

# Creating Slides

Abdul Saboor<sup>1</sup>, Unknown Author<sup>2</sup>

Jan 28, 2023

<sup>1</sup>My University is somewhere in the middle of nowhere

<sup>2</sup>Their 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: 3.1.8 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` <sup>1</sup>

(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, obj)` function to display widget on slides and alternative content in exported slides/report.
- `ipywidgets.HTML` and its subclasses will be displayed as `Slides.alt(widget, value)`. The value of exported HTML will be oldest one.
- 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.

`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  ```multicol 40 60
7  # First column is 40% width
8  If 40 60 was not given, all columns will be of equal width, this paragra

```

```
12 This \{\{var_name\}\} is code from above and will be substituted with the
13 ```
14
15 ```python
```

#### 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.

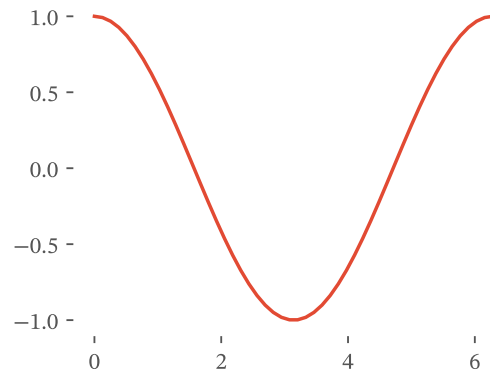
#### Python

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

## Table of Contents

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

# 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
```

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

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!



Plot will be here

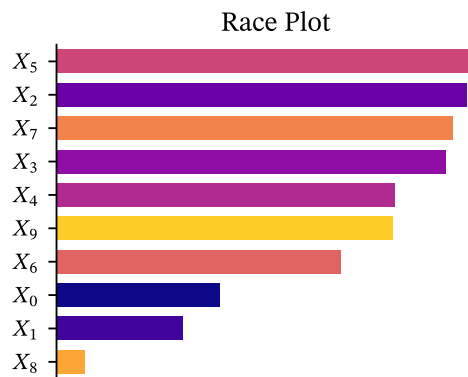
Python

```
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 write([
10     plot_html := ipw.HTML('Plot will be here'),
11     button := ipw.Button(description='Click me to update race plot', layout
12     ], src)
13
14 def update_plot(btn):
15     plot_html.value = race_plot().value #Convert to html string
```

Python

```
1 def race_plot():
2     import numpy as np
3     import matplotlib.pyplot as plt
4
5     x = np.linspace(0,0.9,10)
6     y = np.random.random((10,))
7     _sort = np.argsort(y)
8
9     plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd
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)
```

## Dynamic Content without Widgets



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()) # Onl
9 slides.source.from_callable(race_plot).display()

```

Python

```

1 def race_plot():
2     import numpy as np
3     import matplotlib.pyplot as plt
4
5     x = np.linspace(0,0.9,10)
6     y = np.random.random((10,))
7     _sort = np.argsort(y)
8
9     plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd
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)

```

## 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

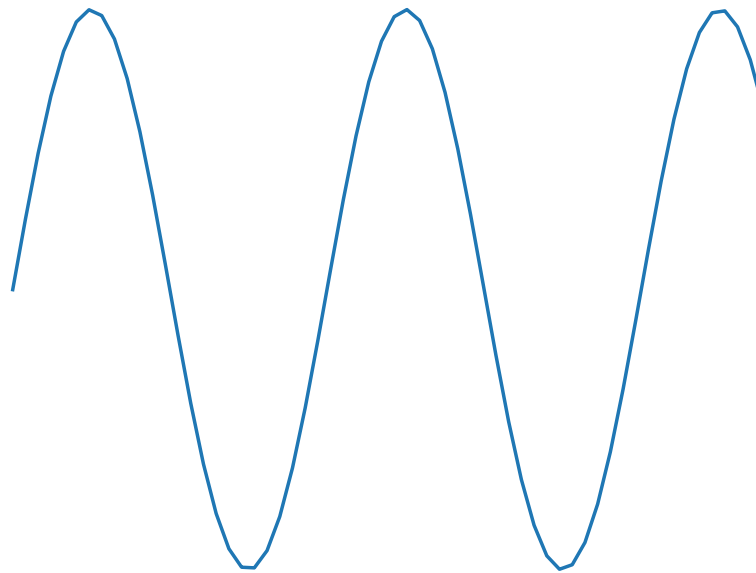
## This is Slide 16.0

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!')
```

