

# examy

Exams, quizzes, and homework in Typst

VERSION 1.1.0

IN THIS MANUAL


- ☑ Questions and numbering
- ☑ Points
- ☑ Cross-references
- ☑ Answer boxes
- ☑ Multiple choice
- ☑ Solutions
- ☑ Rubrics
- ☑ Cover pages
- ☑ Exam metadata
- ☑ Configuration
- ☑ Technical details



ONLINE
siefkenj.github.io/typst-examy

TYPST UNIVERSE
typst.app/universe/package/examy

SOURCE
github.com/siefkenj/typst-examy

# Contents

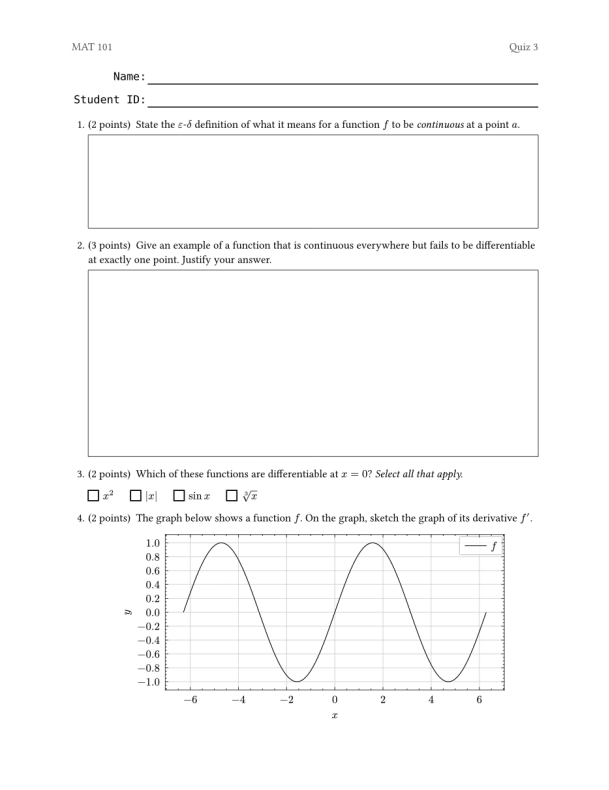

1 Getting started ... 4
  1.1 Installation ... 4
  1.2 A first exam ... 5
  1.3 Basic features ... 6
2 Questions and numbering ... 7
  2.1 Numbering a question ... 8
  2.2 Numbering styles ... 8
  2.3 Height and writing space ... 10
  2.4 Pagebreaks ... 11
  2.5 Columns ... 11
3 Points ... 13
  3.1 The "points" marker ... 13
  3.2 Points tables ... 16
  3.3 Computing with points ... 18
4 Cross-references ... 20
5 Answer boxes ... 22
  5.1 Height and width ... 23
  5.2 Borders ... 23
  5.3 Global settings for answer boxes ... 24
6 Multiple choice ... 26
  6.1 Rows and columns ... 26
  6.2 Round, square, and labelled bubbles ... 27
  6.3 Custom layouts ... 28
7 Solutions ... 30
  7.1 Compiling with and without solutions ... 30
  7.2 Customizing solutions ... 31
  7.3 Solutions inside graphics and other advanced solutions ... 33
8 Rubrics ... 35
  8.1 Visibility ... 36
  8.2 Groups ... 36
  8.3 Customizing the position of rubrics ... 37
  8.4 Listing and exporting rubrics ... 38
9 Cover pages ... 40
  9.1 Title blocks ... 40
  9.2 Name blocks ... 41
  9.3 Putting together a cover page ... 42
10 Exam metadata ... 44
  10.1 Embedding metadata in the PDF ... 44
  10.2 Rubric data ... 44
  10.3 Question regions ... 48
11 Configuration ... 51
  11.1 Reading config values ... 52
  11.2 All `config` options ... 52
12 Technical details ... 58
  12.1 No nested boxes ... 58
  12.2 Step 1: Questions are annotated with metadata ... 58
  12.3 Step 2: Layout ... 58
  12.4 Settings and elembic ... 59
13 Reference ... 60
  `answer-box` element ... 61
  `bubble` alias ... 62
  `choice` function ... 62
  `choices` function ... 64
  `config` element ... 66
  `e` value ... 70
  `exam` function ... 71
  `maketitle` function ... 71
  `name-block` function ... 72
  `num-points` function ... 72
  `num-points-page` function ... 72
  `num-questions` function ... 73
  `page-score-box` function ... 73
  `points-table` function ... 73
  `presets` value ... 74
  `question`, `part`, `subpart` function ... 74
  `rubric` function ... 75
  `rubric-list` function ... 76
  `show-solutions` function ... 76
  `solution` element ... 77
  `square-bubble` alias ... 77
14 Changelog ... 78
  1.1.0 — 2026-11-09 ... 78
  1.0.0 — 2026-10-28 ... 78
  0.4.0 — 2026-10-05 ... 80
  0.3.0 — 2026-09-19 ... 80
  0.2.0 — 2026-07-14 ... 80

## Configuration options

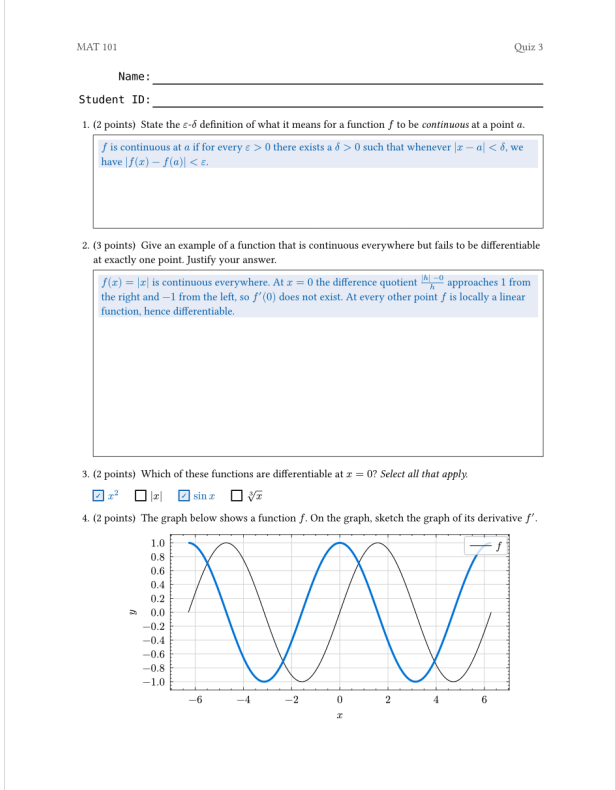

answer-box-default-height ..... 68
answer-box-default-width ..... 69
answer-box-fill ..... 68
answer-box-height ..... 68
answer-box-stroke ..... 68
answer-box-width ..... 68
bonus-points-format ..... 69
bubble-label-font ..... 67
bubble-outset ..... 68
bubble-size ..... 67
bubble-stroke ..... 67
choices-columns ..... 68
choices-enumerate ..... 68
choices-gap ..... 68
choices-inline ..... 68
duration ..... 66
embed-metadata ..... 67
exam-name ..... 66
indent ..... 69
institution ..... 66
numbering ..... 69
numbering-full ..... 69
points-format ..... 69
points-position ..... 70
points-show-totals ..... 70
points-table-bonus ..... 70
points-table-direction ..... 70
points-table-granularity ..... 70
points-table-max-columns ..... 70
points-table-score ..... 70
rubric-position ..... 66
rubric-trace ..... 67
rubric-width ..... 67
show-rubrics ..... 66
show-solutions ..... 66
solution-background-color ..... 67
solution-color ..... 67
strings ..... 69
term ..... 66

# 1 Getting started

A Typst package for writing exams, quizzes, and homework with automatically numbered questions, answer boxes, points accounting, smart cross-references, and solutions that can be toggled on and off. This package follows the spirit of the exam class for LaTeX.




A quiz…




…and solutions from the same source.

## 1.1 Installation

To use examy, import it and start the exam with a show rule:

```
#import "@preview/examy:1.1.0": *

#show: exam
```

A quick-start template is available by choosing examy under *Start from template* in the Typst web app or on the command line by running

```
typst init @preview/examy
```

## 1.2 A first exam

A basic document is built out of an `#exam[..]`. Inside are questions, created with `#question[..]` (see Questions and numbering), along with `#answer-box[..]`s (see Answer boxes) for students to write in and `#solution[..]`s (see Solutions) whose visibility can be toggled to create regular/key versions of a test.

Additionally, examy comes with helpers like `#name-block()` to give students a place to write their name (see Cover pages).

```
#import "@preview/examy:1.1.0": *

#show: exam.with(show-solutions: false)

// A fill-in block (Name / Student ID) at the top of the page
#{
  set text(font: "DejaVu Sans Mono")
  name-block()
}

#question(points: 2)[
  State the definition of a _continuous function_.
  #answer-box(height: 1fr)[
    #solution[A function $f$ is continuous at $a$ if
      $lim_(x -> a) f(x) = f(a)$.]
  ]
]
#question(points: 3)[
  Give an example of a continuous function that is not
  differentiable.
  #answer-box(height: 2fr)[
    #solution[$f(x) = abs(x)$, at $x = 0$.]
  ]
]
```

EXAM

Name: _____

Student ID: _____

1. (2 points) State the definition of a *continuous function.*

2. (3 points) Give an example of a continuous function that is not differentiable.

ANSWER KEY

Name: _____

Student ID: _____

1. (2 points) State the definition of a *continuous function.*

   A function $f$ is continuous at $a$ if $\lim_{x \to a} f(x) = f(a)$.

2. (3 points) Give an example of a continuous function that is not differentiable.

   $f(x) = |x|$, at $x = 0$.

## 1.3 Basic features

**Questions and numbering** Questions, parts, and subparts, and their respective numbering.

**Points** Points associated with each question/part/subpart, as well as points tables.

**Cross-references** Use `@name` to reference a question/part/subpart.

**Answer boxes** Providing a student with space to write. These can have a fixed size or can use available space via fractional units (e.g. `height: 1fr`).

**Multiple choice** Add bubbles for students to answer multiple-choice questions.

**Solutions** Add solutions to questions/parts/subparts that only show up on an answer-key version of the exam.

**Rubrics** Annotate questions and solutions with grading rubrics (positive or negative points), exportable as JSON.

**Cover pages** Functions for creating name/exam info areas.

**Exam metadata** Export rubrics and where each question is on the page as JSON, or attach them to the PDF.

**Configuration** Configuration options that affect the look and feel of your test.

**Reference** Complete documentation for functions provided by examy.

# 2 Questions and numbering

A document is built out of an `#exam[..]` (alternatively `#show: exam`). Inside are questions created with the commands `#question[..]`, `#part[..]`, and `#subpart[..]`, used to create a question hierarchy. Questions/parts/subparts can be assigned points, heights, etc.

```
#show: exam

#question[
  Let $f(x) = x^2 sin(1/x)$ for $x != 0$ and let $f(0) = 0$.
  #part(points: 2)[
    Show that $f$ is continuous at $x = 0$.
    #answer-box(height: 1fr)[]
  ]
  #part(points: 3)[
    Is $f$ differentiable at $x = 0$? Justify your answer.
    #subpart[Compute the difference quotient.]
    #subpart[Take its limit.]
    #answer-box(height: 1fr)[]
  ]
]
```

1. Let $f(x) = x^2 \sin\left(\frac{1}{x}\right)$ for $x \neq 0$ and let $f(0) = 0$.

   (a) (2 points) Show that $f$ is continuous at $x = 0$.

   (b) (3 points) Is $f$ differentiable at $x = 0$? Justify your answer.

      i. Compute the difference quotient.

      ii. Take its limit.

A part's numbering is determined *only* by how deeply it is nested, so `#question[#question[..]]` and `#question[#part[..]]` are numbered the same. The aliases `question`/`part`/`subpart` are provided for convenience.

## 2.1 Numbering a question

By default, questions automatically increment their number. Providing a `number: ..` argument to a `question`/`part`/`subpart` overrides this behaviour. A numerical value will cause numbering to skip to that value; a non-numerical value or general content will pause the counter and show that content. To omit a number completely, you can set `number: none`.

```
#question[An automatically numbered question.]
#question[Another one.]
#question(number: 10)[An integer sets the number.]
#question[…and numbering continues from it.]
#question(number: "★")[Content is shown verbatim.]
#question(number: none)[An unnumbered question.]
#question[The automatic counter ignores the previous two.]
```

1. An automatically numbered question.
2. Another one.

10. An integer sets the number.
11. …and numbering continues from it.

★ Content is shown verbatim.

An unnumbered question.

12. The automatic counter ignores the previous two.

## 2.2 Numbering styles

The numbering style of each level is set for the document via the global `config`'s `numbering` option. It takes an array of numbering patterns (e.g., `("1.", "(a)", "(i)")`) or numbering strings following the same format as an `enum`'s `numbering` argument (e.g., `"1.a.i."`). Questions/parts/subparts are labelled hierarchically with the left-most numbering pattern being assigned to the outer-most question/part/subpart.

```
#show: exam.with(
  numbering: ("Problem 1.", "(A)", "i."),
)

#question[
  The first problem.
  #part[A part.]
  #part[Translate the question statement of @later to German.]
]
#question[
  #part(label: <later>)[Where is the nearest grocery store?]
  #part[Relative to your answer in part @later, where is the nearest train station?]
]
```

Problem 1. The first problem.

(A) A part.

(B) Translate the question statement of Problem 2 (A) to German.

Problem 2. (A) Where is the nearest grocery store?

(B) Relative to your answer in part (A), where is the nearest train station?

To force each question/part/subpart to be labelled with a globally unique identifier, use `numbering-full: true`. This is similar to setting the `full` argument on an `enum`.

```
#show: exam.with(numbering: "1.a.i.", numbering-full: true)

#question[
  #part[A part.]
  #part(label: <second>)[Another, with #subpart[a subpart.]]
]
#question[See @second.]
```

1.a. A part.

1.b. Another, with

1.b.i. a subpart.

2. See 1.b.

To number some questions/parts/subparts differently, set `numbering` with a `config` rule inside the exam. It takes the same values as the global `numbering` option and applies to the questions/parts/subparts after the rule, up to the end of its scope.

> *Note.* Setting the `numbering config` rule does not change the question's `number`. Set `number` on the question/part/subpart if you additionally want the counting in the question label to reset, as the bonus question below does.

```
#question[What is $3 + 4$?]
#[
  #show: e.set_(config, numbering: ("Bonus 1.", "(i)"))
  #question(number: 1)[
    #part[Name a prime number larger than 100.]
    #part(label: <largest>)[Name the largest prime number you know.]
  ]
]
#question[Is your answer to @largest really prime?]
```

1. What is $3 + 4$?

Bonus 1. (i) Name a prime number larger than 100.

(ii) Name the largest prime number you know.

2. Is your answer to Bonus 1 (ii) really prime?

The `indent` option sets how far each part and subpart is indented from the question/part it is in (by default, `1.5em`). A question/part/subpart given `indent` itself uses its own.

## 2.3 Height and writing space

To allow students a space to write, use `#answer-box(height: ..)[]`. However, if you need a question and all its part/subparts to cumulatively take up a fixed amount of space, you can provide a `height: ..` argument directly on a question.

```
#question(height: 4.5cm, points: 4)[
  Sketch the graph of $f(x) = x^3 - x$.
  #part[
    Mark its zeros.
    #answer-box(height: 1fr)[]
  ]
  #part[
    Mark its turning points.
    #answer-box(height: 1fr)[]
  ]
]
#question[The next question starts below the first one's 4.5cm.]
```

1. (4 points) Sketch the graph of $f(x) = x^3 - x$.

   (a) Mark its zeros.

   (b) Mark its turning points.

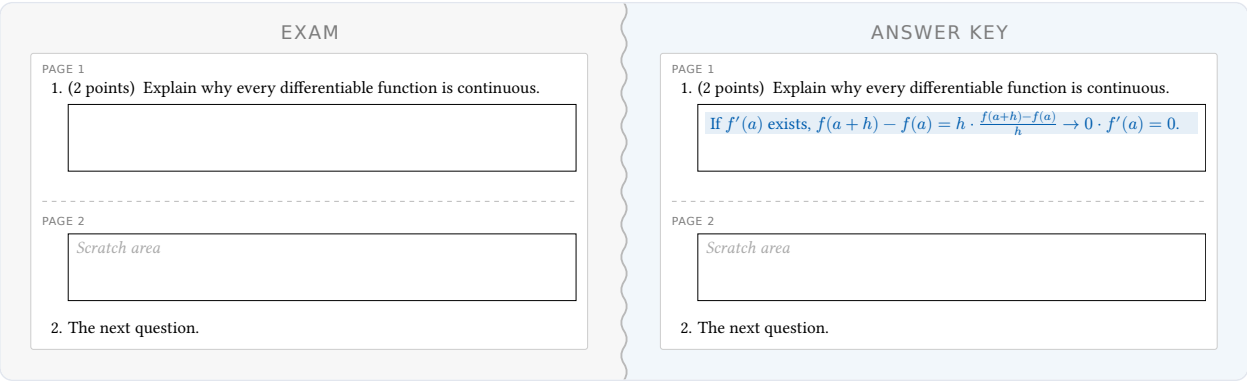
2. The next question starts below the first one's 4.5cm.

