Creating Slides

Abdul Saboor[1], Unknown Author[2]

Feb 12, 2023

[1]My University is somewhere in the middle of nowhere
[2]Their University is somewhere in the middle of nowhere

👉 **Read instructions in left panel**

Contents

⩊ Show Code

Introduction

To see how commands work, use `Slides.docs()` to see the documentation. Here we will focus on using all that functionality to create slides.

<div style="text-align:center">

📝 **Note**

This is inline markdown parsed by magic

</div>

Version: 3.2.6 as executed from below code in markdown.

Python

```python
1  # get the slides instance under a python block in Markdown file, we will use it later t
2  myslides = get_slides_instance()
3  import ipyslides as isd
4  version = myslides.version
5  %xmd #### This is inline markdown parsed by magic {.note .warning}
```

I was added at end by a given proxy, see the how it was done at the end of the slides

IPySlides Online Running Sources

> 📝 Note
>
> - [Edit on Kaggle](#)
> - Launch example Notebook `launch binder`
> - Watch a [Youtube Video](#)

---

1. Add references like this per slide. Use slides.cite() or in markdown cite`key` to add citations generally. ↵

Contents

IPython Display Objects

Any object with following methods could be in `write` command:

`_repr_pretty_`, `_repr_html_`, `_repr_markdown_`, `_repr_svg_`, `_repr_png_`, `_repr_jpeg_`, `_repr_latex_`, `_repr_json_`, `_repr_javascript_`, `_repr_pdf_` Such as IPython.display.[HTML,SVG,Markdown,Code] etc. or third party such as `plotly.graph_objects.Figure`.

Plots and Other Data Types

These objects are implemented to be writable in `write` command:

`matplotlib.pyplot.Figure`, `altair.Chart`, `pygal.Graph`, `pydeck.Deck`, `pandas.DataFrame`, `bokeh.plotting.Figure`, `IPython.display.Image` Many will be extentended in future. If an object is not implemented, use `display(obj)` to show inline or use library's specific command to show in Notebook outside write.

Interactive Widgets

Any object in `ipywidgets` <u>**Link to ipywidgtes right here using textbox command**</u>

or libraries based on ipywidgtes such as bqplot, ipyvolume, plotly's `FigureWidget` [1]

(reference at end) can be included as well.

Commands which do all Magic!

Slides.**write**(*objs, widths=None)

Write `objs` to slides in columns. To create rows in a column, wrap objects in a list or tuple.
You can optionally specify `widths` as a list of percentages for each column.

Write any object that can be displayed in a cell with some additional features:

- Strings will be parsed as as extended markdown that can have citations/python code blocks/Javascript etc.
- Display another function in order by passing it to a lambda function like `lambda: func()`. Only body of the function will be displayed/printed. Return value will be ignored.
- Dispaly IPython widgets such as `ipywidgets` or `ipyvolume` by passing them directly.
- Display Axes/Figure form libraries such as `matplotlib`, `plotly altair`, `bokeh`, `ipyvolume` ect. by passing them directly.
- Display source code of functions/classes/modules or other languages by passing them directly or using `Slides.source` API.
- Use `Slides.alt(widget, func)` function to display widget on slides and alternative content in exported slides/report, function should return possible HTML representation of widget.
- `ipywidgets.HTML` and its subclasses will be displayed as `Slides.alt(widget, html_converter_func)`. The value of exported HTML will be most recent.
- Other options include but not limited to:
  - Output of functions in `ipyslides.utils` module that are also linked to `Slides` object.
  - PIL images, SVGs etc.
  - IPython display objects such as Image, SVG, HTML, Audio, Video, YouTubeVideo, IFrame, Latex, Markdown, JSON, Javascript, etc.
  - Any object that has a `_repr_html_` method, you can create one for your own objects/third party objects by:
    - `Slides.serializer` API.
    - `IPython.core.formatters` API for third party libraries.

