CLP-2 Lecture Slides

The slides in this folder are written to correspond to CLP-2 by Feldman, Rechnitzer, and Yeager. The slides and their source files can be found online at https://github.com/ecyeager/CLP2Slides.

VERSIONS

Most slides have three versions: *lecture*, *handout*, and *full*.

lecture These slides are intended for display and annotation by the instructor during lecture. They include animations, pauses, and space to write solutions to questions. Solutions themselves are generally omitted.

handout The handout mode is used to generate slides that students can access before lecture and use to take notes. Most animations and pauses are suppressed. Solutions to questions are also suppressed, so students can work in class without spoilers.

full These slides are similar to lecture slides, but in addition to including space to write your own solution, they also contain typed solutions. They may be posted to students after class; they may be used by lecturers preparing class; and they may be used during class instead of the 'lecture' version.

Organization

PDFs can be found in the `pdfs` folder, while the folder `src` has their source files.

Slides for a single section are compiled in files named with the section, a short description, and the version type. (For example, `1_2Properties_Handout.pdf` is the handout version of Section 1.2, Basic properties of the definite integral.)

Miscellaneous Files

The file `StudentWorkbook.pdf` has slides for the whole course, with answers removed, formatted for printing. Students who wish to write on paper (rather than annotating PDFs on a tablet) can print out the file and have a workbook to follow for the entire semester.

The file `NotesOnly.pdf` has extra notes to the instructor about the various sections.

Compiling Options

There are many files, so we included several options for compiling them efficiently. The files can be compiled using pdflatex (in the usual way you probably compile LaTeX); using a Python script to compile many files in one go; and using a Python script with parallel processing and latexmk to quickly compile many files in one go.

Compiling Without Python

In the folder `src`, you'll find a list of .tex files corresponding to each section, StudentWorkbook, NotesOnly, and ReadMe.

To change the version of a section file (lecture, handout, full), change which headers are commented out in the .tex file.

Individual .tex files are compiled with pdflatex. From the `src` folder, `pdflatex -output-directory=../pdfs <filename>` will put the PDF into the folder `pdfs`.

Compiling With Python

Running `buildPDFs_parallel.py` or `buildPDFs_slow.py` will compile the slides and sort the PDFs into their directory. By default, they will compile all versions of all files. This may take a long time: each section is included in lecture, handout, and full versions of single sections, as well as the student workbook file, so most sections are compiled 4 times.

BUILDPDFS_SLOW.PY VS. BUILDPDFS_PARALLEL.PY

`buildPDFs_parallel.py` uses parallel processing to speed things up. It also uses latexmk, which requires Perl to be installed. The parallel file has the option to delete auxiliary files. You only need to run `buildPDFs_parallel.py` once.

`buildPDFs_slow.py` compiles files one-at-a-time using pdflatex. It is slower, but might work out-of-the-box on more machines – for example, Windows machines without Perl installed. The repository does not track auxiliary files, so using this option you will initially have to compile twice to generate the navigation bars in the headers and to make the internal references work.

Compiling With Python

Both buildPDF files start with options to specify which files should be compiled. Set each variable to True or False in the .py file.

variable	**value**	**behaviour**
`print_lecture`	True	print lecture versions of slides
	False	do not print lecture versions
`print_handout`	True	print handout versions of slides
	False	do not print handout versions
`print_full`	True	print full versions of slides
	False	do not print full versions

variable	value	behaviour
`print_sections`	False	print no single sections
	True	print all single sections from the list
`print_readme`,	False	do not print the corresponding file (ReadMe, NotesOnly, or StudentWorkbook)
`print_notesonly`, `print_studentworkbook`	True	print the corresponding file
`delete_aux` (in parallel file only)	False	all auxiliary files (.aux, .log, .toc, .nav, etc.) are kept
	True	all auxiliary files that are generated when the program runs will be deleted.

Below the compilation choices are lists of sections. These are lists of exactly which sections to compile. If you would like to omit some (but not all) of these files, comment them out of the list.

If you have already set `print_sections=False`, there is no need to comment out the sections. None of them will be compiled.

There are no further customization options after the lists of sections.

A Note on Repetition

The author has observed in her 100-level classes the usefulness of repeated concrete examples. At this stage in their development, many students have a hard time applying theorems in the way mathematicians usually want to state them, but are adept at noticing patterns.

Active Learning

A common technique in introductory classes is to provide periodic opportunities for students to engage on their own (and with their neighbours) with the material.

There will sometimes be several questions visible on the same slide. In class, the author encourages students to work on the first one or two questions, and to move on to the others only if they are "super speedy" and finish the target question(s) early. This balances out different student needs. Those who need more time can have it, because those who need less time are still able to stay engaged.

Posting Notes

What exactly to give students access to is a very personal decision. We do not know the best solution. Students who do not have access to lecture notes may be more motivated to synthesize content in their own words. Students who do have access to lecture notes may feel less pressure during class.

The author's personal practice in 100-level courses is to give students a handout without solutions to write on in class, and post the full version with solutions online after class. Providing the handout means that students don't spend time and energy copying down things like definitions that are easily found in the textbook. Withholding solutions during class enables the active learning described on the previous slide. Letting students know that solutions will be posted later encourages students to copy down only what seems important to them in the moment, rather than trying to reproduce every line exactly.

Writing out computations

In the author's personal experience with 100-level courses, writing out computations in real time is generally preferable to displaying finished computations and then talking about them. We have taken pains to provide enough space on the slides to at least start a computation, but since many computations may take more than one page, using a PDF annotating software that can quickly provide blank pages is helpful.

For instructors who would like to display finished computations, the full versions of the slides may be used in class. These contain the same animations and pauses that the lecture version does. (They also include extra space to write. This gives you the flexibility to write solutions on your own sometimes, and display solutions sometimes, according to your preference and ability.)

ATTRIBUTION

Attribution of included images information is generally found at the end of each file. Most images are sourced from https://thenounproject.com/. While the author has paid for a license that does not require attribution, attribution is nonetheless included for anyone who would re-use the images.

Lecturer Cues

The lower right-hand corner of the slides holds icons that will be useful for the lecturer but that are not necessary for students to pay attention to. These are meant to be relatively unobtrusive. They generally do not appear in handout mode.

This slide has a box letting you know that the current question has its answer written on the next slide. If you click forward, you may give it away.

No answer

This slide has the opposite cue: the answer to the current question does *not* appear on the slides.

More space

The notebook icon in the corner tells you that there is more room on the next slide. This is used when a slide has a question with little room after it to work.

Usually, the question is sharing the slides with a theorem or a similar example. You can see the problem together with the other content in order to talk about the problem before you write anything down, or to give students a chance to start working on their own. When it's time to work the problem on screen, you can advance to the next slide, hiding the extra information to open up space.

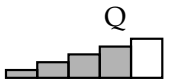
No more space

By contrast, the crossed-out notebook icon on this page indicates that there will not be extra space to write your work on the slides.

Question Set

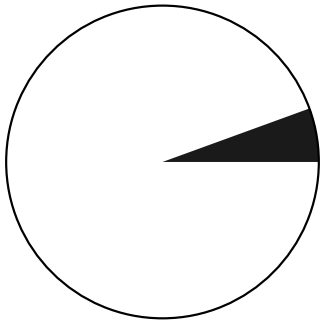
The staircase status bar is used for sets of related questions. The status bar on this slide shows the 4th of 5 questions in a set.

The slides are designed to have slightly more questions than would probably be covered in class. This prevents lectures from running short, and provides extra practice for students who want it. The author does not generally present every single problem in a file.



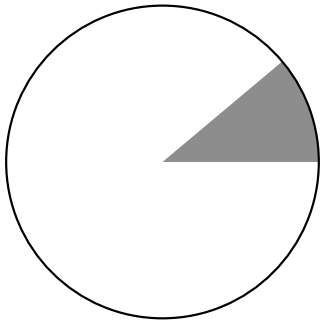
Status Bar

These frames have a flipbook-style animation. The status bar in the corner tells you where you are in that flipbook, so you know when to stop clicking.



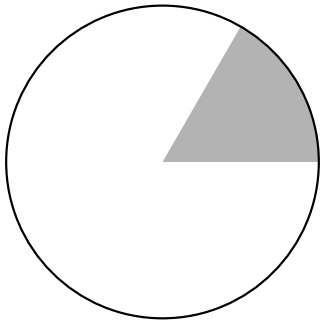
Status Bar

These frames have a flipbook-style animation. The status bar in the corner tells you where you are in that flipbook, so you know when to stop clicking.



Status Bar

These frames have a flipbook-style animation. The status bar in the corner tells you where you are in that flipbook, so you know when to stop clicking.



Unobtrusive Notes

References to the text are sometimes put next to the page number. Content on the slides is similar, but usualy not identical, to the reference. The references can help you track which part of the text corresponds to the current slide.

Keen students can investigate the references in the text if they want a longer written explanation, or a similar example to practice on. For that reason, these show up the handouts as well as in beamer mode.

Text in this colour in the full or lecture versions of the slides does not show up in the handout version. So, this colour signals that students will only see this text on your screen, and not on their redacted copies.

Sometimes things that look like answers are included in the student handouts. For example, if a question is only included to motivate a theorem, its solution might be included for them, because the question serves more as exposition than as practice. These would be coloured normally.

If you just want to use the slides as they are, you can stop here.

The remaining sections describe how to use the custom macros, and are only necessary if you intend to edit the slides.

VERSIONS

The three versions described in the first section are generated using beamer modes and the spoilers toggle (discussed later).

version	**mode**	**spoilers**
lecture	beamer	false
handout	handout	false
full	beamer	true

The usual overlay specifications for modes, e.g. `\onslide<1-3|handout:0>{content}`, will suffice to differentiate between handout and the other two versions.

SPOILERS

You can set the toggle ‘spoilers’ to true or false in the preamble. This controls content that might be an unwanted hint for students thinking about an example on their own. Full solutions also tagged as spoilers.

```
\settoggle{spoilers}{true}
\settoggle{spoilers}{false}
```

The toggle is automatically set to false for lecture and handout versions, and to true for full versions.

Spoilers in overlays

There are versions of `\only`, `\onslide`, and `\alert` that have spoiler-specific behaviour.

`\sonly` ('spoilers-only') works like `\only`, but will only show content if the spoilers tag is true, and colour it.

`\nsonly` ('no-spoilers-only') works like `\only`, but will only show content if the spoilers tag is false.

`\sonslide` ('spoilers-onslide') works like `\onslide`, but will only show content if the spoilers tag is true, andcolour it.

`\nsonslide` ('no-spoilers-onslide') works like `\onslide`, but will only show content if the spoilers tag is false.

`\salert` ('spoilers-alert') works like `\alert`, but will only show content if the spoilers tag is true.

More complicated overlays

The macros on the previous slide can be used with overlay specifications as usual: for example, you may write `\sonly<1-3>{content}`.

To differentiate between all three versions, there are macros that take the overlay specifications as mandatory arguments. You may *not* use the usual overlay syntax (with angle brackets) with these.

`\snshonly{`*a*`}{`*b*`}{`*c*`}{content}` ('spoiler/no spoiler/handout only') will print 'content' on slide(s) *a* in full version, on slide(s) *b* in lecture version, and on slide(s) *c* in handout version. (To suppress it from one version entirely, ask it to print on slide 0.)

`\snshonslide{`*a*`}{`*b*`}{`*c*`}{content}` behaves similarly, but as `onslide` instead of `only`.

