Creating Slides

Abdul Saboor[1], Unknown Author[2]
Feb 08, 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.

📝 Note

This is inline markdown parsed by magic

Version: 3.2.3 as executed from below code in markdown.

Python

```python
# get the slides instance under a python block in Markdown file, we will
myslides = get_slides_instance()
import ipyslides as isd
version = myslides.version
%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

- 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`

Link to ipywidgtes right here using textbox command

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, obj)` function to display widget on slides and alternative content in exported slides/report, obj can be a function to return possible HTML representation of widget.
- `ipywidgets.HTML` and its subclasses will be displayed as `Slides.alt(widget, lambda w: w.value)` where w is given widget. 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`

    - 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.

Note

- `write` is a robust command that can handle most of the cases. If nothing works, `repr(obj)` will be displayed.
- You can avoid `repr(obj)` by `lambda: func()` e.g. `lambda: plt.show()`.
- A single string passed to `write` is equivalent to `parse` command.
- You can add mini columns inside a column by markdown syntax or `Slides.cols`, but content type is limited in that case.

Slides.**parse**(xmd, display_inline=True, rich_outputs=False)

Parse extended markdown and display immediately. If you need output html, use `display_inline = False` but that won't execute python code blocks. Precedence of content return/display is `rich_outputs = True` > `display_inline = True` > `parsed_html_string`.

Example

```
```python run var_name
#If no var_name, code will be executed without assigning it to any varia
import numpy as np
```
Normal Markdown {.report-only}
```multicol 40 60
# First column is 40% width
If 40 60 was not given, all columns will be of equal width, this paragra
{.info}
+++
# Second column is 60% wide
This \{\{var_name\}\} is code from above and will be substituted with the
```

```python
# This will not be executed, only shown
```
```

ℹ Info

- Each block can have class names (speparated with space or .) after all other options such as `python .friendly` or `multicol .Sucess.info`.
  - For example, `python .friendly` will be highlighted with friendly theme from pygments.
  - Pygments themes, however, are not supported with `multicol`.
  - You need to write and display CSS for a custom class.
- The block with `::: class_type` syntax accepts extra classes in quotes, for example `::: multicol "Success" "info"`.
- There are three special CSS classes `report-only`, `slides-only` and `export-only` that control appearance of content in different modes.

⚠ Alert

Nested blocks are not supported.

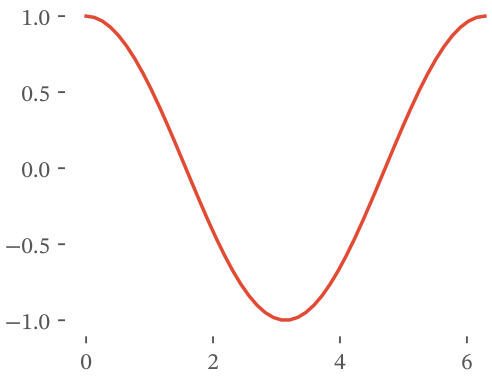
ℹ Info

- Find special syntax to be used in markdown by `Slides.xmd_syntax`.
- Use `Slides.extender` or `ipyslides.xmd.extender` to add markdown extensions.

Python

```
with last.proxies[0].capture():
    write([slides.classed(slides.doc(write,'Slides'),'block-green'), slid
 + 1 more lines ...
```

Table of Contents



Python

```python
import numpy as np, matplotlib.pyplot as plt
plt.rcParams['svg.fonttype'] = 'none' # Global setting, enforce same font
x = np.linspace(0,2*np.pi)
with plt.style.context('ggplot'):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    _ = ax.plot(x,np.cos(x))
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-d
    df = df.describe() #Small for display
except:
    df = '### Install `pandas` to view output'
```

Writing Plotly Figure

Install `plotly` to view output

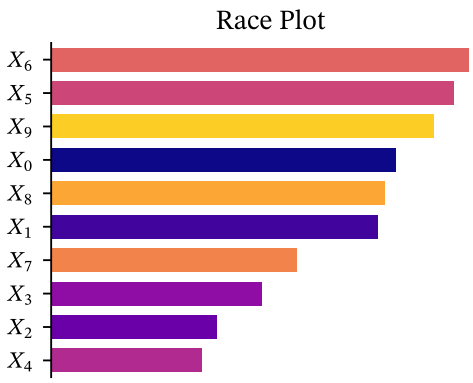
Python

```python
try:
    import plotly.graph_objects as go
    fig = go.Figure()
    fig.add_trace(go.Bar(y=[1,5,8,9]))
except:
    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!