*Warning.* A question/part/subpart with a `height` is laid out as a single block, so it cannot break across pages, and the height of its part/subparts cannot be shared with other questions. Usually, you want to use something like `#answer-box(width: 100%, stroke: none, height: ..)[]` instead.

## 2.4 Pagebreaks

`#pagebreak()` works inside questions, parts, and subparts. The question/part/subpart continues on the next page with the same indentation.

```
#question(points: 2)[
  Explain why every differentiable function is continuous.
  #answer-box(width: 100%, height: 1.5cm)[
    #solution[If $f'(a)$ exists,
      $f(a + h) - f(a) = h dot (f(a + h) - f(a)) / h
        -> 0 dot f'(a) = 0$.]
  ]
  #pagebreak()
  #answer-box(width: 100%, height: 1.5cm)[
    #text(fill:gray)[_Scratch area_]
  ]
]
#question[The next question.]
```




## 2.5 Columns

Questions, parts, and subparts can go inside `#columns`. They are numbered and given their points as usual, and a question's text outside the columns stays across the page.

```
#question[
  Find the derivative of each function.
  #columns(2)[
    #part(points: 1)[
      $f(x) = x^3$
      #answer-box(width: 100%, height: 1cm)[]
    ]
    #part(points: 1)[
      $g(x) = sin(x)$
      #answer-box(width: 100%, height: 1cm)[]
    ]
    #colbreak()
    #part(points: 1)[
      $h(x) = e^(2x)$
      #answer-box(width: 100%, height: 1cm)[]
    ]
    #part(points: 1)[
      $k(x) = ln(x)$
      #answer-box(width: 100%, height: 1cm)[]
    ]
```

```
  ]
]
#question(points: 2)[State the chain rule.]
```

1. Find the derivative of each function.

   (a) (1 point) $f(x) = x^3$

   (c) (1 point) $h(x) = e^{2x}$

   (b) (1 point) $g(x) = \sin(x)$

   (d) (1 point) $k(x) = \ln(x)$

2. (2 points) State the chain rule.

Typst fills the first column before it starts the next, so use `#colbreak()` where the next column should start. Columns continue onto the next page by themselves. A `#pagebreak()` inside them moves what comes after it to the next page, where the same columns start again from the first column.

An answer box with a fractional height, such as `height: 1fr`, shares what is left of its own column down to the bottom of the page. Whatever follows the columns then starts on the next page. A question or part given a `height` has no next page to move to, so inside one, the columns must come last, and no other box with a fractional height can come before them.

Points and rubrics in a margin go in the margin of the page, whichever column they come from. Each stays level with its question if it can, and otherwise moves down until it is clear of the others. Those from different columns can then mix, out of the order of the questions, so `points-position: "start"` is usually clearer inside columns.

# 3 Points

Supplying a question/part/subpart with `points: x` will cause an "(x points)" annotation to be shown next to the question/part/subpart. Points defined this way will show up in the `#points-table()` as well as related functions like `#num-points()`, `#num-questions()`, etc.

```
#show: exam

This exam has #context num-questions() questions worth #context num-points() points.

#align(center, points-table())

#question(points: 2)[A question worth two points.]
#question[
  Points on parts roll up to their question.
  #part(points: 1)[One point.]
  #part(points: 3)[Three points.]
]
#question(points: 4)[
  Bonus points are tallied separately and excluded from the
  totals.
  #part(points: 2, intent: "bonus")[
    *Bonus:* not counted above.
  ]
]
```

This exam has 3 questions worth 10 points.

| Question: | 1 | 2 | 3 | Total |
|---|---|---|---|---|
| Points: | 2 | 4 | 4 | 10 |

1. (2 points) A question worth two points.
2. Points on parts roll up to their question.
   (a) (1 point) One point.
   (b) (3 points) Three points.
3. (4 points) Bonus points are tallied separately and excluded from the totals.
   (a) (2 bonus points) **Bonus:** not counted above.

## 3.1 The "points" marker

A question/part/subpart with a value for `points: ..` will display the number of points for that question. The display can be customized in several ways.

### 3.1.1 Customizing the position of "points"

The global `config`'s `points-position` option sets where the points badge is placed:

`"start"` (default) Directly after the question number.
`"margin-left"` **or** `"margin-right"` In the page margin, level with the question number.
`"end"` Flush right, at the end of the question (including any answer boxes), but before any part/subparts.

```
#show: exam.with(
  points-position: "end",
)

#question(points: 2)[What is $2 + 2$?]
#question[
  #part(points: 1)[
    Name a prime number.
    #answer-box(height:1fr)[]
  ]
  #part(points: 3)[Prove there are infinitely many.]
]
```

1. What is $2 + 2$? (2 points)
2. (a) Name a prime number.

   (1 point)

   (b) Prove there are infinitely many. (3 points)

### 3.1.2 Customizing the display and spelling of "points"

The display of points can be fully customized by setting the `points-format` and `bonus-points-format` options of the global `config`.

```
#show: exam.with(
  points-format: p => [(#p #if p == 1 [Punkt] else [Punkte])],
  bonus-points-format: p => [
    (#p #if p == 1 [Bonuspunkt] else [Bonuspunkte])
  ],
)

#question(points: 1)[Ein Punkt.]
#question(points: 2)[
  Zwei Punkte.
  #part(points: 1, intent: "bonus")[Ein Bonuspunkt.]
]
```

1. (1 Punkt) Ein Punkt.
2. (2 Punkte) Zwei Punkte.
   (a) (1 Bonuspunkt) Ein Bonuspunkt.

### 3.1.3 Question totals

With `points-show-totals: true`, a question/part/subpart will show subtotals with the cumulative number of points of all descendants. These additional subtotals are for display only and do not show up in `#num-points()` or the points table.

```
#show: exam.with(points-show-totals: true)

#question[
  A question whose parts have points.
  #part(points: 2)[Two points.]
  #part[
    #subpart(points: 1)[My subpart]
    #subpart(points: 1)[My other subpart]
  ]
]
#question(points: 4)[
  A question with points *and* points on its parts has no
  single total.
  #part(points: 1)[One point.]
]
```

1. (4 points) A question whose parts have points.
   (a) (2 points) Two points.
   (b) (2 points)
      i. (1 point) My subpart
      ii. (1 point) My other subpart
2. examy: this question has points of its own (4) and points on its parts (1); with `points-show-totals`, give points to one or the other. A question with points **and** points on its parts has no single total.
   (a) (1 point) One point.

With `points-show-totals`, a question with `points` of its own that also has a part with `points` has no single total, so a red warning is shown in place of its points.

## 3.2 Points tables

A `#points-table()` displays point totals for each question. Multiple points tables can be included in an exam, and a points table can be customized via the `points-table-..` options on the global `config` or by passing arguments directly to `#points-table()`.

```
#show: exam

#points-table(
  // A row per question (the default is "horizontal").
  direction: "vertical",
  // An empty Score row for the grader.
  score: true,
  // A Bonus row, when there are bonus points.
  bonus: true,
)

#question(points: 2)[First.]
#question(points: 3)[Second.]
#question(points: 5)[
  Third.
  #part(points: 1, intent: "bonus")[Bonus.]
]
```

| Question | Points | Bonus | Score |
|---|---|---|---|
| 1 | 2 | | |
| 2 | 3 | | |
| 3 | 5 | 1 | |
| Total | 10 | 1 | |

1. (2 points) First.
2. (3 points) Second.
3. (5 points) Third.
   (a) (1 bonus point) Bonus.

By default, a points table has a column per question. Specifying `granularity:` as `"part"` or `"subpart"` will create columns for each part or subpart instead, and `"page"` a column for each page (of the questions/parts/subparts that start on it).

```
#show: exam

#points-table(granularity: "part")

#question(points: 2)[What is $2 + 2$?]
#question[
  #part(points: 3)[Factor $x^2 - 1$.]
]
```

```
#question(points: 4)[
  Solve $x^2 = 4$.
  #part(points: 5)[Check both answers.]
]
```

| Question: | 1 | 2 (a) | 3 | 3 (a) | Total |
|---|---|---|---|---|---|
| Points: | 2 | 3 | 4 | 5 | 14 |

1. (2 points) What is $2 + 2$?
2. (a) (3 points) Factor $x^2 - 1$.
3. (4 points) Solve $x^2 = 4$.
   (a) (5 points) Check both answers.

### 3.2.1 Localizing the points table

Examy includes some fixed strings which can be localized via the `strings` option in the global `config`, including those that show up in the points table.

```
#show: exam.with(
  strings: (
    question: [Aufgabe],
    points: [Punkte],
    total: [Summe],
  ),
  points-format: p => [(#p #if p == 1 [Punkt] else [Punkte])],
)

#points-table()

#question(points: 2)[What is $2 + 2$?]
#question(points: 3)[Factor $x^2 - 1$.]
```

| Aufgabe: | 1 | 2 | Summe |
|---|---|---|---|
| Punkte: | 2 | 3 | 5 |

1. (2 Punkte) What is $2 + 2$?
2. (3 Punkte) Factor $x^2 - 1$.

*Note.* The `points` string affects the word “Points” as displayed in the points table. To change the display of “points” in a question, use `points-format`.

## 3.3 Computing with points

`#num-questions()`, `#num-points()` and `#num-points-page()` return numbers, which can be shown or computed with anywhere in the document. They read the exam's points, so they must be called inside of a `context` (e.g. `#context num-points()`).

`num-points` and `num-points-page` count the regular points by default. Give them an `intent` (`"bonus"` or `"practice"`, as on a question/part/subpart) to count those points instead.

```
#show: exam

#context [
  This exam is worth #num-points() points over
  #num-questions() questions.

  You have 50 minutes: about
  #calc.round(50 / num-points(), digits: 1) minutes per point.

  A pass is #calc.ceil(num-points() * .6) points or more.

  #if num-points(intent: "bonus") > 0 [
    There are also #num-points(intent: "bonus") bonus points.
  ]
]

#question(points: 4)[Differentiate $x^3 sin x$.]
#question(points: 6)[
  Find $integral_0^1 x e^x dif x$.
  #part(points: 2, intent: "bonus")[
    *Bonus:* find $integral_0^1 x^2 e^x dif x$.
  ]
]
```

This exam is worth 10 points over 2 questions.

You have 50 minutes: about 5 minutes per point.

A pass is 6 points or more.

There are also 2 bonus points.

1. (4 points) Differentiate $x^3 \sin x$.
2. (6 points) Find $\int_0^1 xe^x \, \mathrm{d}x$.
   (a) (2 bonus points) **Bonus:** find $\int_0^1 x^2 e^x \, \mathrm{d}x$.

### 3.3.1 Showing points per page

You can use `#num-points-page()` in a page header or footer (or anywhere else on the page) to show the number of points on the current page, subtotalling each question/part/subpart appearing on that page.

This can be used to achieve effects like adding a "Score: ____ / 7" box on the footer of each page.

```
#set page(footer: align(right, page-score-box()))

// Or one of your own:
#set page(footer: context if num-points-page() > 0 [
  Page total:
  #box(width: 2em, stroke: (bottom: .5pt))
  out of #num-points-page()
])
```

# 4 Cross-references

To reference a question/part/subpart, add a `label: <name>` argument and use Typst's `@name` syntax to refer to it. By default, a reference is rendered as the shortest unambiguous string.

```
#question[
  Let $f(x) = x^2$.
  #part(points: 2, label: <continuity>)[
    Show that $f$ is continuous at $0$.
  ]
  #part[
    Using your proof in @continuity, show that $f$ is
    continuous at every point.
  ]
]
#question[
  Redo @continuity for $g(x) = x^3$.
]
```

1. Let $f(x) = x^2$.
   (a) (2 points) Show that $f$ is continuous at 0.
   (b) Using your proof in (a), show that $f$ is continuous at every point.
2. Redo 1 (a) for $g(x) = x^3$.

Notice how in 1 (b), the reference `@continuity` is rendered as "(a)" while in question 2 the same reference is rendered as "1 (a)".

To make every reference show the full number (e.g. "1 (a)" wherever it is), use the `numbering-full: true` option.

```
#show: exam.with(numbering-full: true)

#question[
  Let $f(x) = x^2$.
  #part(points: 2, label: <continuity>)[
    Show that $f$ is continuous at $0$.
  ]
  #part[
    Using your proof in @continuity, show that $f$ is
    continuous at every point.
  ]
]
#question[
  Redo @continuity for $g(x) = x^3$.
]
```

1. Let $f(x) = x^2$.

   1 (a) (2 points) Show that $f$ is continuous at 0.

   1 (b) Using your proof in 1 (a), show that $f$ is continuous at every point.

2. Redo 1 (a) for $g(x) = x^3$.

Questions can be labelled too, as in `#question(label: <q-limits>)[..]`, and a reference to one shows its number. References follow the numbering style set with `numbering` (see Questions and numbering).

> *Note.* Labels are given as an argument, `label: <name>`, rather than written after the question as `#question[..] <name>`, because examy takes questions apart to lay them out (see Technical details).

# 5 Answer boxes

An `#answer-box[..]` provides an area for students to write in. It can have a set height in fixed units (`2cm`, `1in`, ...) or *fractional* units (`1fr`, `2fr`, ...). When the height is set as a fractional unit, an `#answer-box[..]` will negotiate with all other items on the page and adjust its height based on the available room.

```
#question[
  State the Mean Value Theorem.
  #answer-box(width: 100%, height: 1cm)[]
]
#question[
  Let $f(x) = x^3 - 3x$.
  #part[
    Find $f'(x)$.
    #answer-box(height: 1fr)[]
  ]
  #part[
    Find and classify the critical points of $f$.
    #answer-box(height: 2fr)[
      Critical points: #answer-box(height: 1cm, width: 1fr)[]

      Classification(s):
    ]
  ]
]
```

1. State the Mean Value Theorem.

2. Let $f(x) = x^3 - 3x$.

   (a) Find $f'(x)$.

   (b) Find and classify the critical points of $f$.

   Critical points:

   Classification(s):

Answer boxes can also be used mid-sentence.

```
#question[
  The derivative of $x^3$ at $x = 2$
  is #answer-box(height: 1.4em)[#solution[12]].
]
```

EXAM

1. The derivative of $x^3$ at $x = 2$ is [ ].

ANSWER KEY

1. The derivative of $x^3$ at $x = 2$ is [12].

## 5.1 Height and width

Without a height set, an answer box will size to its content (excluding any `#solution[..]`s), with an empty answer box taking up a `default-height` (normally set to `2cm`).

```
#question[
  What is $6 times 7$?
  #answer-box(width: 100%)[#solution[42]]
]
#question[
  Is $sqrt(2)$ rational?
  #answer-box(width: 100%)[Explain your reasoning:]
]
```

EXAM

1. What is $6 \times 7$?

2. Is $\sqrt{2}$ rational?
   Explain your reasoning:

ANSWER KEY

1. What is $6 \times 7$?
   42

2. Is $\sqrt{2}$ rational?
   Explain your reasoning:

Setting `width: 100%` or `height: 1fr` (or another fractional unit) makes an answer box act like a block. Otherwise, it is placed as inline content, centered on the current sentence's baseline.

A fractional width, such as `width: 1fr`, fills the rest of the line, so a box can run from a prompt to the end of its line, as in `Critical points: #answer-box(width: 1fr)[]` above. Two boxes in one line share it in proportion to their fractions.

> *Note.* It is good practice to explicitly set a `height: ..` on an answer box.

