Creating Slides

Abdul Saboor[1], Unknown Author[2]

Jan 28, 2023

[1]My University is somewhere in the middle of nowhere
[2]Their University is somewhere in the middle of nowhere

👈 **Read instructions in left panel**

Contents

Show Code

Introduction

To see how commands work, use `Slides.docs()` to see the documentation. Here we will focus on using all that functionality to create slides.

Note

This is inline markdown parsed by magic

Version: 3.1.8 as executed from below code in markdown.

Python

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

I was added at end by a given proxy, see the how it was done at the end of the slides

IPySlides Online Running Sources

📝 Note

- Edit on Kaggle
- Launch example Notebook launch binder
- Watch a Youtube Video

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

Contents

IPython Display Objects

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

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

Plots and Other Data Types

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

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

Interactive Widgets

Any object in `ipywidgets` Link to ipywidgtes right here using textbox command

or libraries based on ipywidgtes such as bqplot, ipyvolume, plotly's `FigureWidget` [1]

(reference at end) can be included as well.

Commands which do all Magic!

Slides.**write**(*objs, widths=None)

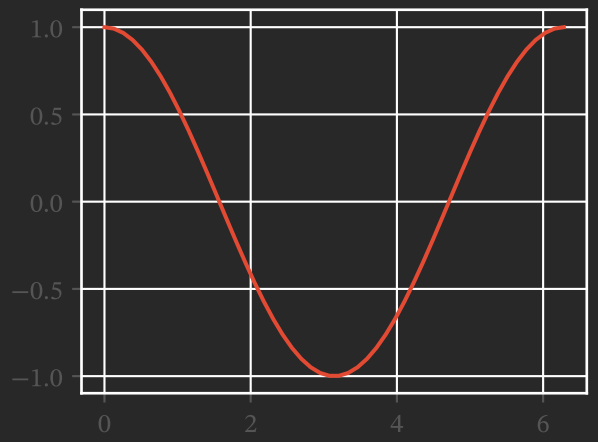
Write `objs` to slides in columns. To create rows in a column, wrap objects in a list or tuple.
You can optionally specify `widths` as a list of percentages for each column.

Write any object that can be displayed in a cell with some additional features:

- Strings will be parsed as as extended markdown that can have citations/python code blocks/Javascript etc.
- Display another function in order by passing it to a lambda function like `lambda: func()`. Only body of the function will be displayed/printed. Return value will be ignored.
- Dispaly IPython widgets such as `ipywidgets` or `ipyvolume` by passing them directly.
- Display Axes/Figure form libraries such as `matplotlib`, `plotly` `altair`, `bokeh`, `ipyvolume` ect. by passing them directly.
- Display source code of functions/classes/modules or other languages by passing them directly or using `Slides.source` API.
- Use `Slides.alt(widget, obj)` function to display widget on slides and alternative content in exported slides/report.
- `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

Table of Contents

Plotting with Matplotlib



Python

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

Writing Pandas DataFrame

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Python

```python
try:
    import pandas as pd
    df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/maste
    df = df.describe() #Small for display
except:
    df = '### Install `pandas` to view output'
```

Writing Plotly Figure

Install `plotly` to view output

Python

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

Interactive Apps with Widgets

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

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

Plot will be here

Python

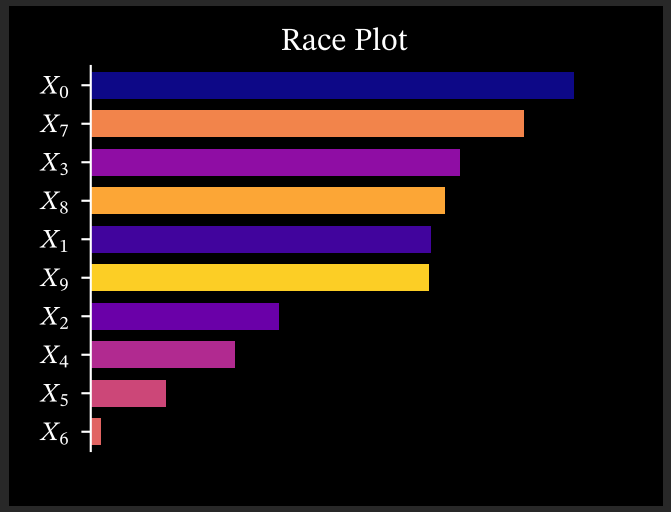
```python
import ipywidgets as ipw

write('''
    ## Interactive Apps with Widgets
    Use `ipywidgets`, `bqplot`,`ipyvo
     ::: note-tip
         Export to Slides/Report to se
    ''')
write([
    plot_html := ipw.HTML('Plot will
    button := ipw.Button(description=
    ], src)

def update_plot(btn):
    plot_html.value = race_plot().val
```

