

► Show Code

Creating Slides

Assuming you have `ls = ipyslides.LiveSlides()`

- Proceed to create slides:
 - `%slide integer` on cell top auto picks slide and `%title` auto picks title page.
 - `%slide integer -m` can be used to create slide from full markdown (extended one).
 - You can use context managers like `with ls.slide(): ...` and `with ls.title(): ...` in place of `%slide` and `%title` respectively.

Python

```
1 import ipyslides as isd
2 ls = isd.LiveSlides()
3 ls.set_animation(main='zoom')
```

Python

```
1 %%title
2 # create a rich content title page
```

Slide 1

ⓘ This is inline markdown parsed by magic

Version: 1.8.3 as executed from below code in markdown.

Python

```
1 import ipyslides as isd
2 version = isd.__version__
3 %xmd ##### This is inline markdown parsed by magic {.Note .Warning}
```

I am created using `with slides.slide(1)` context manager, so I overwrite the previous slide, and you can not see my full code, but part of it using `LiveSlides.source.context` context manager.

I am Alerted and I am *colored and italic text*

Python

```
1 slides.parse_xmd(s1.markdown) #s1 was assigned as `s0, s1, s2 = slides.from_markdown...` in start.
2 write('##### I am created using `with slides.slide(1)` context manager, '
```

► Show Code

Slide 2

Created using `%slide 2 -m` with markdown only

¹ Reference to this will show at end

Column A

Sub column A

Sub column B

Column B

That version from last slide is still in memory. See it is there 1.8.3

I was added at end using `'s2.insert_markdown'`

I am created using `@slides.frames`

Python

```
1 slides.write(obj)
2 slides.notify_later()(lambda: 'That is a notification which shows you can use decorator this way as
well')
```

IPySlides Online Running Sources

ⓘ Launch as voila slides (may not work as expected¹) [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. ↵

Python

```
1 slides.write(obj)
2 slides.notify_later()(lambda: 'That is a notification which shows you can use decorator this way as well')
```

IPython Display Objects

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

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

Plots and Other Data Types

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

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

Interactive Widgets

Any object in `ipywidgets` [Link to ipywidgtes right here using textbox command](#)

or libraries based on ipywidgtes such as `bqplot`, `ipyvolume`, `plotly`'s `FigureWidget`²(reference at end) can be included in `iwrite` command as well as other objects that can be passed to `write` with caveat of Javascript.

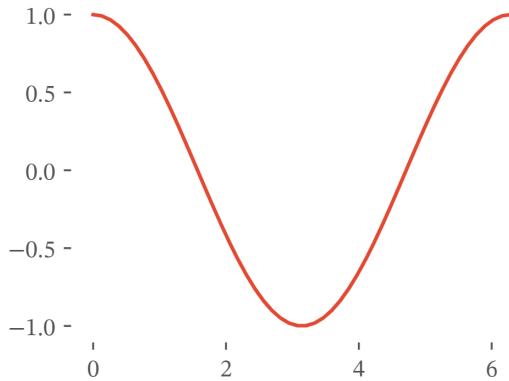
Commands which do all Magic!

`LiveSlides.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 explicitly 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 `LiveSlides.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.

Plotting with Matplotlib



Python

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

Watching Youtube Video?



Python

```
1 write(f"### Watching Youtube Video?")
2 write(YouTubeVideo('Z3iR551KgpI',width='100%',height='266px'))
3 @slides.notify_later()
4 def push():
5     t = time.localtime()
6     return f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm_min:02}'
7
```

