▶ Show Code

Creating Slides

Abdul Saboor[1], Unknown Author[2]
Dec 04, 2022

[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

Table of 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

HBox(children=(HTML(value='', _dom_classes=('GoToHtml',)), Button(description='Skip 5 Slides', style=ButtonSty...

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.

This is inline markdown parsed by magic

Version: 2.1.7 as executed from below code in markdown.

Python

```
# get the slides instance under a python block in Markdown file, we will use it later to run a cell magic.
myslides = get_slides_instance()
import ipyslides as isd
version = isd.__version__
%xmd #### This is inline markdown parsed by magic {.Note .Warning}
```

I was added at end using `s2.insert_markdown`

IPySlides Online Running Sources

Launch as voila slides (may not work as expected [1]) launch binder

Edit on Kaggle

Launch example Notebook launch binder

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

Table of 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

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 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]

1.

This is refernce to FigureWidget using slides.cite command

(reference at end) can be included in iwrite command as well as other objects that can be passed to write with caveat of Javascript.

Commands which do all Magic!

Slides.**write**(*columns, width_percents=None, className=None)

Writes markdown strings or IPython object with method _repr_<html,svg,png,...>_ in each column of same with. If width_percents is given, column width is adjusted. Each column should be a valid object (text/markdown/html/ have *repr or to* method) or list/tuple of objects to form rows or explictly call rows.

- Pass int,float,dict,function etc. Pass list/tuple in a wrapped list for correct print as they used for rows writing too.
- Give a code object from Slides.source.context[from_...] to it, syntax highlight is enabled.
- Give a matplotlib figure/Axes to it or use ipyslides.objs_formatter.plt2html().
- Give an interactive plotly figure.
- Give a pandas dataframe df or df.to_html().
- Give any object which has to_html method like Altair chart. (Note that chart will not remain interactive, use display(chart) if need interactivity like brushing etc.)
- Give an IPython object which has _repr_<repr>_ method where is one of ('html','markdown','svg','png','jpeg','javascript','pdf','pretty','json','latex').
- Give a function/class/module (without calling) and it will be displayed as a pretty printed code block.
- Give a registered object using @Slides.serializer.registor decorator.

If an object is not in above listed things, obj.__repr__() will be printed. If you need to show other than **repr**, use display(obj) outside write command or use methods specific to that library to show in jupyter notebook.

If you give a className, add CSS of it using format_css function and provide it to write function. Get a list of already available classes using slides.css_styles. For these you dont need to provide CSS.

Note: Use keep_format method to bypass markdown parser, for example keep_format(altair_chart.to_html()). Note: You

Each obj in columns could be an IPython widget like ipywidgets,bqplots etc or list/tuple (or wrapped in rows function) of widgets to display as rows in a column. Other objects (those in write command) will be converted to HTML widgets if possible. Object containing javascript code may not work, use write command for that.

If you give a className, add CSS of it using format_css function and provide it to iwrite function. Get a list of already available classes using slides.css_styles. For these you dont need to provide CSS.

Returns: writer, columns as reference to use later and update. rows are packed in columns.

Examples:

```
writer, x = iwrite('X') # writer = iwrite('X'); x = writer.cols[0] # gives same result
writer, (x,y) = iwrite('X','Y')
writer, (x,y) = iwrite(['X','Y']) # One column with two rows
writer, (x,y),z = iwrite(['X','Y'],'Z')
#We unpacked such a way that we can replace objects with new one using `grid.update`
new_obj = writer.update(x, 'First column, first row with new data') #You can update same `new_obj` with it's own widget methods.
```

Slides.**parse_xmd**(extended_markdown, 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 variable
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 paragraph will be inside info block due to class at bottom
{.Info}
+++
# Second column is 60% wide
This {{var_name}} is code from above and will be substituted with the value of var_name
```

```python
# This will not be executed, only shown
```
|| Inline-column A || Inline-column B ||
````

Each block can have class names (speparated with space or .) (in 1.4.7+) 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. Aynthing with class name 'report-only' will not be displayed on slides, but appears in document when Slides.export.<export_function> is called.

Note: Nested blocks are not supported.