Python

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

Dynamic Content without Widgets

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



A Silly Plot

Python

```python
write('''
    ## Dynamic Content without Widget
    Use refresh button below to updat
    ''')

def display_plot(): return race_plot(

write(lambda: slides.on_refresh(displ
slides.source.from_callable(race_plot
```

Python

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

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

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

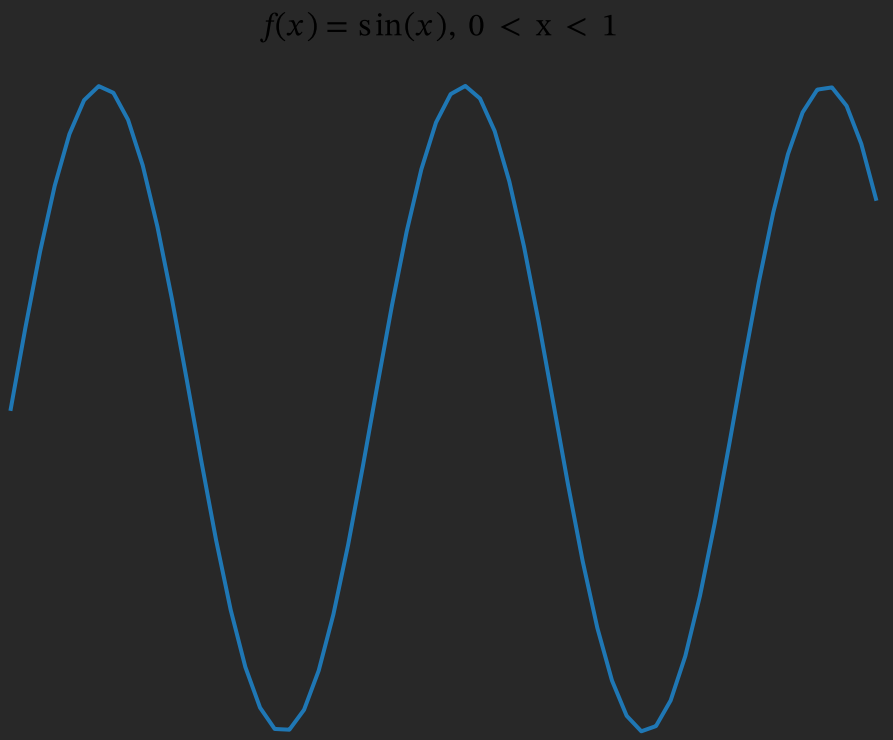
Contents

This is Slide 16.0