## 5.2 Borders

An answer box's border is set with `stroke`, which accepts anything `box`'s `stroke` does.

```
#question[
  Sketch the graph of $y = sin x$.
  #answer-box(width: 100%, height: 1.5cm, stroke: none)[]
]
```

```
#question[
  The capital of France is #answer-box(
    width: 4cm,
    height: 1em,
    stroke: (bottom: .5pt),
  )[]
]
```

1. Sketch the graph of $y = \sin x$.

2. The capital of France is _____

## 5.3 Global settings for answer boxes

Exam-wide defaults can be set for answer boxes via the `answer-box-*` options of the global `config`.

```
#show: exam.with(
  answer-box-height: 1fr,
  answer-box-width: 100%,
  answer-box-stroke: (paint: blue, dash: "dashed"),
)

#question[
  Find the derivative of $x^3 + 2x$.
  #answer-box[]
]
#question[
  Is $x^3 + 2x$ increasing? Explain.
  #answer-box(height: 1.5cm)[_Only 1.5cm available for your answer. Be concise!_]
]
#question[
  Find the critical points of $x^3 - 3x$.
  #answer-box[]
]
```

1. Find the derivative of $x^3 + 2x$.

2. Is $x^3 + 2x$ increasing? Explain.

   *Only 1.5cm available for your answer. Be concise!*

3. Find the critical points of $x^3 - 3x$.

Global settings can be overridden on a case-by-case basis.

```
#show: exam.with(answer-box-stroke: (bottom: .5pt))

#question[
  $2 + 2 =$ #answer-box(height: 1em)[] and
  $3 + 3 =$ #answer-box(height: 1em, stroke: .5pt)[]
]
```

1. $2 + 2 =$ _____ and $3 + 3 =$ ☐

# 6 Multiple choice

Multiple choice questions can be written with `#choices[..]` for an automatically formatted list, or with `#choice[..]` and `#bubble()` for finer-grained control.

A `#choices[..]` accepts a mixed list with both `-` and `+` starting a line or an array of content. Lines starting with `+` are recorded as correct choices. Alternatively, correct choices can be given with `correct:`, by their 0-based index (`correct: 1`), by their label when the bubbles are labelled (`correct: "B"`), or as an array of these (`correct: (0, 2, 3)`).

```
#question(points: 1)[
  What is the capital of Canada? _Select one._
  #choices[
    - Toronto
    + Ottawa
    - Montreal
    - Vancouver
  ]
]
#question(points: 1)[
  Which cities lie on Canada's southern border? _Select all that apply._
  #choices(
    bubble: "square",
    correct: (0,2,3),
    ([Toronto], [Ottawa], [Montreal], [Vancouver])
  )
]
```

EXAM

1. (1 point) What is the capital of Canada? *Select one.*

   ◯ Toronto
   ◯ Ottawa
   ◯ Montreal
   ◯ Vancouver

2. (1 point) Which cities lie on Canada's southern border? *Select all that apply.*

   ☐ Toronto
   ☐ Ottawa
   ☐ Montreal
   ☐ Vancouver

ANSWER KEY

1. (1 point) What is the capital of Canada? *Select one.*

   ◯ Toronto
   ◉ Ottawa
   ◯ Montreal
   ◯ Vancouver

2. (1 point) Which cities lie on Canada's southern border? *Select all that apply.*

   ☑ Toronto
   ☐ Ottawa
   ☑ Montreal
   ☑ Vancouver

## 6.1 Rows and columns

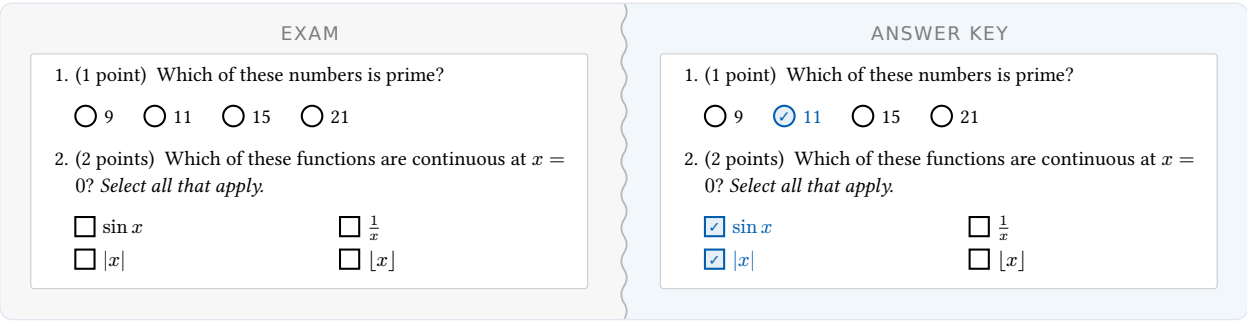
By default, choices are listed one per line, but they can also be laid out in a paragraph or in columns.

`inline: true` Run the choices side by side, wrapping like words. This suits short choices.

`columns: n` Lay the choices out in `n` equal columns, filled row by row.

`gap` The space between choices. Inline, `gap: 1fr` spreads the choices across the line.

```
#question(points: 1)[
  Which of these numbers is prime?
  #choices(inline: true, correct: 1, (9, 11, 15, 21))
]
#question(points: 2)[
  Which of these functions are continuous at $x = 0$?
  _Select all that apply._
  #choices(bubble: "square", columns: 2)[
    + $sin x$
    - $1/x$
    + $abs(x)$
    - $floor(x)$
  ]
]
```

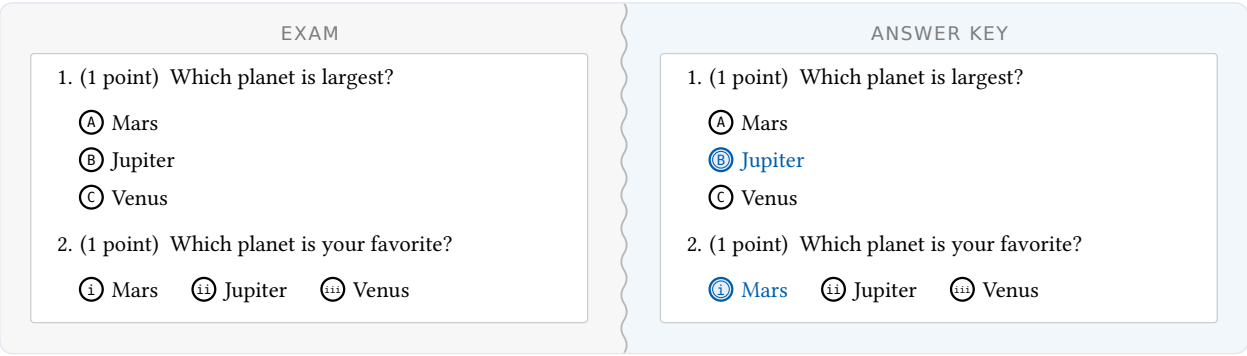



## 6.2 Round, square, and labelled bubbles

The shape of a choice's bubble can be set using `bubble: "circle"` (default) or `bubble: "square"` (often used for select-all-that-apply questions).

Setting `enumerate: true` will label the bubbles with the letters A, B, C, … in sequence. For custom lettering, you can supply a numbering string, an array, or a function that determines the label.

```
#question(points: 1)[
  Which planet is largest?
  #choices(enumerate: true)[
    - Mars
    + Jupiter
    - Venus
  ]
]
#question(points: 1)[
  Which planet is your favorite?
  #choices(enumerate: "i", inline: true)[
    + Mars
    - Jupiter
    - Venus
  ]
]
```




If you would like to set exam-wide defaults for choices, use the global `config`.

```
#show: exam.with(choices-enumerate: "a", bubble-stroke: .06em)
```

Relevant options are:

**`choices-enumerate`** Label each bubble in a choice `#choices[..]` with letters.

**`choices-inline`**, **`choices-columns`**, **`choices-gap`** Effects the layout `choices`.

**`bubble-label-font`** The font of letters used when `choices-enumerate` is set to true (by default, the monospace).

**`bubble-size`** The size of every round bubble (by default, `1em`). Square bubbles are drawn a little smaller, in the same proportion as by default.

**`bubble-stroke`** The outline of every bubble, as a stroke: a thickness (by default, `.1em`), a color, or both (e.g. `.06em + gray`).

**`bubble-outset`** The space around every bubble, as `outset` gives it.

A single `choice` takes `size`, `stroke`, and `outset` itself, and a single `choices` takes those and `label-font` for all its bubbles. Each wins over the global option, and a `stroke` that gives only a color (e.g. `stroke: blue`) keeps the global thickness. Both also take `solution-color` and `solution-background-color`, which set the colors of a correct bubble on the answer key in place of the global `config` options of the same names.

## 6.3 Custom layouts

To fully customize layouts, you can use the `#choice[..]` function directly to draw choices one at a time, or the `#bubble()` and `#square-bubble()` functions to draw bubbles without associated content.

```
#question(points: 1)[
  Let $f(x) = 1/x$. Which statement is true?
  #table(
    columns: (auto, 1fr),
    align: (center + horizon, left),
    [#bubble()], [$f$ is increasing on $(0, 1)$.],
    [#bubble(correct: true)], [$f$ is decreasing on $(0, 1)$.],
  )
]
#question[
  #choice[Agree] #h(1em)
  #choice(correct: true)[Disagree] #h(1em)
  #choice[Unsure]
]
```

EXAM

1. (1 point) Let $f(x) = \frac{1}{x}$. Which statement is true?

| | |
|---|---|
| ◯ | $f$ is increasing on $(0, 1)$. |
| ◯ | $f$ is decreasing on $(0, 1)$. |

2. ◯ Agree ◯ Disagree ◯ Unsure

ANSWER KEY

1. (1 point) Let $f(x) = \frac{1}{x}$. Which statement is true?

| | |
|---|---|
| ◯ | $f$ is increasing on $(0, 1)$. |
| ✓ | $f$ is decreasing on $(0, 1)$. |

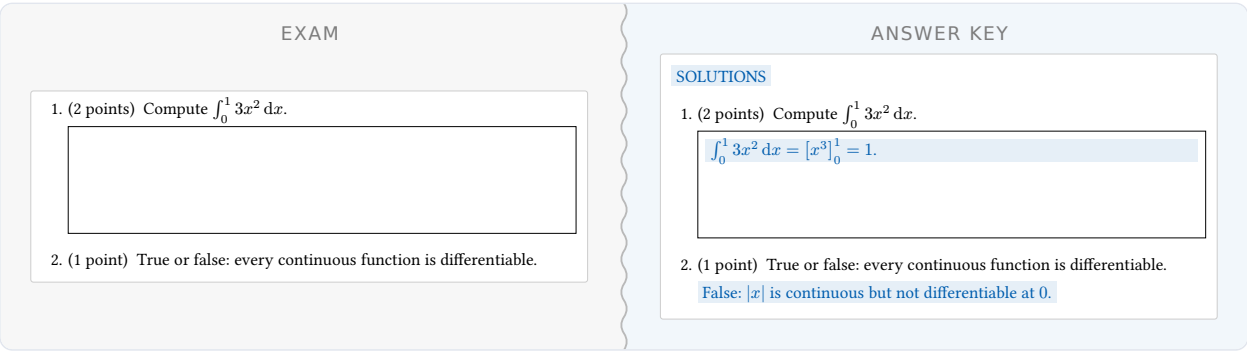
2. ◯ Agree ✓ Disagree ◯ Unsure

# 7 Solutions

Wrapping content in `#solution[..]` ensures that it is only shown in an "answer key" version of an exam, which means you can use a single source to render both an exam and its answer key. Solutions can be placed anywhere in an exam, but are often placed in an `#answer-box[..]`.

```
#solution[SOLUTIONS]

#question(points: 2)[
  Compute $integral_0^1 3 x^2 dif x$.
  #answer-box(width: 100%, height: 2.4cm)[
    #solution[$integral_0^1 3 x^2 dif x = lr([x^3])_0^1 = 1$.]
  ]
]
#question(points: 1)[
  True or false: every continuous function is differentiable.
  #solution[False: $|x|$ is continuous but not
    differentiable at $0$.]
]
```




To toggle whether solutions are visible, set `show-solutions` on the global `config`.

```
#show: exam.with(show-solutions: true)
```

or

```
#show: e.set_(config, show-solutions: true)
```

## 7.1 Compiling with and without solutions

There are three ways to create solution/no-solution versions of an exam: (1) explicitly set `show-solutions:` as `true`/`false` and recompile, (2) use a command-line flag, or (3) Typst's experimental bundle option.

### 7.1.1 Manually setting the flag

By default, solutions are shown in an exam. To create a no-solutions version, use

```
#show: exam.with(
  show-solutions: false
)
```

Putting `show-solutions` on its own line allows you to easily comment/uncomment the line to produce multiple exam versions.

### 7.1.2 Command-line flag

The `show-solutions` input given on the command line overrides the document's `show-solutions` setting, so it can be used to create solution/no-solution versions regardless of the current exam state.

```
# whatever the exam has set
typst compile exam.typ
# the exam (solutions off)
typst compile --input show-solutions=false exam.typ
# the answer key (solutions on)
typst compile --input show-solutions=true exam.typ
```

### 7.1.3 Experimental bundle option

Typst's (experimental) *bundle* export can produce multiple PDFs during a single compile. To use it, save the exam as content, and make one `document` of each version, setting `show-solutions` around it with `e.set_`.

```
#let quiz = [
  #show: exam

  #question[..]
  #question[..]
]

#document("quiz-nosolutions.pdf", {
  show: e.set_(config, show-solutions: false)
  quiz
})
#document("quiz-solutions.pdf", {
  show: e.set_(config, show-solutions: true)
  quiz
})
```

```
typst compile --features bundle -f bundle bundle.typ out/
# writes out/quiz-nosolutions.pdf and out/quiz-solutions.pdf
```

> *Note.* The documents of a bundle share their labels, so each version of the exam has the same labelled questions. A reference to a question/part/subpart refers to the one in its own version. Other labelled content in the exam, such as a heading or a figure, is in every version too, and a reference to it is an error.

## 7.2 Customizing solutions

### 7.2.1 Colors

The colors of solutions can be set via the global `config` with:

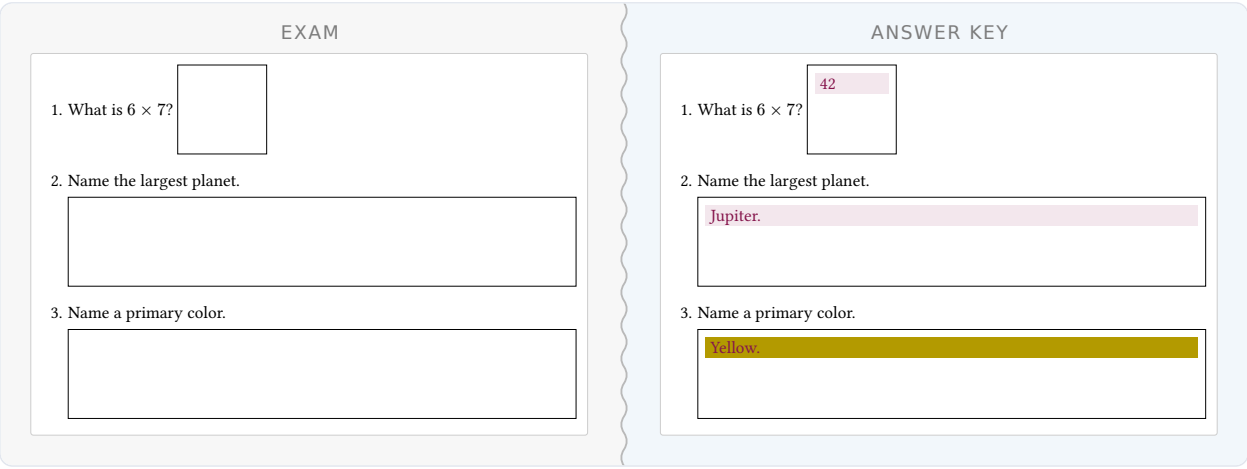
**`solution-color`** The color of a solution's text. The highlight behind it is derived from it, unless set too.
**`solution-background-color`** The color of the highlight. The text color is derived from it, unless set too.

If only one is set, the other is inferred by lightening or darkening the set color.

A single solution takes its own `color` and `background-color`, which win over the global options for that solution, as in the last question below.

```
#show: exam.with(solution-color: maroon)

#question[What is $6 times 7$? #answer-box[#solution[42]]]
#question[
  Name the largest planet.
  #answer-box(width: 100%)[#solution[Jupiter.]]
]
#question[
  Name a primary color.
  #answer-box(width: 100%)[
    #solution(background-color: yellow.darken(30%))[Yellow.]
  ]
]
```




### 7.2.2 The `otherwise:` argument

Normally, when a solution is not rendered, nothing is shown. Setting its `otherwise: ..` value to content will show that content when a solution is not shown. This can be used, for example, to show lined space on a no-solutions version of an exam, but replace it with just a solution on the solutions version.

```
#let lines = block(width: 100%, for _ in range(3) {
  v(1.6em)
  line(length: 100%)
})

#question[
  Differentiate $f(x) = x^2$.
  #answer-box(width: 100%, stroke: none)[
    #solution(otherwise: lines)[By the power rule,
      $f'(x) = 2x$.]
  ]
]
#question[
  #answer-box(width: 100%, stroke: none)[
    The capital of Canada is #solution(
      otherwise: box(width: 3cm, stroke: (bottom: .5pt)),
```

```
    )[Ottawa].
  ]
]
```

EXAM

1. Differentiate $f(x) = x^2$.

2. The capital of Canada is _____.

ANSWER KEY

1. Differentiate $f(x) = x^2$.

   By the power rule, $f'(x) = 2x$.