VERSIONING AND VISUAL CUES

Visual cues are sensitive to mode and the spoiler toggle. They may be used with overlay specifications, e.g. `\AnswerYes<1>` or `\QuestionBar<1-2>{3}{5}`.

The prefixes `s` and `ns` will cause them to require spoilers to be true or false, respectively, to be displayed. (See upcoming slides for details.) The modes vary by icon, and were chosen to best fit the context in which they are used. (For example, since handouts don't contain solutions, they never contain 'AnswerYes' marks. So, 'AnswerYes' is only ever printed in beamer mode, not handout mode.)

VERSIONING AND VISUAL CUES

The tables below details all commands that take ‘s’ and ‘ns’ prefixes, and when each results in a visible mark.

command	mode	spoiler	versions
\AnswerYes \AnswerNo	beamer	true	full
\nsAnswerYes \nsAnswerNo	beamer	false	lecture
\MoreSpace \NoSpace	beamer	any	lecture, full
\sMoreSpace \sNoSpace	beamer	true	full
\nsMoreSpace \nsNoSpace	beamer	false	lecture

VERSIONING AND VISUAL CUES

The tables below details all commands that take ‘s’ and ‘ns’ prefixes, and when each results in a visible mark.

command	mode	spoiler	versions
\StatusBar	beamer	any	lecture, full
\sStatusBar	beamer	true	full
\nsStatusBar	beamer	false	lecture
\QuestionBar	any	any	lecture, full, handout
\sQuestionBar	any	true	full
\nsQuestionBar	any	false	lecture, handout
\AnswerBar	beamer	true	full
\nsAnswerBar	beamer	false	lecture

Sets of Questions

Sometimes there are sets of questions on a particular topic. You can put these inside a `QuestionSet` environment, then give questions inside `\SetQuestion{}` and answers inside `\SetAnswer{}`. The answers will be shown or hidden according to the printsolutions toggle. Since handout mode sets the printsolutions toggle to false, answers are automatically hidden in handouts.

Each question (in both modes) and each answer (in beamer mode) will show up on one overlay. Question and answer staircase status bars will be printed accordingly. If the set has more than about 7 questions, these bars might overrun the page.

More Flexible Sets of Questions

The `QuestionSet` environment takes care of a lot of things for you (like making the items show up on different slides in a handout, and counting how many there are total). The trade-off is that it might be too restrictive. It can only be in one frame environment, for example, and each question will be on a new overlay.

A more general way of getting the staircase Q&A status bar is to use `\QuestionBar{<a>}{<b>}` and `\AnswerBar{<a>}{<b>}`. This will put a staircase status bar showing <a> of <b> total. You'll probably want to wrap these in an `\only` overlay specification to control where they show up. Unlike `QuestionSet`, the total size of the status bar is fixed, so you don't have to worry about overrunning margins.

```
\QuestionBar{2}{7}
```

Lecturer cues in the corner won't be placed correctly if you have `\centering` on the page, so use `\begin{center} \end{center}` instead, and put the commands outside.

The macros in this category are:

- `\MoreSpace`, `\NoSpace`
- `\StatusBar`
- `\QuestionBar`, `\AnswerBar`
- `\AnswerYes`, `\AnswerNo`

To have them show up on only certain slides, you should use overlays, such as `\AnswerBar<3>{2}{2}`. Do *not* use `\onslide` (it won't work) or `\only` (the spacing might be wrong on other overlays).

Slide Clicks

The command `\StatusBar{<a>}{<b>}` will place a small status bar in the lower-right hand corner of the frame, starting at slide number <a> and ending at slide number <b>.

Sometimes using `\pause` causes errors, so it's best to use explicit overlay commands like `\onslide`.

Map of Contents

The mindmap tables of contents aren't automatically generated from the sections in a document. Rather, they are saved in the file `header_maps.sty`.

Each chapter has its own map. You can highlight any number of sections in a chapter. Since numbers don't work in TeX as variable names, sections are labeled with letters: 1 is a or A, 2 is b or B, etc.

The command `\mapofcontentsA{\ab,\ac}` will show the map of contents for Chapter 1 (that's the "A" in the command itself) with sections 1.2 and 1.3 highlighted (that's the list in the argument). The command `\mapofcontentsC{}` will show the map of contents for Chapter 3 (that's the "C" in the command) with no sections highlighted (since the argument is empty).

Referencing the Textbook

`\eref{text}{<label>}` and `\eref{prob}{<label>}` allow you to reference labelled items in the textbook and problem book, respectively. The folder XR has the aux files from these texts. When the texts are updated, copying the new aux files to that folder will make all the references in the slides update when they are re-compiled.

`\eref{text1}{<label>}` will allow you to reference the CLP–1 textbook.

`\unote{<content>}` (where 'u' stands both for 'under' and 'unobtrusive') will put <content> next to the footline page number. This format is often used to reference the textbook and problem book.

ATTRIBUTION

The index is used to cite copyrighted content. The standard format for setting up a citation of an image is:

```
\index{\includegraphics[height=3mm]{<file.png>} <picture title (link)> by <artist (link)> is licensed under <license information (link)>}
```

The command `\LastPage` prints an index of copyrighted works included in the file so far. If none have been cited with the `\index` command, it will print nothing.

`\CCBY , \CCBYtwo , \CCBYthree \CCBYNCNDfour,` and `\CCBYNCSAtwo` display and link to their respective licenses.

Check Our Work

\CheckFrame{< 1 >}{< 2 >} will print < 1 >, then pause and print < 2 >. The content < 2 > is hidden from the handout.

To get the same brown text and frame title, without the proscribed overlays, use \checkframe{}.

Check Our Work

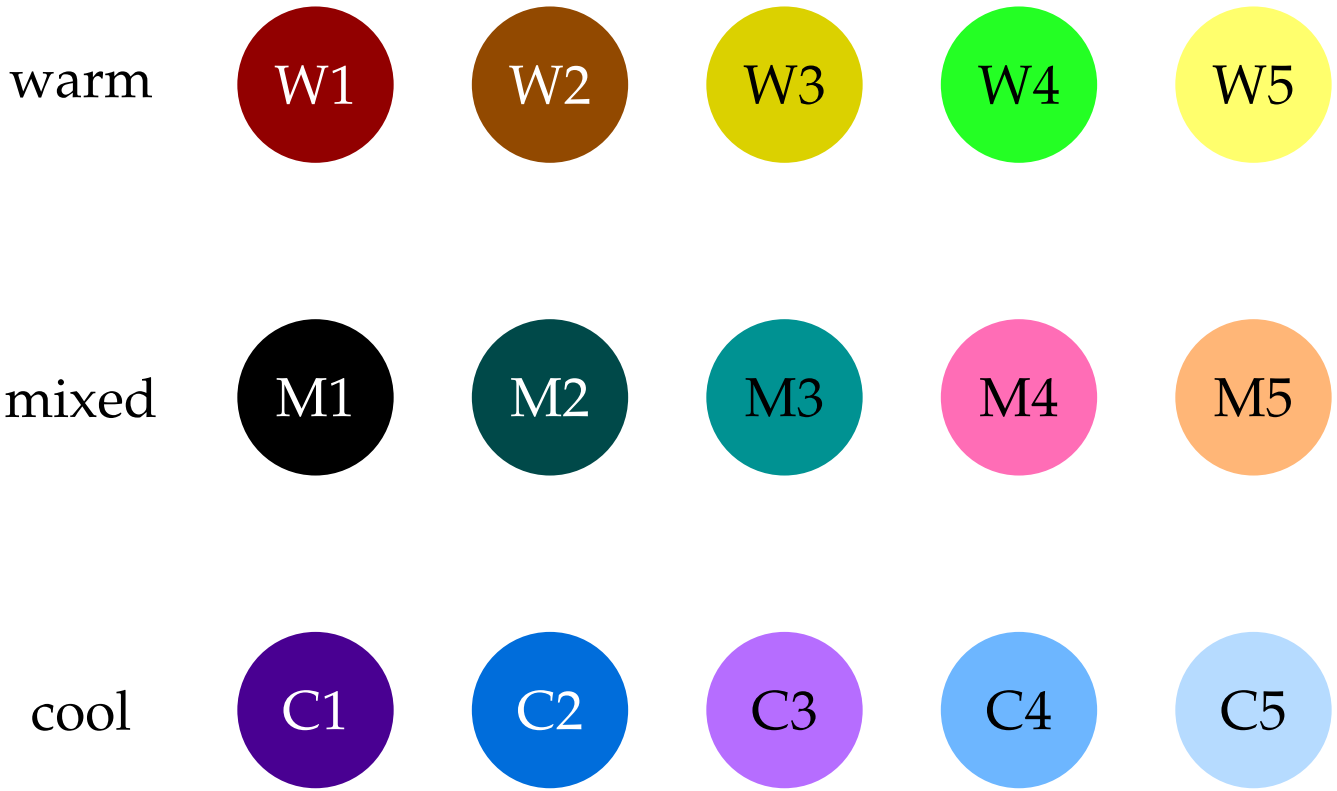
\CheckFrame{< 1 >}{< 2 >} will print < 1 >, then pause and print < 2 >. The content < 2 > is hidden from the handout.

To get the same brown text and frame title, without the proscribed overlays, use \checkframe{}.

These frames are used extensively when antidifferentiation is first introduced. In class, the author usually works through the first few in detail, then afterwards uses them as an opportunity to remind students how they can (and should) easily check antiderivatives by differentiating. The purpose of the unusual text colour is to make it clear to students what exactly is being skipped.

Colour Scheme

The following named colours are used in graphics.



CHOICE TABLE FOR INTEGRATION BY PARTS

When performing integration by parts, one must choose a function to be differentiated, and a function to be antidifferentiated. When this technique is first introduced, the two most obvious choices are presented in a table.

Suppose one were trying to use integration by parts to evaluate $\int f(x)g(x)\,\mathrm{d}x$. One option is to differentiate f and antidifferentiate g; another option would be to differentiate g and antidifferentiate f.

The syntax of the macro is:

```
\IBP{<variable>}{<f>}{<f'>}{<F>}{<g>}{<g'>}{<G>}{<slide>}
```

All entries are inputted as text (no dollar signs). In the above scenario, the variable is x; f' is the derivative of f; F is the antiderivative of f, etc. The last argument gives the overlay number on which the table will first appear.

CHOICE TABLE FOR INTEGRATION BY PARTS

The macro `\IBP{x}{f}{f'}{F}{g}{g'}{G}{1}` generates the table below:

Option A:

$u(x) = f$	
$v'(x) = g$	

Option B:

$u(x) = g$	
$v'(x) = f$	

The derivatives and antiderivatives are hidden in the handout, and only show up on the second and third slide here, to give instructor and/or students time to compute them on their own, if desired.

CHOICE TABLE FOR INTEGRATION BY PARTS

The macro `\IBP{x}{f}{f'}{F}{g}{g'}{G}{1}` generates the table below:

Option A:

$u(x) = f$	$u'(x) = f'$
$v'(x) = g$	$v(x) = G$

Option B:

$u(x) = g$	
$v'(x) = f$	

The derivatives and antiderivatives are hidden in the handout, and only show up on the second and third slide here, to give instructor and/or students time to compute them on their own, if desired.

CHOICE TABLE FOR INTEGRATION BY PARTS

The macro `\IBP{x}{f}{f'}{F}{g}{g'}{G}{1}` generates the table below:

Option A:

$u(x) = f$	$u'(x) = f'$
$v'(x) = g$	$v(x) = G$

Option B:

$u(x) = g$	$u'(x) = g'$
$v'(x) = f$	$v(x) = F$

The derivatives and antiderivatives are hidden in the handout, and only show up on the second and third slide here, to give instructor and/or students time to compute them on their own, if desired.