A Silly Plot

Python

```python
import ipywidgets as ipw

write('''
    ## Interactive Apps with Widgets section`Interactive Widgets`
    Use `ipywidgets`, `bqplot`,`ipyvolume`, `plotly Figurewidget` etc. to
     ::: note-tip
         Export to Slides/Report to see what happens to this slide and nex
    ''')
plot_html = ipw.HTML('Plot will be here')
button = ipw.Button(description='Click me to update race plot',layout=ipw

write([plot_html,button], src)

def update_plot(btn):
    plot_html.value = race_plot().value #Convert to html string

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

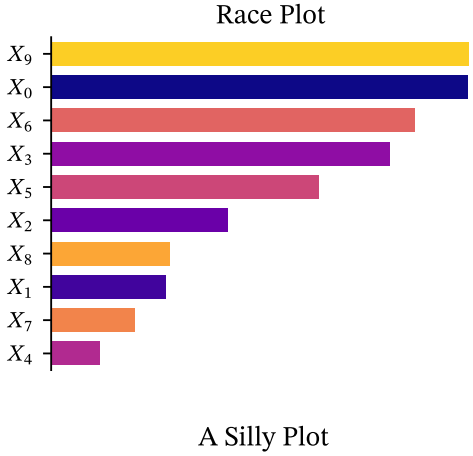
Python

```python
def race_plot():
    import numpy as np
    import matplotlib.pyplot as plt

    x = np.linspace(0,0.9,10)
    y = np.random.random((10,))
    _sort = np.argsort(y)

    plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd
    with plt.style.context(plot_theme):
```

```
    for s in ['right','top','bottom']:
        ax.spines[s].set_visible(False)

    ax.set(title='Race Plot', ylim = [-0.05,0.95], xticks=[],yticks=[c fo
    return plt2html(fig, transparent=False, caption='A Silly Plot')
```

Dynamic Content without Widgets

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



Python

```
write('''
    ## Dynamic Content without Widgets
    Use refresh button below to update plot! Compare with previous slide!
    ''')

def display_plot(): return race_plot().display()

write(lambda: slides.on_refresh(display_plot), rslide.get_source()) # Onl
slides.source.from_callable(race_plot).display()
```

Python

