# How to Use the notes Class

**Yale University, Summer 2018**

*Taught by Jackson Petty.*

## Contents

# 1 Introduction

The `notes` class is designed to be a simple, adaptable document class to aid in typesetting lecture notes. It can be used by students who wish to transcribe lectures, instructors who wish to create an outline of the material they wish to cover, or by speakers who want to create a handout to accompany their lecture. The development of the class arose out of my need to easily and consistently format a wide variety of my notes that I took as a student at Yale University. As such, the features best reflect the needs of a student who is enrolled *primarily* in science and mathematics courses, but who also takes classes in the humanities and social sciences.

This class is closely complimented by the `pset` class, which is designed to format problem sets in a very similar manner to the `notes` class, while providing a more customized interface designed for science and mathematics homework assignments.

# 2 Formatting

## 2.1 The Title Page

The `notes` class comes with an inline title page. On the first page of the document, there are four primary lines. The first line contains the title of the document, set by the `\title{}` command in the document preamble. The second line is for the subtitle of the document, set by the `\subtitle{}` command. By default, the subtitle of the document is set to be `CourseID, Place, Term Year`, where the fields are set by the `\courseid{}`, `\place{}`, `\term{}`, and `\year{}` commands, respectively. The third line is the attribution line, where the instructor and scribe can be displayed. You can set thse values with the `\speaker{}` and `\sribe{}` commands. There are also commands which allow you to specify the email addresses of the speaker and scribe, which will appear in the footer of the title page if set. Use the `\speakeremail{}` and `\scribeemail{}` commands to do this.

## 2.2 Fonts

All body text is set, by default, in the Libertine font family. If you want to use the more traditional Computer Modern fonts, simply pass the `[cmu]` option to the class at the be-

ginning of the document preamble. If you want to use a completely different font family altogether, either load the package you wish to use (if using `pdflatex`) or use the `\setmainfont{}` command (if using `xelatex` or `lualatex`).

By default, the math fonts are left unchanged. This is entirely because of personal preference. I think that the Computer Modern fonts are much, much better than almost all other fonts when it comes to typesetting mathematics. I also like the visual distinction between "math" and "text" that this achieves when using a non-Computer Modern font for the body text. However, if you wish to change this, load any package you want in the document preamble to change the math font.

### 2.3 Notes

You'll probably have noticed by now that the pages are asymmetric; this is intentional, and the wider margin on the right hand side of the page is to allow notes to be placed there, next to the referring text. This makes it easier to see what a note is referencing rather than having to move your eyes to the bottom of the page and back up again. If you want to write a quick margin note, just use the `\note{}` command. You can also use the `\footnote{}` command, which will add a footnote marker to the referencing text and to the beginning of the footnote[1].

*It's super easy to write yourself notes!*

[1] *This is a footnote!*

## 3 Mathematics

### 3.1 Theorem-like Environments

The cornerstone of mathematics is the *theorem*. Along with proofs and definitions, these chunks of text often require special formatting to differentiate them from the surrounding text. `notes` groups these environments into several different *styles*, which reflect both the semantic meaning of each environment and the degree to which the text should be distinct from regular body text.

#### 3.1.1 Proofs and Solutions

The simplest kind of special environment is the proof. It represents a succinct answer to a specific problem, and often is very closely related to the body text of the document. As such, the styling of the `proof` environment is very similar to regular text; it contains an italic head, and is ended with a QED marker which is by default a black square.

*Proof.* I think, therefore I am. ■

By default, the head of the `proof` environment is the word *Proof* in italics, but you can

specify an optional argument which allows you to rename to proof to add specificity.

*Proof of Problem 1.* This follows trivially from Lemma 2.3. ■

`notes` also provides the `solution` environment, which is identical to the `proof` environment but has the word *Solution* in italics in the head. You could of course recreate this with the optional argument to the `proof` environment (that in fact is how it is defined), but this saves time and keystrokes.

*Solution.* Simply integrate the expression. ■

### 3.1.2 Theorems, Lemmas, Corollaries, and Conjectures

The *theorem* style is similar to the *proof* style, but a bit more visually distinct. The head is bold, and the body of the theorem is italic. This helps to differentiate a theorem from a preceding proof, or from surrounding text. *Theorem* style environments are also uniquely numbered, so you can reference the theorem later on.