CHOICE TABLE FOR INTEGRATION BY PARTS

The macro `\IBP{x}{f}{f'}{F}{g}{g'}{G}{1}` generates the table below:

Option A:

$u(x) = f$	$u'(x) = f'$
$v'(x) = g$	$v(x) = G$

$$\to \int G \cdot f' \, \mathrm{d}x$$

Option B:

$u(x) = g$	$u'(x) = g'$
$v'(x) = f$	$v(x) = F$

$$\to \int F \cdot g' \, \mathrm{d}x$$

The derivatives and antiderivatives are hidden in the handout, and only show up on the second and third slide here, to give instructor and/or students time to compute them on their own, if desired.

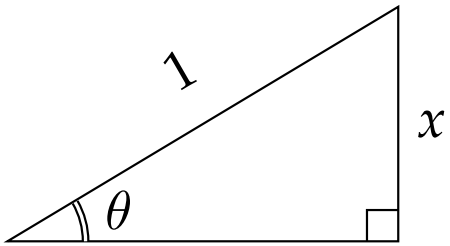
Right Triangle

When we use a trigonometric substitution, we often have to convert between trigonometric functions. For example, suppose we use the substitution $x = \sin\theta$, but then our final answer includes $\tan\theta$. A triangle is often helpful for deciding that $\tan\theta = \frac{x}{\sqrt{1-x^2}}$.

The command
`\TrigTri{<angle>}{<adjacent>}{<opposite>}{<hypotenuse>}`
will generate a right triangle with one angle labeled <angle>, the adjacent side to that angle labelled <adjacent>, etc.

Right Triangle

The command `\TrigTri{\theta}{}{x}{1}` generates the triangle below:



Visualizing Series

The `\weights` macro is intended to impress upon students the relationship between sequences and series. It uses the visual metaphor of weights being added to a scale. Before describing its implementation, we'd like to justify its prevalence, because it is both prominent and repetitive in the slides.

For many years, the author dealt with students who, nearing the final exam, would confuse *sequences* with *series*. The concepts aren't hard to tell apart, and students wouldn't confuse them when they were *first* introduced. It was only after they'd been using them for a while that the lines seemed to blur.

Visualizing Series

The author's theory of the source of the end-of-course confusion around sequences is that learners are sometimes manipulating expressions without keeping the *meaning* of those expressions top-of-mind. One becomes engrossed in the mechanics of, say, the ratio test, and forgets what exactly is being tested.

Since implementing the repetitive display of a visual metaphor, the frequency with which the author was asked the difference between a sequence and a series has decreased dramatically (albeit anecdotally). Explanations about manipulations of partial sums have also become easier, using the shared metaphor of weights on a scale.

VISUALIZING SERIES

The visual metaphor for a series is that the terms of the series are weights (or helium balloons, if they're negative) on a scale. Putting down one weight at a time and noting the reading on the scale generates the sequence of partial sums.

The `\weights` macro should be called inside a tikzpicture environment. It takes three arguments. The first is a list of the sizes of weights; the second is a list of labels for those weights; and the third is a sequence of partial sums. All are entered as lists.

Visualizing Series

First argument of \weights: sizes

The first argument specifies the relative sizes of weights that should be displayed.

The inputted sizes are scaled to make the icons (weights / balloons) fit on the scale. So inputting `1,.5,.25` should give the same result as inputting `4,2,1`. There is a minimum space between each icon, so if you put too many terms here, the icons will overrun the right margin of the frame. There is also a maximum size cutoff for each icon, so in a series whose terms rapidly decay, *the first several icons may be the same size*.

There are two reasons why these sizes are entered separately from their labels. In addition to specifying formatting, it is often desirable to take some poetic license with the sizes for clarity. A series whose terms decay quickly is hard to sketch realistically.

Visualizing Series

Second argument of \weights: labels

The second argument gives labels to be added to the weights. Each label disappears as the weight drops to the scale.

These will be interpreted as being in math mode, so enter `\frac12` instead of `$\frac12$`.

You do not need to have as many labels as there are weights. Indeed, you can leave the second argument entirely blank, if you choose.

Visualizing Series

Third argument of \weights: partial sums

The third argument gives the sequence of partial sums. Partial sums are displayed one-by-one on the scale, and stored on the right-hand side of the frame, to emphasize the "sequence" part of "sequence of partial sums."

If you leave the third argument blank, the partial sums will be computed using the list of weight sizes (the first argument in the macro), and truncated to 4 decimal places.

By default, the first partial sum is S_1. `\setpartialstart{n}` will make the first partial sum S_n. (n must be an integer, not a parameter.) The default starting index will reset at the end of the frame, or you can use `\resetpartialstart` to reset it.

Visualizing Series: example 1

Everything specified; realistic weight sizes, but rapid decay

The next frame shows the result of:

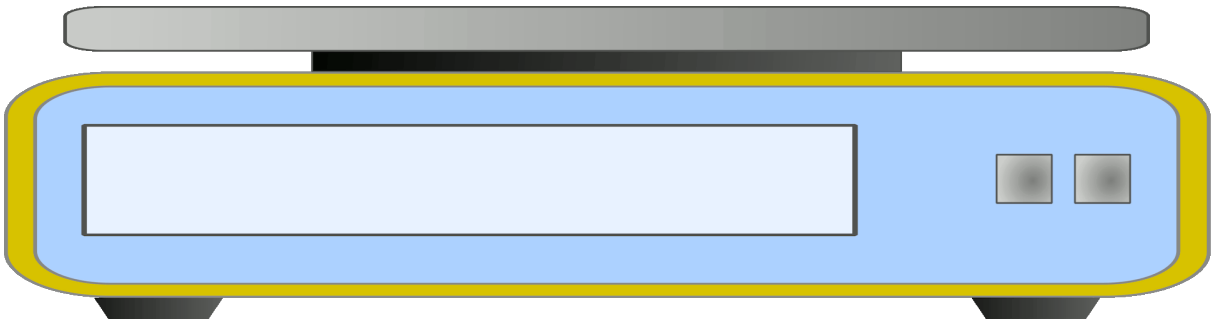
```
\begin{tikzpicture}
\weights{4,2,1}{\frac12,\frac14,\frac18}{.5,.75,.875}
\end{tikzpicture}
```

If we're trying to represent $\frac{1}{2} + \frac{1}{4} + \frac{1}{8}$, the weight sizes (first argument) might seem to be realistic, because they are proportional to the actual summands.

Because the terms of the sum decay rapidly, displaying the sizes as given makes the first icon huge. So, the first icon is constrained by a maximum size, making it the same size as the second icon. So in the actual display, the sizes of the icons don't correspond well to the summands.