2. The capital of Canada is Ottawa .

## 7.3 Solutions inside graphics and other advanced solutions

A `#solution[..]` is wrapped in context, and as such, its output cannot be used as the input for most functions, like those that draw graphics.

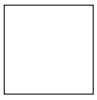
An alternative is to pass `#exam[..]` a function of the form `solutions-only => ...` The first argument is a function which returns its content only when solutions are rendered and returns its `otherwise:` argument otherwise. This needs `#exam(solutions-only => [..])` around the questions; with `#show: exam`, the exam is given content, not a function.

```
#exam(solutions-only => [
  #question[
    Mark the top and right sides of the box.

    #box(width: 2cm, height: 2cm, stroke: (
      rest: .5pt,
      top: solutions-only(2pt + blue, otherwise: .5pt),
      right: solutions-only(2pt + blue, otherwise: .5pt),
    ))
  ]
])
```

EXAM

1. Mark the top and right sides of the box.



ANSWER KEY

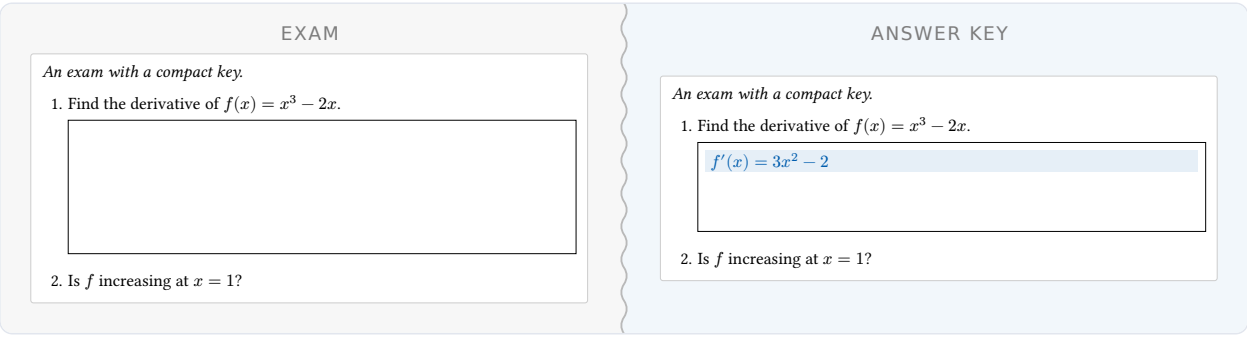
1. Mark the top and right sides of the box.



This can also be used to achieve effects like changing a question's height between solution and no-solution versions.

```
#exam(solutions-only => [
  _An exam with a compact key._
  #question[
    Find the derivative of $f(x) = x^3 - 2x$.
    // Room to work on the exam; on the key, only as tall as the answer.
    #answer-box(
```

```
      width: 100%,
      height: solutions-only(auto, otherwise: 3cm),
    )[#solution[$f'(x) = 3x^2 - 2$]]
  ]
  #question[Is $f$ increasing at $x = 1$?]
])
```




The `solutions-only` function behaves in the following way:

**`solutions-only(x)`** `x` on the key, and nothing on the exam.
**`solutions-only(x, otherwise: y)`** `x` on the key, and `y` on the exam.

# 8 Rubrics

Rubrics indicate how points should be awarded when grading a question, and examy provides a `#rubric(points: ..)[..]` for that purpose. By default, rubrics are shown in the margin and take up no space in the exam. They can be nested in groups, and they can be authored anywhere in an exam and will point back to the specific location where they are authored.

All rubric items in a particular question/part/subpart are grouped together and shown in a single box in the margin. Rubric items set in the `rubric: ..` option of a question/part/subpart will not have a traceback line. To hide or show the traceback line of a single rubric item, give it `trace: false` or `trace: true`; `config`'s `rubric-trace` sets this for all of them.

```
#show: exam.with(show-rubrics: auto, rubric-width: 4.8cm)

#question(points: 1, rubric: [
  #rubric(points: 1)[Name]
  Make sure it is correctly spelled.
])[
  Spell your name.
]

#question(points: 3, rubric: [
  #rubric(points: 0)[Not a derivative]
  #rubric(points: 0)[Blank]
])[
  Compute $f'(x)$ for $f(x) = x^2 e^x$.
  #solution[
    By the product rule,#rubric(points: 2)[Uses the product rule]
    $f'(x) = 2x e^x + x^2 e^x = x e^x (x + 2)$.#rubric(points: 1)[Factors]
  ]
]
```

EXAM

1. (1 point) Spell your name.
2. (3 points) Compute $f'(x)$ for $f(x) = x^2 e^x$.

ANSWER KEY

1. (1 point) Spell your name.
2. (3 points) Compute $f'(x)$ for $f(x) = x^2 e^x$.

   By the product rule, $f'(x) = 2xe^x + x^2 e^x = xe^x(x + 2)$.

**Question 1** (1 point)
+1 Name
*Make sure it is correctly spelled.*

**Question 2** (3 points)
+2 Uses the product rule
+1 Factors
0 Not a derivative
0 Blank

A rubric item's `points` can be positive or negative. No accounting/checking is done by examy, and it is the author's responsibility to make sure that rubric points add up to the correct total.

## 8.1 Visibility

Rubrics are hidden by default and can be shown by setting `show-rubrics` on the global `config`.

`show-rubrics: false` (default) Rubrics are hidden.
`show-rubrics: true` Rubrics are shown.
`show-rubrics: auto` Rubrics are shown whenever solutions are shown.

As with solutions, a command-line setting can be used to override the exam's settings.

```
# the answer key, with its rubric, for the graders
typst compile --input show-solutions=true --input show-rubrics=true exam.typ
```

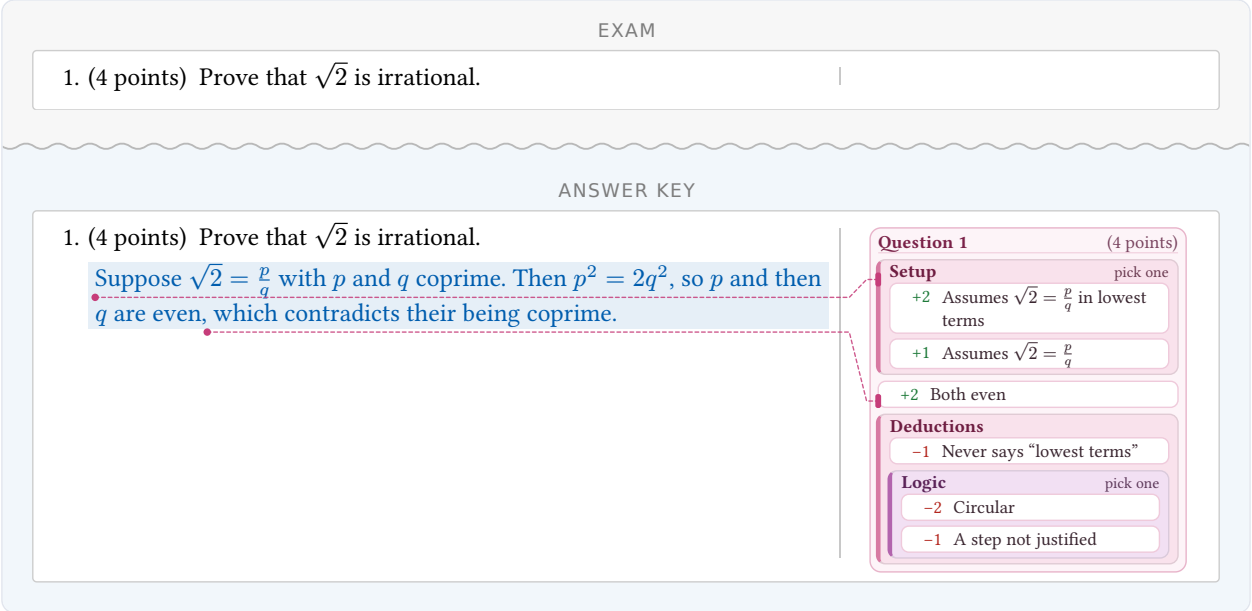
## 8.2 Groups

A `#rubric[..]` that contains other `#rubric[..]`s creates a group. Its first paragraph is the group's title and the `#rubric`s inside are items. Any other non-rubric text will show up as a note.

A rubric's `annotation` argument supplies a short note for the grader, useful for instructions like `annotation: [pick one]` on a group of mutually exclusive rubric items.

```
#show: exam.with(show-rubrics: auto, rubric-width: 4.8cm)

#question(points: 4, rubric: [
  #rubric[*Deductions*
    #rubric(points: -1)[Never says "lowest terms"]
    #rubric(annotation: [pick one])[Logic
      #rubric(points: -2)[Circular]
      #rubric(points: -1)[A step not justified]
    ]
  ]
])[
  Prove that $sqrt(2)$ is irrational.
  #solution[
    #rubric(annotation: [pick one])[Setup
      #rubric(points: 2)[Assumes $sqrt(2) = p/q$ in lowest terms]
      #rubric(points: 1)[Assumes $sqrt(2) = p/q$]
    ]Suppose $sqrt(2) = p/q$ with $p$ and $q$ coprime. Then
    $p^2 = 2q^2$, so $p$ and then $q$ are even,#rubric(points: 2)[Both even]
    which contradicts their being coprime.
  ]
]
```




## 8.3 Customizing the position of rubrics

The `rubric-position` option sets where each rubric is placed.

`"margin-right"` (default) Place the rubric in the right margin.
`"margin-left"` Place the rubric in the left margin.
`"end"` Place the rubric at the end of the question/part/subpart where the rubric is placed.

When set in the margins, a rubric takes up no additional space. When `rubric-position: "end"` is set, the rubrics will take up space on the page, so a rubric vs. no-rubric rendering will have different spacings.

```
#show: exam.with(show-rubrics: auto, rubric-position: "end")

#question(points: 2, rubric: [
  #rubric(points: 1)[States the hypotheses]
  #rubric(points: 1)[States the conclusion]
])[
  State the Mean Value Theorem.
  #answer-box(width: 100%, height: 1cm)[#solution[Let $f:[a,b] -> RR$ ...]]
]
#question(points: 1, rubric: rubric(points: 1)[Anything drawn])[
  Draw a picture.
]
```

EXAM

1. (2 points) State the Mean Value Theorem.

2. (1 point) Draw a picture.

ANSWER KEY

1. (2 points) State the Mean Value Theorem.

   Let $f : [a, b] \to \mathbb{R}$ ...

   **Question 1** (2 points)

   +1 States the hypotheses

   +1 States the conclusion

2. (1 point) Draw a picture.

   **Question 2** (1 point)

   +1 Anything drawn

## 8.4 Listing and exporting rubrics

To display all rubrics in a list, use the `#rubric-list()` function. It always displays the rubrics, regardless of whether `show-rubrics` is set.

```
#show: exam

#question(points: 2, rubric: [
  #rubric(points: 1)[Uses $A = pi r^2$]
  #rubric(points: 1)[Gets $9 pi$]
])[What is the area of a circle of radius 3?]
#question(points: 3, rubric: [
  #rubric(points: 2)[Uses the chain rule]
  #rubric(points: 1)[Simplifies]
  #rubric(points: -1, annotation: [once only])[Sign error]
])[Differentiate $sin(x^2)$.]

*Grading guide*
#rubric-list()
```

1. (2 points) What is the area of a circle of radius 3?
2. (3 points) Differentiate $\sin(x^2)$.

**Grading guide**

**Question 1** (2 points)

- +1 Uses $A = \pi r^2$
- +1 Gets $9\pi$

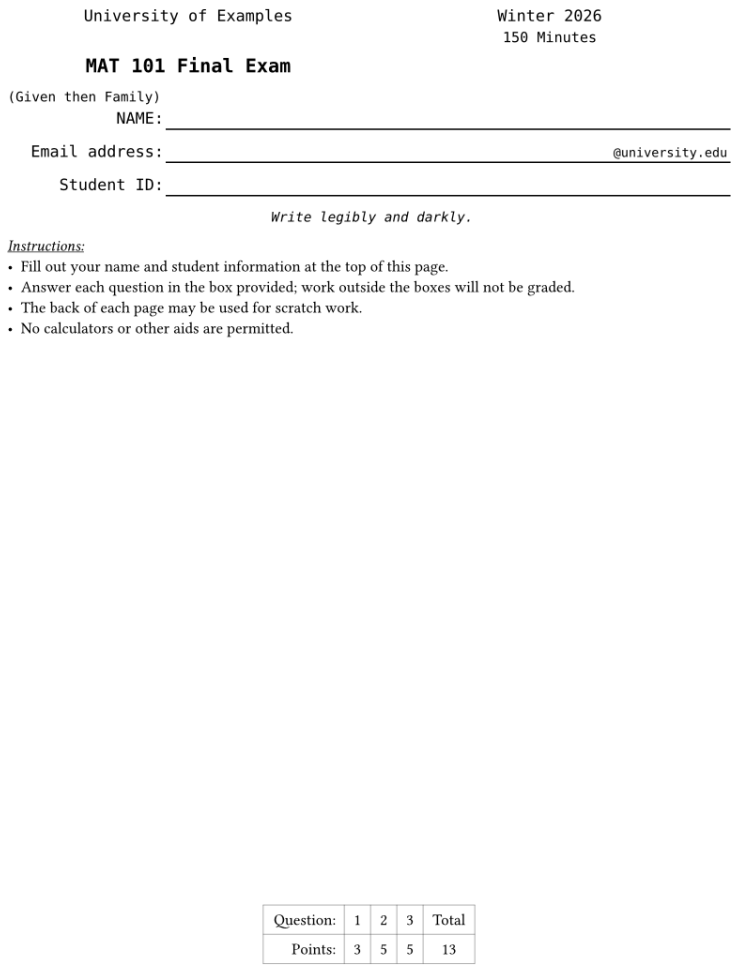
**Question 2** (3 points)

- +2 Uses the chain rule
- +1 Simplifies
- −1 Sign error — once only

Rubrics can also be exported as JSON data, or attached to the exam's PDF for a grading tool (see Exam metadata).

# 9 Cover pages

A cover page is ordinary content placed before the first question. Examy provides `#maketitle()`, `#name-block()`, and `#points-table()`, which may be used to construct a typical cover page.



University of Examples

Winter 2026
150 Minutes

**MAT 101 Final Exam**

(Given then Family)
NAME: _____

Email address: _____ @university.edu

Student ID: _____

*Write legibly and darkly.*

*Instructions:*
- Fill out your name and student information at the top of this page.
- Answer each question in the box provided; work outside the boxes will not be graded.
- The back of each page may be used for scratch work.
- No calculators or other aids are permitted.

| Question: | 1 | 2 | 3 | Total |
|---|---|---|---|---|
| Points: | 3 | 5 | 5 | 13 |

The cover of `examples/final-exam.typ`.

## 9.1 Title blocks

A `#maketitle()` renders the exam's details, which are set with the global `config`. It includes the `institution`, `exam-name`, `term`, and `duration`. Each detail can be overridden by passing arguments directly to `#maketitle()` (e.g. `#maketitle(term: [Summer 2026])`).

The exam's `duration` can be a Typst `duration` such as `duration(minutes: 150)`, or content.

```
#show: exam.with(
  institution: [University of Examples],
  exam-name: [MAT 101 Final Exam],
  term: [Winter 2026],
  duration: duration(minutes: 150),
)

#maketitle()
```

University of Examples

Winter 2026

150 Minutes

**MAT 101 Final Exam**

## 9.2 Name blocks

A `#name-block()` renders a fill-in block, with a Name and a Student ID row by default.

> *Note.* `#maketitle()` and `#name-block()` are typeset in the document's font. Use a `set` rule to change the font, e.g. `#{ set text(font: "DejaVu Sans Mono"); name-block() }`.

### 9.2.1 Customizing

Name blocks can be customized by setting the `fields: (..)` argument. The `fields` argument expects an array of content or dictionaries consisting of `(prefix: .., suffix: ..)`. `(prefix: .., suffix: ..)` items will show before and after an underline.

```
#set text(font: "DejaVu Sans Mono")

// The default block: a Name row and a Student ID row.
#name-block()

#v(1.5em)

// Custom rows.
#name-block(fields: (
  (prefix: [#text(size: .85em)[(Given then Family)] \ NAME:]),
  (prefix: [Email address:], suffix: [`@university.edu`]),
  (prefix: [Student ID:]),
  {
    set align(center)
    text(size: .85em)[_Write legibly and darkly._]
  },
))
```

Name: _____

Student ID: _____

(Given then Family)
NAME: _____

Email address: _____ @university.edu

Student ID: _____

*Write legibly and darkly.*

### 9.2.2 Presets

Institution-specific presets are available in `presets`. For example, the University of Toronto standard can be typeset with `#name-block(fields: presets.utoronto.name-fields)`.