```
def race_plot():
    import numpy as np
    import matplotlib.pyplot as plt

    x = np.linspace(0.0,9,10)
```

```
8
9       plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd
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[
13
14      for s in ['right','top','bottom']:
15          ax.spines[s].set_visible(False)
16
17      ax.set(title='Race Plot', ylim = [-0.05,0.95], xticks=[],yticks=[c fo
18      return plt2html(fig, transparent=False, caption='A Silly Plot')
```

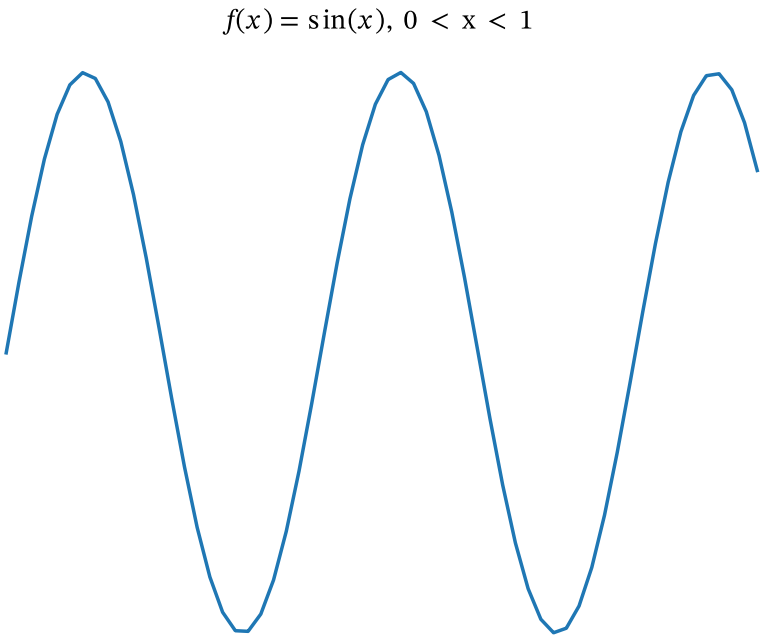
Contents

This is Slide 16.0

and we are animating matplotlib

Python

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



Python

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

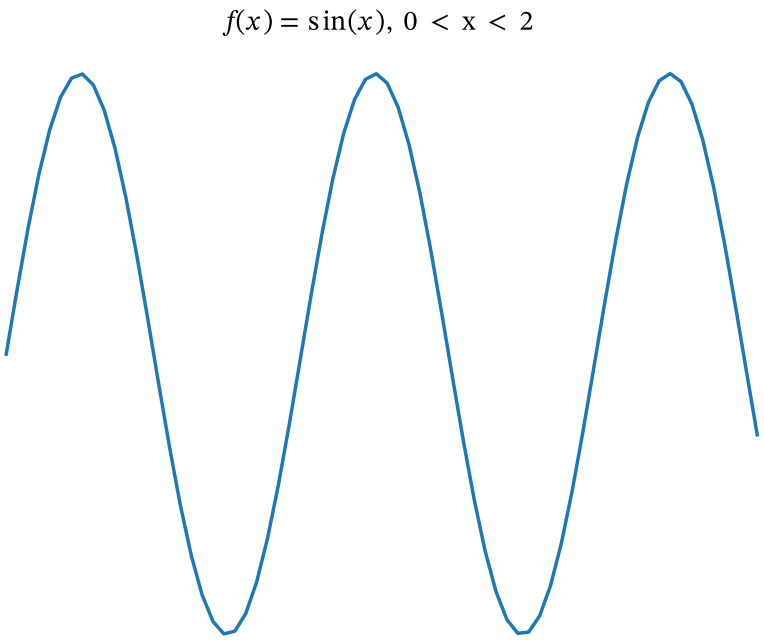
2

This is Slide 16.1

and we are animating matplotlib

Python

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

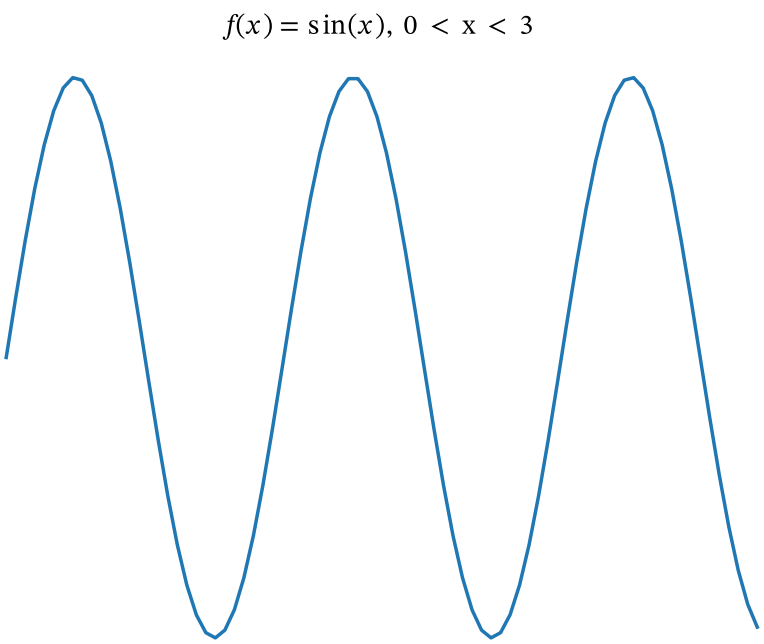


This is Slide 16.2

and we are animating matplotlib

Python

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

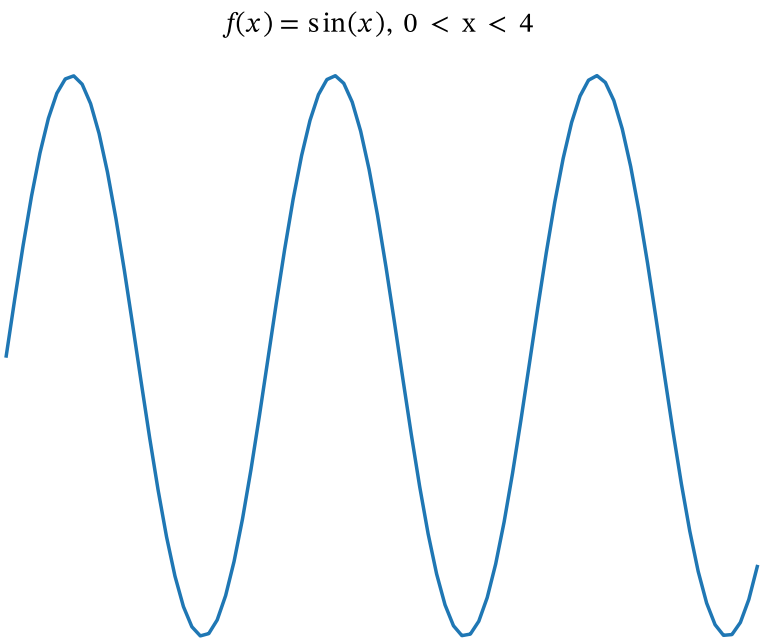


This is Slide 16.3

and we are animating matplotlib

Python