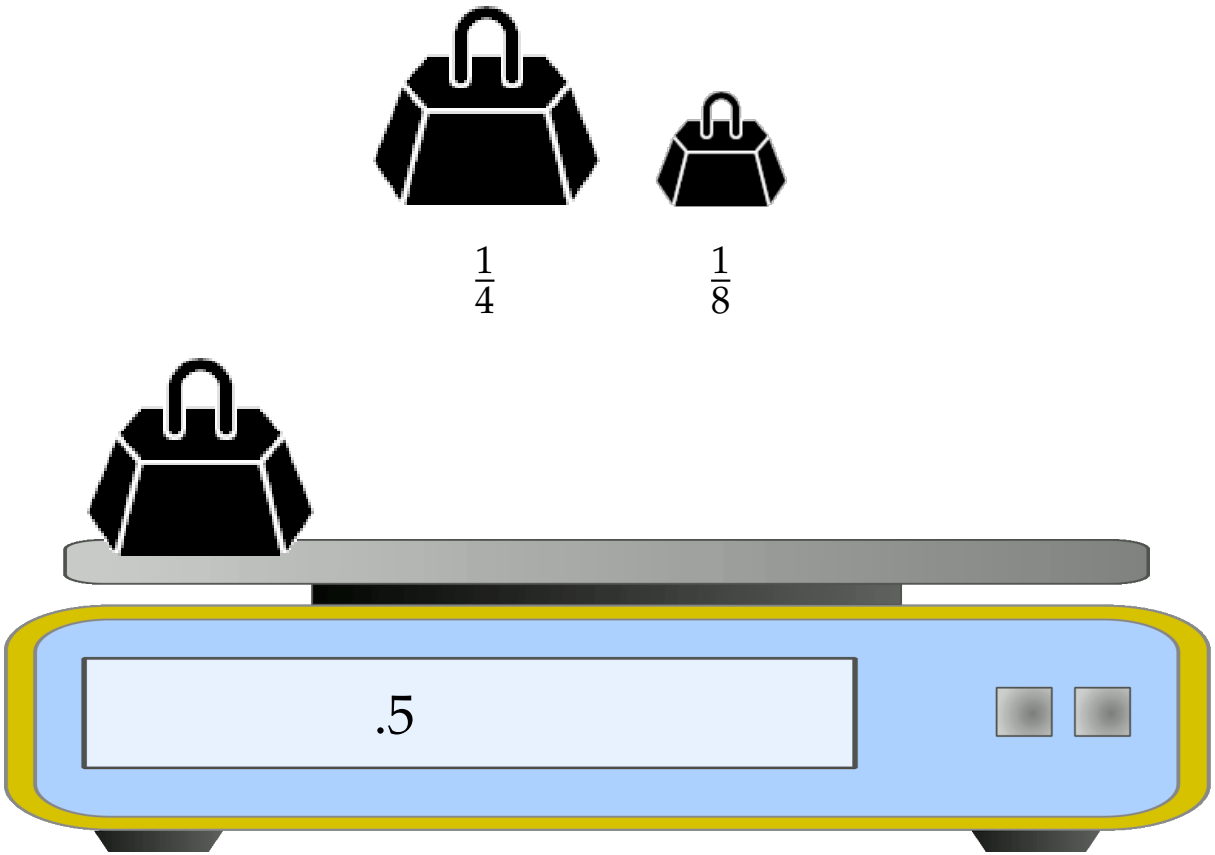
Visualizing Series: example 1

$\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$



Visualizing Series: example 1

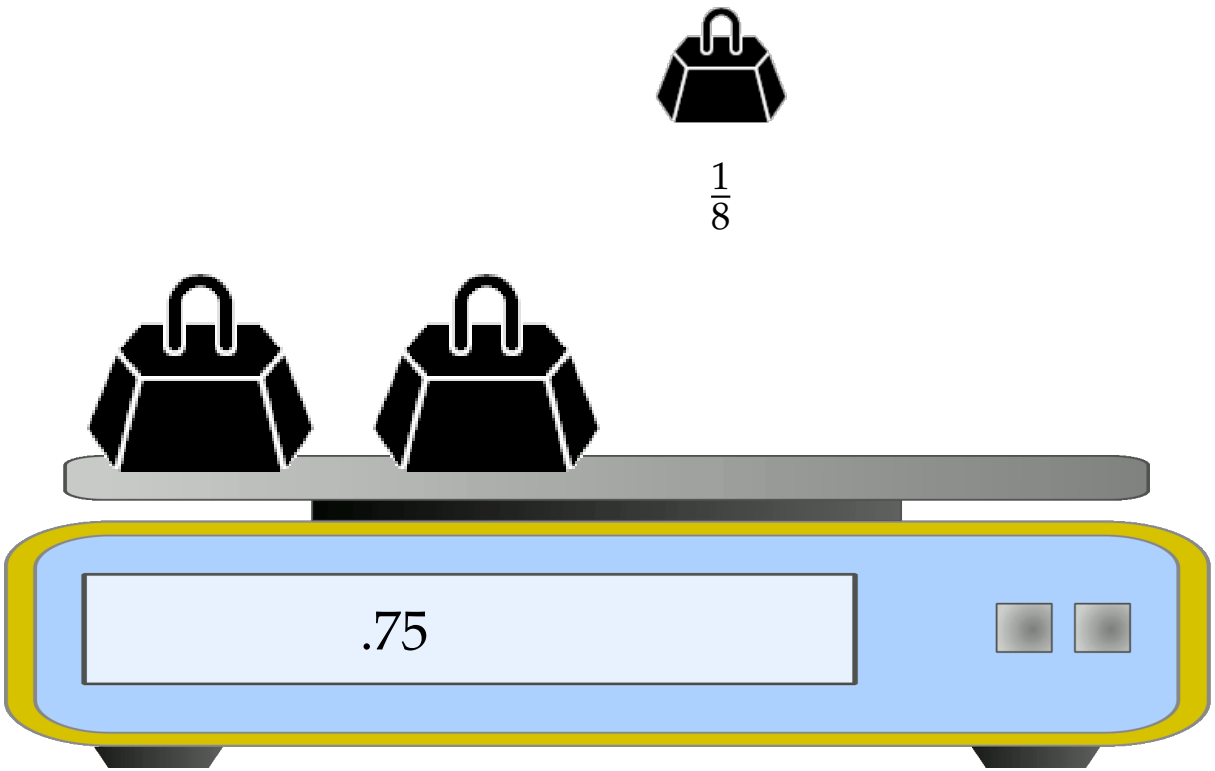
$$S_1 = .5$$



Visualizing Series: example 1

$$S_1 = .5$$

$$S_2 = .75$$

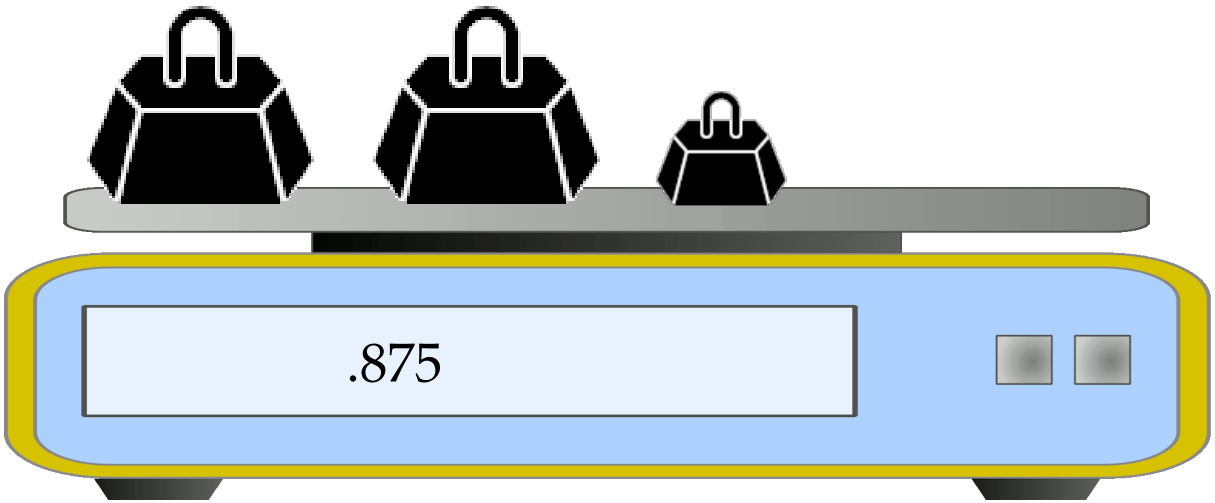


Visualizing Series: example 1

$$S_1 = .5$$

$$S_2 = .75$$

$$S_3 = .875$$



Visualizing Series: example 2

Unspecified partial sums

The next frame shows the result of:

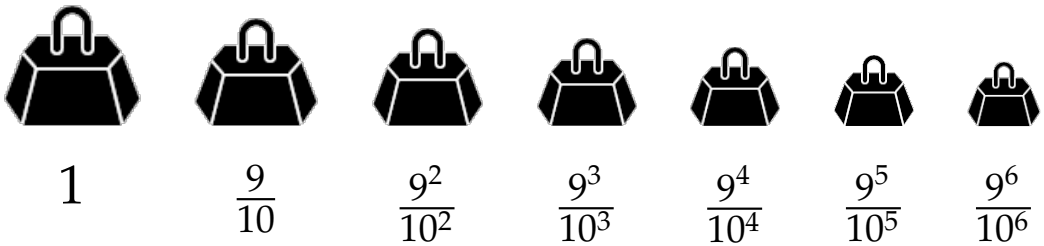
```
\setpartialstart{0} \begin{tikzpicture}
\weights{1,.9,.81,.729,.6561,.59049,.531441}
{1,\frac9{10},\frac{9^2}{10^2},\frac{9^3}{10^3},\frac{9^4}{10^4},\frac{9^5}{10^5
{}
\end{tikzpicture}
```

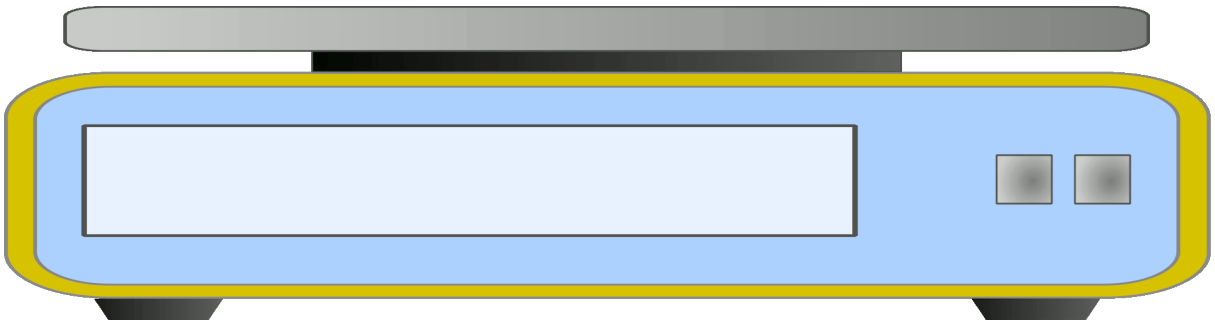
The first partial sum is labeled S_0, instead of S_1.

The weights (first argument) are written accurately, so we can leave the partial sums (third argument) blank. Note the partial sums are all printed with 4 digits.

Because the series decays slowly, it's possible to show all weights with different sizes, without any weight overrunning the system's maximum and being scaled down.

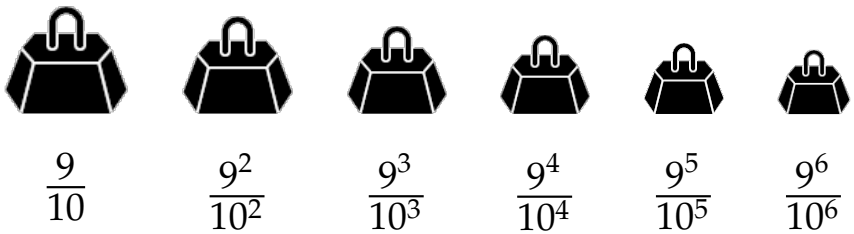
Visualizing Series: example 2

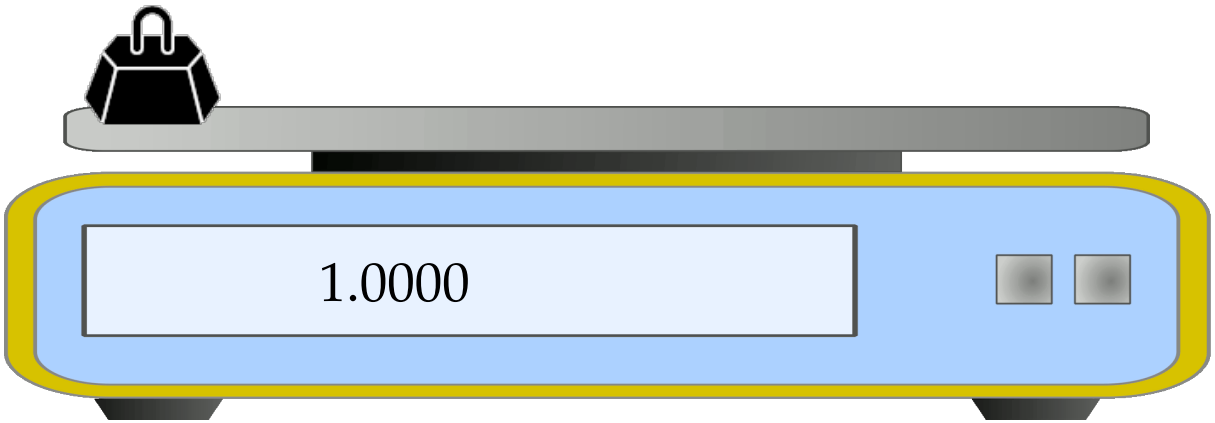




Visualizing Series: example 2

$$S_0 = 1.0000$$

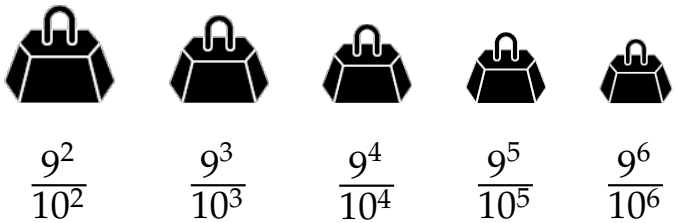




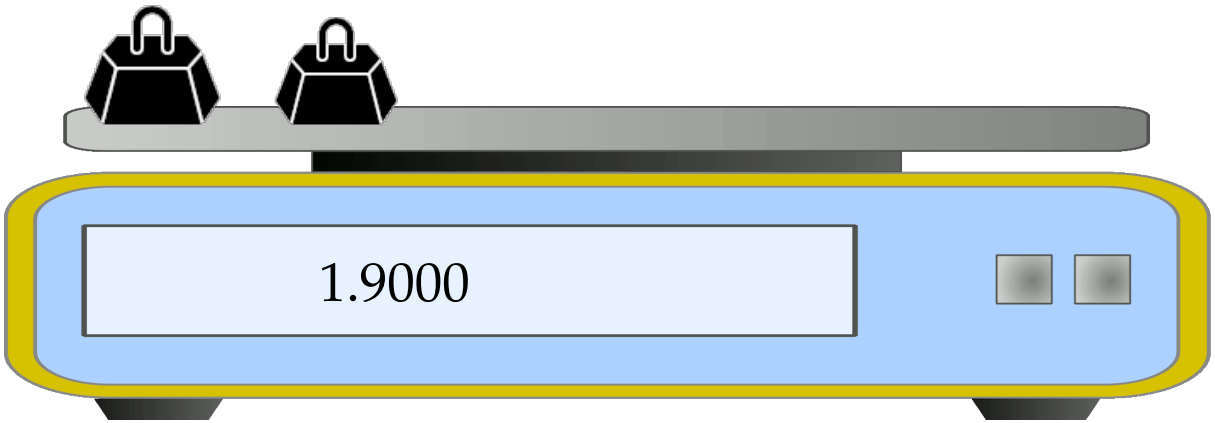
Visualizing Series: example 2

$$S_0 = 1.0000$$

$$S_1 = 1.9000$$



$\frac{9^2}{10^2}$ $\frac{9^3}{10^3}$ $\frac{9^4}{10^4}$ $\frac{9^5}{10^5}$ $\frac{9^6}{10^6}$