Table of Contents

Plotting with Matplotlib
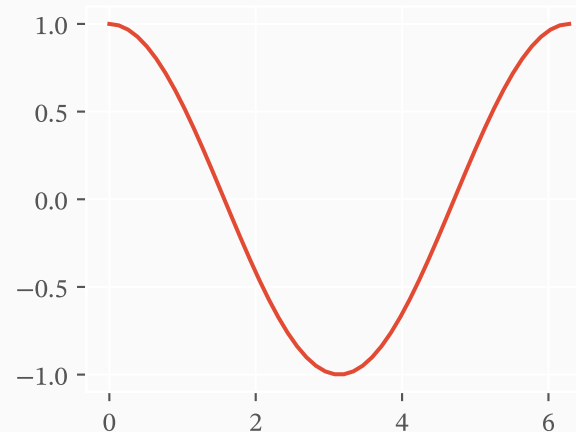
Python

```python
1  import numpy as np, matplotlib.pyplot as plt
2  plt.rcParams['svg.fonttype'] = 'none' # Global setting, enforce same fonts as presentat
3  x = np.linspace(0,2*np.pi)
4  with plt.style.context('ggplot'):
5      fig, ax = plt.subplots(figsize=(3.4,2.6))
6      _ = ax.plot(x,np.cos(x))
7  write([ax, s.focus_lines([1,3,4])])
```

Writing Pandas DataFrame

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Python

```python
try:
    import pandas as pd
    df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris
    df = df.describe() #Small for display
except:
    df = '### Install `pandas` to view output'
```

Writing Plotly Figure

## Install `plotly` to view output

Python

```python
1  try:
2      import plotly.graph_objects as go
3      fig = go.Figure()
4      fig.add_trace(go.Bar(y=[1,5,8,9]))
5  except:
6      fig = '### Install `plotly` to view output'
```

Interactive Apps with Widgets

Use `ipywidgets`, `bqplot`,`ipyvolume`, `plotly Figurewidget` etc. to show live apps like this!

> 💡 Tip
>
> Export to Slides/Report to see what happens to this slide and next slide!

Race Plot

A Silly Plot
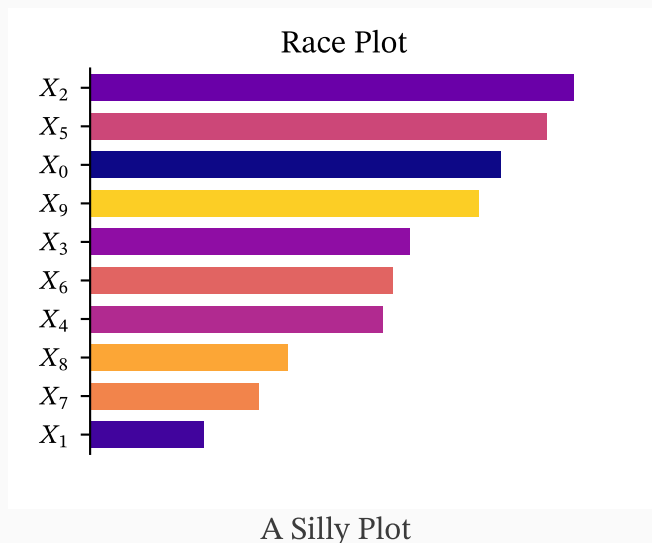
Python

```python
import ipywidgets as ipw

write('''
    ## Interactive Apps with Widgets sec
    Use `ipywidgets`, `bqplot`,`ipyvolum
    ::: note-tip
        Export to Slides/Report to see w
    ''')
plot_html = ipw.HTML('Plot will be here'
button = ipw.Button(description='Click n

write([plot_html,button], src)

def update_plot(btn):
    plot_html.value = race_plot().value

button.on_click(update_plot)
update_plot(None) #Initialize plot
```