## 9.3 Putting together a cover page

With `#show: exam` at the top of the document, the cover page is part of the exam, so the options given to the exam apply to it too.

```
#show: exam.with(
  institution: [University of Examples],
  exam-name: [MAT 101 Final Exam],
  term: [Winter 2026],
  duration: duration(minutes: 150),
)

// The title and name blocks in the monospaced font bundled with Typst;
// the questions keep the document's font.
#{
  set text(font: "DejaVu Sans Mono")
  maketitle()
  name-block(fields: (
    (prefix: [#text(size: .85em)[(Given then Family)] \ NAME:]),
    // ..more fields
  ))
}

#underline[_Instructions:_]
- Fill out your name and student information at the top of this
  page.
- Answer each question in the box provided; work outside the
  boxes will not be graded.
- No calculators or other aids are permitted.

#v(1fr)
#align(center, points-table())
#pagebreak()

#question[..]
```

The repository's examples include a minimal quiz, a complete exam with custom name fields, and an exam using the University of Toronto preset.

# 10 Exam metadata

Examy can embed data about the rubric and questions of an exam directly into the compiled PDF as an attached JSON file with Markdown strings. This data can then be read by tools that process the PDF.

> *Note.* Typst syntax is converted to Markdown and math is converted to LaTeX when embedded into the PDF. This is primarily applicable to embedded rubric data.

## 10.1 Embedding metadata in the PDF

The global `config`'s `embed-metadata` option can be used to attach data to the PDF as JSON files.

```
#show: exam.with(embed-metadata: true)
```

`embed-metadata: false` (default) Nothing is attached.

`embed-metadata: "rubric"` The rubrics are attached as `examy-rubric.json`.

`embed-metadata: "question-regions"` Where each question is on the page is attached as `examy-regions.json`.

`embed-metadata: true` Both are attached.

`embed-metadata: auto` Question regions are always attached and rubric information is only attached when solutions are shown.

As with solutions, a command-line setting can be used to override the exam's settings. Its value is written without quotes.

```
# the answer key, with its rubrics attached, for a grading tool
typst compile --input show-solutions=true --input embed-metadata=true exam.typ
```

Attaching data does not change the layout of the exam. If a document holds several exams, the second exam's files are named `examy-rubric-2.json` and `examy-regions-2.json`, and so on.

> *Warning.* If `"rubric"` data is embedded in an exam, it can be read even when `show-solutions: false` is set. This means that if a student has access to a version of the exam without solutions but with the rubric metadata, they might have access to information that you didn't want to share. Use `embed-metadata: auto` to attach the rubrics only when solutions are shown.

## 10.2 Rubric data

Rubric data for an exam can be embedded regardless of whether rubrics are shown. Alternatively, rubric data can be accessed directly via the command line or displayed in a document with a query command.

```
typst eval --in exam.typ --pretty 'query(<examy-rubric>).first().value' > examy-rubric.json
```

```
#show: exam.with(exam-name: "Example Exam")
#set text(size: 0.8em)
```

```
#question(points: 2, rubric: [
  #rubric(points: 1)[Uses $A = pi r^2$]
  #rubric(points: 1)[Gets $9 pi$]
])[What is the area of a circle of radius 3?]
#question(points: 3, rubric: [
  #rubric(points: 1)[Simplifies]
  #rubric(points: -1, annotation: [once only])[Sign error]
])[Differentiate $sin(x^2)$.]

*Rubric as JSON*
#context {
  // Get the rubric data as JSON
  let data = query(<examy-rubric>).first().value
  raw(json.encode(data), lang: "json", block: true)
}
```

1. (2 points) What is the area of a circle of radius 3?
2. (3 points) Differentiate $\sin(x^2)$.

**Rubric as JSON**


```
{
  "format": "examy-rubric",
  "exam-name": "Example Exam",
  "rubrics": [
    {
      "address": [
        0
      ],
      "display-name": "1",
      "level": 1,
      "points": 2,
      "bonus-points": null,
      "rubric": [
        {
          "type": "item",
          "points": 1,
          "description": "Uses $A = \\pi r^{2}$",
          "annotation": null
        },
        {
          "type": "item",
          "points": 1,
          "description": "Gets $9 \\pi$",
          "annotation": null
        }
      ]
    },
    {
      "address": [
        1
      ],
      "display-name": "2",
      "level": 1,
      "points": 3,
      "bonus-points": null,
      "rubric": [
        {
          "type": "item",
          "points": 1,
          "description": "Simplifies",
          "annotation": null
        },
        {
          "type": "item",
          "points": -1,
          "description": "Sign error",
          "annotation": "once only"
        }
      ]
    }
  ]
}
```


Using `.first().value` retrieves the rubric from the first exam in the document (if your document only has one exam, then the first exam will be your exam), and `.at(1).value`, `.at(2).value`, etc. can be used to retrieve rubrics from other exams.

A TypeScript schema for the embedded `examy-rubric.json` is shown below. Questions with no points value will be given a `null` value for the `points` field.

```ts
export type Question = {
  /**
   * Position among its siblings at each level, from 0. E.g. `[0, 2]`
   * for question 1 (c).
   */
  "address": number[];
  /** Label as printed on the exam. E.g. `"1 (c)"`. */
  "display-name": string;
  /** `1` for a question, `2` for a part, and `3` for a subpart. */
  "level": 1 | 2 | 3;
  /**
   * Points given to it directly, not counting its parts; `null` if
   * none are given.
   */
  "points": number | null;
  /**
   * Bonus points given to it directly, not counting its parts; `null`
   * if none are given.
   */
  "bonus-points": number | null;
};

export type RubricItem = {
  "type": "item";
  /**
   * Points the item awards (negative for a deduction); `null` if none
   * are given.
   */
  "points": number | null;
  /** Markdown representation of the item's text, with LaTeX math. */
  "description": string;
  /**
   * Markdown representation of the item's annotation, with LaTeX
   * math; `null` if none.
   */
  "annotation": string | null;
};

export type RubricNote = {
  "type": "note";
  /** Markdown representation of the note's text, with LaTeX math. */
  "text": string;
};

export type RubricGroup = {
  "type": "group";
  /**
   * Markdown representation of the group's title (its first
   * paragraph), with LaTeX math.
   */
  "title": string;
  /**
   * Markdown representation of the text from the title to the first
   * item; `null` if none.
   */
  "note": string | null;
  /**
   * Markdown representation of the group's annotation. E.g.
   * `"pick one"`; `null` if none.
   */
```

```
  "annotation": string | null;
  /** The items, groups, and notes in the group, in order. */
  "items": RubricEntry[];
};

/** An entry of a rubric: an item, a group of entries, or a note. */
export type RubricEntry = RubricItem | RubricGroup | RubricNote;

/** The rubric of a question/part/subpart. */
export type Rubric = Question & {
  /** The rubric's items, groups, and notes, in order. */
  "rubric": RubricEntry[];
};

export type ExamyRubric = {
  /**
   * Identifies the data. An incompatible later version will have
   * another `format`.
   */
  "format": "examy-rubric";
  /**
   * Markdown representation of the global `config`'s `exam-name`;
   * `null` if it is not set.
   */
  "exam-name": string | null;
  /** Each question/part/subpart that has a rubric, in order. */
  "rubrics": Rubric[];
};
```

## 10.3 Question regions

Data about the region that each question occupies can be embedded in the PDF as `examy-regions.json` or retrieved from the command line via a Typst query command.

```
typst eval --in exam.typ --pretty 'query(<examy-regions>).first().value' > examy-regions.json
```

All positions are in points from the top left of the page (rounded to one decimal place). Question regions overlap the regions of their parts, and the regions for `#answer-box[..]`s contained in a question/part/subpart are included in an `answer-boxes` array.

If any `name-block` lines are present, their regions are included in `name-block-regions`.

```
#show: exam.with(exam-name: "Example Exam", embed-metadata: true)
#set text(size: 0.8em)

#name-block()

#question(points: 2, rubric: [
  #rubric(points: 1)[Uses $A = pi r^2$]
  #rubric(points: 1)[Gets $9 pi$]
])[What is the area of a circle of radius 3?]
#question(points: 3, rubric: [
  #rubric(points: 1)[Simplifies]
  #rubric(points: -1, annotation: [once only])[Sign error]
])[
  Differentiate $sin(x^2)$.
  #answer-box(width: 100%, height: 1cm)[]
```

```
]

*Regions as JSON*
#context {
  // Get the regions data as JSON
  let data = query(<examy-regions>).first().value
  // Two columns: the first is filled to this height, then the second
  block(height: 17.5cm, columns(2, raw(json.encode(data), lang: "json", block: true)))
}
```

*Its output is shown in the online documentation.*

A TypeScript schema for the embedded `examy-regions.json` is shown below. `Question` is the same as in the rubric data.

```
/**
 * A point on a page, in points (pt) from the page's top left, to a
 * tenth of a point.
 */
export type Position = {
  /** Page number, counting from 1. */
  "page": number;
  "x": number;
  "y": number;
};

export type Question = {
  /**
   * Position among its siblings at each level, from 0. E.g. `[0, 2]`
   * for question 1 (c).
   */
  "address": number[];
  /** Label as printed on the exam. E.g. `"1 (c)"`. */
  "display-name": string;
  /** `1` for a question, `2` for a part, and `3` for a subpart. */
  "level": 1 | 2 | 3;
  /**
   * Points given to it directly, not counting its parts; `null` if
   * none are given.
   */
  "points": number | null;
  /**
   * Bonus points given to it directly, not counting its parts; `null`
   * if none are given.
   */
  "bonus-points": number | null;
};

/** A page the exam is on, in points (pt). */
export type Page = {
  /** Page number, counting from 1. */
  "page": number;
  "width": number;
  "height": number;
  /**
   * The page's margins. With mirrored margins, `left` and `right` are
   * this page's.
   */
  "margin": { "top": number; "bottom": number; "left": number; "right": number };
```

```
};

/** A line of a `#name-block()`. */
export type NameBlockRegion = {
  /**
   * Markdown representation of the line's prefix. E.g. `"Name:"`;
   * `null` if none.
   */
  "display-name": string | null;
  /** Top left of the line. */
  "start": Position;
  /** Bottom right of the line. */
  "end": Position;
};

/** An answer box, to the outside of its outline as it is drawn. */
export type AnswerBox = {
  /** Top left of the box. */
  "top": Position;
  /** Bottom right of the box. */
  "bottom": Position;
};

/** Where a question/part/subpart is on the page. */
export type QuestionRegion = Question & {
  /** Top left of its label. */
  "start": Position | null;
  /**
   * Bottom right of everything in it, its parts and answer boxes too.
   */
  "end": Position | null;
  /** The answer boxes in its own text, not its parts'. */
  "answer-boxes": AnswerBox[];
};

export type ExamyRegions = {
  /**
   * Identifies the data. An incompatible later version will have
   * another `format`.
   */
  "format": "examy-regions";
  /**
   * Markdown representation of the global `config`'s `exam-name`;
   * `null` if it is not set.
   */
  "exam-name": string | null;
  /** Each page the exam is on, in order. */
  "pages": Page[];
  /** Each line of the exam's `#name-block()`, in order. */
  "name-block-regions": NameBlockRegion[];
  /** Each question/part/subpart, in order. */
  "question-regions": QuestionRegion[];
};
```

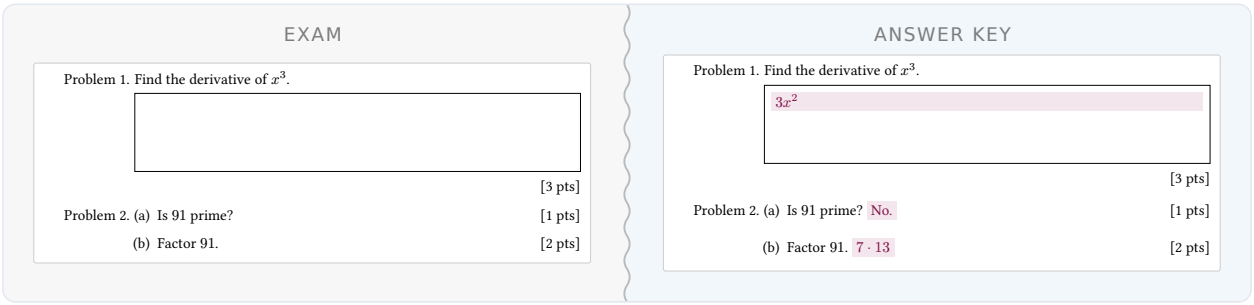
# 11 Configuration

Global `config` options can be set either directly on an exam, using `#show: exam.with(..)` or using elembic's `#show: e.set_(config, ..)` mechanism.

```
// Instead of using exam.with(..), you could uncomment the following lines
//
// #show: e.set_(config,
//   numbering: ("Problem 1.", "(a)"),
//   points-position: "end",
//   points-format: p => [[#p pts]],
//   solution-color: maroon,
// )

#show: exam.with(
  numbering: ("Problem 1.", "(a)"),
  points-position: "end",
  points-format: p => [[#p pts]],
  solution-color: maroon,
)

#question(points: 3)[
  Find the derivative of $x^3$.
  #answer-box(width: 100%)[#solution[$3x^2$]]
]
#question[
  #part(points: 1)[Is $91$ prime? #solution[No.]]
  #part(points: 2)[Factor $91$. #solution[$7 dot 13$]]
]
```

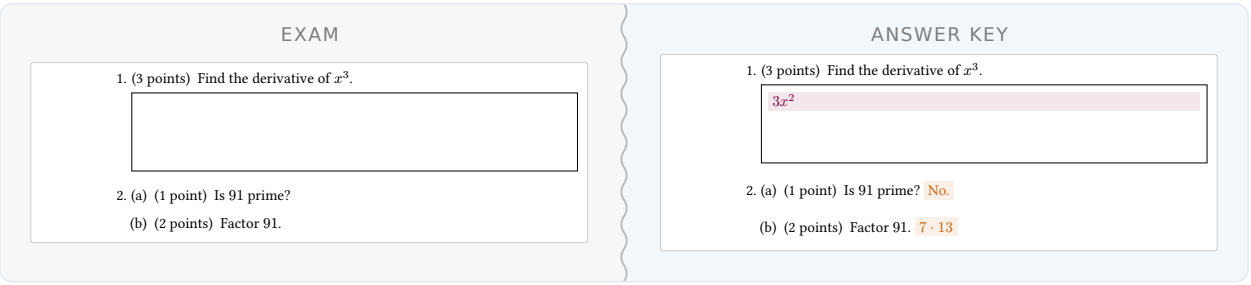



Inside of an exam, values can be set with elembic's `#show: e.set_(..)` mechanism. Such values will affect the exam from that point onwards.

> *Warning.* A few options cannot be changed in the middle of an exam: `numbering-full` (it would make references ambiguous), `strings`, and `show-rubrics`. Set these with `#show: exam.with(..)`.

```
#show: exam.with(
  solution-color: maroon,
)
```

```typ
#question(points: 3)[
  Find the derivative of $x^3$.
  #answer-box(width: 100%)[#solution[$3x^2$]]
]
// Solution color will change from now onward
#show: e.set_(config, solution-color: orange.darken(20%))
#question[
  #part(points: 1)[Is $91$ prime? #solution[No.]]
  #part(points: 2)[Factor $91$. #solution[$7 dot 13$]]
]
```




## 11.1 Reading config values

The global config can be read with elembic's `e.get(get => ..)` mechanism. The following example shows how you might create a fully custom cover page with the exam's name and term.

```typ
#show: exam.with(
  exam-name: [MAT 101 Quiz 3],
  term: [Fall 2026],
)

#e.get(get => [
  *#text(fill: blue)[#get(config).exam-name]*, #get(config).term
])
```

**MAT 101 Quiz 3**, Fall 2026

> *Note.* To check whether solutions are shown, use `#show-solutions(get)` instead of reading the value directly from `config` (see Solutions). This ensures that settings passed on the command line via `--input show-solutions=..` are taken into account.

## 11.2 All `config` options

### Solutions

**`show-solutions`** `bool` default `true`

Whether to show solutions. The `--input show-solutions=..` command-line option overrides this setting.

**solution-color** default: `auto` | color

The primary color for solutions; i.e., the color of the text of a solution or the outline and check mark of a filled-in bubble. Setting to `auto` derives it from `solution-background-color`.