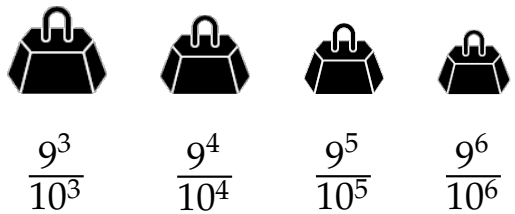


Visualizing Series: example 2

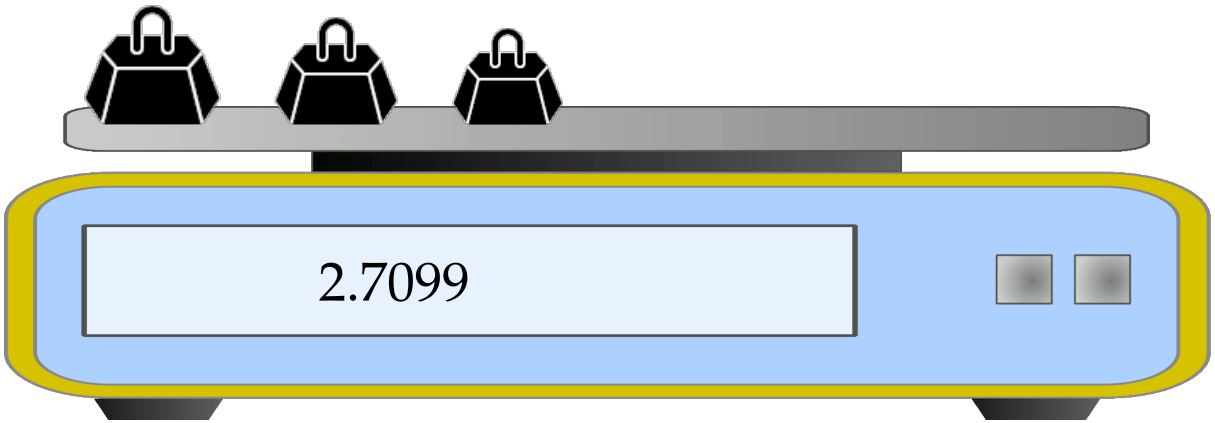
$$S_0 = 1.0000$$

$$S_1 = 1.9000$$

$$S_2 = 2.7099$$



$\frac{9^3}{10^3}$ $\frac{9^4}{10^4}$ $\frac{9^5}{10^5}$ $\frac{9^6}{10^6}$



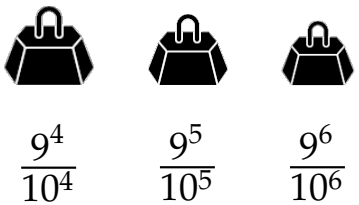
Visualizing Series: example 2

$$S_0 = 1.0000$$

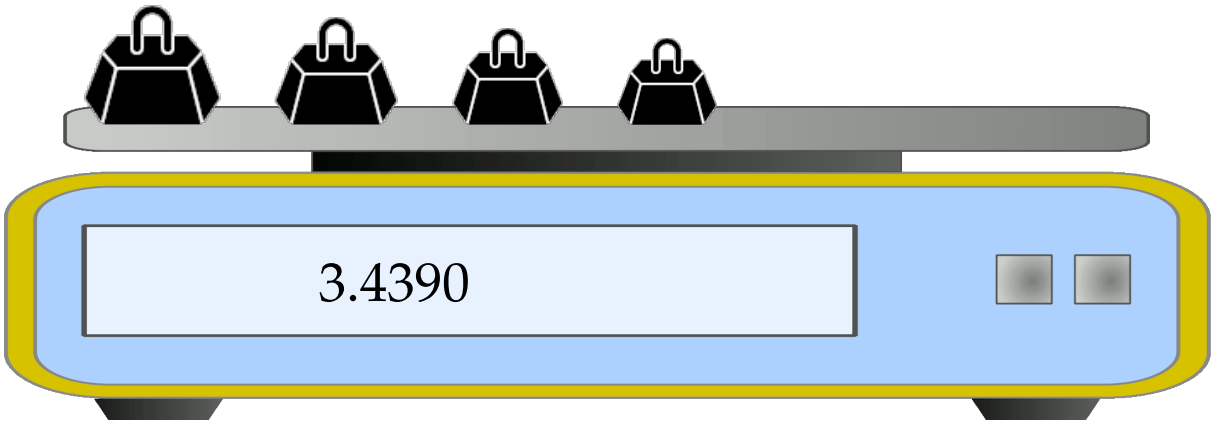
$$S_1 = 1.9000$$

$$S_2 = 2.7099$$

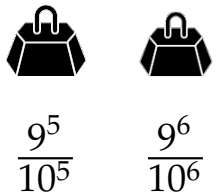
$$S_3 = 3.4390$$



$\frac{9^4}{10^4}$ $\frac{9^5}{10^5}$ $\frac{9^6}{10^6}$



Visualizing Series: example 2



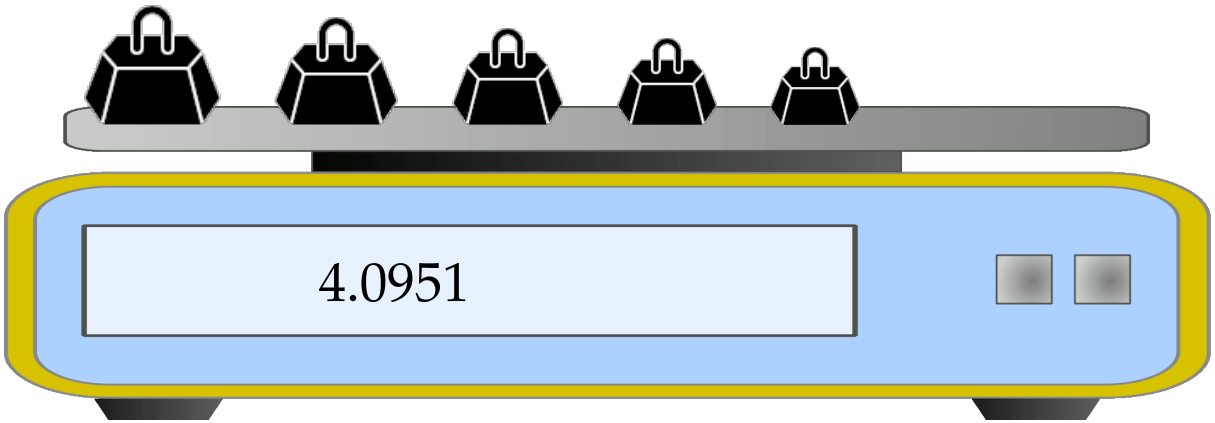
$$S_0 = 1.0000$$

$$S_1 = 1.9000$$

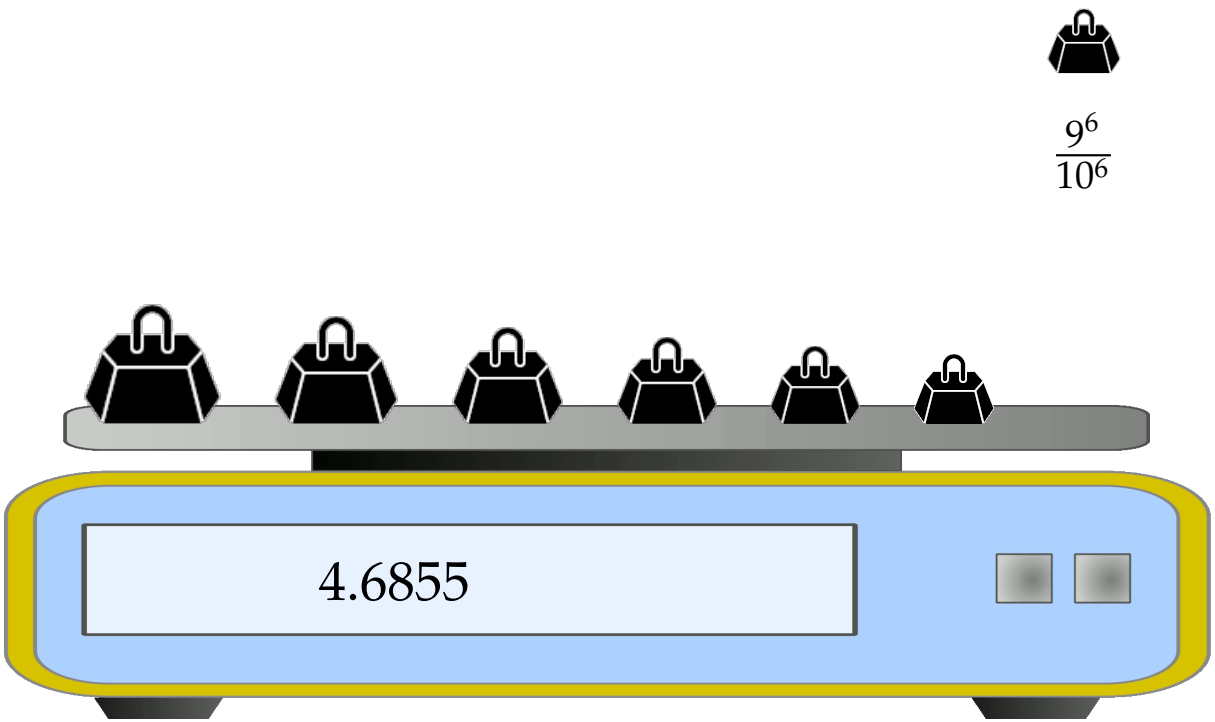
$$S_2 = 2.7099$$

$$S_3 = 3.4390$$

$$S_4 = 4.0951$$



Visualizing Series: example 2



$$S_0 = 1.0000$$

$$S_1 = 1.9000$$

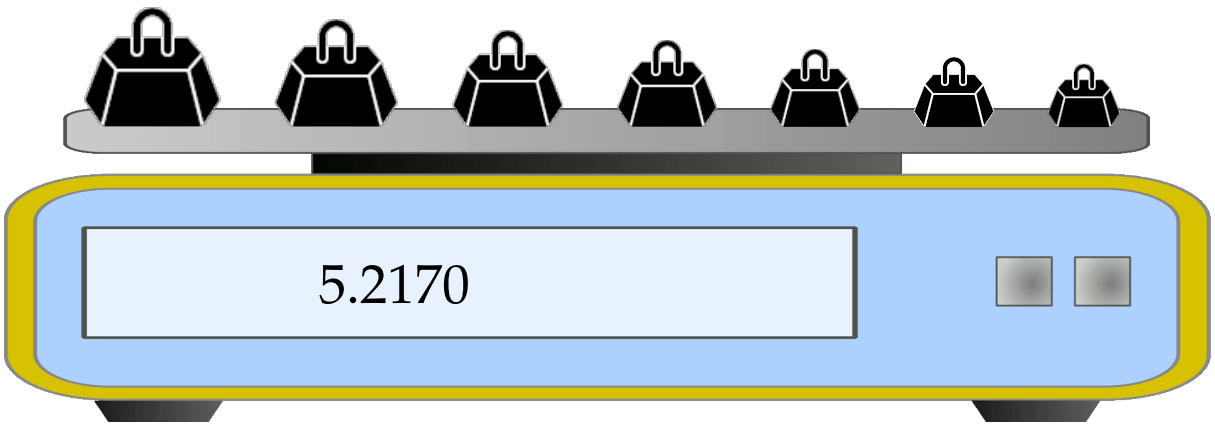
$$S_2 = 2.7099$$

$$S_3 = 3.4390$$

$$S_4 = 4.0951$$

$$S_5 = 4.6855$$

Visualizing Series: example 2



$$S_0 = 1.0000$$