Data Tables

Here is Table

h1	h2	h3
d1	d2	d3
r1	r2	r3

Python

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

Writing Pandas DataFrame

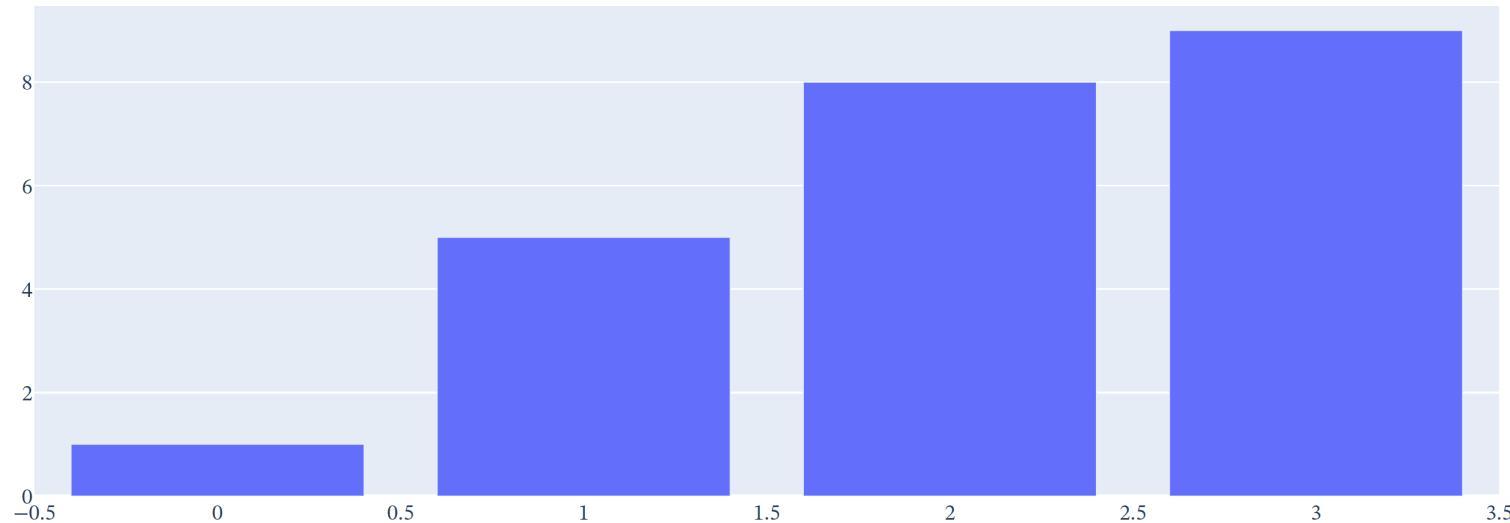
	sepal_length	sepal_width	petal_length	petal_width
count	150.0000 00	150.0000 00	150.0000 00	150.0000 00
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Python

Writing Altair Chart

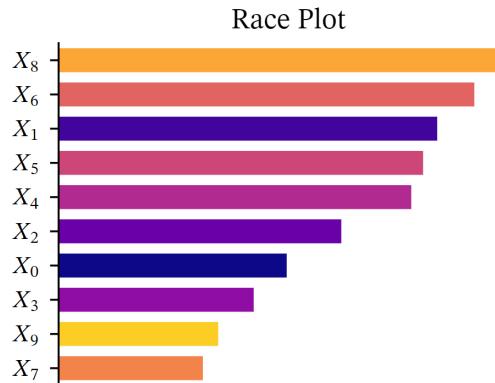
ⓘ May not work everywhere, needs javascript

Writing Plotly Figure



Interactive Apps on Slide

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



Click me to update race plot

Check out this app

Python

```
1 import ipywidgets as ipw
2 import numpy as np, matplotlib.pyplot as plt
3
4 write('## Interactive Apps on Slide\nUse `ipywidgets`,  
`bqplot`, `ipyvolume` , `plotly Figurewidget` etc. to show live  
apps like this!')
5 writer, (plot,button, _), code = slides.iwrite([
6     '## Plot will be here! Click button below to activate it!',  

7     ipw.Button(description='Click me to update race  
plot', layout=ipw.Layout(width='max-content')),  

8     "[Check out this app]  
(https://massgh.github.io/pivotpy/Widgets.html#VasprunApp)", src.  
focus_lines([4,5,6,7,*range(24,30)]))  

9
10 def update_plot():
11     x = np.linspace(0, 0.9, 10)
12     y = np.random.random((10,))
13     _sort = np.argsort(y)
14
15     fig,ax = plt.subplots(figsize=(3.4, 2.6))
16     ax.barh(x,y[_sort],height=0.07,color=plt.cm.get_cmap('plasma')
```