This function was added in 1.4.6

New in 1.7.2

If an object does not render as you want, use display(object) or register it as you want using @Slides.serializer.register decorator

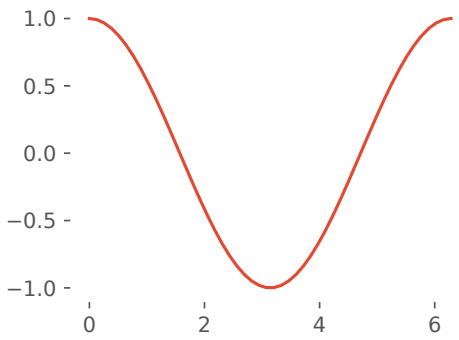
Python

```
write([slides.doc(write,'Slides'), slides.doc(iwrite,'Slides'), slides.doc(slides.parse_xmd,'Slides')])
write("#### If an object does not render as you want, use `display(object)` or register it as you want using `@Slides.serializer.register` decorator")
```

Table of 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

Plotting with Matplotlib



Python

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

```
import pandas as pd
 + 2 more lines ...
df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
 + 7 more lines ...
df = df.describe() #Small for display
```

Writing Altair Chart

ⓘ May not work everywhere, needs javascript

Python

```
 + 1 more lines ...
import altair as alt
alt.themes.enable('dark')
df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
chart = alt.Chart(df,width=300,height=260).mark_circle(size=60).encode(
    x='sepal_length',
    y='sepal_width',
    color='species',
    size = 'petal_width',
    tooltip=['species', 'sepal_length', 'sepal_width','petal_width','petal_length']
    ).interactive()
 + 1 more lines ...
```

Writing Plotly Figure

Python

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

Interactive Apps on Slide

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

HBox(children=(VBox(children=('

Plot will be here! Click button below to activate it!

', Button(descrip...

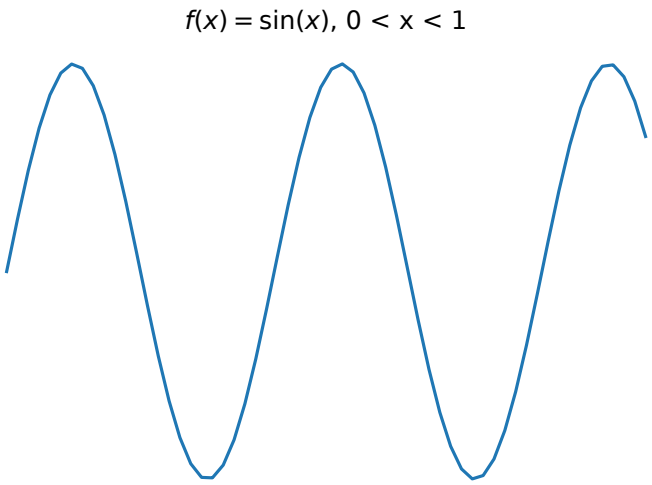
HBox(children=(HTML(value='', _dom_classes=('GoToHtml',)), Button(description='Skip All Next Frames', style=Bu...

This is Slide 15.0

and we are animating matplotlib

Python

```
fig, ax = plt.subplots()
+ 6 more lines ...
```



Python

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

2

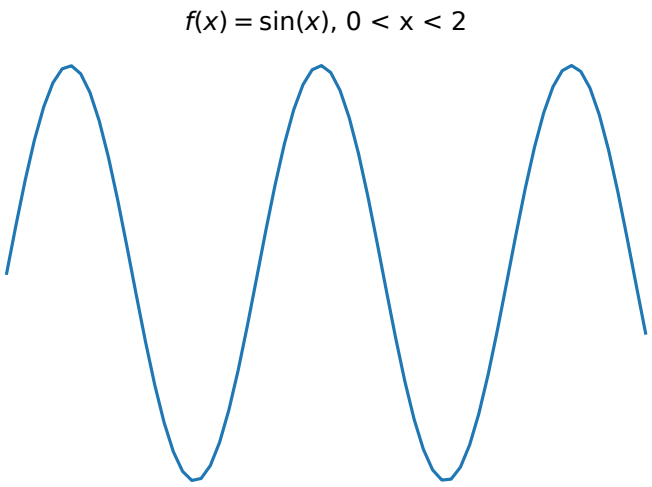
2. Set citation for key 'This'.