**Theorem 1** (Banach Fixed-Point)**.** *Let $(X, d)$ be a non-empty complete metric space and let $f : X \to X$ be a contraction. Then there exists a unique fixed point $x_0 \in X$ such that $f(x_0) = x_0$.*

*Proof.* Let's consider the following scenario, without a loss of generality. ■

In addition to the `theorem` environment, the `lemma`, `corollary`, and `conjecture` environments use *theorem* style formatting, and each are separately numbered.

**Corollary 1.** *Now that we've proven Theorem 1 we can make the following note.*

**Lemma 1.** *This is a lemma; it's useful to break up your arguments into lemmas to make them easier to manage.*

**Conjecture 1.** *I'll bet the moon is made of cheese.*

### 3.1.3 Definition and Notation

One step up from the *theorem* style environments is the *definition* style. These environments need to be visually distinct enough to stand out on a page as it is scrolling by or being flipped though so that you can quickly refer to them at a glance. *Definition* style environments are indented by $1\,\mathrm{cm}$ to preserve the visual distinction from surrounding text.

> **Definition 1** (Group)**.** A *group* is an ordered pair $(S, \star)$, where $S$ is a set and $\star$ is a binary operation on $S$.

The `notes` class also provides the `notation` and `abuse` environments to represent *Notation* and *Abuse of Notation*, respectively, and which follow the *definition* style.

> **Notation** ($\hookrightarrow$)**.** Given a map $f$, $f : A \hookrightarrow B$ means that $f$ is *injective* from $A$ into $B$.

> **Abuse of Notation.** Consider that $\sin^2(x) = (\sin x)^2$ while $\sin^{-1}(x) = \arcsin x$. What does $\sin^{-2}(x)$ mean? Who knows.

### 3.1.4 Problems and Exercises

Although more useful in the `pset` class, the `problem` and `exercise` environments provide an easy way to call attention to problems which should be completed. These are very distinct from the surrounding text, and so are boxed with a light grey background. It is often useful for them to contain the `parts` environment, to specify multi-step problems. This introduces the `\part[]` command, which takes an optional parameter which acts as a reference label so you can cite the part later in the document. In essence, `\part[pt:4:a] ...` is short for `\part\label{pt:4:a} ...`

**Problem 1.** Find the square root of 144.

*Solution.* The square root of 144 is 12. ■

**Problem 2** (Taylor)**.**

(a) This is the first part.

(b) This is the second part.

**Exercise 3.** Find some fossils, because fossils are cool.

### 3.1.5 Examples

The most visually distinct environment is the `example` environment. This is almost identical to the `problem` environment except that the background is gold instead of grey and there is a line break after the head.

**Example 1.**
Puffins are a type of bird.

### 3.2 Math Commands

The `notes` class also provides several commands for commonly used mathematical expressions, such as blackboard bold fonts for sets and fields.

| | | |
|---|---|---|
| $\mathbb{R}$ | `\R` | Real Numbers |
| $\mathbb{C}$ | `\C` | Complex Numbers |
| $\mathbb{Z}$ | `\Z` | Integers |
| $\mathbb{Q}$ | `\Q` | Rational Numbers |
| $\mathbb{N}$ | `\N` | Natural Numbers |
| $\mathbb{F}$ | `\F` | Generic Fields |

## 4 Chemistry

The `notes` class loads the `mhchem` package to aid in the typesetting of chemical formulae.

$$\mathrm{CO_2 + C \longrightarrow 2\,CO}$$

It also loads the `chemfig` package to aid in the creation of chemical structure diagrams. To enable this feature, you must pass the `[diagram]` option to the class at the beginning of your document preamble.

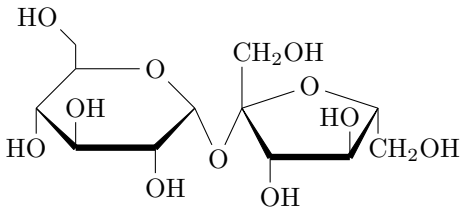


Figure 1: A chemical structure diagram.

