

Creating Slides

Abdul Saboor¹, Unknown Author²
Feb 12, 2023

¹My University is somewhere in the middle of nowhere

²Their University is somewhere in the middle of nowhere

 **Read instructions in left panel**

Contents

1. **Introduction**
2. Variety of Content Types to Display
3. Plotting and DataFrame
4. Interactive Widgets
5. Simple Animations with Frames
6. Controlling Content on Frames
7. Miscellaneous Content
8. Custom Objects Serilaization
9. Code to Generate Slides

≡ 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: 3.2.6 as executed from below code in markdown.


Python

```
1 # get the slides instance under a python block in Markdown file, we will
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 
- Watch a [Youtube Video](#)

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

Contents

1. Introduction
2. **Variety of Content Types to Display**
3. Plotting and DataFrame
4. Interactive Widgets
5. Simple Animations with Frames
6. Controlling Content on Frames
7. Miscellaneous Content
8. Custom Objects Serilaization
9. Code to Generate Slides

IPython Display Objects

Any object with following methods could be inwrite 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
extended 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 `ipywidgets` right here using `textbox` command

or libraries based on `ipywidgets` 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 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.
- Display IPython widgets such as `ipywidgets` or `ipyvolume` 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.

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

```
1 ```python run var_name
2 #If no var_name, code will be executed without assigning it to any varia
3 import numpy as np
4 ```
5 # Normal Markdown {.report-only}
6 ```
```

```

9  {.info}
10  +++
11  # Second column is 60% wide
12  This \({var_name}\) is code from above and will be substituted with the
13  ```
14
15  ```python
16  # This will not be executed, only shown
17  ```
18  || Inline-column A || Inline-column B ||

```

i Info

- Each block can have class names (separated 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

```

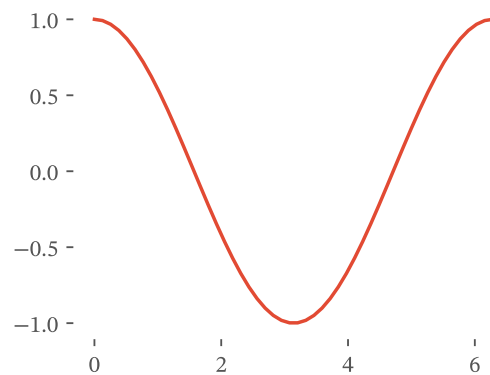
1  with last.proxies[0].capture():
2      write([slides.classed(slides.doc(write, 'Slides'), 'block-green'), slic
3  + 1 more lines ...

```

Table of Contents

- 4. Interactive Widgets
- 5. Simple Animations with Frames
- 6. Controlling Content on Frames
- 7. Miscellaneous Content
- 8. Custom Objects Serilaization
- 9. Code to Generate Slides

Plotting with Matplotlib



Python

```
1 import numpy as np, matplotlib.pyplot as plt
2 plt.rcParams['svg.fonttype'] = 'none' # Global setting, enforce same font
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

```
1 try:
2     import pandas as pd
3     df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-c
4     df = df.describe() #Small for display
5 except:
6     df = '### Install `pandas` to view output'
```

Writing Plotly Figure

Install plotly to view output

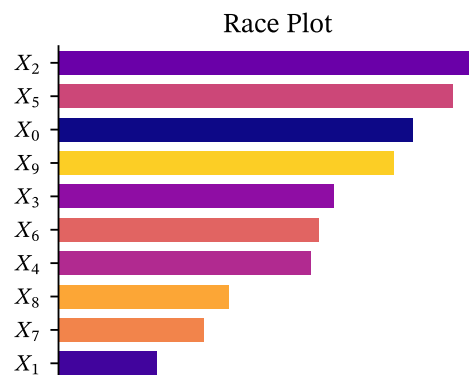
Python

```
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

💡 Tip

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



A Silly Plot

Python