```
 + 3 more lines ...
ax.set_title(f'$f(x)=\sin(x)$, 0 < x < {idx+1}')
 + 2 more lines ...
```

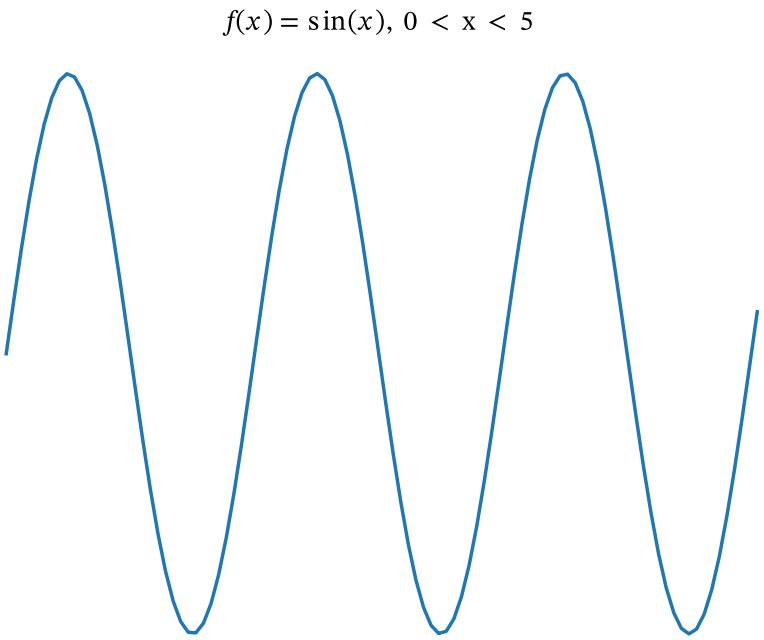


This is Slide 16.4

and we are animating matplotlib

Python

```
 + 4 more lines ...
ax.set_axis_off()
 + 1 more lines ...
```



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

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

Frames with

3

4

Python

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

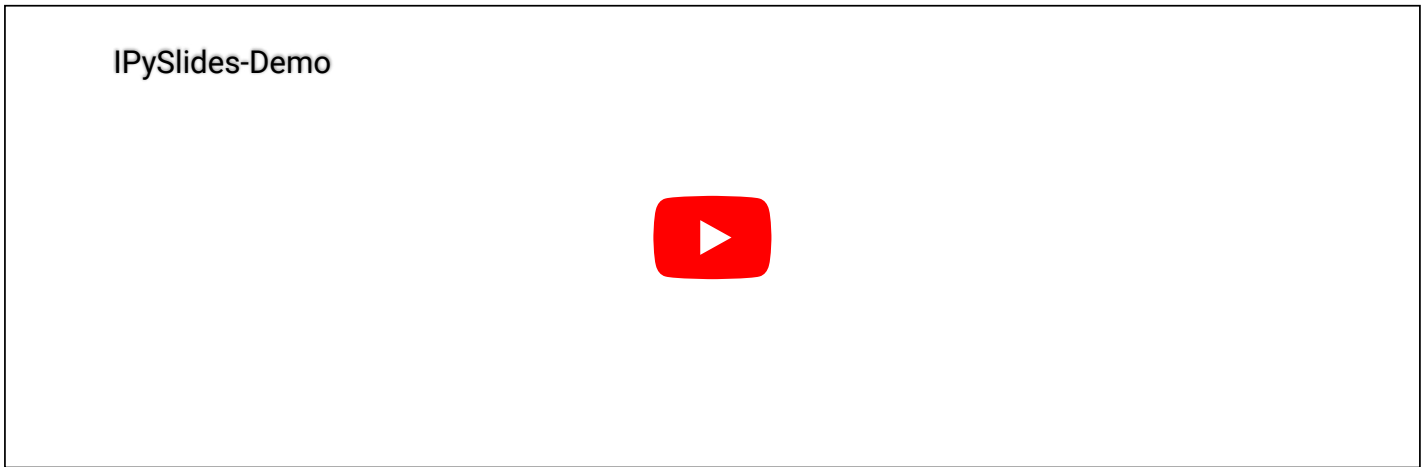
Displaying image from url from somewhere in Kashmir (کشمیر)

Python

```
backward_skipper.display()
forward_skipper.set_target()
slides.format_css({'.goto-button .fa.fa-minus': slides.icon('arrow',color
slides.write('## Displaying image from url from somewhere in Kashmir colo
try:
    slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d
except:
    slides.write('Could not retrieve image from url. Check internt connec
s.get_source().display()
```

Watching Youtube Video?

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



Python