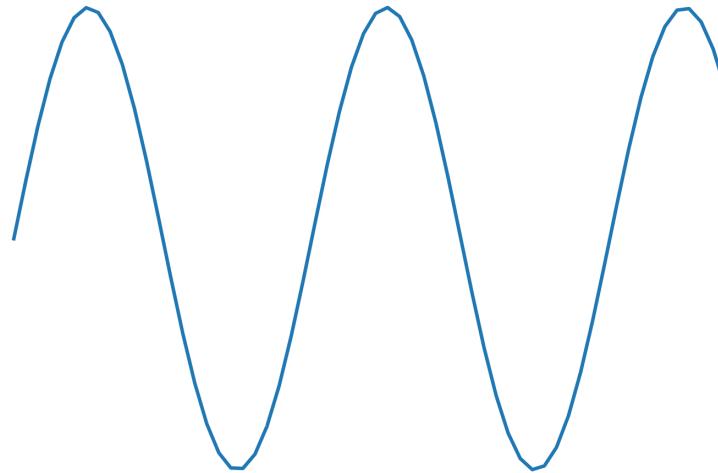
This is Slide 14.1

and we are animating matplotlib

Python

```
1 fig, ax = plt.subplots()  
2 + 6 more lines ...
```

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



Python

```
1 + 5 more lines ...  
2 slides.notes.insert(f'## This is under @frames decorator, so it will be shown only in first frame')  
3 slides.notify_later()(lambda: f'This is under @frames decorator, so it will be shown only in first  
frame')
```

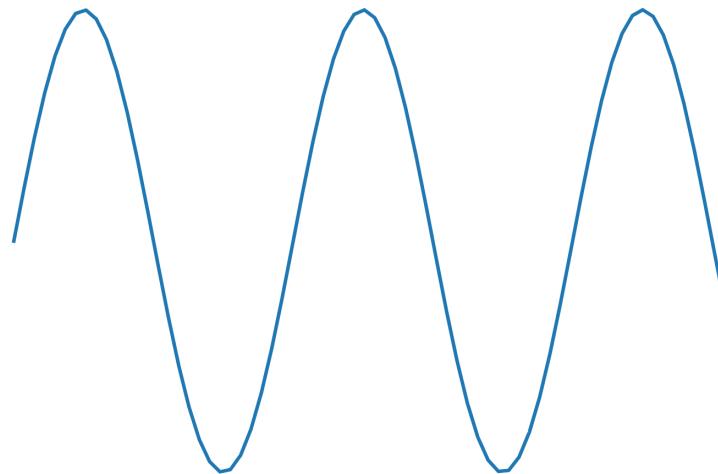
This is Slide 14.2

and we are animating matplotlib

Python

```
1 + 1 more lines ...
2 x = np.linspace(0, obj+1, 50+10*(obj
 - 13))
3 + 5 more lines ...
```

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



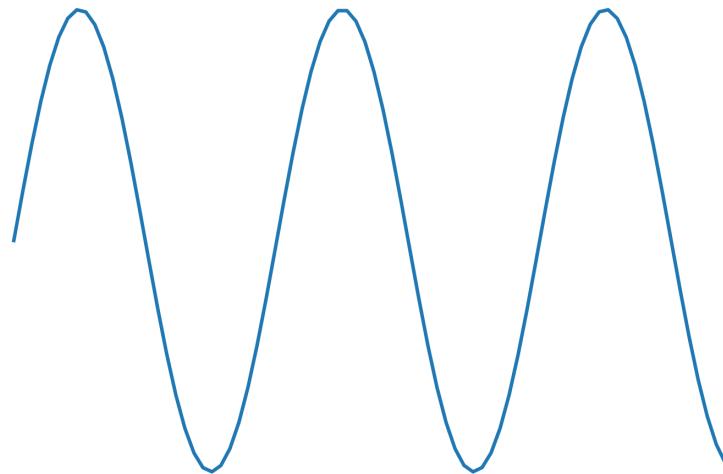
This is Slide 14.3

and we are animating matplotlib

Python

```
1 + 2 more lines ...
2 ax.plot(x, np.sin(x));
3 + 4 more lines ...
```

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



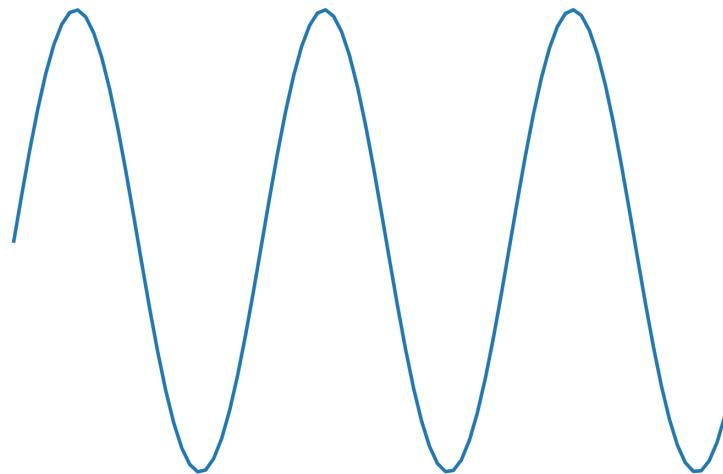
This is Slide 14.4

and we are animating matplotlib

Python

```
1 + 3 more lines ...
2 ax.set_title(f'$f(x)=\sin(x)$, 0 <
   x < {obj - 13}')
3 + 3 more lines ...
```