```
1 import ipywidgets as ipw
2
3 write('''
4     ## Interactive Apps with Widgets section`Interactive Widgets`
5     Use `ipywidgets`, `bqplot`, `ipyvolume`, `plotly Figurewidget` etc. to
6     ::: note-tip
7     Export to Slides/Report to see what happens to this slide and nex
8     ''')
9 plot_html = ipw.HTML('Plot will be here')
10 button = ipw.Button(description='Click me to update race plot', layout=ipw
11
12 write([plot_html, button], src)
13
14 def update_plot(btn):
15     plot_html.value = race_plot().value #Convert to html string
16
17 button.on_click(update_plot)
18 update_plot(None) #Initialize plot
```

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,))
```

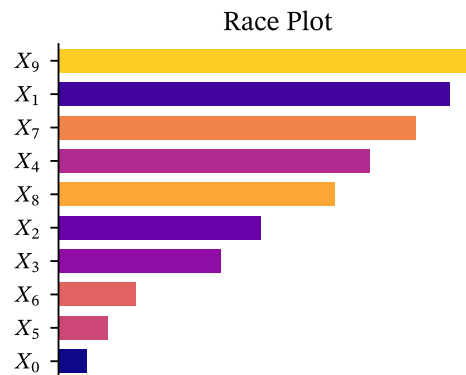
```

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 for c in x])
18     return plt2html(fig, transparent=False, caption='A Silly Plot')

```

Dynamic Content without Widgets

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



A Silly Plot

Python

```

1 write('''
2     ## Dynamic Content without Widgets
3     Use refresh button below to update plot! Compare with previous slide!
4     ''')
5
6 def display_plot(): return race_plot().display()
7
8 write(lambda: slides.on_refresh(display_plot), rslide.get_source()) # Only
9 slides.source.from_callable(race_plot).display()

