i < 10 goto
    outer
STORE R1, @R4        ; store sum
HALT
\end{assembly}
```

`\begin{java}[`⟨*listings options*⟩`]...\end{java}`

Tragically-common shorthand environment for a listing of Java code.
⟨*listings options*⟩ is passed directly to the `listings` package.

`\begin{scheme}[`⟨*listings options*⟩`]...\end{scheme}`

Shorthand environment for a listing of Scheme code, useful for COSI 121b. Requires the `scheme` package option to be loaded.
⟨*listings options*⟩ is passed directly to the `listings` package.

`\begin{ganttschedule}[`⟨*total cell count*⟩`]...\end{ganttschedule}`

An environment for drawing Gantt charts indicating process scheduling. The mandatory argument indicates how small the grid should be; `19` subdivides the line into 19 cells.
To use the `ganttschedule` environment, make sure to use the `gantt` package option.
Within a `ganttschedule`, use the `\burst` command to indicate an active process (i.e. a process burst).

NOTE The charts `ganttschedule` draws aren't actually really proper Gantt charts, which can indicate parallel activities; however, that's what Liuba calls them, so that's what they're called here.

`\pid{`⟨*pid*⟩`}{`⟨*burst length*⟩`}`

Draw a burst for process ⟨*pid*⟩ of time length ⟨*burst length*⟩.

NOTE The Gantt chart packages (notably `tikz`) don't play nicely with `ltxguidex`, the package this documentation is written in; for a typeset example, see section 5.

```
1  \begin{ganttschedule}{19}
2      \pid{2}{1}
```

```
    \pid{4}{1}
    \pid{3}{2}
    \pid{5}{5}
    \pid{1}{10}
\end{ganttschedule}
```

NOTE Because `ganttschedule` relies on `tikz`, `fp`, and `calc`, it can add significantly to document compile times. If you intend to use the `ganttschedule` environment, make sure to use the `gantt` class option or set `gantt` in `\bpsset`. If you fail to include the `gantt` option, you will see an error message:

```
! Package brandeis-problemset Error: ganttschedule enviornment
   not loaded in preamble.

See the brandeis-problemset package documentation for
   explanation.
Type  H <return>  for immediate help.
l.4 \burst
    {1}{1}
? H
Did you mean to use the 'gantt' option for the
   brandeis-problemset document class?
```

3.1 General formatting commands

`\newacronym[`⟨*command*⟩`]{`⟨*acronym text*⟩`}`
`\newacronyms{`⟨*acronym list*⟩`}`

Creates a new acronym. If ⟨*command*⟩ isn't given, the text of the macro will be used instead; `\newacronym{cfg}` would define a command `\cfg` which typesets as "CFG". If the resulting command already exists, it will be redefined. For `\newacronyms`, the ⟨*acronym list*⟩ is a comma-delimited list of acronyms.

A CFG describes a context-free language…
The SPARC had a unique CPU…

```
\newacronym[\xyz]{cfg}
A \xyz\ describes a context-free
   language\dots

\newacronyms{sparc, cpu}
The \sparc\ had a unique
   \cpu\dots
```

`\ac{`⟨*acronym*⟩`}`

Typesets an acronym. The ⟨*acronym*⟩ should be lowercase (e.g. `\ac{cpu}` rather than `\ac{CPU}`). Currently, `\ac` simply delegates to `\textsc`. In the future, I'd like to support a bit of letterspacing; "for abbreviations and acronyms in the midst of normal text, use spaced small caps."[5]

[5] *The Elements of Typographic Style* by Robert Bringhurst, 2nd. ed, § 3.2.2

`\Sc{⟨text⟩}`

An abbreviation for `\textsc`.

`\Rm{⟨text⟩}`

An abbreviation for `\textrm`.

`\Up{⟨text⟩}`

An abbreviation for `\textup`.

`\Bf{⟨text⟩}`

An abbreviation for `\textbf`.

`\It{⟨text⟩}`

An abbreviation for `\textit`.

`\Tt{⟨text⟩}`

An abbreviation for `\texttt`.

4 A class and a package

As a user, you'll likely only need the `brandeis-problemset` document class. However, a *package* named `brandeis-problemset` is also provided. The class styles an entire document, while the package only provides commands. This allows — for example — loading the package for the examples in this document without messing up our titles, headers, and so on.

5 Example

A brief example usage of `brandeis-problemset` follows. For a longer, more in-depth example, see `example.tex` in the `brandeis-problemset` repository.

```
\documentclass[gantt]{brandeis-problemset}
\author{Rebecca Turner}
\bpsset{
    coursenumber=21a,
    instructor=Dr.\ Liuba Shrira,
    duedate=2018-10-20,
    number=3,
}
\newcommand{\io}{\ac{io}}
\newcommand{\cpu}{\ac{cpu}}
\begin{document}
```

```
\begin{problem}
    Write an assembly program!
\end{problem}

\begin{assembly}
        LOAD  R1, $200      ; A = (program location) + 200
        LOAD  R2, =1        ; i = 1
\end{assembly}

\begin{problem}
    What does this algorithm do? Analyze its worst-case running time
       and
    express it using big-O notation.

\begin{pseudocode}[Foo]
Foo(a, n)
    Input:  two integers, a and n
    Output: a^n
    k <- 0
    b <- 1
    while k < n do
        k <- k + 1
        b <- b * a
    return b
\end{pseudocode}
\end{problem}

$\Rm{Foo}(a, n)$ computes $a^n$, and will run in $O(n)$ time always.