This is Slide 15.1

and we are animating matplotlib

Python

```
+ 1 more lines ...
x = np.linspace(0,obj+1,50+10*(idx+1))
+ 5 more lines ...
```



2

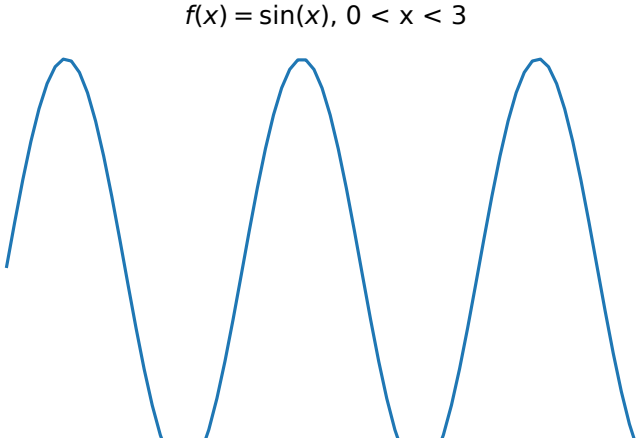
2. Set citation for key 'This'.

This is Slide 15.2

and we are animating matplotlib

Python

```
+ 2 more lines ...
ax.plot(x,np.sin(x));
+ 4 more lines ...
```



2

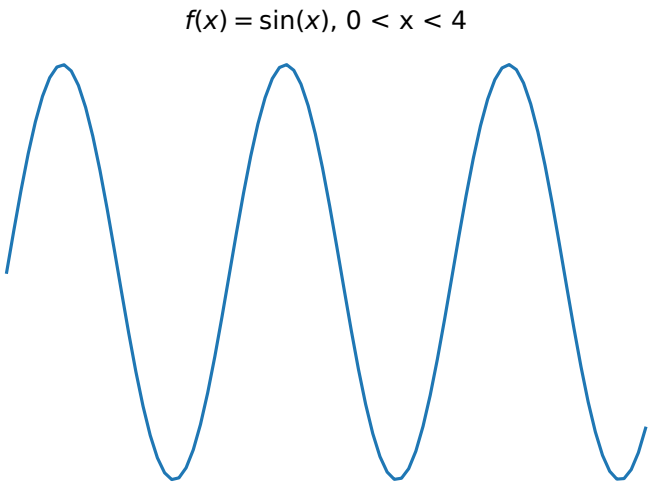
2. Set citation for key 'This'.

This is Slide 15.3

and we are animating matplotlib

Python

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



2

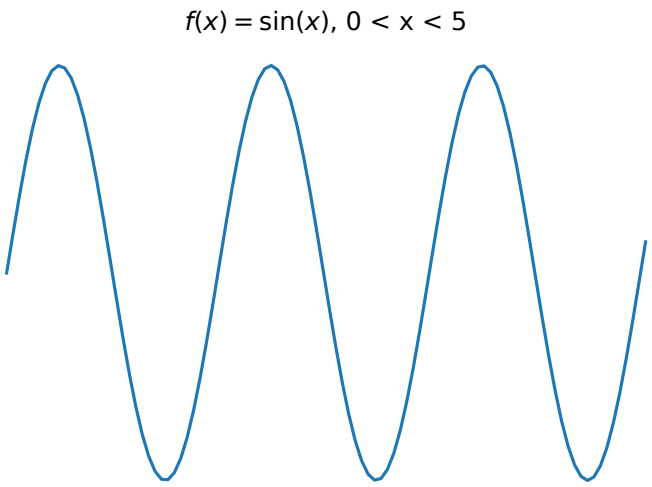
2. Set citation for key 'This'.

This is Slide 15.4

and we are animating matplotlib

Python

```
+ 4 more lines ...
ax.set_axis_off()
+ 2 more lines ...
```



2

2. Set citation for key 'This'.

Table of 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

Frames with

repeat = False

1

Frames with

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

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

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

3

4

Python

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

HBox(children=(HTML(value='', _dom_classes=('GoToHtml',)), Button(description='Skip All Previous Frames', styl...

Displaying image from url from somewhere in Kashmir (کشمیر)

Python

```
slides.goto_button(slides.running.number - 5, 'Skip All Previous Frames')
slides.write('## Displaying image from url from somewhere in Kashmir color[crimson]`(کشمیر)` section`Miscellaneous Content`')
try:
  slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d61fe/master/pass/top-image%20(1).jpg').display()
except:
  slides.write('Could not retrieve image from url. Check internt connection!',className='Error')
s.display()
```

Watching Youtube Video?