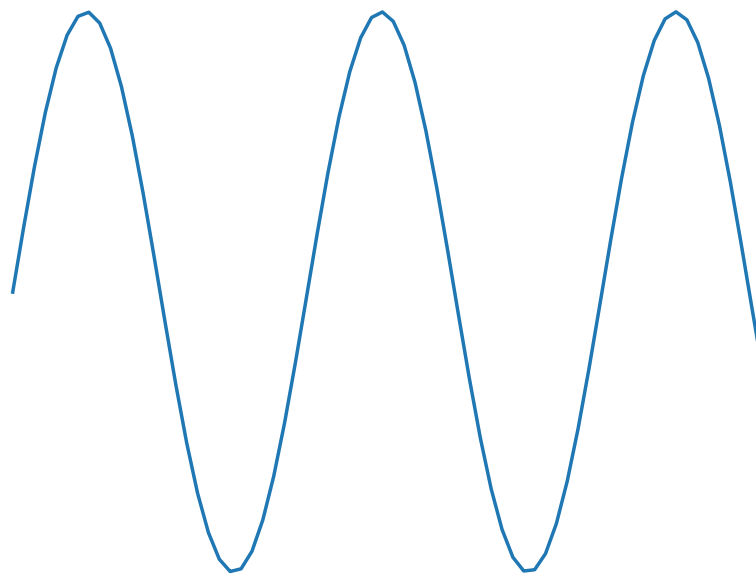
## This is Slide 16.1

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 ...
```

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



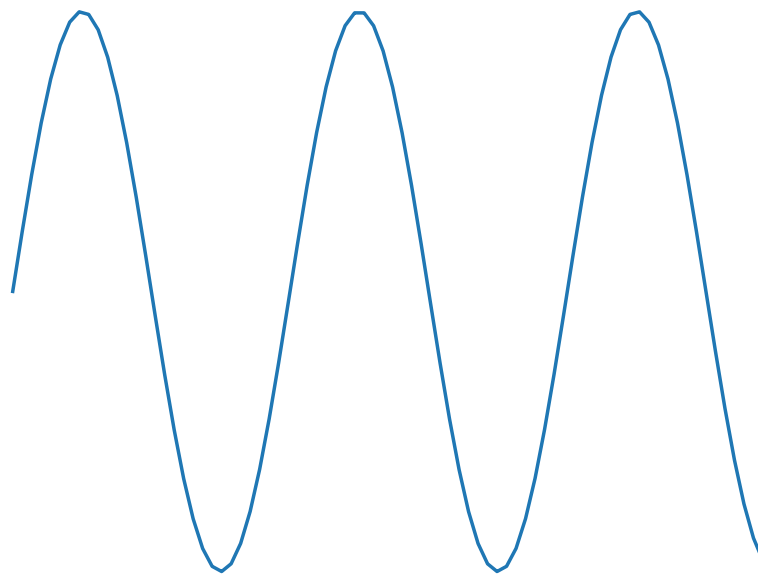
## This is Slide 16.2

and we are animating matplotlib

Python

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

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



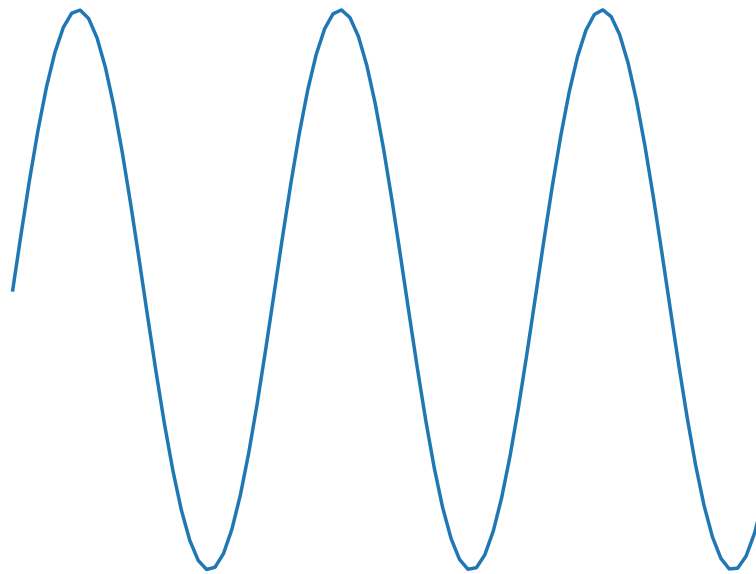
## This is Slide 16.3

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 ...
```