```

Python

```

1 def race_plot():
2     import numpy as np

```

```

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.
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

1. Introduction
2. Variety of Content Types to Display
3. Plotting and DataFrame
4. Interactive Widgets
5. **Simple Animations with Frames**
6. Controlling Content on Frames
7. Miscellaneous Content
8. Custom Objects Serilaization
9. Code to Generate Slides

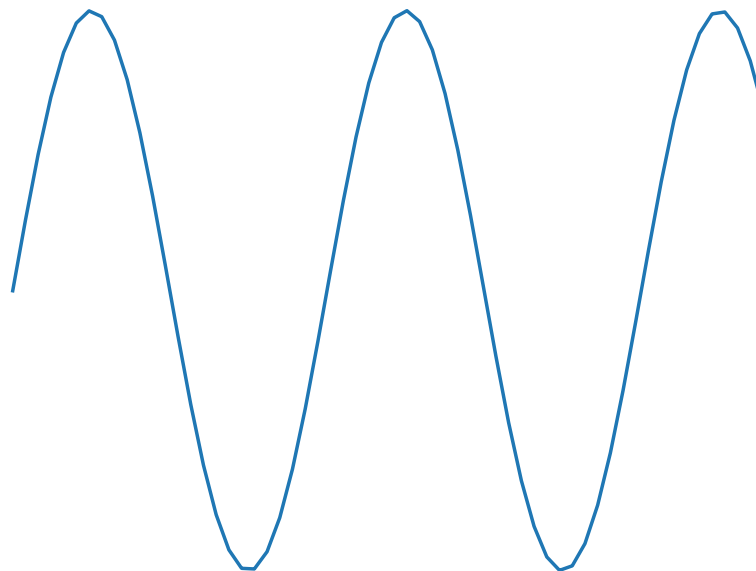
This is Slide 16.0

and we are animating matplotlib

Python

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

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



Python

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

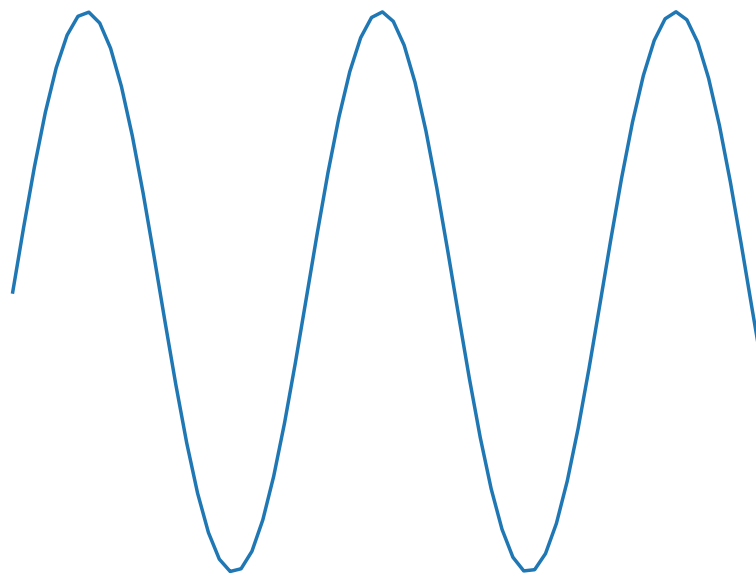
This is Slide 16.1

and we are animating matplotlib

Python

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

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



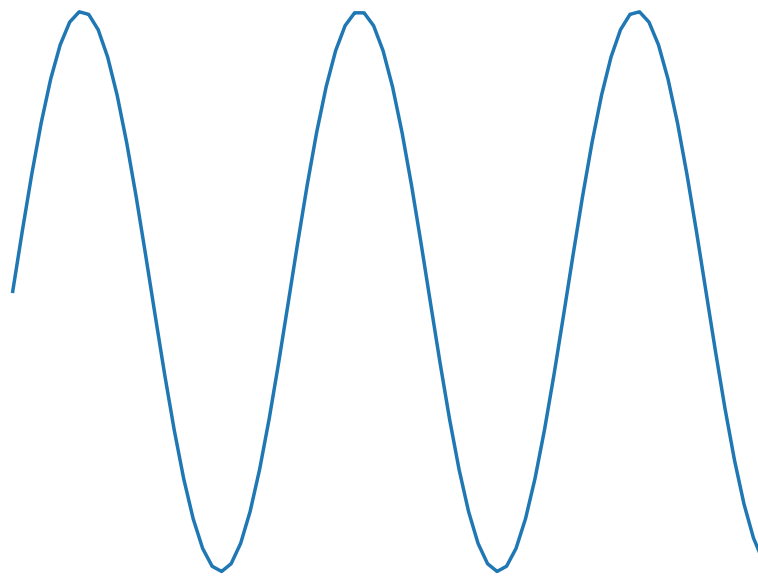
This is Slide 16.2

and we are animating matplotlib

Python

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

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



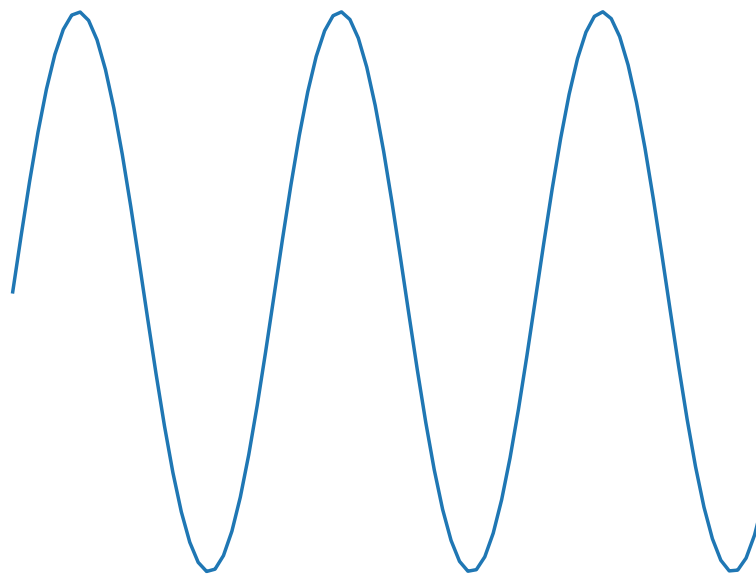
This is Slide 16.3

and we are animating matplotlib

Python

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

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



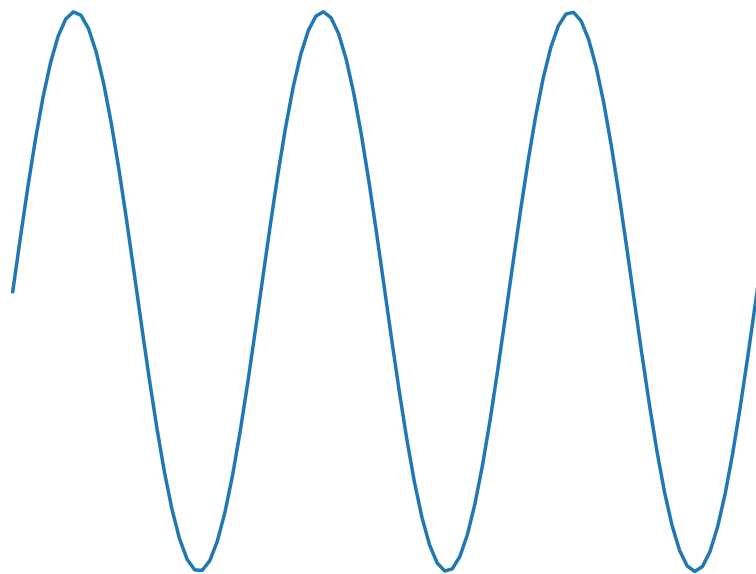
This is Slide 16.4

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

1. Introduction
2. Variety of Content Types to Display
3. Plotting and DataFrame
4. Interactive Widgets
5. Simple Animations with Frames
6. **Controlling Content on Frames**
7. Miscellaneous Content
8. Custom Objects Serilaization

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

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

Frames with

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

3

4

Python

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

Displaying image from url from somewhere in Kashmir (کشمیر)



Python