Python

```
write(f"### Watching Youtube Video?")
write(YouTubeVideo('Z3iR551KgpI',width='100%',height='266px'))
@slides.notify_later()
def push():
  t = time.localtime()
  return f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm_min:02}'

s.display() # s = source.context(style='vs', className="Youtube")
```

Here is Table

h1	h2	h3
d1	d2	d3
r1	r2	r3

Python

```
write('## Data Tables')
# Remember myslides variable was assigned in a python block
# in markdown just in start. Magic!
write(myslides.block_r('Here is Table','<hr/>','''
 |h1|h2|h3|
 |---|---|---|
 |d1|d2|d3|
 |r1|r2|r3|
 '''))
s.focus_lines([3,4,5,6]).display()
```

$\LaTeX$ in Slides

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.

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

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

Built-in CSS styles

Python

```
slides.css_styles.display()
slides.write('Info',className='Info')
slides.write('Warning',className='Warning')
slides.write('سارے جہاں میں دھوم ہماری زباں کی ہے۔',className='Right RTL')
```

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

className	Formatting Style
'Center'	------Text------
'Left'	Text----------
'Right'	----------Text
'RTL'	------ اردو عربی
'Info'	Blue Text
'Warning'	Orange Text
'Success'	Green Text
'Error'	Red Text
'Note'	Text with info 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.
'Block'	Block of text/objects
'Block-[color]'	Block of text/objects with specific background color from red, green, blue, yellow, cyan, magenta and gray.
'RawText'	Text will not be formatted and will be shown as it is.

سارے جہاں میں دھوم ہماری زباں کی ہے۔

Python

```
slides.rows(
  '## Can skip `write` commnad sometimes',
  slides.cols('### Column A','### Column B',className='Info'),
  '||### Column C {.Warning}||### Column D {.Success}||',
).display()
slides.write('----') # In Python < 3.8, context manager does not properly handle end of code block, so use this to end context
```

Can skip write commnad sometimes

Column A

Column B

Column C

Column D

Table of 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

```
    return f'<span style="color:{color};">{obj}</span>'

slides.write(*range(10))
```

This is all code to generate slides

e:\research\ipyslides\ipyslides_demo.py

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

from ipyslides.writers import write, iwrite
from ipyslides.formatters import libraries, __reprs__
from ipyslides._base.intro import logo_svg

markdown_str = """# Creating Slides
class`Center`
alert`Abdul Saboor`sup`1`, Unknown Authorsup`2`