$$S_1 = 1.9000$$

$$S_2 = 2.7099$$

$$S_3 = 3.4390$$

$$S_4 = 4.0951$$

$$S_5 = 4.6855$$

$$S_6 = 5.2170$$

VISUALIZING SERIES: EXAMPLE 3

NEGATIVE TERMS, LABEL SIZES

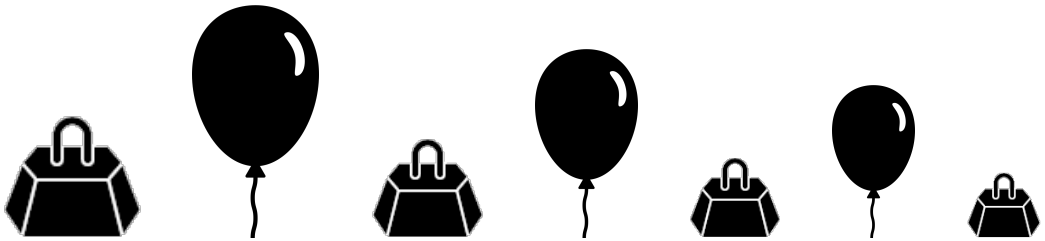
The next frame shows the result of:

```
\begin{tikzpicture}
\weights{1,-.9,.81,-.729,.6561,-.59049,.531441}
{\displaystyle{\left(-\frac9{10}\right)^0},
\textstyle{\left(-\frac9{10}\right)^1},
\scriptstyle{\left(-\frac9{10}\right)^2},
\text{\scriptsize $\left(-\frac9{10}\right)^3$},
\text{\tiny $\left(-\frac9{10}\right)^4$}}
{}
\end{tikzpicture}
```

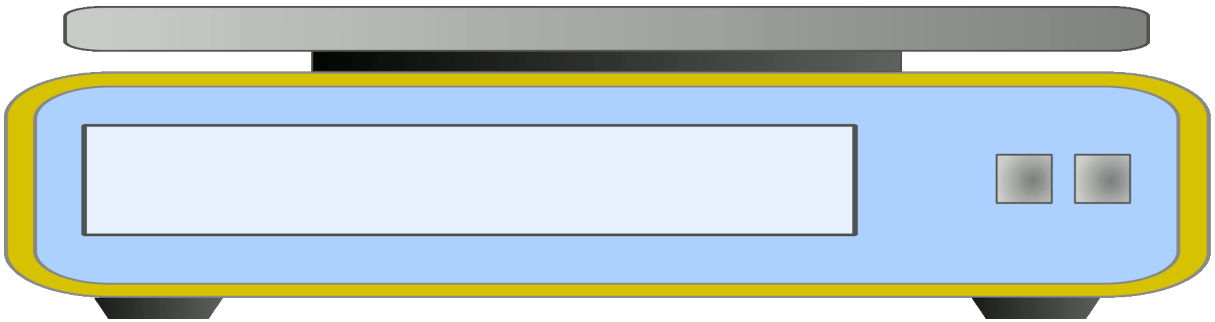
The negative terms are shown as helium balloons.

Various commands are shown that can change the sizes of the labels.

Visualizing Series: example 3

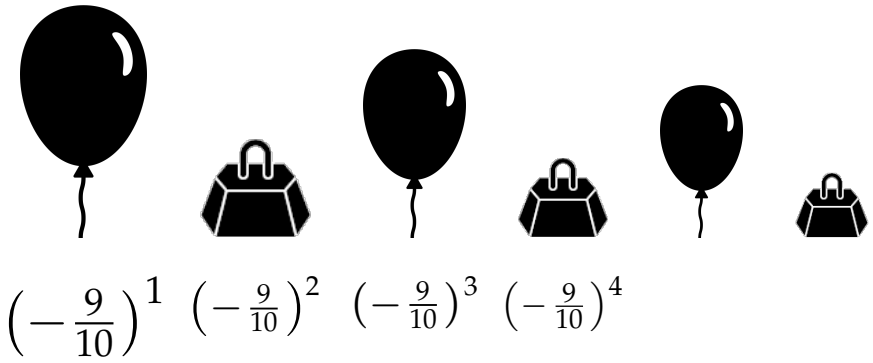


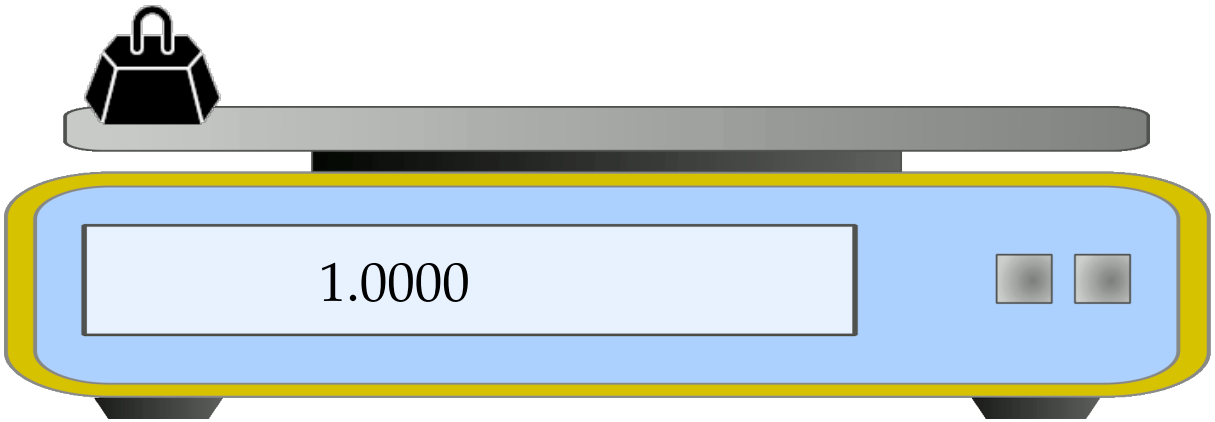
$$\left(-\frac{9}{10}\right)^0 \left(-\frac{9}{10}\right)^1 \left(-\frac{9}{10}\right)^2 \left(-\frac{9}{10}\right)^3 \left(-\frac{9}{10}\right)^4$$