```
1 backward_skipper.display()  
2 forward_skipper.set_target()
```

```

6     slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d
7 except:
8     slides.write('Could not retrieve image from url. Check internet connec
9 s.get_source().display()

```

Watching Youtube Video?

Want to do some drawing instead? Click on pencil icon and draw something on [tldraw](#)!

IPySlides-Demo



Python

```

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



Click to do nothing

Python

```
1 write('## Block API\nNew `block` API is as robust as `write` command. On
2 slides.block_red(
3     [
4         '### Table',
5         '''
6         |h1 |h2 |h3 |
7         |---|---|---|
8         |d1 |d2 |d3 |
9         |r1 |r2 |r3 |
10        ''',
11     ],
12     [
13         '### Widgets',
14         ipw.Checkbox(description='Select to do nothing',indent=False),
15         slides.alt(ipw.IntSlider(),lambda w: f'<input type="range" min="0'
16         ipw.Button(description='Click to do nothing'),
17         'proxy`[Paste Widgets Screenshot Here]`'
18     ]
19 )
20 s.get_source().display()
```

L^AT_EX in Slides

⚠ Alert

Use `$` `$` or `$$` `$$` to display latex in Markdown, or embed images of equations *L^AT_EX* needs time to load, so keeping it in view until it loads would help.