```python
write(f"### Watching Youtube Video?")
write('**Want to do some drawing instead?**\nClick on pencil icon and dra

write(YouTubeVideo('thgLGl14-tg',width='100%',height='266px'))

@slides.on_load
def push():
    t = time.localtime()
    slides.notify(f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm
    slides.set_overlay_url('https://tldraw.com')

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

Python

```python
write('## Block API\nNew `block` API is as robust as `write` command. On
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="
        ipw.Button(description='Click to do nothing'),
    ]
)
s.get_source().focus_lines([3,4,5,6]).display()
```

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.

$$\int_1^1 \quad 1 \quad ,$$

Python

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

Built-in CSS styles

Use any or combinations of these styles in className argument of writing fun

className	Formatting Style
'align-center'	——————Text——————
'align-left'	Text————————
'align-right'	————————Text
'rtl'	—————— اردو عربی
'info'	Blue text. Icon ℹ️ 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
'export-only'	Hidden on main slides, but will appear in exported sli
'jupyter-only'	Hidden on exported slides/report, but will appear on m
'page-break'	Report will break page in print after object with this
'block'	Block of text/objects
'block-[color]'	Block of text/objects with specific background color f
	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-chil

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 dis
with slides.suppress_stdout(): # suppress stdout from register fuction be
    @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
def demo(self):
    "Demo slides with a variety of content."
    self.close_view() # Close any previous view to speed up loading 10x f
    self.clear() # Clear previous content

    with self.set_dir(os.path.split(__file__)[0]):
```

```
        N = raw_source.count('auto.') + raw_source.count('\n---') + 1 # (
        self.create(*range(N)) # Create slides first, this is faster
        self.shell.run_line_magic('run', file) # Run demo in same namespa

    return self #_demo.demo(self) # Run demo
```

e:\research\ipyslides\ipyslides_demo.py

```
# Author: Abdul Saboor
# This demonstrates that you can generate slides from a .py file too, whi
import time

from ipyslides.core import Slides
from ipyslides.writer import write
from ipyslides.formatters import libraries, __reprs__, plt2html
from ipyslides._base.intro import logo_svg


slides = Slides() # It reurns running slides instance or creates a new on

auto = slides.AutoSlides() # Does not work inside Jupyter notebook (shoul

slides.settings.set_footer('Author: Abdul Saboor عبدالصبور')
slides.settings.set_logo(logo_svg,width=60) # This is by defualt a logo o
slides._citation_mode = 'global' # This could be changed by other functio
slides.set_citations({
        'pf': 'This is refernce to FigureWidget using `slides.cite` comma
        'This': 'I was cited for no reason',
    })

slides.run_cell("""
%%title -m
# Creating Slides
::: align-center
    alert`Abdul Saboor`sup`1`, Unknown Authorsup`2`
    center`today```
    ::: text-box
        sup`1`My University is somewhere in the middle of nowhere
        sup`2`Their University is somewhere in the middle of nowhere
<h4 style=""color:green;""> 👈 Read instructions in left panel</h4>
""")

#Demo for loading slides from a file as text block
```

```
---
proxy`something will be here in start`
# 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.
```python run source
# get the slides instance under a python block in Markdown file, we will
myslides = get_slides_instance()
import ipyslides as isd
version = myslides.version
%xmd #### This is inline markdown parsed by magic {.note .warning}
```
Version: {{version}} as executed from below code in markdown.
{{source}}
proxy`something will be here in end`

IPySlides Online Running Sources
::: note
    - [Edit on Kaggle](https://www.kaggle.com/massgh/ipyslides)
    - Launch example Notebook [![Binder](https://mybinder.org/badge_logo.
    - Watch a [Youtube Video](https://www.youtube.com/watch?v=ytfWIYbJteE

[^1]: Add references like this per slide. Use slides.cite() or in markdow

""", trusted=True)

slides.shell.user_ns['write'] = write #Inject variable in IPython shell
slide s2 has proxies to be filled in later
p1, p2 = s2.proxies
with p1.capture():
    s2.get_source().display(collapsed = True)

with p2.capture():
    slides.write(f'alert`I was added at end by a given proxy, see the how

*others, last = auto.from_markdown(f"""
section`Variety of Content Types to Display` toc`### Contents`