Python

```python
1  def race plot():
```

Dynamic Content without Widgets

Use refresh button below to update plot! Compare with previous slide!
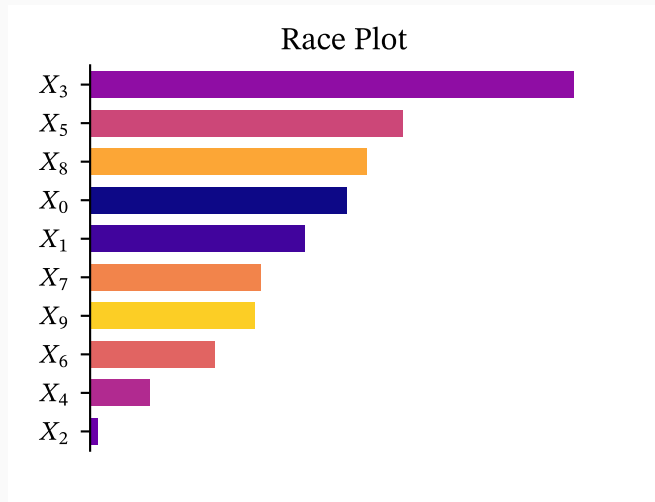
Race Plot

A Silly Plot

Python

```
1  write('''
2      ## Dynamic Content without Widgets
3      Use refresh button below to update p
4      ''')
5
6  def display_plot(): return race_plot().c
7
8  write(lambda: slides.on_refresh(display_
9  slides.source.from_callable(race_plot).c
```

Python

```
1  def race_plot():
2      import numpy as np
3      import matplotlib.pyplot as plt
4
5      x = np.linspace(0,0.9,10)
6      y = np.random.random((10,))
7      _sort = np.argsort(y)
8
9      plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd.value else 'de
10     with plt.style.context(plot_theme):
11         fig,ax = plt.subplots(figsize=(3.4,2.6))
12         ax.barh(x,y[_sort],height=0.07,color=plt.cm.get_cmap('plasma')(x[_sort]))
13
```

Contents

This is Slide 16.0

$$f(x) = \sin(x),\ 0 < x < 1$$

and we are animating matplotlib

Python

```python
1  fig, ax = plt.subplots()
2    + 5 more lines ...
```
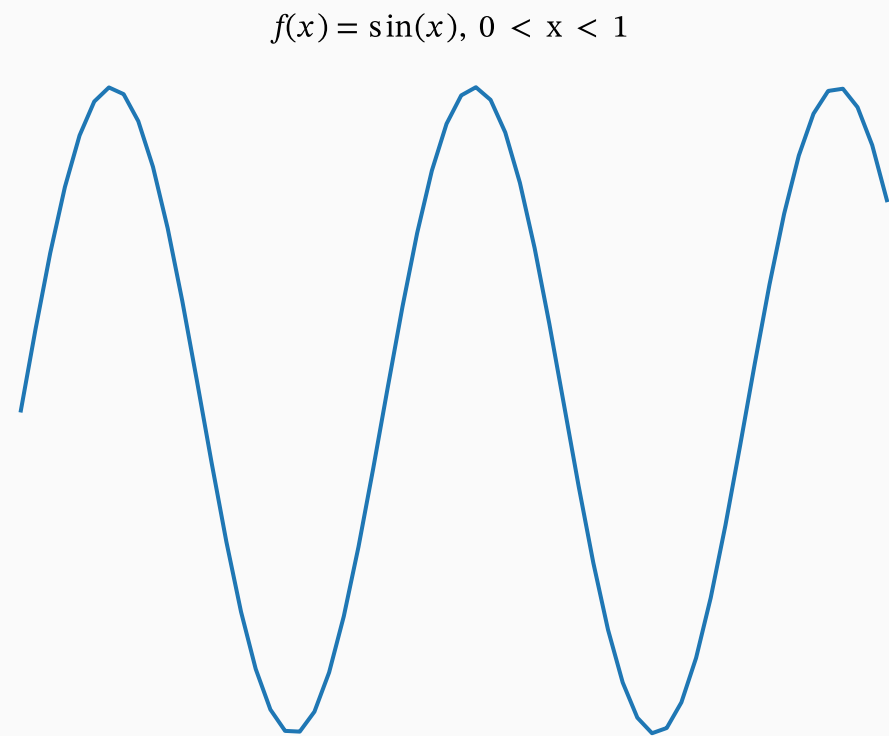