## 5 Physics

The `notes` class loads the `physics` class, which provides many useful document commands for formatting physics notation. Supplementing this is the `\unit{}` command,

which formats a given symbol as a unit vector by setting the symbol in a bold font and adding a hat above it.

$$\texttt{\textbackslash unit\{v\}} \to \hat{\mathbf{v}}, \qquad \texttt{\textbackslash unit\{\textbackslash alpha\}} \to \hat{\boldsymbol{\alpha}}$$

## 5.1 Units

The `notes` class also loads the `siunitx` package to aid in the automatic formatting of units. It also defines the following custom non-SI units which still prove useful in physics and engineering.

| | |
|---|---|
| mi | `\mile` |
| gallon | `\gallon` |
| lb | `\pound` |
| atm | `\atmosphere` |
| °F | `\fahrenheit` |
| at | `\atom` |
| molecule | `\molecule` |
| cal | `\calorie` |
| Cal | `\Calorie` |
| in | `\inch` |

## 5.2 Feynman Diagrams

The `notes` class loads the `tikz-feynman` package to aid in the typesetting of Feynman diagrams. In order to load this package, you must pass the `[diagram]` option to the class at the beginning of your document preamble.

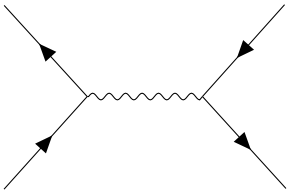

Figure 2: A Feynman diagram showing a photon exchange between two fermion.

Note that in order to get almost any use of this package, you must compile your document with `lualatex`.

### 5.3 Circuits

The `notes` class also loads the `circuitikz` package to aid in the typesetting of circuits. In order to load this package, you must pass the `[diagram]` option to the class at the beginning of your document preamble.

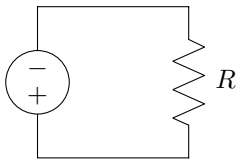


Figure 3: A simple circuit.

Unlike with Feynman diagrams, the `circuitikz` package is fully compatible with all LaTeX engines, so there are no restrictions on how you compile your document. The `notes` class loads `circuitikz` with the `[american]` and `[siunitx]` options. It also makes the line width of all bipole components equal to the width of the 'wires', so there is no visual disparity between the two.

## 6 Linguistics

The `notes` class provides several commands to help you typeset linguistics notation. However, it is *highly* recommended that you compile any such documents with `xelatex` or `lualatex`, which will provide native support for all IPA characters, as well as allow you to specify separate fonts for IPA characters.

### 6.1 IPA Fonts

If you compile your documents with `xelatex` or `lualatex`, the `notes` class will define a font switch named `\ipafont`, which is by default the same as the body text — in this document, the `\ipafont` has been set to Charis SIL. If you wish to change this to a different font, simply use the `\setipafont{}` command. In order to typeset a block of text in the `\ipafont`, simply use one of the following methods.

*Charis SIL and Doulos SIL are both fully IPA compatible choices*

`{\ipafont 'lɑːtɛx}` → ˈlɑːtɛx

`\ipa{'lɑːtɛx}` → ˈlɑːtɛx

Notice that, with `xelatex` or `lualatex`, you can input the IPA symbol directly as a character. This makes documents filled with such symbols much easier to read and write.

### 6.2 Phonemes, Allophones, and Orthography

In many cases, linguists must distinguist between a particular character representing a phoneme, an allophone, or whether the character is an orhtographic symbol. The `\phon{}`, `\allo{}`, and `\orth{}` commands make this easy.

`\phon{a}` → /a/

`\allo{a}` → [a]

`\orth{a}` → ⟨a⟩

These commands automatically typeset any argument text in the `\ipafont`.

### 6.3 Linguistic Rules

It is often useful to document sound changes. When these changes are conditional, it is common to separate the sound change from the condition with a large slash. The `notes` class defines the `\where` command to fill such a need. Consider the change where a reconstructed /*m/ becomes a /ɱ/ when it is word-initial and prevocalic. This would be written succinctly as /*m/ → /ɱ/ / #_V, which is produced by typing `\phon{*m} \to ~\phon{ɱ} \where~$\#$\_V`.

## 7 Computer Science

### 7.1 Source Code

The `notes` class uses the `minted` package to allow for the typesetting of source code. This package requires that the `pygments` Python package be installed on your system. It also requries you to compile your document using the `-shell-escape` flag, which allows the TEX engine to run arbitrary code. In order to use this feature, you must pass the `[code]` option to the class at the beginning of the document preamble. If you are compiling your document with `xelatex`, it is also recommended that you use the `-8bit` flag when compiling, as there is an open bug which causes tab literals to be misprinted; if you use tabs for indentation in your source code, this can cause very visible problems unless the `-8bit` flag is used.

### 7.2 Algorithms

The `notes` class also supports typesetting algorithms by loading the `algorithm` and `algpseudocode` packages. In order to use these features, you must pass the `[code]` option to the class at the beginning of the document preamble.