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

```
such as color[fg=navy,bg=skyblue] ipython.display.[html,svg,markdown,code
---
## Plots and Other **Data**{{style='color:var(--accent-color);'}} Types
#### These objects are implemented to be writable in `write` command:
{', '.join([f"`{lib['name']}.{lib['obj']}`" for lib in libraries])}
Many will be extentended in future. If an object is not implemented, use
command to show in Notebook outside color[fg=teal,bg=whitesmoke]`write`.
---
## Interactive Widgets
### Any object in `ipywidgets`{slides.textbox('<a href="https://ipywidget
or libraries based on ipywidgtes such as color[red]`bqplot`,color[green]`
can be included as well.
{{.warning}}
---
## Commands which do all Magic!
proxy`Add functions here`
""", trusted=True)


with slides.source.context(auto_display = False) as s:
    with last.proxies[0].capture():
        write([slides.classed(slides.doc(write,'Slides'),'block-green'),
        s.show_lines([0,1]).display()


auto.from_markdown('section`Plotting and DataFrame` toc```')

# Matplotlib
with auto.slide() as sl:
    write('## Plotting with Matplotlib')
    with slides.source.context(auto_display = False) as s:
        import numpy as np, matplotlib.pyplot as plt
        plt.rcParams['svg.fonttype'] = 'none' # Global setting, enforce s
        x = np.linspace(0,2*np.pi)
        with plt.style.context('ggplot'):
            fig, ax = plt.subplots(figsize=(3.4,2.6))
            _ = ax.plot(x,np.cos(x))
        write([ax, s.focus_lines([1,3,4])])

    sl.set_css({'background':'linear-gradient(to right, #FFDAB9 0%, #F0E6

# Plotly and Pandas DataFrame only show if you have installed
```

```
        df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seabo
        df = df.describe() #Small for display
    except:
        df = '### Install `pandas` to view output'

with auto.slide():
    write(['## Writing Pandas DataFrame', df, source])

with slides.source.context(False) as s:
    try:
        import plotly.graph_objects as go
        fig = go.Figure()
        fig.add_trace(go.Bar(y=[1,5,8,9]))
    except:
        fig = '### Install `plotly` to view output'

with auto.slide():
    write(('## Writing Plotly Figure',fig, s))

def race_plot():
    import numpy as np
    import matplotlib.pyplot as plt

    x = np.linspace(0,0.9,10)
    y = np.random.random((10,))
    _sort = np.argsort(y)

    plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd
    with plt.style.context(plot_theme):
        fig,ax = plt.subplots(figsize=(3.4,2.6))
        ax.barh(x,y[_sort],height=0.07,color=plt.cm.get_cmap('plasma')(x[

    for s in ['right','top','bottom']:
        ax.spines[s].set_visible(False)

    ax.set(title='Race Plot', ylim = [-0.05,0.95], xticks=[],yticks=[c fo
    return plt2html(fig, transparent=False, caption='A Silly Plot')


# Interactive widgets.
with auto.slide():
```

```
169
170         write('''
171             ## Interactive Apps with Widgets section`Interactive Widgets`
172             Use `ipywidgets`, `bqplot`,`ipyvolume`, `plotly Figurewidget`
173              ::: note-tip
174                 Export to Slides/Report to see what happens to this slide
175             ''')
176         plot_html = ipw.HTML('Plot will be here')
177         button = ipw.Button(description='Click me to update race plot',la
178 
179         write([plot_html,button], src)
180 
181         def update_plot(btn):
182             plot_html.value = race_plot().value #Convert to html string
183 
184         button.on_click(update_plot)
185         update_plot(None) #Initialize plot
186 
187     slides.source.from_callable(race_plot).display()
188 
189 with auto.slide() as rslide:
190     write('''
191         ## Dynamic Content without Widgets
192         Use refresh button below to update plot! Compare with previous sl
193         ''')
194 
195     def display_plot(): return race_plot().display()
196 
197     write(lambda: slides.on_refresh(display_plot), rslide.get_source()) #
198     slides.source.from_callable(race_plot).display()
199 
200 auto.from_markdown('section`Simple Animations with Frames` toc`### Conten
201 
202 forward_skipper = slides.goto_button('Skip All Next Frames')
203 backward_skipper = slides.goto_button('Skip Previous Frames', icon='minus
204 # Animat plot in slides