Python

```python
1    + 5 more lines ...
2  slides.notes.insert(f'## This is under @frames decorator!')
```
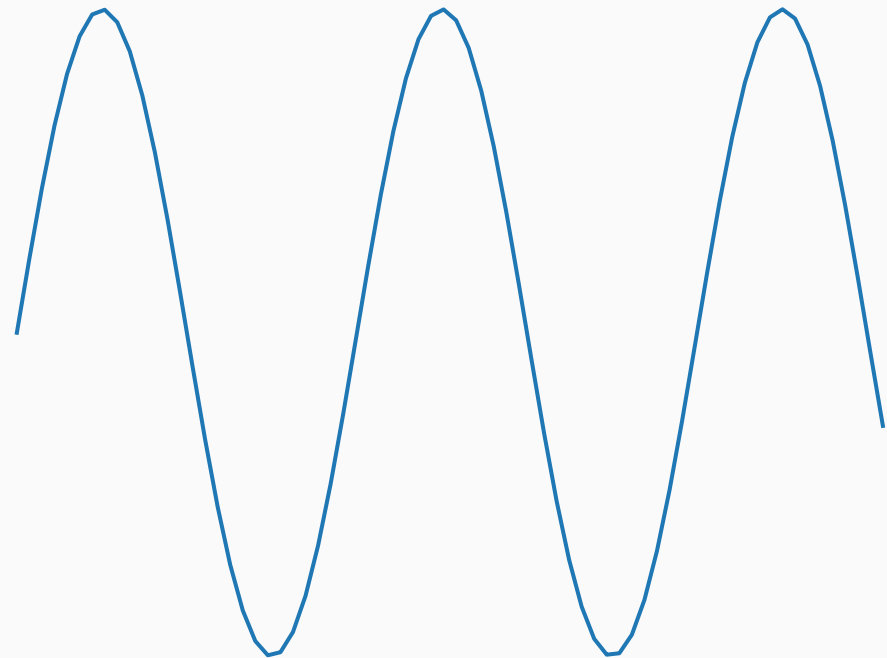
2

This is Slide 16.1

and we are animating matplotlib

Python

```
1    + 1 more lines ...
2  x = np.linspace(0,obj+1,50+10*(
3    + 4 more lines ...
```

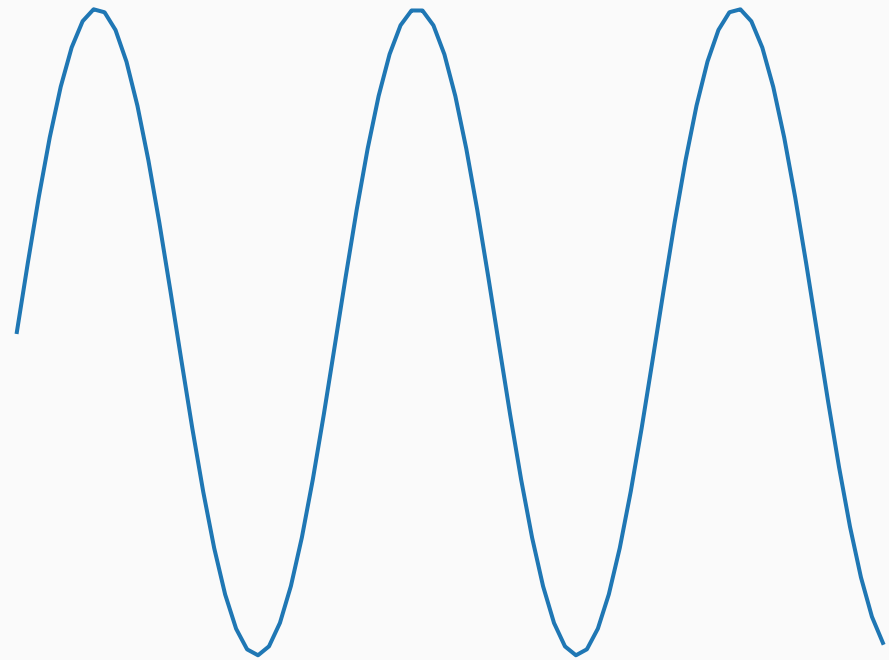$$f(x) = \sin(x),\ 0 < x < 2$$

2

This is Slide 16.2

and we are animating matplotlib

Python

```
1   + 2 more lines ...
2 ax.plot(x,np.sin(x));
3   + 3 more lines ...
```

