GROUP PROJECT 2.1, FLAVOR A

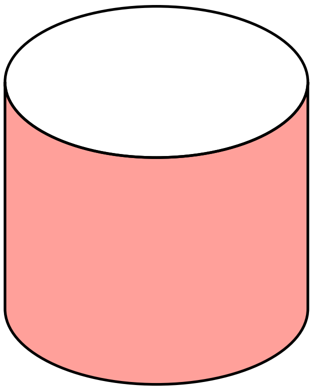
Some Group

Jane **Doe**
`12345678`

~~Jane~~John **Smith**
`10010010`

Alex **kʷikʷəƛ̓**
9999^{9999}

```
(
  "Jane Doe",
  "John Smith",
  "Alex kʷikʷəƛ\u{313}",
)
```



$12\,\mathrm{km}\ \lim\limits_{x\to 0}^{\text{wut}}$

---

1. (3 points) one
    (a) (Extra, no) two
        i. (1 point) three

User Manual: Drawing

Drawing

As we are doing math, inevitably we will need to draw some graphs. Typst has some native drawing abilities, but they are very limited. There is an ad hoc Typst drawing library, a package actually, called “cetz”, with its graphing companion “cetz-plot”. Simply

```
#import drawing: *
```

to let the template import them for you.

For general drawing techniques, refer to the cetz documentation. For graphing, download and refer to the cetz-plot manual.

There are other drawing packages available, but not imported by this template, here is a brief list:
- fletcher: nodes & arrows;
- jlyfish: Julia integration;
- neoplot: Gnuplot integration.

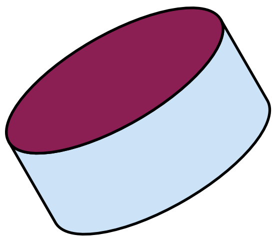
Find more visualization packages here.

Template Helpers

Besides importing the drawing packages, the `drawing` module also provides some helper functions.

For example, the `cylinder()` function draws an upright no-perspective cylinder.

```
#cetz.canvas({
  import cetz.draw: *
  group({
    rotate(30deg)
    cylinder(
      (0, 0), // Center
      (1.618, .6), // Radius: (x, y)
      2cm / 1.618, // Height
      fill-top: maroon.lighten(5%), // Top color
      fill-side: blue.transparentize(80%), // Side color
    )
  })
})
```



Other helps: `introduction`, `getting-started`, `setup`, `author`, `caveats`.

Author

The `author()` function is to be used as an argument of the `setup()` function, providing an author dictionary. It takes the first name, last name, and student number as arguments. For example,

```
#show: setup.with(
  author("Jane", "Doe", 12345678),
  // ...
)
```

supplies

```
{
  name: {
    first: "Jane",
    last: "Doe",
  },
  id: 12345678,
}
```

And in the PDF metadata there will be a “Jane Doe” in the authors field, student number not included.

What if your last name is kʷikʷəƛ̓, that happens to type…

```
k\u{02b7}ik\u{02b7}\u{0259}\u{019b}\u{0313}
```

Well, you still call `author()`:

```
author("Alex", "k\u{02b7}ik\u{02b7}\u{0259}\u{019b}\u{0313}", 12345678)
```

If the PDF viewer gets lucky, the name will show up in file information as normal, but more often then not, it becomes something like `Alex kʷikʷəƛ\u{313}` due to incompatibility. You are recommended to provide a plain text version of the full name with argument `strname` that must be a `str`:

```
author(
  "Alex",
  "k\u{02b7}ik\u{02b7}\u{0259}\u{019b}\u{0313}",
  12345678,
  strname: "Alex Coquitlam" // This sure will have no display issue.
)
```

If `strname` is set, it will be used in the PDF metadata instead of the displayed name.

In some more extreme cases, `strname` would be a necessity, rather than a backup. Take name Ga~~***lli***~~$\overline{\mathcal{leo}}$ as an example. The name is so special that it cannot be converted to plain text. In this case, you must provide a `strname` to avoid incomprehensible PDF metadata.

```
author(
  [#underline(text(fill: purple)[Ga])#strike[*_lli_*]#overline($cal("leo")$)],
  "Smith",
  12345678,
  strname: "Gallileo Smith"
)
```

Other helps: `introduction`, `getting-started`, `setup`, `caveats`, `drawing`.