\begin{problem}[number=5.4]
    Consider the following set of processes, with the length of the
    \cpu\ burst given in milliseconds:

    \begin{center}
        \begin{tabu} to 0.25\linewidth{X[1,$]rr}
            \Th{Process} & \Th{Burst time} & \Th{Priority} \\
            P_1 & 10 & 3 \\
            P_2 & 1 & 1 \\
            P_3 & 2 & 3 \\
            P_4 & 1 & 4 \\
            P_5 & 5 & 2 \\
        \end{tabu}
    \end{center}%$

    Draw a Gantt chart to illustrate the execution of these processes
    using the \ac{sjf} scheduling algorith.
\end{problem}

\begin{ganttschedule}{19}
    \pid{2}{1}
    \pid{4}{1}
```

```
    \pid{3}{2}
    \pid{5}{5}
    \pid{1}{10}
\end{ganttschedule}
\end{document}
```

6 Changelog

Coming soon... Rebecca Turner

Added

- Support for COSI 130b, including regular expressions and context-free grammars.

0.5.5 Rebecca Turner (2019-04-10) — Reordered package loading to prevent `mathtools` from breaking `\overbrace` and `\underbrace` commands.

0.5.4 Rebecca Turner (2019-04-02) — Fixed hanging indent in `description`-lists; the hanging indent was accidentally removed through a tweak to hang the list item markers for the `itemize` and `enumerate` lists.

0.5.3 Rebecca Turner (2019-03-30) — Commands such as `\texttt` won't automatically cause errors when used in the optional arguments of the `problem` and `subproblem` environments. More complex commands may still cause issues, however.

0.5.2 Rebecca Turner (2019-03-13) — Scheme code highlighting erroneously highlighted `cbr`, `cabr`, etc. rather than `cdr`,`cadr`, etc. These keywords have been renamed.

0.5.1 Rebecca Turner (2019-03-09) — Distribution erroneously excluded `brandeis-problemset.sty`.

0.5.0 Rebecca Turner (2019-03-06)

Added

- `\newacronym` and `\newacronyms` commands.
- Added `toc` option to the `problem` and `subproblem` environments.

Changed

- Cleaned up internals; improved option system, split `brandeis-problemset` into a class and a package, renamed commands and lengths to use the `bps@` prefix more consistently.
- `brandeis-problemset` now has a modular and much more conservative approach; far fewer packages are loaded and features are loaded only upon request to a much greater extent.
- `subproblem` is now an environment, not a command, with an interface matching the `problem` environment.
- The `assembly` and `pseudocode` options now define languages for the `listings` package, rather than just providing environments to use the languages.

Removed

- The following commands have been removed in favor of the `\bpsset` command, which encompasses their functionality entirely: `\duedate{`⟨*date*⟩`}`, `\instructor{`⟨*name*⟩`}`, `\course{`⟨*name*⟩`}`, `\coursenumber{`⟨*number*⟩`}`, `\assignment{`⟨*name*⟩`}`, and `\problemsetnumber{`⟨*number*⟩`}`.
- Removed dependencies: `environ` and `titlesec`. No longer unconditionally loaded: `hyperref`, `xcolor`, `comment`, `listings`, `multirow`, `booktabs`, `longtable`, `tabu`.

Deprecated

- The `\problemsetsetup` command, which has been renamed `\bpsset`.

0.4.4 Rebecca Turner (2019-02-14)

Changed

- Changed Times body copy font from `tex-gyre`'s Termes to the newer `stix2-otf` (for XeLaTeX or LuaTeX) and `stix2-type1` (for other TeX engines) — the STIX2 fonts are somewhat unique amongst Times-likes in that they contain small caps.
- Redefined `\Re` to print in blackboard-bold.

0.4.3 Rebecca Turner (2019-01-20) — Fixed typos in license file, fixed distributed documentation `.pdf`.

0.4.2 Rebecca Turner (2019-01-19)

Added

- `author` and `date` keys added to `\problemsetsetup` to simplify class-wide configuration.

Fixed

- Fixed definitions for `\duedate`, `\instructor`, etc. to avoid spurious errors due to undefined commands.

Changed

- Translated documentation to the new `ltxguidex` document class for added beauty.
- Re-licensed `brandeis-problemset` to the LPPL v1.3c for easy transfer of maintenence in the future.

0.4.1 Rebecca Turner (2019-01-03) — Updated `scheme` environment to properly recognize all primitive functions, added syntax coloring to all code.

0.4.0 Rebecca Turner (2018-12-20)

Added

- `solution` environment and `solutions` class option.
- `scheme` shorthand environment and `scheme` class option.

Fixed

- Boolean class options being overwritten by keys defined for `\problemsetsetup`.
- Title-formatting errors

Removed

- Assignment- and course-specific class options `duedate`, `assignment`, `instructor`, and `course`. These settings should be configured with either `\problemsetsetup` or their specific commands. (`\duedate`, `\instructor`, etc.).

0.3.0 Rebecca Turner (2018-10-24)

Added

- This changelog.
- Support for `\parts` and referencing problems.
- Options to `problem` environment: `part`, `label`, and `partlabel`.
- `\maketitle` (contrast with `\maketitlepage`).

0.2.0 Rebecca Turner (2018-10-20)

Changed

- Class renamed to from `problemset` to `brandeis-problemset`.

Added

- A license header.
- `ganttschedule` environment.
- Additional keywords for `pseudocode` environment: `and`, `or`, `nil`, and `len`.
- `\ac` command for acronyms.
- An example document.

0.1.0 Rebecca Turner (2018-10-19) — Initial beta as `problemset`.