$$f(x) = \sin(x),\ 0 < x < 3$$

2

This is Slide 16.3

and we are animating matplotlib

Python

```
1   + 3 more lines ...
2 ax.set_title(f'$f(x)=\sin(x)$,
3   + 2 more lines ...
```

$$f(x) = \sin(x), \; 0 < x < 4$$

2

This is Slide 16.4

and we are animating matplotlib

Python

```
1    + 4 more lines ...
2  ax.set_axis_off()
3    + 1 more lines ...
```
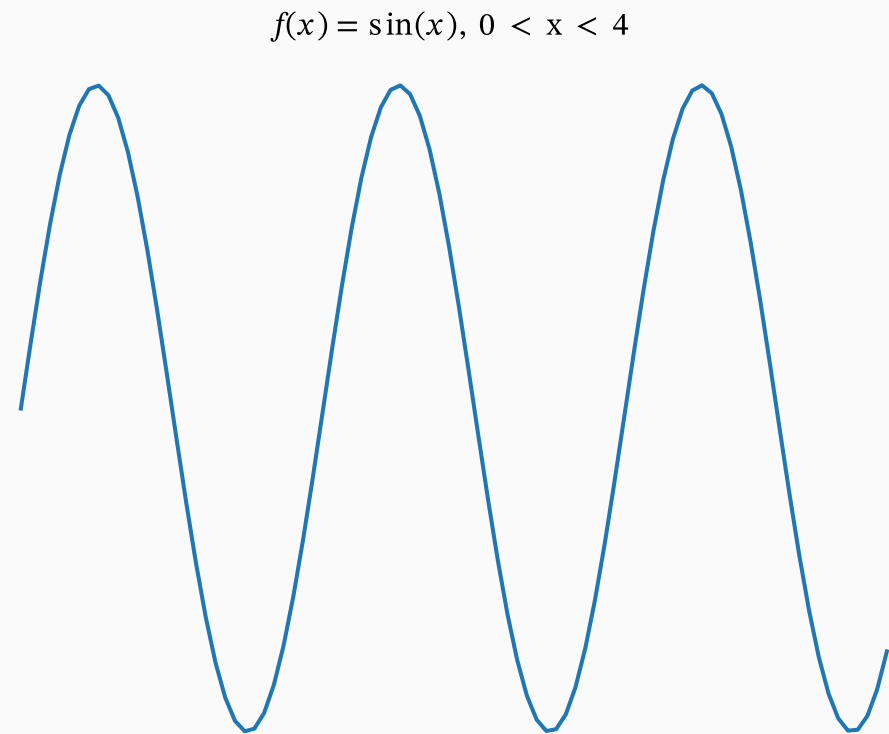
$f(x) = \sin(x),\ 0 < x < 5$
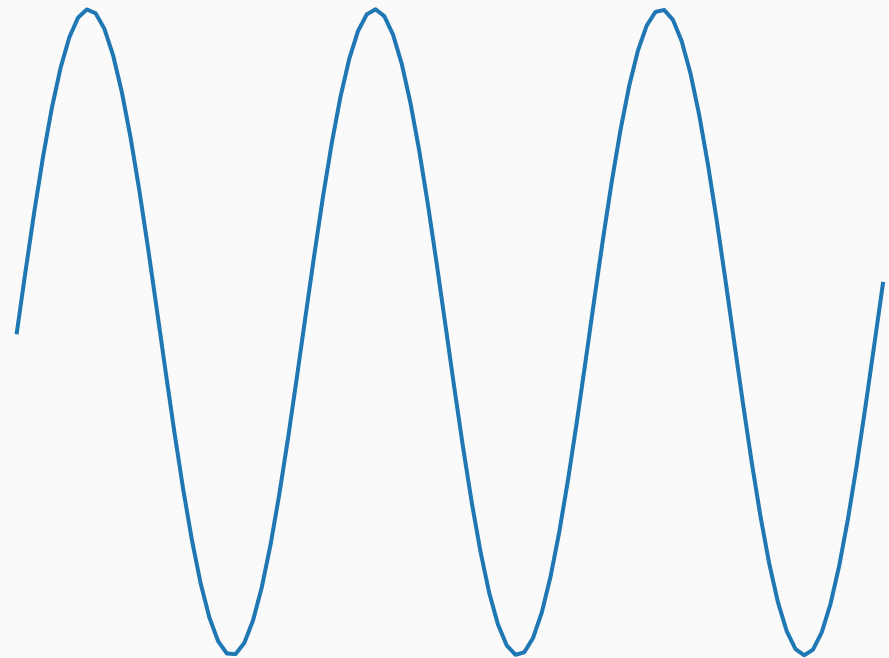
2