205 @auto.frames(*range(14,19))
206 def func(obj,idx):
207     if idx == 0:
208         forward_skipper.display()
209         backward_skipper.set_target()
```

```
        x = np.linspace(0,obj+1,50+10*(idx+1))
        ax.plot(x,np.sin(x));
        ax.set_title(f'$f(x)=\sin(x)$, 0 < x < {idx+1}')
        ax.set_axis_off()
        slides.notes.insert(f'## This is under @frames decorator!')

    slides.write([f'### This is Slide {slides.running.number}.{idx}\n and
                  s.show_lines([idx])
                  ],ax,widths=[40,60])
    if idx == 0: #Only show source code of first frame
        s.show_lines([5]).display()
    slides.write(slides.cite('This'))

auto.from_markdown('section`Controlling Content on Frames` toc`### Conten

# Frames structure
boxes = [f'<div style="background:var(--hover-bg);width:auto;height:2em;p
@auto.frames(*boxes, repeat=False)
def f(obj,idx):
    slides.write('# Frames with \n#### `repeat = False`')
    slides.write(obj)
@auto.frames(*boxes, repeat=True,frame_height='100%')
def f(obj,idx):
    slides.running.set_animation(None) #Disable animation for showing bul
    slides.write('# Frames with \n#### `repeat = True` and Fancy Bullet L
    slides.bullets(obj, marker='💘').display()

@auto.frames(*boxes, repeat=[(0,1),(2,3)])
def f(obj,idx):
    with slides.source.context(auto_display = False) as s:
        slides.write('# Frames with \n#### `repeat = [(0,1),(2,3)]`')
        slides.write(*obj)

    s.display()

with auto.slide() as s:
    backward_skipper.display()
    forward_skipper.set_target()
    slides.format_css({'.goto-button .fa.fa-minus': slides.icon('arrow',c
    slides.write('## Displaying image from url from somewhere in Kashmir
    try:
```

```
256         slides.write('Could not retrieve image from url. Check internt co
257     s.get_source().display()
258 
259 # Youtube
260 from IPython.display import YouTubeVideo
261 with auto.slide() as ys: # We will use this in next %%magic
262     write(f"### Watching Youtube Video?")
263     write('**Want to do some drawing instead?**\nClick on pencil icon and
264 
265     write(YouTubeVideo('thgLGl14-tg',width='100%',height='266px'))
266 
267     @slides.on_load
268     def push():
269         t = time.localtime()
270         slides.notify(f'You are watching Youtube at Time-{t.tm_hour:02}:{
271         slides.set_overlay_url('https://tldraw.com')
272 
273     ys.get_source().display()
274 
275 
276 with auto.slide() as s:
277     write('## Block API\nNew `block` API is as robust as `write` command.
278     slides.block_red(
279         [
280             '### Table',
281             '''
282             |h1 |h2 |h3 |
283             |---|---|---|
284             |d1 |d2 |d3 |
285             |r1 |r2 |r3 |
286             ''',
287         ],
288         [
289             '### Widgets',
290             ipw.Checkbox(description='Select to do nothing',indent=False)
291             slides.alt(ipw.IntSlider(),lambda w: f'<input type="range" m
292             ipw.Button(description='Click to do nothing'),
293         ]
294     )
295     s.get_source().focus_lines([3,4,5,6]).display()
296 
```

```
Use `$ $` or `$$ $$` to display latex in Markdown, or embed images of equ
$\LaTeX$ needs time to load, so keeping it in view until it loads would h
{.note-warning}

$$\int_0^1\\frac{1}{1-x^2}dx$$
''', trusted=True)

with auto.slide(), slides.source.context():
    slides.write('## Built-in CSS styles')
    slides.css_styles.display()

auto.from_markdown('section`Custom Objects Serilaization` toc`### Content

with auto.slide() as some_slide:
    slides.write('## Serialize Custom Objects to HTML\nThis is useful for
    with slides.suppress_stdout(): # suppress stdout from register fuctio
        @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()

with auto.slide():
    slides.write('## This is all code to generate slides section`Code to
    slides.source.from_callable(slides.demo).display()
    slides.source.from_file(__file__).display()

with auto.slide():
    slides.write('Slides keep their full code if they are not made by @fr
    slides.get_source().display()


with auto.slide() as bib_slide:
    slides.write('citations`## Reference via Markdown\n———`')
    bib_slide.get_source().display()


slides.navigate_to(0) # Go to title slide
```