Visualizing Series: example 3

$$S_1 = 1.0000$$

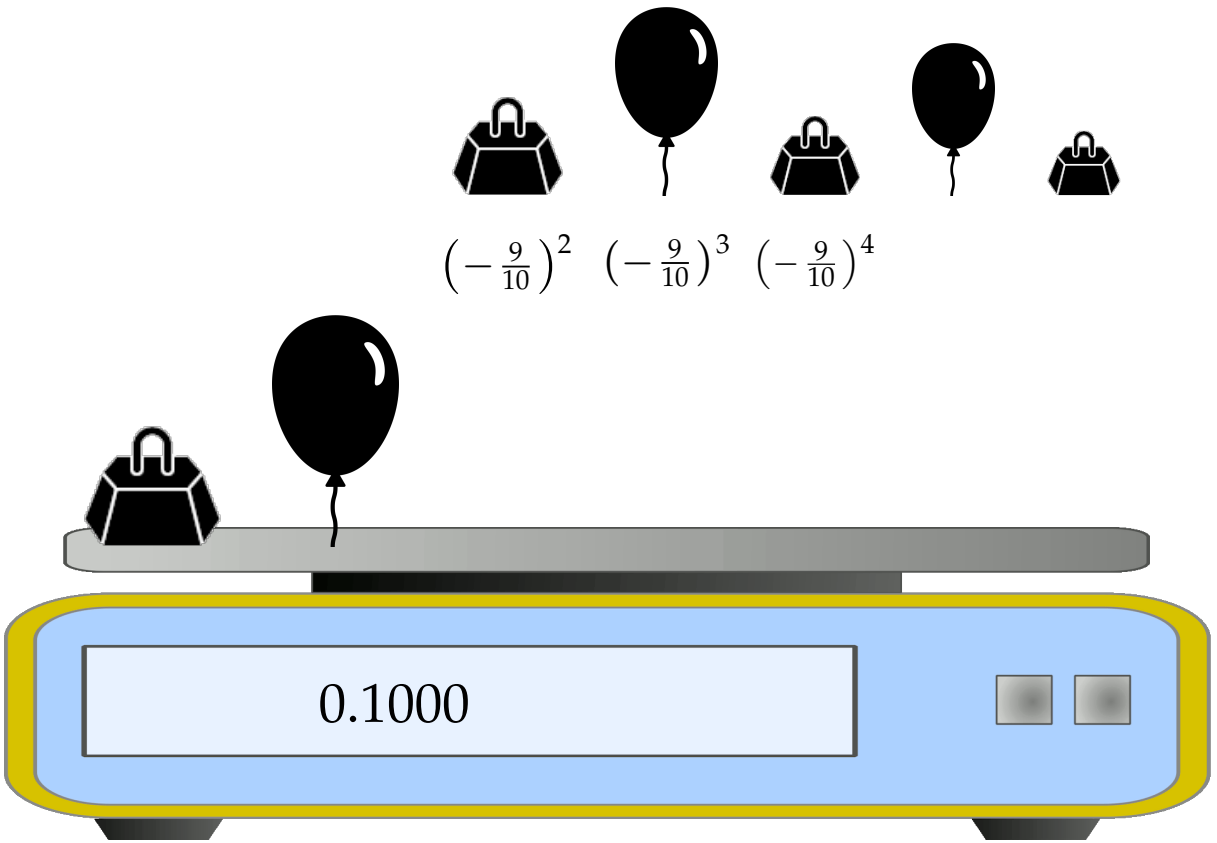




Visualizing Series: example 3

$S_1 = 1.0000$

$S_2 = 0.1000$

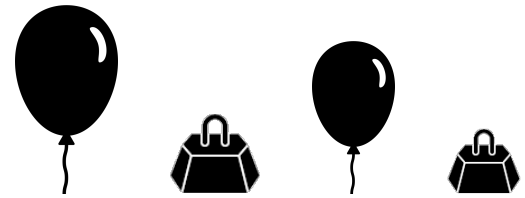


Visualizing Series: example 3

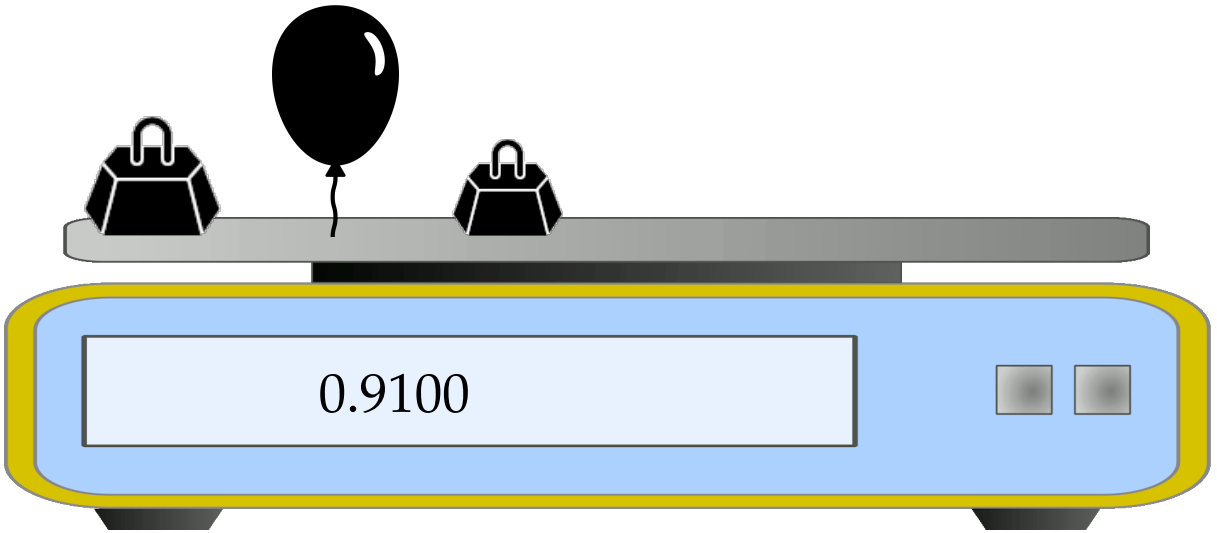
$$S_1 = 1.0000$$

$$S_2 = 0.1000$$

$$S_3 = 0.9100$$



$\left(-\frac{9}{10}\right)^3$ $\left(-\frac{9}{10}\right)^4$



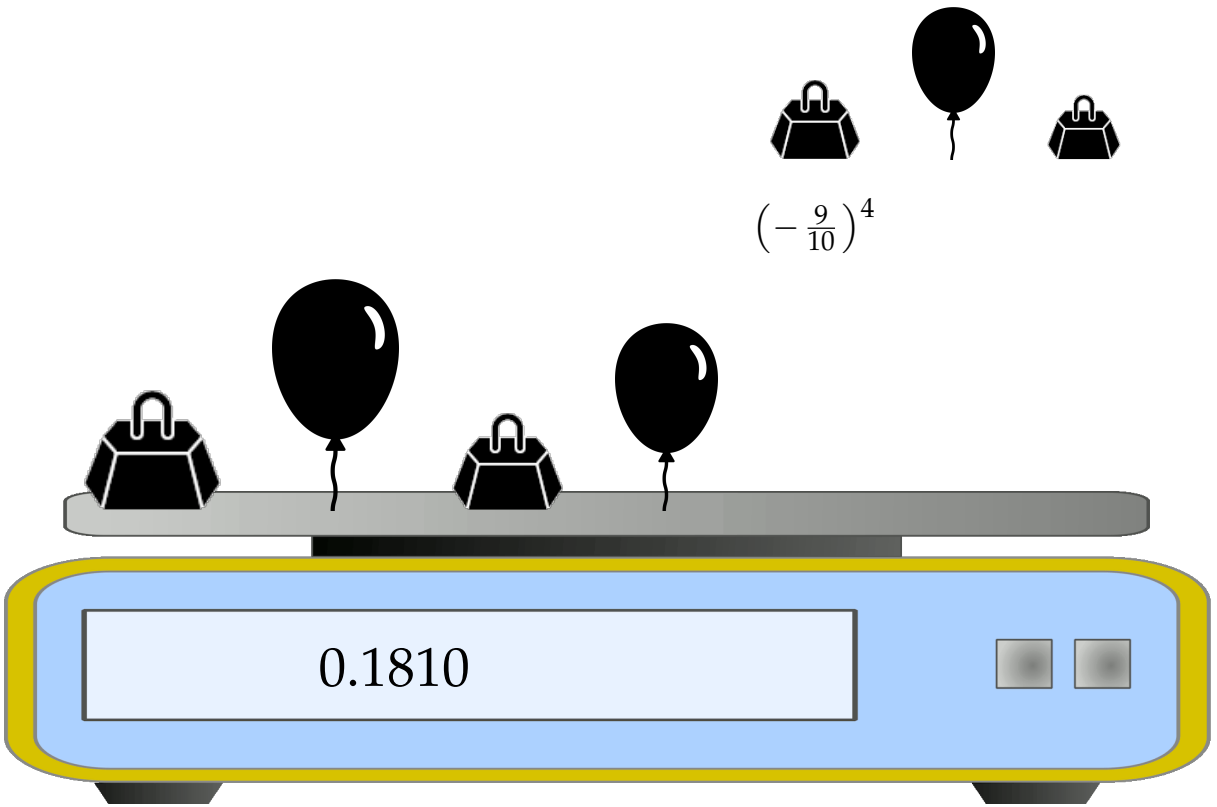
Visualizing Series: example 3

$$S_1 = 1.0000$$

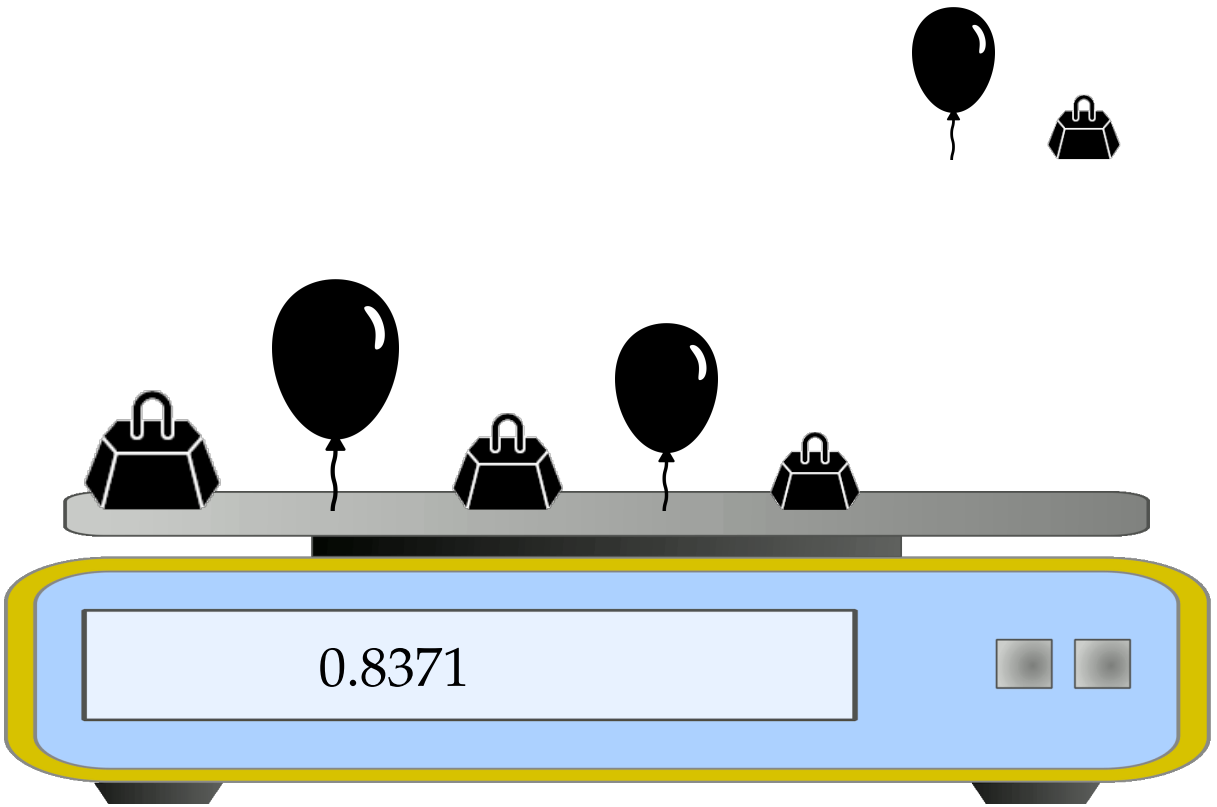
$$S_2 = 0.1000$$

$$S_3 = 0.9100$$

$$S_4 = 0.1810$$



Visualizing Series: example 3



$$S_1 = 1.0000$$

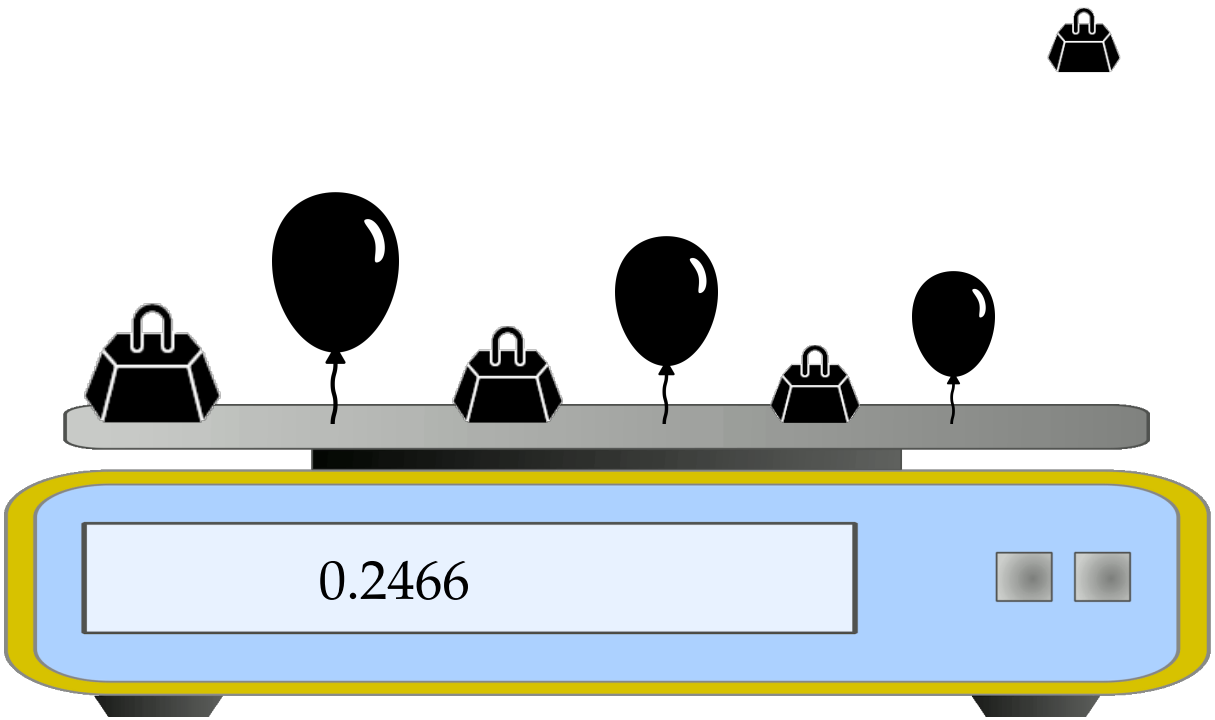
$$S_2 = 0.1000$$

$$S_3 = 0.9100$$

$$S_4 = 0.1810$$

$$S_5 = 0.8371$$

Visualizing Series: example 3



$$S_1 = 1.0000$$

$$S_2 = 0.1000$$

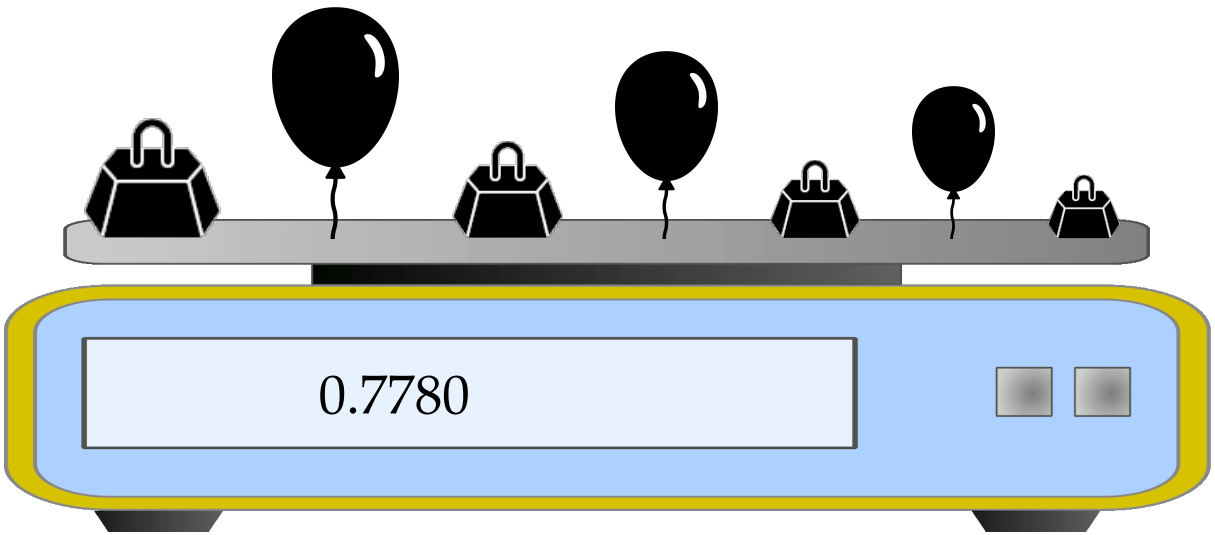
$$S_3 = 0.9100$$

$$S_4 = 0.1810$$

$$S_5 = 0.8371$$

$$S_6 = 0.2466$$

Visualizing Series: example 3



$$S_1 = 1.0000$$

$$S_2 = 0.1000$$

$$S_3 = 0.9100$$

$$S_4 = 0.1810$$

$$S_5 = 0.8371$$

$$S_6 = 0.2466$$

$$S_7 = 0.7780$$

VISUALIZING SERIES: FURTHER DETAILS

- In handout mode, there is no animation. The slide shows the labelled weights hovering above the scale, and displays the sequence of partial sums.
- To shorten the animation, use overlay commands, e.g. `\begin{frame}<1,10->`.
- Infinite sums are represented as finite sums, so it's nice to include a verbal reminder that we imagine this process continuing on.
- When specifying labels, note the entire entry gets put into math mode, so simply writing `\small` won't make the labels smaller.
  - The easiest way to change sizes is to use math styles (like `\displaystyle` or `\scriptstyle`).
  - Alternately, you can put the entry inside a text bracket, then specify the text size, then put it back into math mode. For example, `\text{\tiny $\frac12$}`

Hippo Stacks

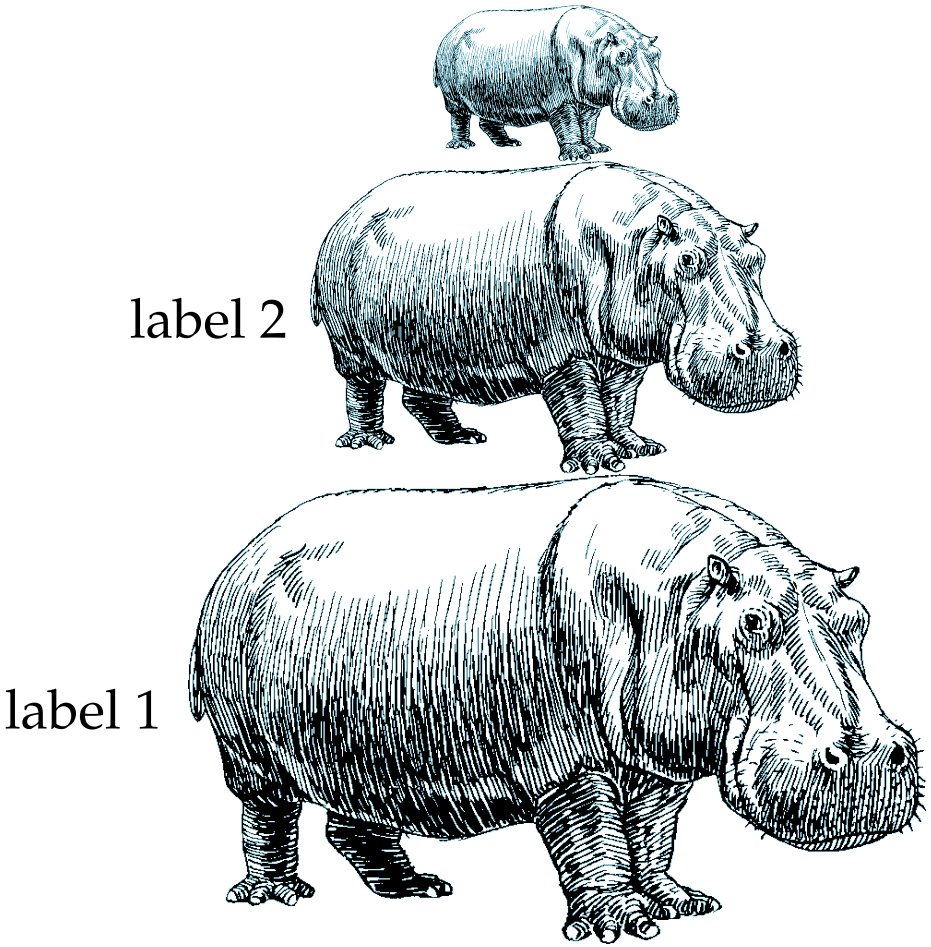
The weights-on-a-scale visualization doesn't lend itself nicely to comparing series, so we provide a second visual metaphor: stacks of hippos. The height of each hippo represents the term being added, so the height of the stack of hippos is a sum.