**solution-background-color** default: `auto` | color

The background color for a solution. Setting to `auto` derives it from `solution-color` by lightening.

### Cover pages

**institution** default: `none` | content

The institution name shown by `#maketitle()`.

**exam-name** default: `none` | content

The name of the exam (e.g. Midterm 1) shown by `#maketitle()`.

**term** default: `none` | content

The term of the exam (e.g. Fall 2026) shown by `#maketitle()`.

**duration** default: `none` | duration | content

The length of the exam shown by `#maketitle()`. `Duration`s are shown in minutes (e.g. `duration(hours: 1, minutes: 50)` shows as "110 Minutes"). Pass content for complete control over what is shown (e.g., pass `[1 h 50 min]`).

### Numbering and indentation

**numbering** `str | function | (str | function)[]` default: `("1.", "(a)", "i.")`

Determines how questions, parts, and subparts are numbered. This option behaves like `enum`'s `numbering`, taking a numbering pattern (e.g. `"1.a.i."`) or a function that generates a label. In addition, `numbering` also accepts an array of numbering patterns (e.g. `("Problem 1.", "(a)", "i.")`), with the left-most pattern applying to the highest level (almost always the `#question[..]`).

A `config` rule inside an exam (e.g. `#show: e.set_(config, numbering: ("Bonus 1.", "(i)"))`) sets the numbering of the questions/parts/subparts after it in its scope. Its left-most pattern still applies to questions, even when the rule is inside a question.

**numbering-full** `bool` default: `false`

Similar to `enum`'s `full`, if set, ensures that questions/parts/subparts get globally unique identifiers (e.g., part (a) of question 1 is labelled "1 (a)" by default, or "1.a." with `numbering: "1.a.i."`). This also affects `@name` references to questions.

**indent** length default: 1.5em

How far each question/part/subpart is indented from the question/part it is in (a question, from the left edge of the text).

## Points

**points-format** default: auto | function

A function `(points) => [content]` that determines how points are visibly displayed. Defaults to a function that renders `(X points)`. Other examples include `p => [[#p]]` or `p => [(#p Punkte)]`.

**bonus-points-format** default: auto | function

A function `(bonus-points) => [content]` that determines how bonus points are visibly displayed. Defaults to a function that renders `(X bonus points)`.

**strings** dictionary(content) default:

```
(
  question: [Question],
  page: [Page],
  points: [Points],
  bonus: [Bonus],
  score: [Score],
  total: [Total],
)
```

Words that can be localized to different languages. These include `question`, `page`, `points`, `bonus`, `score`, and `total`. Any subset of these may be set.

Note: the words in points badges are set by `points-format` and `bonus-points-format`.

**points-position** default: "start" | "margin-left" | "margin-right" | "end"

Determines where the points are visibly displayed.

`"start"`: before the question text.

`"margin-left"` or `"margin-right"`: in the page margin. Those of every column go in that margin, stacked.

`"end"`: right-aligned, after the question text.

**points-show-totals** bool default: false

When set to `true`, a question/part/subpart whose parts/subparts have points shows a total even when it is not given points itself: the sum of the points of its parts/subparts. One with no points of its own and none in its parts/subparts still shows none.

For example, `#question[#part(points: 2)[]]` shows `1. (2 points)`, with `(a) (2 points)` on the line below.

When set, a question that is assigned points while its parts/subparts are too shows a warning in red in place of its points.

**points-table-direction** default `"horizontal"` | `"vertical"`

Determines the layout of `#points-table()`: `"horizontal"` (a column per question) or `"vertical"` (a row per question).

**points-table-granularity** default `"question"` | `"part"` | `"subpart"` | `"page"`

How points are grouped into the cells of a `#points-table()`: one cell per `"question"` (the default), `"part"`, `"subpart"`, or `"page"`.

**points-table-score** `bool` default `false`

Whether `#points-table()` includes an empty row for a grader to fill in.

**points-table-bonus** `bool` default `false`

Whether `#points-table()` includes a row for the bonus points (only visible if the exam has bonus points).

**points-table-max-columns** default `none` | `int`

The maximum number of columns in a horizontal `#points-table()`. A wider table wraps to multiple rows. Use `none` for no limit.

## Answer boxes

**answer-box-height** default `auto` | `length` | `ratio` | `relative` | `fraction`

The default height of an `#answer-box[..]`. Setting to `auto` sizes a non-empty box to fit its contents, excluding `#solution[..]`s, and gives an empty box `answer-box-default-height`.

**answer-box-width** default `auto` | `length` | `ratio` | `relative` | `fraction`

The default width of an `#answer-box[..]`. Setting to `auto` makes a box `answer-box-default-width` wide, or full width if its height is fractional (e.g. `1fr`).

**answer-box-stroke** `length` | `color` | `gradient` | `tiling` | `stroke` | `dictionary` | `none` default `0.5pt`

The outline stroke of an `#answer-box[..]`, specified like a `box`'s `stroke` (e.g. `(bottom: .5pt)` for a single line to write on).

**answer-box-fill** default `none` | `color` | `gradient` | `tiling`

The background fill of an `#answer-box[..]`, specified like a `box`'s `fill` (e.g. `luma(245)`).

**answer-box-default-height** `length` default `2cm`

The height of an empty `#answer-box[..]` with an otherwise unspecified height.

**answer-box-default-width** `length` default: `2cm`

The width of an `#answer-box[..]` with a width of `auto`, unless its height is a fractional unit (e.g. `2fr`).

### Multiple choice

**bubble-size** default: `auto` `| length`

The diameter of bubbles drawn by `#choice[..]`, `#bubble()`, `#square-bubble()`, and `#choices[..]`. Setting to `auto` uses `1em` (which scales with the surrounding text).

**bubble-stroke** `length | color | gradient | tiling | stroke | dictionary` default: `0.1em`

The outline of bubbles drawn by `#choice[..]`, `#bubble()`, `#square-bubble()`, and `#choices[..]`, specified like a `circle`'s `stroke`: a thickness (e.g. `.15em`), a color (e.g. `blue`), or both (e.g. `.15em + blue`). A thickness left out is `.1em`, and a color left out is black. On the answer key, a correct bubble's outline takes the solution color.

**bubble-outset** `length | dictionary` default: `(right: 0.4em)`

The space around each bubble, specified like `#choice[..]`'s `outset`: a length (the gap after the bubble) or a dictionary of sides (`left`, `right`, `top`, `bottom`, `x`, `y`, `rest`).

**bubble-label-font** `str | array` default: `"DejaVu Sans Mono"`

The font of the labels inside bubbles (see `choices-enumerate`).

**choices-enumerate** `bool | str | function | array` default: `false`

Determines how `#choices[..]` labels its bubbles. Setting to `false` results in no labels; `true` results in A, B, C, …. A numbering pattern, function, or array of labels can be used (e.g., `"a"` or `("x", "y", "z")`).

**choices-inline** `bool` default: `false`

Show `#choices[..]` inline in a paragraph. If you would like a grid of choices, consider using `choices-columns` instead.

**choices-columns** default: `none` `| int`

Split `#choices[..]` into the specified number of columns. Setting to `none` results in a single column (i.e., each choice on its own line). Cannot be combined with `choices-inline`.

**choices-gap** default: `auto` `| length | fraction`

The space between items in a `#choices[..]` block. Setting to `auto` uses the usual line spacing for choices on separate lines, and `1.5em` for choices side by side. A fraction (e.g. `1fr`) spreads side-by-side choices across the line.

## Rubrics

**show-rubrics** `auto | bool` default `false`

Whether to show rubrics (see `#rubric[..]`). Setting to `auto` shows rubrics whenever solutions are shown. The `--input show-rubrics=..` command-line option overrides this setting.

**rubric-position** default `"margin-right"` | `"margin-left"` | `"end"`

Determines where rubrics are displayed.

`"margin-left"` or `"margin-right"`: in the page margin, next to the question/part/subpart they belong to (see `rubric-width`). Those of every column go in that margin, stacked.

`"end"`: after the question/part/subpart, taking up space on the page, so an exam with visible rubrics is laid out differently from one without.

**rubric-trace** default `auto` | `bool`

Whether to draw a dotted line from each rubric item written in a question's text to the rubric. Setting to `auto` draws lines only for rubrics in the margin. A rubric item's own `trace` overrides this.

**rubric-width** default `auto` | `length`

The width of rubrics in the margin. Setting to `auto` fills the margin.

## Exam metadata

**embed-metadata** `bool | auto | "rubric" | "question-regions"` default `false`

Whether to embed extra metadata into the PDF (as attached JSON files). The `--input embed-metadata=..` command-line option overrides this setting.

`true`: Embed all metadata.

`false`: Don't embed metadata.

`auto`: Embed the question regions, and the rubric data only when solutions are shown.

`"rubric"`: Embed rubric data as `examy-rubric.json` (the same data as `<examy-rubric>`).

`"question-regions"`: Embed rectangles indicating where each question/part/subpart starts and ends, where its answer boxes are, where the lines of a `#name-block()` are, and the size and margins of each page, as `examy-regions.json`.

# 12 Technical details

Examy is written entirely in Typst, heavily relying on elembic for managing settings and options. Understanding this document is not needed for using examy, but might be of interest.

## 12.1 No nested boxes

A straightforward implementation of examy would wrap each question/part/subpart in a box and apply an appropriate inset. Unfortunately, `#pagebreak()` cannot work inside a box and, worse still, boxes can only negotiate fractional units if they are peers of each other. So, the box approach would make it impossible to have a part and a subpart both share the remaining space on the page.

Instead, examy flattens all question/part/subpart contents and tags the start and end of each block with `metadata`. The flat stream is then parsed, indentation level is tracked, and peer boxes are produced, so that `#pagebreak()` and `fr` units can negotiate space.

## 12.2 Step 1: Questions are annotated with metadata

`question`, `part`, and `subpart` don't draw anything themselves. Each one returns its body unchanged but between two invisible markers (`metadata` values) that record the points, number, label, rubric, etc. associated with the question/part/subpart.

Additionally, a test is inserted into the content stream. This test is later removed if the exam is well-formed, but it allows the query `typst eval --in exam.typ 'query(<examy-warning>)'` to report information about a malformed exam.

## 12.3 Step 2: Layout

`#exam[]` receives the question stream (with `metadata` tags already inserted). It then runs through the following steps.

**Tokenize** The question stream is scanned for examy's `metadata` and `#pagebreak()`s. The stream is split to give special handling to breaks as well as `set` and `show` rules. Warning markers are also removed from the stream. Everything else is kept whole. From here on, a question/part/subpart inside a box (e.g., list item, box, table, etc.) is an error.

**Parse** The tokens become a tree of questions/parts/subparts with their numbers, points, and options precomputed. This allows for a global analysis of the exam (e.g., for use in `#points-table()`) without appealing to `context` or `query`s.

**Plan** A question with no `#pagebreak()` and no `fr` answer box is drawn using nested blocks. Any other question is split into peer blocks, each indented to appear as if it is nested.

**Render** Each block is drawn with its number and points badge. Labelled questions/parts/subparts get a Typst label that can be used for `@name` references, and the point information recorded for `#points-table()`, `#num-points()`, etc.

*Note.* The downside of this parsing is that "click back" in tools like tinymist can be broken since the source stream has been parsed and reconstructed. (Usually it works to click a sub-boxed item, like an equation.)

## 12.4 Settings and elembic

The global `config` object and question references are controlled by elembic, which is re-exported as `e`.

**Settings** The global `config` is an elembic element that is never drawn; its fields are examy's options. `#show: e.set_(config, ..)` works like a Typst `set` rule, applying until the end of its block, and `#show: exam.with(..)` applies its options the same way. Options are read with `e.get`, which, like Typst's `context`, gives the values in effect where it is placed. This is why `#show-solutions(get)` takes `get`.

The exam reads its options where it starts. A `config` rule inside the exam applies to the questions in its scope, except for `numbering-full`, `strings`, and `show-rubrics`, which hold for the whole exam.

**References** Each labelled question/part/subpart leaves an invisible elembic element as its anchor. Elembic lets an element decide how references to it are drawn, so a reference can show "1 (a)" or just "(a)" depending on where it is. `#show: e.prepare()` sets this up. The exam applies it to its own content, so a document only needs it for references from outside the exam.

While `solution` and `answer-box` are elembic elements, `#exam[..]`, `#question[..]`, `#part[..]`, and `#subpart[..]` are plain Typst functions. However, during the *Render* phase of examy, these items are typeset using elembic elements (and so can respond to elembic's set rules).

# 13 Reference

To import all examy functions, use

```
#import "@preview/examy:1.1.0": *
```

A complete list is documented below.

The default value of each argument is shown in a blue box (e.g. default `false`). Many arguments take their default from the global `config`. Their box also shows a gear and the name of the `config` option they inherit from (e.g. default `auto` | `bubble-size`). Setting such an argument to `auto` uses the value of that `config` option, so changing the option changes the default everywhere.

> *Note.* Many arguments on individual elements can be set on the global `config` instead of specifying them on the element.

| | |
|---|---|
| `answer-box` | A box for students to write their answers in. |
| `bubble` | A round bubble (for "select one" multiple choice). |
| `choice` | A single bubble (round or square) followed by the choice it labels. |
| `choices` | Used for creating multiple-choice questions. |
| `config` | Global configuration of package options. |
| `e` | A re-export of elembic, mainly for `e.set_` and `e.get` (e.g. `#show: e.set_(config, ..)`). |
| `exam` | Define an exam with questions/parts/subparts. |
| `maketitle` | Displays the exam's title block (like LaTeX's `\maketitle`), using `config`'s `institution`, `exam-name`, `term`, and `duration`. |
| `name-block` | A block of lines for students to fill in their name and other details. |
| `num-points` | The total number of points in the exam: by default its regular points, not counting bonus and practice points. |
| `num-points-page` | The number of points on a page (of the questions/parts/subparts that start on it): by default its regular points, on the current page. |
| `num-questions` | The number of questions in the exam. |
| `page-score-box` | A box for the grader to write the page's score in (e.g. "Score: ____ / 7"). |
| `points-table` | A table of the points per question, with a total. |
| `presets` | Settings for specific institutions. |
| `question`, `part`, `subpart` | Declare a question, part, or subpart, numbered "1.", "(a)", and "i." respectively. |
| `rubric` | A rubric item, or a group of items, for grading a question/part/subpart. |
| `rubric-list` | Lists every rubric of the exam. |
| `show-solutions` | Returns whether solutions are shown. |

`solution` — A solution to a question/part/subpart, shown only when solutions are shown (see `config`'s `show-solutions`).

`square-bubble` — A square bubble, used for "select all that apply".

## `answer-box` element

```
#answer-box(
  body,
  solution: none,
  width: auto,
  height: auto,
  baseline: 50% - .33em,
  default-height: auto,
  default-width: auto,
  stroke: auto,
  fill: auto,
  radius: 0pt,
  outset: 0pt,
  clip: false,
)
```

A box for students to write their answers in.

### Arguments

**`body`** `content | str` *required · positional*

The content of the box (e.g. a prompt or a `#solution[..]`). Can be blank (e.g., `#answer-box[]`).

**`solution`** default: `none` | `content`

A solution shown at the end of the box (e.g. `#answer-box(solution: [42])[]`). This is equivalent to writing `#solution[..]` as the last item of content in `body`.

**`width`** default: `auto` (`answer-box-width`) | `length | ratio | relative | fraction`

The width of the box. Set to a fraction (e.g. `1fr`) to fill the rest of the line. Setting to `auto` uses `config`'s `answer-box-width`.

**`height`** default: `auto` (`answer-box-height`) | `length | ratio | relative | fraction | none`

The height of the box. Setting to `auto` uses `config`'s `answer-box-height`. Can be any length unit including fractional units (e.g. `1fr`), along with a special value:

`none`: if non-empty, fits the box's content, ignoring `#solution[..]`s, whatever `config`'s `answer-box-height` is. This value ensures an exam and answer key have the same layout.

An empty box with height `none` gets a height of `default-height`.

**`baseline`** `length | ratio | relative` default: `50% - .33em`

The vertical position of a box placed inline in a sentence.

**default-height** default: auto | answer-box-default-height | length

The height of the box when it is empty and its height is otherwise unspecified. Setting to `auto` uses `config`'s `answer-box-default-height`.

**default-width** default: auto | answer-box-default-width | length

The width of the box when its width is `auto`, unless its height is a fractional unit (e.g. `2fr`). Setting to `auto` uses `config`'s `answer-box-default-width`.

**stroke** default: auto | answer-box-stroke | length | color | gradient | tiling | stroke | dictionary | none

The outline stroke of the box, specified like a `box`'s `stroke` (e.g. `(bottom: .5pt)` for a single line to write on). Setting to `auto` uses `config`'s `answer-box-stroke`.