Contents

Frames with

`repeat = False`

1

Frames with

`repeat = False`

2

Frames with

`repeat = False`

3

Frames with

`repeat = False`

4

Frames with

repeat = True **and Fancy Bullet List**

💘 **1**

Frames with

repeat = True and Fancy Bullet List

💘 | 1

💘 | 2

Frames with

repeat = True and Fancy Bullet List

❤️ 1

❤️ 2

❤️ 3

Frames with

repeat = True and Fancy Bullet List

💘 | | 1 |
💘 | | 2 |
💘 | | 3 |
💘 | | 4 |

Frames with

repeat = [(0,1),(2,3)]

1	2

Python

```python
slides.write('# Frames with \n### `repeat = [(0,1),(2,3)]`')
slides.write(*obj)
```

Frames with

repeat = [(0,1),(2,3)]

3	4

Python

```python
1  slides.write('# Frames with \n### `repeat = [(0,1),(2,3)]`')
2  slides.write(*obj)
```

Displaying image from url from somewhere in Kashmir (كشمير)

Python

```python
backward_skipper.display()
forward_skipper.set_target()
slides.format_css({'.goto-button .fa.fa-minus': slides.icon('arrow',color='crimson',rot
slides.write('## Displaying image from url from somewhere in Kashmir color[crimson]`(كشمير
try:
    slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d61fe/master/pas
```

Watching Youtube Video?

Want to do some drawing instead? Click on pencil icon and draw something on <u>tldraw</u>!

IPySlides-Demo

Python

```python
 1  write(f"### Watching Youtube Video?")
 2  write('**Want to do some drawing instead?**\nClick on pencil icon and draw something on
 3
 4  write(YouTubeVideo('thgLGl14-tg',width='100%',height='266px'))
 5
 6  @slides.on_load
 7  def push():
 8      t = time.localtime()
 9      slides.notify(f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm_min:02}')
10      slides.set_overlay_url('https://tldraw.com')
11
12  ys.get_source().display()
```

Block API

New `block` API is as robust as `write` command. On top of it, it makes single unit of related content.

Table

h1	h2	h3
d1	d2	d3
r1	r2	r3

Widgets

☑ Select to do nothing

32

Click to do nothing

Python

```python
write('## Block API\nNew `block` API is as robust as `write` command. On top of it, it n
slides.block_red(
    [
        '### Table',
        '''
        |h1 |h2 |h3 |
        |---|---|---|
        |d1 |d2 |d3 |
        |r1 |r2 |r3 |
        ''',
    ],
    [
        '### Widgets',
        ipw.Checkbox(description='Select to do nothing',indent=False),
        slides.alt(ipw.IntSlider(),lambda w: f'<input type="range" min="{w.min}" max="{
```

$\LaTeX$ in Slides

> ⚠️ Alert
>
> Use $ $ or $$ $$ to display latex in Markdown, or embed images of equations $\LaTeX$ needs time to load, so keeping it in view until it loads would help.

$$\int_0^1 \frac{1}{1-x^2} dx$$

```
1  slides.write('## Built-in CSS styles')
2  slides.css_styles.display()
```