today``

class`TextBox`
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>
---
section`Introduction`
---
# 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 use it later to run a cell magic.
myslides = get_slides_instance()
import ipyslides as isd
version = isd.__version__
%xmd #### This is inline markdown parsed by magic {.Note .Warning}
```
Version: {{version}} as executed from below code in markdown.
{{source}}

IPySlides Online Running Sources
Launch as voila slides (may not work as expected [^1])[![Binder](https://mybinder.org/badge_logo.svg)]
(https://mybinder.org/v2/gh/massgh/ipyslides-voila/HEAD?
urlpath=voila%2Frender%2Fnotebooks%2Fipyslides.ipynb)
{.Note .Error}

[Edit on Kaggle](https://www.kaggle.com/massgh/ipyslides)
{.Note .Warning}

Launch example Notebook [![Binder](https://mybinder.org/badge_logo.svg)]
```

```
generally.

"""

def demo(slides_instance):
  "We can import `ipyslides.Slides` and use it to create slides, but we will use the slides instance passed to this function."
  slides = slides_instance
  slides.close_view() # Close any previous view to speed up loading 10x faster on average
  slides.clear() # Clear previous content

  # Create shortcut for autoslide/autoframes, but DO NOT do it in Notebook, you may get countless slides there with each run.
  def autoslide(*args,**kwargs): return slides.slide(slides.auto_number, *args, **kwargs)
  def autoframes(*args,**kwargs): return slides.frames(slides.auto_number, *args, **kwargs)

  slides.settings.set_footer('Author: Abdul Saboor عبدالصبور')
  slides.settings.set_logo(logo_svg,width=60) # This is by defualt a logo of ipyslides
  slides._citation_mode = 'global' # This could be changed by other functions
  slides.set_citations({'pf': 'This is refernce to FigureWidget using `slides.cite` command'})

  #Demo for loading slides from a file or text block
  s0, s1, s2, *others = slides.from_markdown(0,markdown_str, trusted=True)

  section_slides = {'1':s1} # We collect all slides that have section to update at end. Very useful

  with s0.insert(0):
    s0.source.display(collapsed = True)

  slides.shell.user_ns['write'] = write #Inject variable in IPython shell

  # Insert source of slide 2
  with s2.insert(0):
    s2.source.display(collapsed = True)
    slides.goto_button(slides.running.number+5, 'Skip 5 Slides')

  s2.insert_markdown({-1: f'alert`I was added at end using \`s2.insert_markdown\``'})

  #Now generate many slides in a loop
  __contents = ["section`Variety of Content Types to Display`",
  f"""
 ## IPython Display Objects
 #### Any object with following methods could be in`write` command:
 {', '.join([f'`_repr_{rep}_`' for rep in __reprs__])}
 Such as color[navy_skyblue]`IPython.display.[HTML,SVG,Markdown,Code]` etc. or third party such as `plotly.graph_objects.Figure`{{.Warning}}.
 """,
  f"""
 ## 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 `display(obj)` to show inline or use library's specific
 command to show in Notebook outside color[teal_whitesmoke]`write`.
```

```
 ### Any object in `ipywidgets`{slides.textbox('<a href="https://ipywidgets.readthedocs.io/en/latest/">Link to ipywidgtes right here using textbox command</a>')}
 or libraries based on ipywidgtes such as color[red]`bqplot`,color[green]`ipyvolume`,plotly's `FigureWidget` cite`pf`(reference at end)
 can be included in `iwrite` command as well as other objects that can be passed to `write` with caveat of Javascript.
 {{.Warning}}
 """,
  '## Commands which do all Magic!']

  for i, content in enumerate(__contents):
    with autoslide(props_dict = {'':dict(background = 'skyblue')}):
      write(textwrap.dedent(content))
      if i == 4:
        with slides.source.context(auto_display = False) as s:
          write([slides.doc(write,'Slides'), slides.doc(iwrite,'Slides'), slides.doc(slides.parse_xmd,'Slides')])
          write("#### If an object does not render as you want, use `display(object)` or register it as you want using `@Slides.serializer.register` decorator")

        s.show_lines([0,1]).display()
      if i == 0:
        section_slides['2'] = slides.running

  with autoslide() as section_slides['3']:
    slides.write('section`Plotting and DataFrame`')

  # Matplotlib
  with autoslide(props_dict = {'': dict(background='linear-gradient(to right, #FFDAB9 0%, #F0E68C 100%)')}):
    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 fonts as presentation
      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])])

  # Plotly and Pandas DataFrame only show if you have installed
  try:
    with slides.source.context(auto_display = False) as source:
      import pandas as pd
      import altair as alt
      alt.themes.enable('dark')
      df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
      chart = alt.Chart(df,width=300,height=260).mark_circle(size=60).encode(
        x='sepal_length',
        y='sepal_width',
        color='species',
        size = 'petal_width',
        tooltip=['species', 'sepal_length', 'sepal_width','petal_width','petal_length']
        ).interactive()
      df = df.describe() #Small for display
```

```
152
153   with autoslide(props_dict = {'':dict(background='#800000')}):
154     write(('## Writing Pandas DataFrame',df))
155     source.show_lines([0,3,11]).display()
156
157   with autoslide():
158     slides.write(('## Writing Altair Chart\nMay not work everywhere, needs javascript\n{.Note
    .Warning}',chart))
159     source.show_lines(range(1,11)).display()
160
161   try:
162     with slides.source.context(False) as s:
163       import plotly.graph_objects as go
164       fig = go.Figure()
165       fig.add_trace(go.Bar(y=[1,5,8,9]))
166   except:
167     fig = '### Install `plotly` to view output'
168
169   with autoslide():
170     write(('## Writing Plotly Figure',fig))
171     s.display()
172
173   # Interactive widgets.
174   with autoslide():
175     with slides.source.context(auto_display = False) as src:
176       import ipywidgets as ipw
177       import numpy as np, matplotlib.pyplot as plt
178
179       write('## Interactive Apps on color[var(--accent-color)]`Slide`\n Use `ipywidgets`,
    `bqplot`,`ipyvolume` , `plotly Figurewidget` etc. to show live apps like this!')
180       writer, (plot,button, _), code = slides.iwrite([
181         '## Plot will be here! Click button below to activate it! section`Interactive Widgets`',
182         ipw.Button(description='Click me to update race plot',layout=ipw.Layout(width='max-content')),
183         "[Check out this app]