**fill** default: auto | answer-box-fill | color | gradient | tiling | none

The background color of the box, specified like a `box`'s `fill` (e.g. `luma(245)`). Setting to `auto` uses `config`'s `answer-box-fill`.

**radius** length | ratio | relative | dictionary default: 0pt

Same as a `box`'s `radius`.

**outset** length | ratio | relative | dictionary default: 0pt

Same as a `box`'s `outset`.

**clip** bool default: false

Same as a `box`'s `clip`.

## bubble alias

```
bubble
```

A round bubble (for "select one" multiple choice).

Alias for `choice.with(bubble: "circle")`; see `choice`.

## choice function

```
#choice(
  body,
  bubble: "circle",
  correct: false,
  outset: auto,
  size: auto,
```

```
  stroke: auto,
  solution-color: auto,
  solution-background-color: auto,
)
```

A single bubble (round or square) followed by the choice it labels. For a list of choices, use `#choices[..]`.

When solutions are shown, a `correct: true` bubble is filled in to indicate it is a correct choice.

### Arguments

**body** default: `none` | content | str | int | float · *positional*

The text of the choice, shown after the bubble. Leave it out (e.g. `#choice[]`) for a bubble on its own.

**bubble** default: `"circle"` | `"square"`

The shape of the bubble: `"circle"` for "select one", `"square"` for "select all that apply".

**correct** bool default: `false`

Whether this choice is correct; has an effect only when solutions are shown.

**outset** default: `auto` (`bubble-outset`) | length | dictionary

The space around the bubble, specified like a `box`'s `inset`. Sides given replace those of `config`'s `bubble-outset`; the others are kept. Setting to `auto` uses `config`'s `bubble-outset`.

**size** default: `auto` (`bubble-size`) | length

The diameter of a bubble. Setting to `auto` uses `config`'s `bubble-size`.

**stroke** default: `auto` (`bubble-stroke`) | length | color | gradient | tiling | stroke | dictionary

The outline of the bubble, specified like `config`'s `bubble-stroke` (e.g. `.15em`, `blue`, or `.15em + blue`). What it leaves out, its thickness or its color, is `config`'s. Setting to `auto` uses `config`'s `bubble-stroke`.

**solution-color** default: `auto` (`solution-color`) | color

The primary color for a correct bubble; i.e., the color of its outline, check mark, and text when filled in. Setting to `auto` uses `config`'s `solution-color`.

**solution-background-color** default: `auto` (`solution-background-color`) | color

The background color for a correct bubble when filled in. Setting to `auto` uses `config`'s `solution-background-color`.

## `choices` function

```
#choices(
  body,
  correct: none,
  bubble: "circle",
  inline: auto,
  columns: auto,
  gap: auto,
  size: auto,
  outset: auto,
  stroke: auto,
  label-font: auto,
  solution-color: auto,
  solution-background-color: auto,
  enumerate: auto,
)
```

Used for creating multiple-choice questions. Accepts a list, with each list item becoming a `#choice[..]` with a bubble. Correct choices start with `+` and incorrect choices with `-`.

For example, to specify a list of choices, `a`, `b`, `c`, with `c` being the correct choice, use:

```
#choices[
 - a
 - b
 + c
]
```

When solutions are shown, the correct choices are filled in and checked.

### Arguments

**`body`** `content | array` *required · positional*

The choices, as a list (one `-` .. item per choice, or `+` .. for a correct one) or the choices as an array of content.

**`correct`** default: `none` `| int | str | content | array`

The correct choice, by index (`0`-indexed) or, when the bubbles are labelled (see `enumerate`), by label (e.g. `"A"`), or an array of indices/labels. Additive with choices marked with `+`.

**`bubble`** default: `"circle"` `| "square"`

The shape of the bubbles: `"circle"` for "select one", `"square"` for "select all that apply".

**`inline`** default: `auto` (`choices-inline`) `| bool`

Show the choices inline in a paragraph. If you would like a grid of choices, consider using `columns` instead. Setting to `auto` uses `config`'s `choices-inline`.

Note: `inline` and `columns` are taken together: giving either one ignores both of `config`'s `choices-inline` and `choices-columns`.

**`columns`** default: `auto` (`choices-columns`) | `int` | `none`

Split the choices into the specified number of columns. Setting to `none` results in a single column (i.e., each choice on its own line). Cannot be combined with `inline`. Setting to `auto` uses `config`'s `choices-columns`.

**`gap`** default: `auto` (`choices-gap`) | `length` | `fraction`

The space between the choices. A fraction (e.g. `1fr`) spreads side-by-side choices across the line. Setting to `auto` uses `config`'s `choices-gap`.

**`size`** default: `auto` (`bubble-size`) | `length`

The diameter of the bubbles. Setting to `auto` uses `config`'s `bubble-size`.

**`outset`** default: `auto` (`bubble-outset`) | `length` | `dictionary`

The space around each bubble, specified like `#choice[..]`'s `outset`. Sides given replace those of `config`'s `bubble-outset`; the others are kept. Setting to `auto` uses `config`'s `bubble-outset`.

**`stroke`** default: `auto` (`bubble-stroke`) | `length` | `color` | `gradient` | `tiling` | `stroke` | `dictionary`

The outline of the bubbles, specified like `config`'s `bubble-stroke` (e.g. `.15em`, `blue`, or `.15em + blue`). What it leaves out, its thickness or its color, is `config`'s. Setting to `auto` uses `config`'s `bubble-stroke`.

**`label-font`** default: `auto` (`bubble-label-font`) | `str` | `array`

The font of the labels inside the bubbles, relevant when `enumerate: true` is set. Setting to `auto` uses `config`'s `bubble-label-font`.

**`solution-color`** default: `auto` (`solution-color`) | `color`

The primary color for the correct bubbles; i.e., the color of their outlines, check marks, and text when filled in. Setting to `auto` uses `config`'s `solution-color`.

**`solution-background-color`** default: `auto` (`solution-background-color`) | `color`

The background color for the correct bubbles when filled in. Setting to `auto` uses `config`'s `solution-background-color`.

**`enumerate`** default: `auto` (`choices-enumerate`) | `bool` | `str` | `function` | `array`

Determines how the bubbles are labelled. Setting to `false` results in no labels; `true` results in A, B, C, .... A numbering pattern, function, or array of labels can be used (e.g., `"a"` or `("x", "y", "z")`). Setting to `auto` uses `config`'s `choices-enumerate`.

## `config` element

```
config
```

Global configuration of package options. Give options to the exam, e.g. `#show: exam.with(show-solutions: true)`, or set them with `#show: e.set_(config, show-solutions: true)`.

### Fields

**`show-solutions`** `bool` default: `true`

Whether to show solutions. The `--input show-solutions=..` command-line option overrides this setting.

**`institution`** default: `none` | `content`

The institution name shown by `#maketitle()`.

**`exam-name`** default: `none` | `content`

The name of the exam (e.g. Midterm 1) shown by `#maketitle()`.

**`term`** default: `none` | `content`

The term of the exam (e.g. Fall 2026) shown by `#maketitle()`.

**`duration`** default: `none` | `duration` | `content`

The length of the exam shown by `#maketitle()`. `Duration`s are shown in minutes (e.g. `duration(hours: 1, minutes: 50)` shows as "110 Minutes"). Pass content for complete control over what is shown (e.g., pass `[1 h 50 min]`).

**`show-rubrics`** `auto` | `bool` default: `false`

Whether to show rubrics (see `#rubric[..]`). Setting to `auto` shows rubrics whenever solutions are shown. The `--input show-rubrics=..` command-line option overrides this setting.

**`rubric-position`** default: `"margin-right"` | `"margin-left"` | `"end"`

Determines where rubrics are displayed.

`"margin-left"` or `"margin-right"`: in the page margin, next to the question/part/subpart they belong to (see `rubric-width`). Those of every column go in that margin, stacked.

`"end"`: after the question/part/subpart, taking up space on the page, so an exam with visible rubrics is laid out differently from one without.

**rubric-trace** default `auto` | bool

Whether to draw a dotted line from each rubric item written in a question's text to the rubric. Setting to `auto` draws lines only for rubrics in the margin. A rubric item's own `trace` overrides this.

**rubric-width** default `auto` | length

The width of rubrics in the margin. Setting to `auto` fills the margin.

**embed-metadata** `bool | auto | "rubric" | "question-regions"` default `false`

Whether to embed extra metadata into the PDF (as attached JSON files). The `--input embed-metadata=..` command-line option overrides this setting.

`true`: Embed all metadata.

`false`: Don't embed metadata.

`auto`: Embed the question regions, and the rubric data only when solutions are shown.

`"rubric"`: Embed rubric data as `examy-rubric.json` (the same data as `<examy-rubric>`).

`"question-regions"`: Embed rectangles indicating where each question/part/subpart starts and ends, where its answer boxes are, where the lines of a `#name-block()` are, and the size and margins of each page, as `examy-regions.json`.

**solution-color** default `auto` | color

The primary color for solutions; i.e., the color of the text of a solution or the outline and check mark of a filled-in bubble. Setting to `auto` derives it from `solution-background-color`.

**solution-background-color** default `auto` | color

The background color for a solution. Setting to `auto` derives it from `solution-color` by lightening.

**bubble-stroke** `length | color | gradient | tiling | stroke | dictionary` default `0.1em`

The outline of bubbles drawn by `#choice[..]`, `#bubble()`, `#square-bubble()`, and `#choices[..]`, specified like a `circle's stroke`: a thickness (e.g. `.15em`), a color (e.g. `blue`), or both (e.g. `.15em + blue`). A thickness left out is `.1em`, and a color left out is black. On the answer key, a correct bubble's outline takes the solution color.

**bubble-label-font** `str | array` default `"DejaVu Sans Mono"`

The font of the labels inside bubbles (see `choices-enumerate`).

**bubble-size** default `auto` | length

The diameter of bubbles drawn by `#choice[..]`, `#bubble()`, `#square-bubble()`, and `#choices[..]`. Setting to `auto` uses `1em` (which scales with the surrounding text).

**bubble-outset** `length | dictionary` default `(right: 0.4em)`

The space around each bubble, specified like `#choice[..]`'s `outset`: a length (the gap after the bubble) or a dictionary of sides (`left`, `right`, `top`, `bottom`, `x`, `y`, `rest`).

**choices-enumerate** `bool | str | function | array` default `false`

Determines how `#choices[..]` labels its bubbles. Setting to `false` results in no labels; `true` results in A, B, C, .... A numbering pattern, function, or array of labels can be used (e.g., `"a"` or `("x", "y", "z")`).

**choices-inline** `bool` default `false`

Show `#choices[..]` inline in a paragraph. If you would like a grid of choices, consider using `choices-columns` instead.

**choices-columns** default `none` `| int`

Split `#choices[..]` into the specified number of columns. Setting to `none` results in a single column (i.e., each choice on its own line). Cannot be combined with `choices-inline`.

**choices-gap** default `auto` `| length | fraction`

The space between items in a `#choices[..]` block. Setting to `auto` uses the usual line spacing for choices on separate lines, and `1.5em` for choices side by side. A fraction (e.g. `1fr`) spreads side-by-side choices across the line.

**answer-box-height** default `auto` `| length | ratio | relative | fraction`

The default height of an `#answer-box[..]`. Setting to `auto` sizes a non-empty box to fit its contents, excluding `#solution[..]`s, and gives an empty box `answer-box-default-height`.

**answer-box-width** default `auto` `| length | ratio | relative | fraction`

The default width of an `#answer-box[..]`. Setting to `auto` makes a box `answer-box-default-width` wide, or full width if its height is fractional (e.g. `1fr`).

**answer-box-stroke** `length | color | gradient | tiling | stroke | dictionary | none` default `0.5pt`

The outline stroke of an `#answer-box[..]`, specified like a `box`'s `stroke` (e.g. `(bottom: .5pt)` for a single line to write on).

**answer-box-fill** default `none` `| color | gradient | tiling`

The background fill of an `#answer-box[..]`, specified like a `box`'s `fill` (e.g. `luma(245)`).

**answer-box-default-height** `length` default `2cm`

The height of an empty `#answer-box[..]` with an otherwise unspecified height.

**answer-box-default-width** `length` default: `2cm`

The width of an `#answer-box[..]` with a width of `auto`, unless its height is a fractional unit (e.g. `2fr`).

**numbering** `str | function | (str | function)[]` default: `("1.", "(a)", "i.")`

Determines how questions, parts, and subparts are numbered. This option behaves like `enum`'s `numbering`, taking a numbering pattern (e.g. `"1.a.i."`) or a function that generates a label. In addition, `numbering` also accepts an array of numbering patterns (e.g. `("Problem 1.", "(a)", "i.")`), with the left-most pattern applying to the highest level (almost always the `#question[..]`).

A `config` rule inside an exam (e.g. `#show: e.set_(config, numbering: ("Bonus 1.", "(i)"))`) sets the numbering of the questions/parts/subparts after it in its scope. Its left-most pattern still applies to questions, even when the rule is inside a question.

**indent** `length` default: `1.5em`

How far each question/part/subpart is indented from the question/part it is in (a question, from the left edge of the text).

**numbering-full** `bool` default: `false`

Similar to `enum`'s `full`, if set, ensures that questions/parts/subparts get globally unique identifiers (e.g., part (a) of question 1 is labelled "1 (a)" by default, or "1.a." with `numbering: "1.a.i."`). This also affects `@name` references to questions.

**strings** `dictionary(content)` default:

```
(
  question: [Question],
  page: [Page],
  points: [Points],
  bonus: [Bonus],
  score: [Score],
  total: [Total],
)
```

Words that can be localized to different languages. These include `question`, `page`, `points`, `bonus`, `score`, and `total`. Any subset of these may be set.

Note: the words in points badges are set by `points-format` and `bonus-points-format`.

**points-format** default: `auto` `| function`

A function `(points) => [content]` that determines how points are visibly displayed. Defaults to a function that renders `(X points)`. Other examples include `p => [[#p]]` or `p => [(#p Punkte)]`.

**bonus-points-format** default: `auto` `| function`

A function `(bonus-points) => [content]` that determines how bonus points are visibly displayed. Defaults to a function that renders `(X bonus points)`.

**points-position** default `"start"` | `"margin-left"` | `"margin-right"` | `"end"`

Determines where the points are visibly displayed.

`"start"`: before the question text.

`"margin-left"` or `"margin-right"`: in the page margin. Those of every column go in that margin, stacked.

`"end"`: right-aligned, after the question text.

**points-show-totals** `bool` default `false`

When set to `true`, a question/part/subpart whose parts/subparts have points shows a total even when it is not given points itself: the sum of the points of its parts/subparts. One with no points of its own and none in its parts/subparts still shows none.

For example, `#question[#part(points: 2)[]]` shows `1. (2 points)`, with `(a) (2 points)` on the line below.

When set, a question that is assigned points while its parts/subparts are too shows a warning in red in place of its points.

**points-table-direction** default `"horizontal"` | `"vertical"`

Determines the layout of `#points-table()`: `"horizontal"` (a column per question) or `"vertical"` (a row per question).

**points-table-granularity** default `"question"` | `"part"` | `"subpart"` | `"page"`

How points are grouped into the cells of a `#points-table()`: one cell per `"question"` (the default), `"part"`, `"subpart"`, or `"page"`.

**points-table-score** `bool` default `false`

Whether `#points-table()` includes an empty row for a grader to fill in.

**points-table-bonus** `bool` default `false`

Whether `#points-table()` includes a row for the bonus points (only visible if the exam has bonus points).

**points-table-max-columns** default `none` | `int`

The maximum number of columns in a horizontal `#points-table()`. A wider table wraps to multiple rows. Use `none` for no limit.

## `e` value

```
e
```

A re-export of elembic, mainly for `e.set_` and `e.get` (e.g. `#show: e.set_(config, ..)`). Use it rather than importing elembic yourself.

## `exam` function

```
#exam(questions, ..options)
```

Define an exam with questions/parts/subparts. This is the main entry point for examy, and is usually invoked as a show rule (e.g. `#show: exam` or `#show: exam.with(..)`).

This function automatically invokes `e.prepare()`, which elembic needs to make cross references work.

### Arguments

**`questions`** `content | function` *required · positional*

Content of the exam built out of Typst content and `#question[..]` blocks.

Alternatively, a function `(solutions-only) => [..]` that returns the content. `solutions-only(x, otherwise: y)` gives `x` if solutions are shown and `y` otherwise. Unlike `#solution[..]`, which returns a Typst `context`, `solutions-only()` returns what it is given and so can be used anywhere.

**`..options`** *any number*

Any of `config`'s options (e.g. `#show: exam.with(show-solutions: false)`).