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



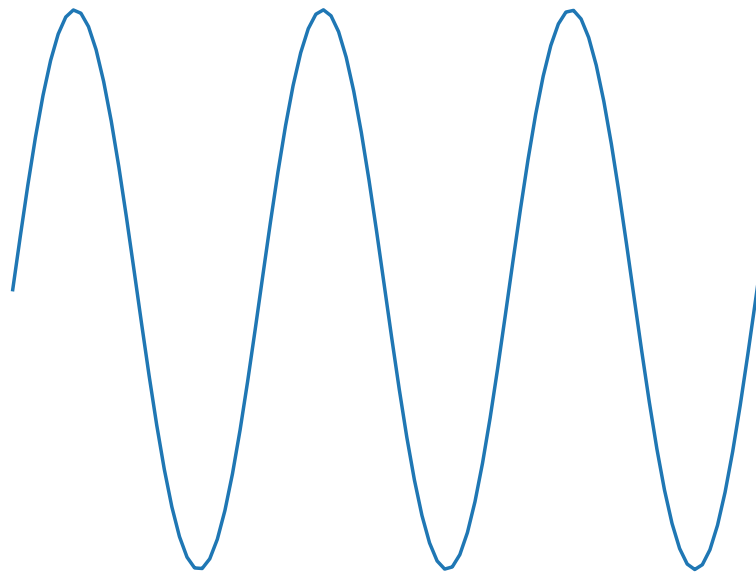
# This is Slide 16.4

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 slides.write('## Displaying image from url from somewhere in Kashmir colo  
2 try:
```

```
6 s.get_source().display()
```

## Watching Youtube Video?

**Want to do some drawing instead?** Click on pencil icon in toolbar above 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 in tool
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()
```

## Data Tables

Here is Table

h1

h2

h3

## Python

```
1 write('## Data Tables')
2 write(slides.block_r('Here is Table', '<hr/>', '''
3     |h1|h2|h3|
4     |---|---|---|
5     |d1|d2|d3|
6     |r1|r2|r3|
7     '''))
8 s.get_source().focus_lines([3,4,5,6]).display()
```

# $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ in Slides

## ⚠ Alert

Use \$ \$ or \$\$ \$\$ to display latex in Markdown, or embed images of equations  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  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.

'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

## 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))
```

# 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')
```



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 ```
12 Version: {{version}} as executed from below code in markdown.
13 {{source}}
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=ytfWIIYbJtel)
6
7 [^1]: Add references like this per slide. Use slides.cite() or in markdown
```



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_`
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, u
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:inl
3 or libraries based on ipywidgets 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('## Plotting with Matplotlib')
2 with slides.source.context(auto_display = False) as s:
3     import numpy as np, matplotlib.pyplot as plt
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))
```

```
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    write([
11        plot_html := ipw.HTML('Plot will be here'),
12        button := ipw.Button(description='Click me to update race plot',1
13    ], src)
14
15    def update_plot(btn):
```

Python: Slide 14

```
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()) # 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 slides.write('## Displaying image from url from somewhere in Kashmir colo
2 try:
3     slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d
4 except:
5     slides.write('Could not retrieve image from url. Check internt connec
```

```

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

```

Python: Slide 23

```

1 write('## Data Tables')
2 write(slides.block_r('Here is Table','<hr/>','''
3     |h1|h2|h3|
4     |---|---|---|
5     |d1|d2|d3|
6     |r1|r2|r3|
7     '''))
8 s.get_source().focus_lines([3,4,5,6]).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)

```

```
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()
3 slides.source.from_file(__file__).display()
```

Python: Slide 29

```
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()
```