Built-in CSS styles

```
Use any or combinations of these styles in className argument of writing functions:
─────────────────────────────────────────────────────────────────────────────────

 className             | Formatting Style
═══════════════════════════════════════════════════════════════════════════════════

 'align-center'        | ———————Text———————
 'align-left'          | Text————————————
 'align-right'         | ————————————————Text
 'rtl'                 | ———————— اردو عربی
 'info'                | Blue text. Icon ⓘ  for note-info class.
 'tip'                 | Blue Text. Icon💡  for note-tip class.
 'warning'             | Orange Text. Icon ⚠️  for note-warning class.
 'success'             | Green text. Icon ✅  for note-success class.
 'error'               | Red Text. Icon⚡  for note-error class.
 'note'                | 📝  Text with note icon.
 'slides-only'         | Text will not appear in exported html report.
 'report-only'         | Text will not appear on slides. Use to fill content in report.
 'export-only'         | Hidden on main slides, but will appear in exported slides/report.
 'jupyter-only'        | Hidden on exported slides/report, but will appear on main slides.
 'page-break'          | Report will break page in print after object with this class.
 'block'               | Block of text/objects
 'block-[color]'       | Block of text/objects with specific background color from red,
                       | green, blue, yellow, cyan, magenta and gray.
 'raw-text'            | Text will not be formatted and will be shown as it is.
 'zoom-self'           | Zooms object on hover, when Zoom is enabled.
 'zoom-child'          | Zooms child object on hover, when Zoom is enabled.
 'no-zoom'             | Disables zoom on object when it is child of 'zoom-child'.
─────────────────────────────────────────────────────────────────────────────────
```

Contents

Serialize Custom Objects to HTML

This is useful for displaying user defined/third party objects in slides

0   1   2   3   4   5   6   7   8   9

Python

```python
slides.write('## Serialize Custom Objects to HTML\nThis is useful for displaying user de
with slides.suppress_stdout(): # suppress stdout from register fuction below
    @slides.serializer.register(int)
    def colorize(obj):
        color = 'red' if obj % 2 == 0 else 'green'
        return f'<span style="color:{color};">{obj}</span>'
    slides.write(*range(10))

some_slide.get_source().display()
```

This is all code to generate slides

Python

```python
 1  def demo(self):
 2      "Demo slides with a variety of content."
 3      self.close_view() # Close any previous view to speed up loading 10x faster on avera
 4      self.clear() # Clear previous content
 5
 6      with self.set_dir(os.path.split(__file__)[0]):
 7          file = '../_demo.py'
 8          raw_source = self.source.from_file(file).raw
 9          N = raw_source.count('auto.') + raw_source.count('\n---') + 1 # Count number of
10          self.create(*range(N)) # Create slides first, this is faster
11          self.shell.run_line_magic('run', file) # Run demo in same namespace
12
13      return self #_demo.demo(self) # Run demo
```

e:\research\ipyslides\ipyslides_demo.py

```python
 1  # Author: Abdul Saboor
 2  # This demonstrates that you can generate slides from a .py file too, which you can impc
 3  import time
 4
 5  from ipyslides.core import Slides
 6  from ipyslides.writer import write
 7  from ipyslides.formatters import libraries, __reprs__, plt2html
 8  from ipyslides._base.intro import logo_svg
 9
10
11  slides = Slides() # It reurns running slides instance or creates a new one
12
```

> 📝 **Note**
>
> Slides keep their full code if they are not made by @frames decorator!

Source Code

Markdown: Slide 0

```
1  # Creating Slides
2  ::: align-center
3      alert`Abdul Saboor`sup`1`, Unknown Authorsup`2`
4      center`today```
5      ::: text-box
6          sup`1`My University is somewhere in the middle of nowhere
7          sup`2`Their University is somewhere in the middle of nowhere
8  <h4 style=""color:green;"> 👆 Read instructions in left panel</h4>
```

Markdown: Slide 1

```
1  section`Introduction` toc`### Contents`
```

Markdown: Slide 2

```
1   proxy`something will be here in start`
2   # Introduction
3   To see how commands work, use `Slides.docs()` to see the documentation.
4   Here we will focus on using all that functionality to create slides.
5   ```python run source
6   # get the slides instance under a python block in Markdown file, we will use it later to
7   myslides = get_slides_instance()
8   import ipyslides as isd
9   version = myslides.version
10  %xmd #### This is inline markdown parsed by magic {.note .warning}
11  ```
```

Reference via Markdown

1. This is refernce to FigureWidget using `slides.cite` command
2. I was cited for no reason

```python
1  slides.write('citations`## Reference via Markdown\n——`')
2  bib_slide.get_source().display()
```