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

Python

```
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 fun

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
'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-child

Contents

1. Introduction
2. Variety of Content Types to Display
3. Plotting and DataFrame
4. Interactive Widgets
5. Simple Animations with Frames
6. Controlling Content on Frames
7. Miscellaneous Content
8. **Custom Objects Serilaization**
9. Code to Generate Slides

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

```
1 slides.write('## Serialize Custom Objects to HTML\nThis is useful for dis
2 with slides.suppress_stdout(): # suppress stdout from register fuction be
3     @slides.serializer.register(int)
4     def colorize(obj):
5         color = 'red' if obj % 2 == 0 else 'green'
6         return f'<span style="color:{color};">{obj}</span>'
7     slides.write(*range(10))
8
9 some_slide.get_source().display()
```


This is all code to generate slides

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 f
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 #
10        self.create(*range(N)) # Create slides first, this is faster
11        self.shell.run_line_magic('run', file) # Run demo in same namesp
12
13        return self #_demo.demo(self) # Run demo
```

e:\research\ipyslides\ipyslides_demo.py

```
1 # Author: Abdul Saboor
2 # This demonstrates that you can generate slides from a .py file too, whi
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 on
12
13 auto = slides.AutoSlides() # Does not work inside Jupyter notebook (shoul
14
15 slides.settings.set_footer('Author: Abdul Saboor عبدالصبور')
16 slides.settings.set_logo(logo_svg,width=60) # This is by default a logo o
17 slides._citation_mode = 'global' # This could be changed by other functio
18 slides.set_citations({
19     'pf': 'This is refernce to FigureWidget using `slides.cite` comma
20     'This': 'I was cited for no reason',
21 })
```

```

25 # Creating Slides
26 ::: align-center
27     alert`Abdul Saboor`sup`1`, Unknown Authorsup`2`
28     center`today``
29     ::: text-box
30         sup`1`My University is somewhere in the middle of nowhere
31         sup`2`Their University is somewhere in the middle of nowhere
32 <h4 style=""color:green;"> 🖱️ Read instructions in left panel</h4>
33 """)
34
35 #Demo for loading slides from a file or text block
36 s1, s2, *others = auto.from_markdown("""
37 section`Introduction` toc`### Contents`
38 ---
39 proxy`something will be here in start`
40 # Introduction
41 To see how commands work, use `Slides.docs()` to see the documentation.
42 Here we will focus on using all that functionality to create slides.
43 ```python run source
44 # get the slides instance under a python block in Markdown file, we will
45 myslides = get_slides_instance()
46 import ipyslides as isd
47 version = myslides.version
48 %xmd ##### This is inline markdown parsed by magic {.note .warning}
49 ```
50 Version: {{version}} as executed from below code in markdown.
51 {{source}}
52 proxy`something will be here in end`
53 ---
54 # IPySlides Online Running Sources
55 ::: note
56     - [Edit on Kaggle](https://www.kaggle.com/massgh/ipyslides)
57     - Launch example Notebook [![Binder](https://mybinder.org/badge_logo.
58     - Watch a [Youtube Video](https://www.youtube.com/watch?v=ytfWIYbJteE
59
60 [^1]: Add references like this per slide. Use slides.cite() or in markdow
61
62 """, trusted=True)
63
64
65 slides.shell.user_ns['write'] = write #Inject variable in IPython shell

```

```

69     s2.get_source().display(collapsed = True)
70
71 with p2.capture():
72     slides.write(f'alert`I was added at end by a given proxy, see the how
73
74
75 *others, last = auto.from_markdown(f"""
76 section`Variety of Content Types to Display` toc`### Contents`
77 ---
78 ## IPython Display Objects
79 ##### Any object with following methods could be in`write` command:
80 {'', '.join([f'`_repr_{rep}`' for rep in __reprs__])}
81 Such as color[fg=navy,bg=skyblue]`IPython.display.[HTML,SVG,Markdown,Code
82 ---
83 ## Plots and Other **Data**{{style='color:var(--accent-color);'}} Types
84 ##### These objects are implemented to be writable in `write` command:
85 {'', '.join([f"{{lib['name']}}.{{lib['obj']}}" for lib in libraries])}
86 Many will be extended in future. If an object is not implemented, use
87 command to show in Notebook outside color[fg=teal,bg=whitesmoke]`write`.
88 ---
89 ## Interactive Widgets
90 ### Any object in `ipywidgets`{slides.textbox('<a href="https://ipywidget
91 or libraries based on ipywidgtes such as color[red]`bqplot`,color[green]`
92 can be included as well.
93 {{.warning}}
94 ---
95 ## Commands which do all Magic!
96 proxy`Add functions here`
97 """, trusted=True)
98
99
100 with slides.source.context(auto_display = False) as s:
101     with last.proxies[0].capture():
102         write([slides.classed(slides.doc(write, 'Slides'), 'block-green'),
103             s.show_lines([0,1]).display()
104
105
106 auto.from_markdown('section`Plotting and DataFrame` toc``')
107
108 # Matplotlib
109 with auto.slide() as sl:

```

```

113     plt.rcParams['svg.fonttype'] = 'none' # Global setting, enforce s
114     x = np.linspace(0,2*np.pi)
115     with plt.style.context('ggplot'):
116         fig, ax = plt.subplots(figsize=(3.4,2.6))
117         _ = ax.plot(x,np.cos(x))
118         write([ax, s.focus_lines([1,3,4])])
119
120     sl.set_css({'background':'linear-gradient(to right, #FFDAB9 0%, #F0E6
121
122 # Plotly and Pandas DataFrame only show if you have installed
123 with slides.source.context(auto_display = False) as source:
124     try:
125         import pandas as pd
126         df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seabo
127         df = df.describe() #Small for display
128     except:
129         df = '### Install `pandas` to view output'
130
131 with auto.slide():
132     write(['## Writing Pandas DataFrame', df, source])
133
134 with slides.source.context(False) as s:
135     try:
136         import plotly.graph_objects as go
137         fig = go.Figure()
138         fig.add_trace(go.Bar(y=[1,5,8,9]))
139     except:
140         fig = '### Install `plotly` to view output'
141
142 with auto.slide():
143     write(['## Writing Plotly Figure',fig, s])
144
145 def race_plot():
146     import numpy as np
147     import matplotlib.pyplot as plt
148
149     x = np.linspace(0,0.9,10)
150     y = np.random.random((10,))
151     _sort = np.argsort(y)
152
153     plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd

```

```

157
158     for s in ['right', 'top', 'bottom']:
159         ax.spines[s].set_visible(False)
160
161     ax.set(title='Race Plot', ylim = [-0.05, 0.95], xticks=[], yticks=[c for c in range(0, 100)])
162     return plt2html(fig, transparent=False, caption='A Silly Plot')
163
164
165 # Interactive widgets.
166 with auto.slide():
167     with slides.source.context(auto_display = False) as src:
168         import ipywidgets as ipw
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', layout={'width': 150})
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 slides
193         ''')
194
195     def display_plot(): return race_plot().display()
196
197     write(lambda: slides.on_refresh(display_plot), rslide.get_source()) #

```

```

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()
210
211     with slides.source.context(auto_display = False) as s:
212         fig, ax = plt.subplots()
213         x = np.linspace(0,obj+1,50+10*(idx+1))
214         ax.plot(x,np.sin(x));
215         ax.set_title(f'$f(x)=\sin(x)$,  $0 < x < \{idx+1\}$ ')
216         ax.set_axis_off()
217         slides.notes.insert(f'## This is under @frames decorator!')
218
219     slides.write([f'### This is Slide {slides.running.number}.{idx}\n and
220                  s.show_lines([idx])
221                  ],ax,widths=[40,60])
222     if idx == 0: #Only show source code of first frame
223         s.show_lines([5]).display()
224     slides.write(slides.cite('This'))
225
226 auto.from_markdown('section`Controlling Content on Frames` toc`### Content
227
228 # Frames structure
229 boxes = [f'<div style="background:var(--hover-bg);width:auto;height:2em;p
230 @auto.frames(*boxes, repeat=False)
231 def f(obj,idx):
232     slides.write('# Frames with \n#### `repeat = False`')
233     slides.write(obj)
234 @auto.frames(*boxes, repeat=True,frame_height='100%')
235 def f(obj,idx):
236     slides.running.set_animation(None) #Disable animation for showing bul
237     slides.write('# Frames with \n#### `repeat = True` and Fancy Bullet L
238     slides.bullets(obj, marker='♥').display()
239
240 @auto.frames(*boxes, repeat=[(0,1),(2,3)])
241 def f(obj,idx):