Source Code

Markdown: Slide 0

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

Markdown: Slide 1

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

Markdown: Slide 2

```
proxy`something will be here in start`
# 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.
```python run source
# get the slides instance under a python block in Markdown file, we will
myslides = get_slides_instance()
import ipyslides as isd
version = myslides.version
%xmd #### This is inline markdown parsed by magic {.note .warning}
```
Version: {{version}} as executed from below code in markdown.
{{source}}
proxy`something will be here in end`
```

Markdown: Slide 3

```
# IPySlides Online Running Sources
::: note
    - [Edit on Kaggle](https://www.kaggle.com/massgh/ipyslides)
    - Launch example Notebook [![Binder](https://mybinder.org/badge_logo
    - Watch a [Youtube Video](https://www.youtube.com/watch?v=ytfWIYbJtel

[^1]: Add references like this per slide. Use slides.cite() or in markdow
```

Markdown: Slide 5

```
## IPython Display Objects
#### Any object with following methods could be in`write` command:
`_repr_pretty_`, `_repr_html_`, `_repr_markdown_`, `_repr_svg_`, `_repr_p
Such as color[fg=navy,bg=skyblue]`IPython.display.[HTML,SVG,Markdown,C
```

Markdown: Slide 6

```
## Plots and Other **Data**{style='color:var(--accent-color);'} Types
#### These objects are implemented to be writable in `write` command:
`matplotlib.pyplot.Figure`, `altair.Chart`, `pygal.Graph`, `pydeck.Deck`,
Many will be extentended in future. If an object is not implemented, u
command to show in Notebook outside color[fg=teal,bg=whitesmoke]`write
```

Markdown: Slide 7

```
## Interactive Widgets
### Any object in `ipywidgets`<span class='text-box' style = 'display:inl
or libraries based on ipywidgtes such as color[red]`bqplot`,color[gree
can be included as well.
{.warning}
```

Markdown: Slide 8

```
## Commands which do all Magic!
proxy`Add functions here`
```

Markdown: Slide 9

```
section`Plotting and DataFrame` toc``
```

Python: Slide 10

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

sl.set_css({'background':'linear-gradient(to right, #FFDAB9 0%, #F0E68C 1
```

Python: Slide 11

```
write(['## Writing Pandas DataFrame', df, source])
```

```
write(('## Writing Plotly Figure',fig, s))
```

Python: Slide 13

```
with slides.source.context(auto_display = False) as src:
    import ipywidgets as ipw

    write('''
        ## Interactive Apps with Widgets section`Interactive Widgets`
        Use `ipywidgets`, `bqplot`,`ipyvolume`, `plotly Figurewidget` etc
         ::: note-tip
             Export to Slides/Report to see what happens to this slide and
        ''')
    plot_html = ipw.HTML('Plot will be here')
    button = ipw.Button(description='Click me to update race plot',layout

    write([plot_html,button], src)

    def update_plot(btn):
        plot_html.value = race_plot().value #Convert to html string

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

slides.source.from_callable(race_plot).display()
```

Python: Slide 14

```
write('''
    ## Dynamic Content without Widgets
    Use refresh button below to update plot! Compare with previous slide!
    ''')

def display_plot(): return race_plot().display()

write(lambda: slides.on_refresh(display_plot), rslide.get_source()) # Onl
slides.source.from_callable(race_plot).display()
```

Markdown: Slide 15

```
section`Simple Animations with Frames` toc`### Contents`
```

Markdown: Slide 17

```
section`Controlling Content on Frames` toc`### Contents`
```

```
forward_skipper.set_target()
slides.format_css({'.goto-button .fa.fa-minus': slides.icon('arrow',color
slides.write('## Displaying image from url from somewhere in Kashmir colo
try:
    slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d
except:
    slides.write('Could not retrieve image from url. Check internt connec
s.get_source().display()
```

Python: Slide 22

```
write(f"### Watching Youtube Video?")
write('**Want to do some drawing instead?**\nClick on pencil icon and dra

write(YouTubeVideo('thgLGl14-tg',width='100%',height='266px'))

@slides.on_load
def push():
    t = time.localtime()
    slides.notify(f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm
    slides.set_overlay_url('https://tldraw.com')

ys.get_source().display()
```

Python: Slide 23

```
write('## Block API\nNew `block` API is as robust as `write` command. On
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="
        ipw.Button(description='Click to do nothing'),
    ]
)
```

```
## $\LaTeX$ in Slides
Use `$ $` or `$$ $$` to display latex in Markdown, or embed images of e
$\LaTeX$ needs time to load, so keeping it in view until it loads woul
{.note-warning}

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

Python: Slide 25

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

Markdown: Slide 26

```
section`Custom Objects Serilaization` toc`### Contents`
```

Python: Slide 27

```
slides.write('## Serialize Custom Objects to HTML\nThis is useful for dis
with slides.suppress_stdout(): # suppress stdout from register fuction be
    @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()
```

Python: Slide 28

```
slides.write('## This is all code to generate slides section`Code to Gene
slides.source.from_callable(slides.demo).display()
slides.source.from_file(__file__).display()
```

Python: Slide 29

```
slides.write('Slides keep their full code if they are not made by @frames
slides.get_source().display()
```

Reference via Markdown

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

```
bib_slide.get_source().display()
```