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



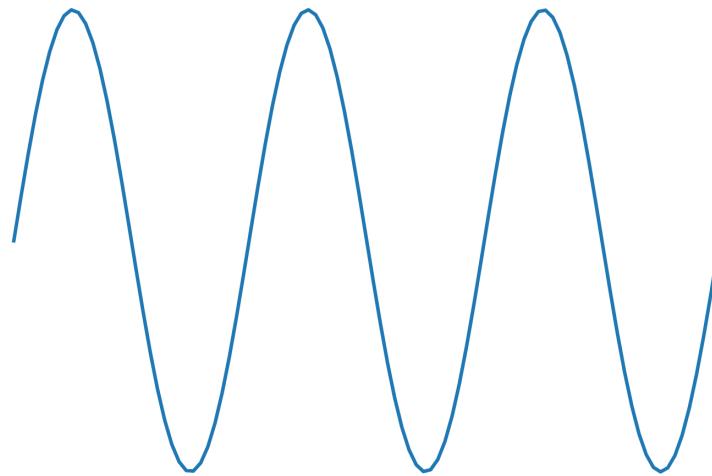
This is Slide 14.5

and we are animating matplotlib

Python

```
1 + 4 more lines ...
2 ax.set_axis_off()
3 + 2 more lines ...
```

$$f(x) = \sin(x), 0 < x < 5$$



Frames with

```
repeat = False
```

1

Frames with

```
repeat = False
```

2

Frames with

```
repeat = False
```

3

Frames with

```
repeat = False
```

4

Frames with

```
repeat = True
```

1

Frames with

```
repeat = True
```

1

2

Frames with

```
repeat = True
```

1

2

3

Frames with

```
repeat = True
```

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)



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

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

className = 'Center' -----Text-----

className = 'Left' Text-----

className = 'Right' -----Text

className = 'RTL' ----- عربی ،،،

className = 'Info' Blue Text

className = 'Warning' Orange Text

className = 'Success' Green Text

className = 'Error' Red Text

className = 'Note' Text with info icon

className = 'slides-only' Text will not appear in exported html with 'build_report'

className = 'report-only' Text will not appear on slides. Useful to fill content in report.

----- 'D1' ----- D1 ----- f ----- / -----

Info

Warning

Can skip `write` command sometimes

Column A

Column C

Column B

Column D



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.serializer.register(int)
2 def colorize(obj):
3     color = 'red' if obj % 2 == 0 else 'green'
4     return f'{obj}' if color == 'red' else f'{obj}'
5
6 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 faster on average
4      self.clear() # Clear previous content
5
6      import runpy
7      file = os.path.join(os.path.dirname(os.path.dirname(__file__)), '_demo.py') # Relative
path to this file
8      slides = runpy.run_path(file, init_globals= {'slides': self})['slides']
9
10     N = len(slides)
11     with slides.slide(N + 1):
12         slides.write('## This is all code to generate slides')
13         slides.write(self.demo)
14         slides.source.from_file(file).display()
```

e:\research\ipyslides\ipyslides_demo.py

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

Source Code

Markdown: Slide 0

```
1 # Creating Slides
2 **Assuming you have `ls = ipyslides.LiveSlides()`**
3
4 - Proceed to create slides:
5   - `%%slide integer` on cell top auto picks slide and `%%title`
    auto picks title page.
6   - `%%slide integer -m`
     can be used to create slide from full markdown (extended one).
7   - You can use context managers like `with ls.slide(): ...` and `with ls.title(): ...`
     in place of `%%slide` and `%%title` respectively.
8
9 ````python
10 import ipyslides as isd
11 ls = isd.LiveSlides()
12 ls.set_animation(main='zoom')
```

Markdown: Slide 2

Author: Abdul Saboor | عبدالصبور | Sep-24-2022 24 / 25

References

¹ This is reference created using markdown

² This is reference to FigureWidget using `slides.cite` command

Python

```
1 slides.write_citations()
```