    (https://massgh.github.io/pivotpy/Widgets.html#VasprunApp)"],src.focus_lines([4,5,6,7,*range(24,30)]))
184
185       def update_plot():
186         x = np.linspace(0,0.9,10)
187         y = np.random.random((10,))
188         _sort = np.argsort(y)
189
190         fig,ax = plt.subplots(figsize=(3.4,2.6))
191         ax.barh(x,y[_sort],height=0.07,color=plt.cm.get_cmap('plasma')(x[_sort]))
192
193         for s in ['right','top','bottom']:
194           ax.spines[s].set_visible(False)
195
196         ax.set(title='Race Plot', ylim = [-0.05,0.95], xticks=[],yticks=[c for c in x],yticklabels=
    [rf'$X_{int(c*10)}$' for c in x[_sort]])
197         writer.update(plot, fig) #Update plot each time
198
199       def onclick(btn):
200         plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd.value else 'default'
201         with plt.style.context(plot_theme):
```

```
206
207     slides.notes.insert('## Something to hide from viewers!')
208
209
210   # Animat plot in slides
211   @autoframes(*range(14,19))
212   def func(obj,idx):
213     slides.write('section`Simple Animations with Frames`')
214     if idx == 0:
215       slides.goto_button(slides.running.number + 4, 'Skip All Next Frames')
216
217     with slides.source.context(auto_display = False) as s:
218       fig, ax = plt.subplots()
219       x = np.linspace(0,obj+1,50+10*(idx+1))
220       ax.plot(x,np.sin(x));
221       ax.set_title(f'$f(x)=\sin(x)$, 0 < x < {idx+1}')
222       ax.set_axis_off()
223       slides.notes.insert(f'## This is under @frames decorator!')
224       slides.notify_later()(lambda: f'This is under @frames decorator!')
225
226     slides.write([f'### This is Slide {slides.running.number}.{idx}\n and we are animating matplotlib',
227             s.show_lines([idx])
228             ],ax,width_percents=[40,60])
229     if idx == 0: #Only show source code of first frame
230       s.show_lines([5,6]).display()
231
232     slides.write(slides.cite('This'))
233
234   with autoslide() as section_slides['4']:
235     slides.write('section`Controlling Content on Frames`')
236
237   # Frames structure
238   boxes = [f'<div style="background:var(--tr-hover-bg);width:auto;height:2em;padding:8px;margin:8px;border-
radius:4px;"><b class="Center">{i}</b></div>' for i in range(1,5)]
239   @autoframes(*boxes, repeat=False)
240   def f(obj,idx):
241     slides.write('# Frames with \n#### `repeat = False`')
242     slides.write(obj)
243
244   @autoframes(*boxes, repeat=True,frame_height='100%')
245   def f(obj,idx):
246     slides.running.set_animation(None) #Disable animation for showing bullets list
247     slides.write('# Frames with \n#### `repeat = True` and Fancy Bullet List')
248     slides.bullets(obj, marker='💘').display()
249
250   @autoframes(*boxes, repeat=[(0,1),(2,3)])
251   def f(obj,idx):
252     with slides.source.context(auto_display = False) as s:
253       slides.write('# Frames with \n#### `repeat = [(0,1),(2,3)]`')
254       slides.write(*obj)
255     s.display()
256
257   with autoslide():
258     with slides.source.context(auto_display = False) as s:
259       slides.goto_button(slides.running.number + 5, 'Skip All Previous Frames')
```

```
        slides.image(f'https://assets.gqindia.com/photos/610d2f12e59aeaf2a52d0ffe/master/pass/top
image%20(1).jpg').display()
      except:
        slides.write('Could not retrieve image from url. Check internt connection!',className='Error')
      s.display()

  # Youtube
  from IPython.display import YouTubeVideo
  with autoslide():
    with slides.source.context(auto_display=False, style='vs',className="Youtube") as s:
      write(f"### Watching Youtube Video?")
      write(YouTubeVideo('Z3iR55lKgpI',width='100%',height='266px'))
      @slides.notify_later()
      def push():
        t=time.localtime()
        return f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm_min:02}'

      s.display() # s = source.context(style='vs', className="Youtube")

  # Data Table
  slides.shell.run_cell(f"""
%%slide {slides.auto_number}
with myslides.source.context(auto_display = False) as s:
write('## Data Tables')
# Remember myslides variable was assigned in a python block
# in markdown just in start. Magic!
write(myslides.block_r('Here is Table','<hr/>','''
|h1|h2|h3|
|---|---|---|
|d1|d2|d3|
|r1|r2|r3|
'''))
s.focus_lines([3,4,5,6]).display()
""")

  slides.from_markdown(slides.auto_number, '''
## $\LaTeX$ in Slides
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.
{.Note .Warning}

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

  with autoslide():
    slides.write('## Built-in CSS styles')
    with slides.source.context():
      slides.css_styles.display()
      slides.write('Info',className='Info')
      slides.write('Warning',className='Warning')
      slides.write('سارے جہاں میں دھوم ہماری زباں کی ہے۔',className='Right RTL')

  with autoslide(),slides.source.context():
```

```
    ).display()
    slides.write('----') # In Python < 3.8, context manager does not properly handle end of code block, so use this to end context

  with autoslide() as section_slides['5']:
    slides.write('section`Custom Objects Serilaization`')

  with autoslide():
    slides.write('## Serialize Custom Objects to HTML\nThis is useful for displaying user defined/third party objects in slides')
    with slides.suppress_stdout(): # suppress stdout from register fuction below
      with slides.source.context(auto_display=False) as s:
        @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))

      s.display()

  with autoslide():
    slides.write('## This is all code to generate slides section`Code to Generate Slides`')
    slides.source.from_file(__file__).display()

  with autoslide():
    slides.write('Slides made by using `from_markdown` or `%%slide` magic preserve their full code\n{.Note.Info}')
    slides.get_source().display()

  with autoslide(props_dict={'':dict(background='#9ACD32')}):
    with slides.source.context():
      slides.write('citations`## Reference via Markdown\n----`',
            ['## Reference via Python API\n----',
             *slides.citations])
      slides.write('Markdown is easier to write and read, but Python API is more powerful.')

  for _, slide in section_slides.items():
    slide.insert_markdown({-1:'toc`## Table of Contents\n----`'})

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

if __name__=='__main__':
  print('WARNING: This file is not meant to be run directly. Import in Jupyter Notebook instead!')
```