```

```

245
246     s.display()
247
248 with auto.slide() as s:
249     backward_skipper.display()
250     forward_skipper.set_target()
251     slides.format_css({'goto-button .fa.fa-minus': slides.icon('arrow', c
252     slides.write('## Displaying image from url from somewhere in Kashmir
253     try:
254         slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2
255     except:
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}:{t
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             '''

```

```

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         'proxy'[Paste Widegets Screenshot Here]`'
294     ]
295 )
296 s.get_source().display()
297
298
299 auto.from_markdown('''
300 ##  $\LaTeX$  in Slides
301 Use '$ $' or '$$ $$' to display latex in Markdown, or embed images of equ
302  $\LaTeX$  needs time to load, so keeping it in view until it loads would h
303 {.note-warning}
304
305  $\int_0^1 \frac{1}{1-x^2} dx$ 
306 ''', trusted=True)
307
308 with auto.slide(), slides.source.context():
309     slides.write('## Built-in CSS styles')
310     slides.css_styles.display()
311
312 auto.from_markdown('section`Custom Objects Serilaization` toc`### Content
313
314 with auto.slide() as some_slide:
315     slides.write('## Serialize Custom Objects to HTML\nThis is useful for
316     with slides.suppress_stdout(): # suppress stdout from register fuctio
317         @slides.serializer.register(int)
318         def colorize(obj):
319             color = 'red' if obj % 2 == 0 else 'green'
320             return f'<span style="color:{color};">{obj}</span>'
321         slides.write(*range(10))
322
323     some_slide.get_source().display()
324
325 with auto.slide():
326     slides.write('## This is all code to generate slides section`Code to
327     slides.source.from_callable(slides.demo).display()
328     slides.source.from_file(__file__).display()
329
330

```



```

333
334
335 with auto.slide() as bib_slide:
336     slides.write('citations`## Reference via Markdown\n——`')
337     bib_slide.get_source().display()
338
339
340 slides.navigate_to(0) # Go to title slide

```



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
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 ```

```

```
14 proxy`something will be here in end`
```

Markdown: Slide 3

```
1 # IPySlides Online Running Sources
2 ::: note
3     - [Edit on Kaggle](https://www.kaggle.com/massgh/ipyslides)
4     - Launch example Notebook [!Binder](https://mybinder.org/badge_logo)
5     - Watch a [Youtube Video](https://www.youtube.com/watch?v=ytfWIYbJte)
6
7 [^1]: Add references like this per slide. Use slides.cite() or in markdown
```

Markdown: Slide 4

```
1 section`Variety of Content Types to Display` toc`### Contents`
```

Markdown: Slide 5

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

Markdown: Slide 6

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

Markdown: Slide 7

```
1 ## Interactive Widgets
2 ### Any object in `ipywidgets`<span class='text-box' style = 'display:inline-block; width: 100px; height: 1em; border: 1px solid black; vertical-align: middle; margin-left: 5px; margin-top: -1px; margin-bottom: -1px; font-family: monospace; font-size: 0.8em; padding: 2px 5px; white-space: nowrap; background-color: #f9f9f9; border-radius: 3px; box-shadow: 0 1px 0px #ccc; transform: rotate(-1deg);'></span>'color[red]` or libraries based on ipywigdtes such as color[red]`bqplot`,color[gree
3 or libraries based on ipywigdtes such as color[red]`bqplot`,color[gree
4 can be included as well.
5 {.warning}
```

Markdown: Slide 8

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

Markdown: Slide 9

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

Python: Slide 10