The macro still needs to be called inside a tikzpicture environment. `\HippoStack` takes two arguments. The first is a list of hippo heights; the second is a list of labels. As with weights on a scale, there can be fewer labels than hippos.

`\VarHippoStack` works the same way, but it produces hippos that are a different colour and facing a different direction.

HIPPO STACKS

```
\begin{tikzpicture}\HippoStack{3,2,1}{label 1,label 2}\end{tikzpicture}
```

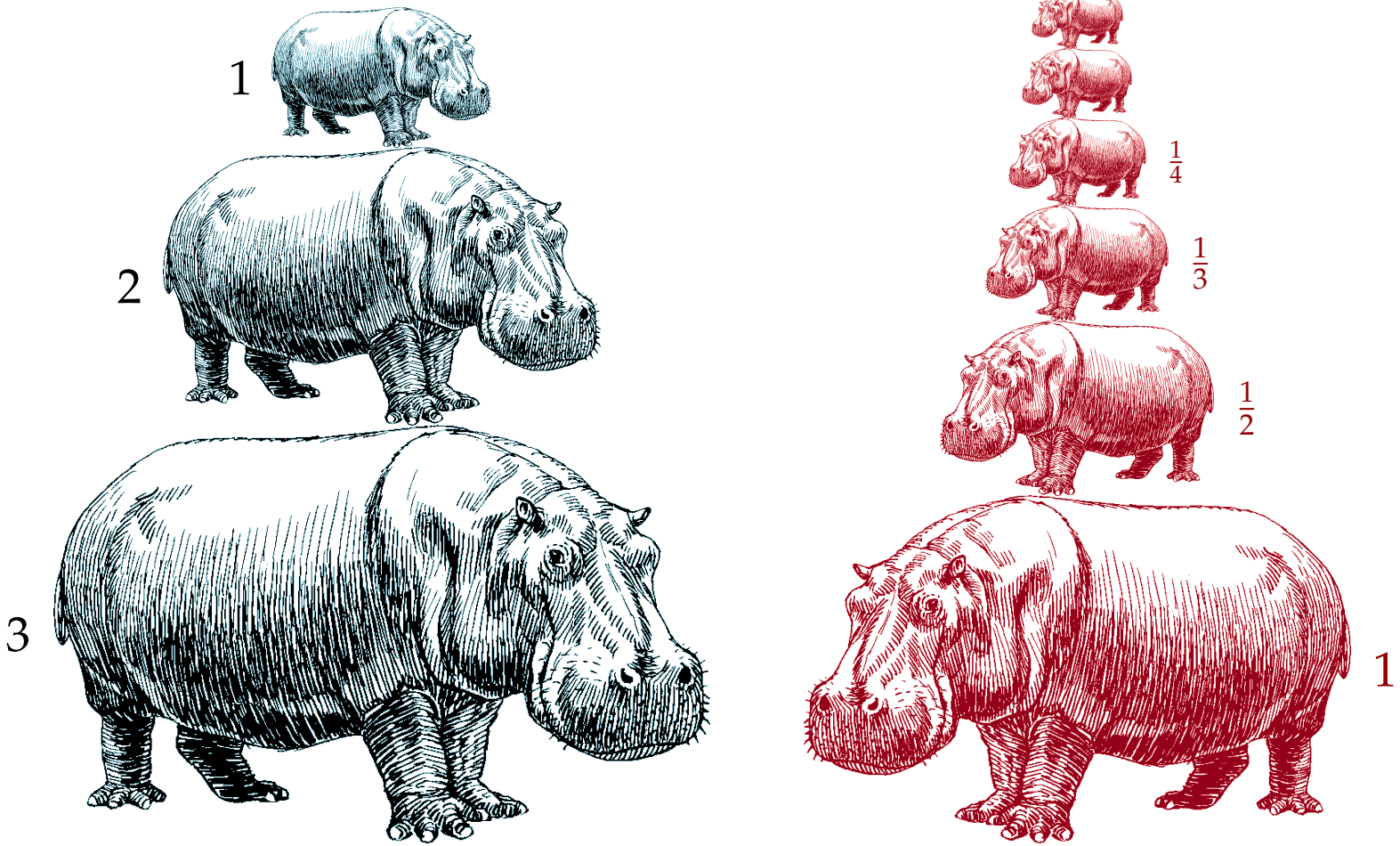


Hippo Stacks

Unlike weights on a scale, the sizes given for hippo stacks are interpreted literally, not scaled to fit the page. Their units are centimetres.

Also unlike weights on a scale, the labels are interpreted as text, not math. So if you want to use math mode, you'll add the dollar signs to each element on the list.

An example using `\smash` to allow the hippos to overrun the frame:



HIPPO STACKS

Further details and tricks:

- To use multiple stacks, you'll have to separate them yourself. You can make two different pictures environments, or use e.g. `\begin{scope}[xshift=5cm] ··· \end{scope}` .
- To place a large number of hippos, one may wish to generate the desired decimals in a spreadsheet, then copy-paste the list to the .tex file.
- The hippo sizes in the slides are, more-or-less, proportional to the series they are meant to represent. We hope this helps students strengthen their intuition.

Substitution Rule

When we did simple examples of the substitution rule, we had special colours for the "inside" function (usually called u) and its derivative. The colour for the differential is meant to stand out more, since it is commonly forgotten by new learners.

- For the inside function: `\su{text}` gives su{text}.
- For just the letter 'u': `\su{}` gives su{}.
- For the derivative of the inside function: `\sdu{text}` gives sdu{text}.
- For just 'du': `\sdu{}` gives sdu{}.

The colour in the commands is overlay-aware.

In later examples, where these functions were not so easily tracked, we didn't use special colours.

Included Work