Slides made by using from_markdown or %%slide magic preserve their full code

Source Code

Markdown: Slide 0

```
# Creating Slides
```

```
class`TextBox`
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`
```

Markdown: Slide 2

```
# 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 use it later to run a cell magic.
myslides = get_slides_instance()
import ipyslides as isd
version = isd.__version__
%xmd #### This is inline markdown parsed by magic {.Note .Warning}
```
Version: {{version}} as executed from below code in markdown.
{{source}}
```

Markdown: Slide 3

```
# IPySlides Online Running Sources
Launch as voila slides (may not work as expected [^1])[![Binder](https://mybinder.org/badge_logo.svg)](https://my
{.Note .Error}

[Edit on Kaggle](https://www.kaggle.com/massgh/ipyslides)
{.Note .Warning}

Launch example Notebook [!
[Binder](https://mybinder.org/badge_logo.svg)](https://mybinder.org/v2/gh/massgh/ipyslides-voila/HEAD?urlpat
{.Note .Success}

[^1]: Add references like this per slide. Use slides.cite() or in markdown cite\`key\` to add citations generally.
```

Python: Slide 22

```
with myslides.source.context(auto_display=False) as s:
  write('## Data Tables')
  # Remember myslides variable was assigned in a python block
  # in markdown just in start. Magic!
  write(myslides.block_r('Here is Table','<hr/>','''
 |h1|h2|h3|
 |---|---|---|
 |d1|d2|d3|
 |r1|r2|r3|
 '''))
  s.focus_lines([3,4,5,6]).display()
```

Markdown: Slide 23

```

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

Python

```python
slides.write('citations`## Reference via Markdown\n----`',
       ['## Reference via Python API\n----',
        *slides.citations])
slides.write('Markdown is easier to write and read, but Python API is more powerful.')
```

Reference via Markdown

1.

This is refernce to FigureWidget using slides.cite command

2. Set citation for key 'This'.

Reference via Python API

1.

This is refernce to FigureWidget using slides.cite command

2. Set citation for key 'This'.

Markdown is easier to write and read, but Python API is more powerful.