## `maketitle` function

```
#maketitle(
  institution: auto,
  exam-name: auto,
  term: auto,
  duration: auto,
)
```

Displays the exam's title block (like LaTeX's `\maketitle`), using `config`'s `institution`, `exam-name`, `term`, and `duration`. Displays nothing if all four are `none`.

Note: use a `set` rule to change the font settings (e.g. `#{ set text(font: "DejaVu Sans Mono"); maketitle() }`).

### Arguments

**`institution`** default: `auto` (⚙ `institution`) | `content` | `none`

The institution name. Setting to `auto` uses `config`'s `institution`; `none` hides it.

**`exam-name`** default: `auto` (⚙ `exam-name`) | `content` | `none`

The name of the exam (e.g. Midterm 1). Setting to `auto` uses `config`'s `exam-name`; `none` hides it.

**`term`** default: `auto` (⚙ `term`) | `content` | `none`

The term of the exam (e.g. Fall 2026). Setting to `auto` uses `config`'s `term`; `none` hides it.

**duration** default: `auto` (`duration`) | duration | content | none

The length of the exam. `Duration`s are shown in minutes (e.g. `duration(hours: 1, minutes: 50)` shows as "110 Minutes"). Pass content for complete control over what is shown (e.g., pass `[1 h 50 min]`). Setting to `auto` uses `config`'s `duration`; `none` hides it.

## name-block function

```
#name-block(
  title: none,
  fields: ((prefix: [Name:]), (prefix: [Student ID:])),
)
```

A block of lines for students to fill in their name and other details.

Note: it uses the document's font. To change it, use a `set` rule (e.g. `#{ set text(font: "DejaVu Sans Mono"); name-block() }`).

### Arguments

**title** default: `none` | content

A heading shown centered above the block.

**fields** (dictionary | content)[] default: `((prefix: [Name:]), (prefix: [Student ID:]))`

The rows of the block. A dictionary `(prefix: .., suffix: ..)` (both optional) draws the prefix, a line to write on, and the suffix. Any other content is shown as is, as a row of its own.

## num-points function

```
#num-points(intent: none)
```

The total number of points in the exam: by default its regular points, not counting bonus and practice points. Must be used inside of a context block, e.g. `#context num-points()` to show it, or `#context calc.ceil(num-points() * .6)` to compute with it.

### Arguments

**intent** default: `none` | "practice" | "bonus"

Which points to count, by their `intent` (see `#question[..]`): `none` for regular points, `"bonus"` for bonus points, or `"practice"` for practice points.

## num-points-page function

```
#num-points-page(page: auto, intent: none)
```

The number of points on a page (of the questions/parts/subparts that start on it): by default its regular points, on the current page. Must be used inside of a context block, e.g. in a page footer, `#set page(footer: context [#num-points-page() points])`.

### Arguments

**page** default: `auto` | `int`

The physical page number (counted from 1). Setting to `auto` uses the current page.

**intent** default: `none` | `"practice"` | `"bonus"`

Which points to count, by their `intent` (see `#question[..]`): `none` for regular points, `"bonus"` for bonus points, or `"practice"` for practice points.

## num-questions function

```
#num-questions()
```

The number of questions in the exam. Must be used inside of a context block, e.g. `#context num-questions()`.

## page-score-box function

```
#page-score-box()
```

A box for the grader to write the page's score in (e.g. "Score: ____ / 7"). Meant for a page footer (e.g. `#set page(footer: align(right, page-score-box()))`). Shows nothing on a page without points. The word "Score" can be localized with `config`'s `strings`.

## points-table function

```
#points-table(
  direction: auto,
  granularity: auto,
  score: auto,
  bonus: auto,
  max-columns: auto,
)
```

A table of the points per question, with a total.

### Arguments

**direction** default: `auto` | `points-table-direction` | `"horizontal"` | `"vertical"`

Determines the layout of the table: `"horizontal"` (a column per question) or `"vertical"` (a row per question). Setting to `auto` uses `config`'s `points-table-direction`.

**granularity** default `auto` | `points-table-granularity` | `"question"` | `"part"` | `"subpart"` | `"page"`

How points are grouped into table cells: one cell per `"question"` (`config`'s default), `"part"`, `"subpart"`, or `"page"`. Setting to `auto` uses `config`'s `points-table-granularity`.

**score** default `auto` | `points-table-score` | `bool`

Whether the table includes an empty row for a grader to fill in. Setting to `auto` uses `config`'s `points-table-score`.

**bonus** default `auto` | `points-table-bonus` | `bool`

Whether the table includes a row for the bonus points (only visible if the exam has bonus points). Setting to `auto` uses `config`'s `points-table-bonus`.

**max-columns** default `auto` | `points-table-max-columns` | `int` | `none`

The maximum number of columns in a horizontal table. A wider table wraps to multiple rows. Use `none` for no limit. Setting to `auto` uses `config`'s `points-table-max-columns`.

## `presets` value

```
presets
```

Settings for specific institutions. Currently only `presets.utoronto.name-fields`, the University of Toronto rows for `#name-block(fields: presets.utoronto.name-fields)`.

## `question`, `part`, `subpart` function

```
#question(
  body,
  points: none,
  intent: none,
  rubric: none,
  number: auto,
  height: auto,
  indent: auto,
  label: none,
)
#part(..)  // the same arguments
#subpart(..)  // the same arguments
```

Declare a question, part, or subpart, numbered "1.", "(a)", and "i." respectively. `#question[..]`, `#part[..]`, and `#subpart[..]` are aliases of the same function, taking the same arguments. The numbering style is chosen by nesting depth, not by function, so `#question[#part[..]]` and `#question[#question[..]]` will render identically.

### Arguments

**body** `content | str` *required · positional*

The text of the question/part/subpart. Can contain nested question/part/subparts.

**points** default: `none` `| int | float`

The number of points shown for this question/part/subpart (e.g. `points: 2` shows "(2 points)"). Points are also shown in `#points-table()` and related functions.

**intent** default: `none` `| "practice" | "bonus"`

Setting to `"bonus"` counts the points as bonus points, which are totaled separately. Setting to `"practice"` leaves the points out of the exam's totals and its points table; `num-points(intent: "practice")` counts them.

**rubric** default: `none` `| content`

Rubric items for the question/part/subpart (see `#rubric[..]`). Rubric items written here are shown after any `#rubric[..]` items in `body`.

**number** default: `auto` `| int | str | content | none`

By default, questions/parts/subparts automatically increment their number. Providing a `number` overrides this behaviour. A numerical value will cause numbering to skip to that value; a non-numerical value or general content will pause the counter and show that content. To omit a number completely, set `number: none`.

**height** default: `auto` `| length | ratio | relative | fraction`

The height of the question/part/subpart, including its parts. A length (e.g. `5cm`, or `50%` of the page), or a fraction (e.g. `1fr`), makes it a single block of that height, which cannot break across pages.

Note: prefer `#answer-box[..]` (e.g. `#answer-box(width: 100%, height: 3cm)[]`) over setting a question's height if your goal is to give students a space to write in.

**indent** default: `auto` `indent` `| length`

How far this question/part/subpart is indented from the question/part it is in (a question, from the left edge of the text). Setting to `auto` uses `config`'s `indent`.

**label** default: `none` `| label`

A label for referencing the question/part/subpart with `@name` (e.g. `label: <derivative>`).

## `rubric` function

```
#rubric(body, points: none, annotation: none, trace: auto)
```

A rubric item, or a group of items, for grading a question/part/subpart. Rubrics can be written in the question's `rubric` argument or anywhere in its body. Rubrics are only shown when `config`'s `show-rubrics` is set and by default take up no space.

A rubric containing other rubrics is a group: its first paragraph is the group's title, and the rubrics inside are its items. Other text is shown as a note.

#### Arguments

**`body`** `content | str` *required · positional*

The criterion for the item, or a group's title and items.

**`points`** default: `none` `| int | float`

The points for this rubric item (positive or negative).

**`annotation`** default: `none` `| content | str`

A short note for the grader, shown at the top of the rubric (e.g. `annotation: [pick one]`).

**`trace`** default: `auto` (`rubric-trace`) `| bool`

Whether to draw a dotted line from where this rubric item is written to the rubric. Setting to `auto` uses `config`'s `rubric-trace`.

An item inside a group is traced with its group: the line runs from where the group is written, and the group's `trace` decides whether it is drawn. The item's own `trace` is not used.

Note: items written in a question/part/subpart's `rubric` argument are never traced.

## `rubric-list` function

```
#rubric-list()
```

Lists every rubric of the exam. This function ignores the value of the global `show-rubrics` option.

## `show-solutions` function

```
#show-solutions(get)
```

Returns whether solutions are shown. Use this instead of reading `show-solutions` directly from `config`, since it takes the `--input show-solutions=..` command-line option into account. Useful for content that `#solution[..]` cannot wrap (e.g. to show a solution curve on a plot). Must be used with elembic's `get` function. E.g., `e.get(get => if show-solutions(get) [..])`.

Note: do not use it to show or hide a question/part/subpart. The exam cannot see a question/part/subpart inside `e.get`, so it is drawn as plain text, unnumbered, under a warning.

### Arguments

**get** `function` *required · positional*

The `get` function returned by elembic. E.g., `e.get(get => ..)`.

## solution element

```
#solution(
  body,
  width: none,
  color: auto,
  background-color: auto,
  otherwise: none,
)
```

A solution to a question/part/subpart, shown only when solutions are shown (see `config`'s `show-solutions`).

### Arguments

**body** `content` *required · positional*

The solution content.

**width** default: `none` | `auto` | `relative`

The width of the solution box. Setting to `auto` fits the content; `none` lets the surrounding content decide (e.g. a solution on its own in an `#answer-box[..]` fills its width).

**color** default: `auto` | `solution-color` | `color`

The color of the text of the solution. Setting to `auto` uses `config`'s `solution-color`.

**background-color** default: `auto` | `solution-background-color` | `color`

The background color for the solution. Setting to `auto` uses `config`'s `solution-background-color`.

**otherwise** default: `none` | `content`

Content shown in place of the solution when solutions are hidden. Can be used to show, for example, lined paper when no solution is visible.

Note: to provide space for a student to write an answer use `#answer-box[..]` instead.

## square-bubble alias

```
square-bubble
```

A square bubble, used for "select all that apply".

Alias for `choice.with(bubble: "square")`; see `choice`.

# 14 Changelog

What has changed in each version of examy, newest first. Before updating, read the version's breaking changes: they can make an exam written for an earlier version look different or fail to compile.

## 1.1.0 — 2026-11-09

### Added

- `embed-metadata` attaches data for grading tools such as Gradescope to the PDF: the rubrics (`"rubric"`, as `examy-rubric.json`), where each question, its answer boxes, and the name block's lines are on the page (`"question-regions"`, as `examy-regions.json`, also left as `<examy-regions>`), or both (`true`). With `auto`, the regions always, and the rubrics only when solutions are shown. `--input embed-metadata=..` sets it from the command line.
- Questions, parts, and subparts can go inside `#columns(..)`. A question's text outside the columns stays across the page. Points and rubrics in a margin go in the page's margin from any column, stacked there, in a page's own columns too; `fr` heights share what is left of their column; and a `#pagebreak()` inside the columns splits them, and the same columns go on on the next page.

### Breaking changes

- The rubric data, `<examy-rubric>`, has a new format: `(format, exam-name, rubrics)` instead of `(exam, questions)`. Each rubric gives its question's `address`, `display-name`, `level`, `points`, and `bonus-points` instead of `number` and `title`, and `points` counts only the points given to it directly, not its parts'. See the manual's "Exam metadata" page for the whole format.

### Fixed

- `#pagebreak(to: "odd")` and `#pagebreak(to: "even")` between or inside questions keep their `to`.
- `#set pagebreak(weak: true)` and `#set colbreak(weak: true)` apply to breaks between or inside questions.
- The text of every question, part, and subpart is a paragraph, in the PDF's tags, so that a screen reader reads each one on its own, and for `par` rules. It was one only when the exam drew something in the margin beside it, or the text had a paragraph break: so `show par` rules and `first-line-indent: (all: true)` now apply to every question, not only to some.

## 1.0.0 — 2026-10-28

### Breaking changes

- `points-table` is a function now. Write `#points-table()` instead of `#points-table`.
- `num-points` and `num-questions` are functions that return a number, and need context: write `#context num-points()` to show it, or compute with it, as in `#context calc.ceil(num-points() * .6)`.
- `maketitle` and `name-block` use the document's font instead of DejaVu Sans Mono. To keep the old look, set the font around them, e.g. `#{ set text(font: "DejaVu Sans Mono"); maketitle() }`.
- An answer box with no `height` fits its content instead of always being 2cm tall. Give it a `height` to keep room to write below a prompt.
- `question`, `part` and `subpart` no longer take `solution:`. Write `#solution[..]` in the body, where it should be drawn.
- `show-solutions` is `true` or `false` (`true` by default); `none` is no longer accepted.

- `exam` is a function, not an elembic element, so `e.set_(exam, ..)` and `e.show_(exam, ..)` no longer apply to it.

### Added

- Documentation site, https://siefkenj.github.io/typst-examy/, with a guide, a reference, and a playground for writing an exam in the browser.
- `#show: exam` makes the rest of the document the exam, and `exam` takes any `config` option: `#show: exam.with(show-solutions: true)`.
- Rubrics: `rubric` items and groups, drawn in the margin or after each question with `show-rubrics`, with a dotted line from where each item is written (`rubric-trace`, or an item's own `trace`), `rubric-list` for a grading guide, and the rubrics as data (`<examy-rubric>`) to export as JSON.
- Points: badges in the margin or at the end (`points-position`), drawn by `points-format` and `bonus-points-format`; totals of parts (`points-show-totals`); points tables by question, part, subpart or page (`granularity`), vertical, with Score and Bonus rows, split over several rows of columns; points per page (`num-points-page()`, `page-score-box`); and bonus and practice totals (`num-points(intent: "bonus")`).
- Numbering: `numbering` and `numbering-full`, and `strings` to translate the words examy prints. A `config` rule inside an exam can number some questions differently, e.g. a "Bonus 1." section.
- Answer boxes: set rules (`#show: e.set_(answer-box, ..)`), `stroke`, `fill`, `radius`, `outset` and `clip`, a fractional width (`width: 1fr`), and `config` options for every box's size and look (`answer-box-*`). Solutions in a box with no `height` take no space, so the key lays out like the exam.
- Solutions: `otherwise:`, shown in a solution's place on the exam, and their own `color` and `background-color`.
- Multiple choice: labels in the bubbles (`enumerate`, `choices-enumerate`), `correct:` by label (`correct: "C"`), and options for the bubbles' size, outline, spacing, label font and colors, and for the layout of every `choices` (`choices-inline`, `choices-columns`, `choices-gap`).
- A `config` rule inside an exam can change `numbering`, `points-format`, `bonus-points-format`, `points-position`, `points-show-totals`, the `rubric-*` options and `indent` for the questions after it, e.g. `#show: e.set_(config, points-position: "end")` before a section of the exam.
- Questions: `height`, `indent` (and `config`'s `indent`), and a warning in red where a question is drawn that no exam lays out.
- `duration` can be content, shown as written: `duration: [1 h 50 min]`.
- Every function checks its arguments, and an error names the argument and the types it takes.
- Template: `typst init @preview/examy`.

### Changed

- The badge of bonus points reads "(2 bonus points)".
- A points table labels each column as a reference to its question reads, and links to it.
- `answer-box`'s `solution:` is drawn like a `#solution[..]` at the end of the box, and no longer stretches to fill a box of set height.

### Deprecated

- `answer-box`'s `default_height` and `default_width`: use `default-height` and `default-width`.
- `presets.utoronto.name_fields`: use `presets.utoronto.name-fields`.
- `solution`'s `boxed`: ignored. Solutions are always drawn in a box.
- `config`'s `show-rubric` and `--input show-rubric=..`: use `show-rubrics`.
- `config`'s `solution-text-color`: use `solution-color`.

### Fixed

- In a document with several exams, such as an exam and its key in one bundle, point counts, points tables and references are about their own exam.

- A set rule inside a question no longer hides the parts after it from the exam.
- A reference in a question's text after one of its parts, between questions, or after the exam reads as from there: `@sub` is "(a) i" in question 1's text after part (a), not "i".
- `num-points-page` in a page footer no longer slows down a long exam.
- Many fixes to the layout of points badges, question numbers, answer boxes, solutions and equations.

### 0.4.0 — 2026-10-05

- Multiple-choice bubbles: `choice`, `choices`.
- Solutions in an answer box widen automatically to fill it.
- Fixed the spacing of points on a question with no body.

### 0.3.0 — 2026-09-19

- `fr` answer boxes on the same page share the leftover space equally.

### 0.2.0 — 2026-07-14

- Redesign of the package.