```
1 write(1## Plotting with Matplotlib)
```

```

4     plt.rcParams['svg.fonttype'] = 'none' # Global setting, enforce same
5     x = np.linspace(0,2*np.pi)
6     with plt.style.context('ggplot'):
7         fig, ax = plt.subplots(figsize=(3.4,2.6))
8         _ = ax.plot(x,np.cos(x))
9         write([ax, s.focus_lines([1,3,4])])
10
11     sl.set_css({'background':'linear-gradient(to right, #FFDAB9 0%, #F0E68C 1

```

Python: Slide 11

```

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

```

Python: Slide 12

```

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

```

Python: Slide 13

```

1 with slides.source.context(auto_display = False) as src:
2     import ipywidgets as ipw
3
4     write('''
5         ## Interactive Apps with Widgets section`Interactive Widgets`
6         Use `ipywidgets`, `bqplot`,`ipyvolume`, `plotly Figurewidget` etc
7         ::: note-tip
8         Export to Slides/Report to see what happens to this slide and
9         ''')
10    plot_html = ipw.HTML('Plot will be here')
11    button = ipw.Button(description='Click me to update race plot',layout
12
13    write([plot_html,button], src)
14
15    def update_plot(btn):
16        plot_html.value = race_plot().value #Convert to html string
17
18    button.on_click(update_plot)
19    update_plot(None) #Initialize plot
20
21 slides.source.from_callable(race_plot).display()

```

Python: Slide 14

```

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

```

```

6 def display_plot(): return race_plot().display()
7
8 write(lambda: slides.on_refresh(display_plot), rslide.get_source()) # Onl
9 slides.source.from_callable(race_plot).display()

```

Markdown: Slide 15

```

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

```

Markdown: Slide 17

```

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

```

Python: Slide 21

```

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

```

Python: Slide 22

```

1 write(f"### Watching Youtube Video?")
2 write('**Want to do some drawing instead?**\nClick on pencil icon and dra
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.tr
10     slides.set_overlay_url('https://tldraw.com')
11
12 ys.get_source().display()

```

Python: Slide 23

```

1 write('## Block API\nNew `block` API is as robust as `write` command. On
2 slides.block_red(
3     [
4         '### Table',
5         '''

```

```

9         |r1 |r2 |r3 |
10        ''' ,
11    ],
12    [
13        '### Widgets',
14        ipw.Checkbox(description='Select to do nothing',indent=False),
15        slides.alt(ipw.IntSlider(),lambda w: f'<input type="range" min="{w}" max="{w+1}" value="{w}">'),
16        ipw.Button(description='Click to do nothing'),
17        'proxy'[Paste Widegets Screenshot Here]''
18    ]
19 )
20 s.get_source().display()

```

Markdown: Slide 24

```

1  ##  $\LaTeX$  in Slides
2  Use '$ $' or '$$ $$' to display latex in Markdown, or embed images of e
3   $\LaTeX$  needs time to load, so keeping it in view until it loads woul
4  {.note-warning}
5
6  $$\int_0^1 \frac{1}{1-x^2} dx$$

```

Python: Slide 25

```

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

```

Markdown: Slide 26

```

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

```

Python: Slide 27

```

1  slides.write('## Serialize Custom Objects to HTML\nThis is useful for dis
2  with slides.suppress_stdout(): # suppress stdout from register fuction be
3      @slides.serializer.register(int)
4      def colorize(obj):
5          color = 'red' if obj % 2 == 0 else 'green'
6          return f'<span style="color:{color};">{obj}</span>'
7      slides.write(*range(10))
8
9  some_slide.get_source().display()

```

Python: Slide 28

```

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

```

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

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()
```