and we are animating matplotlib

Python

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



Python

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

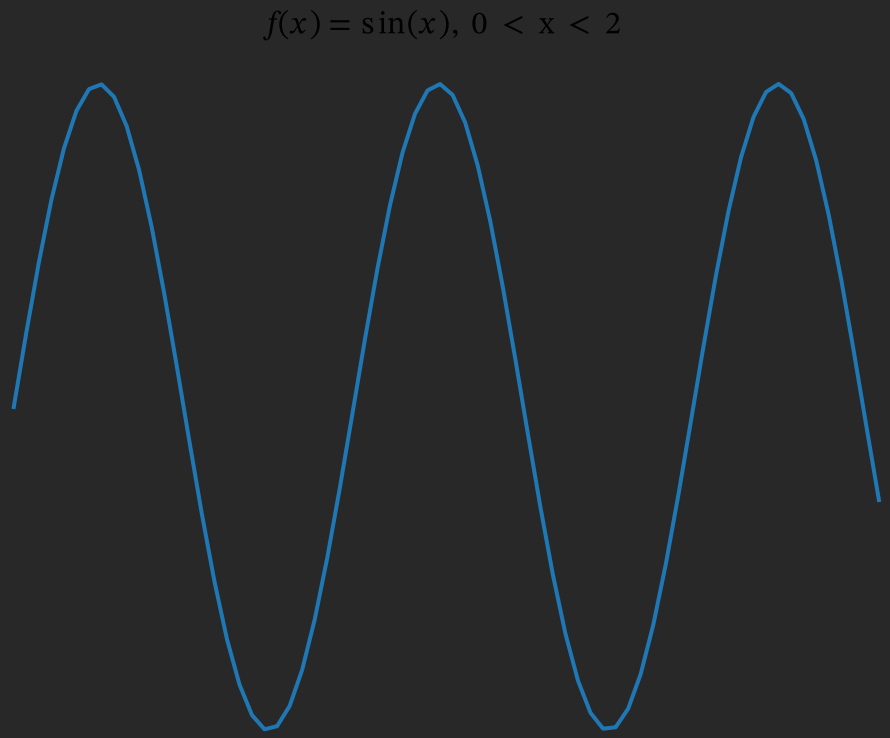
2

This is Slide 16.1

and we are animating matplotlib

Python

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



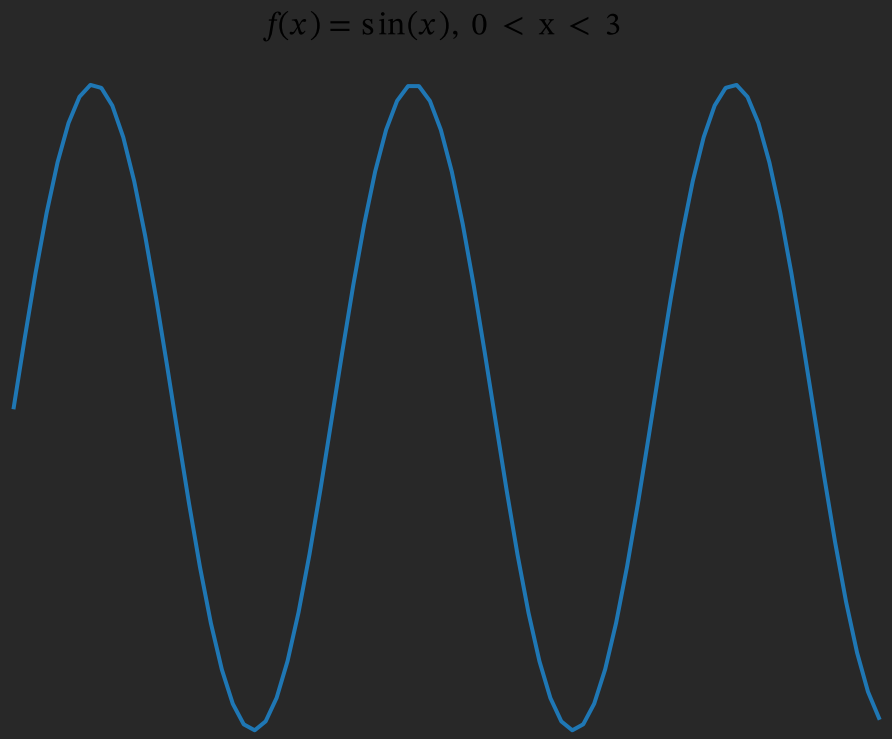
2

This is Slide 16.2

and we are animating matplotlib

Python

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



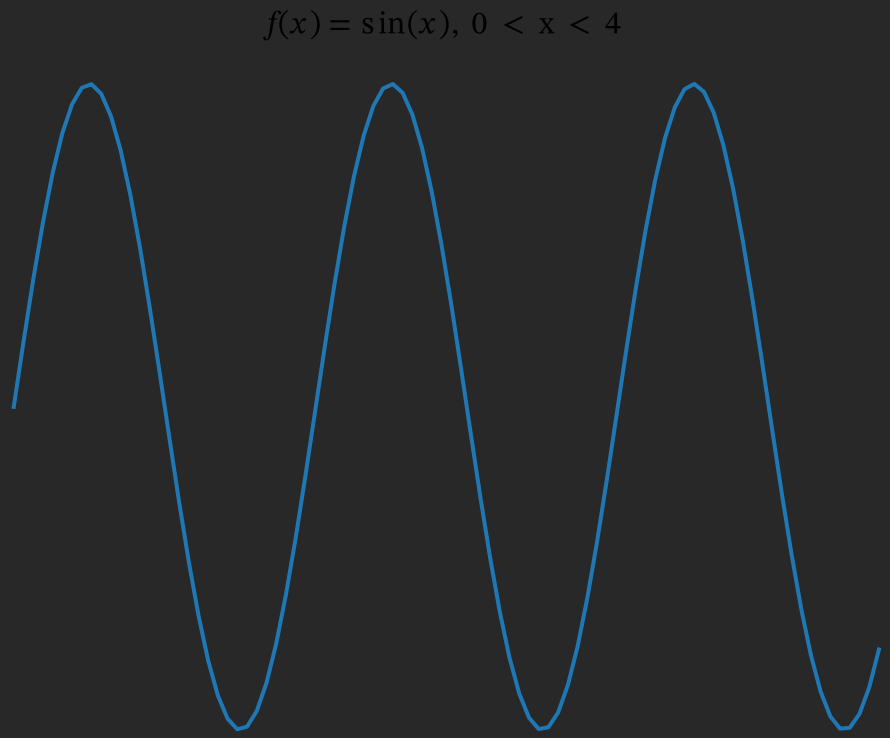
2

This is Slide 16.3

and we are animating matplotlib

Python

```
  + 3 more lines ...
ax.set_title(f'$f(x)=\sin(x)$
  + 2 more lines ...
```



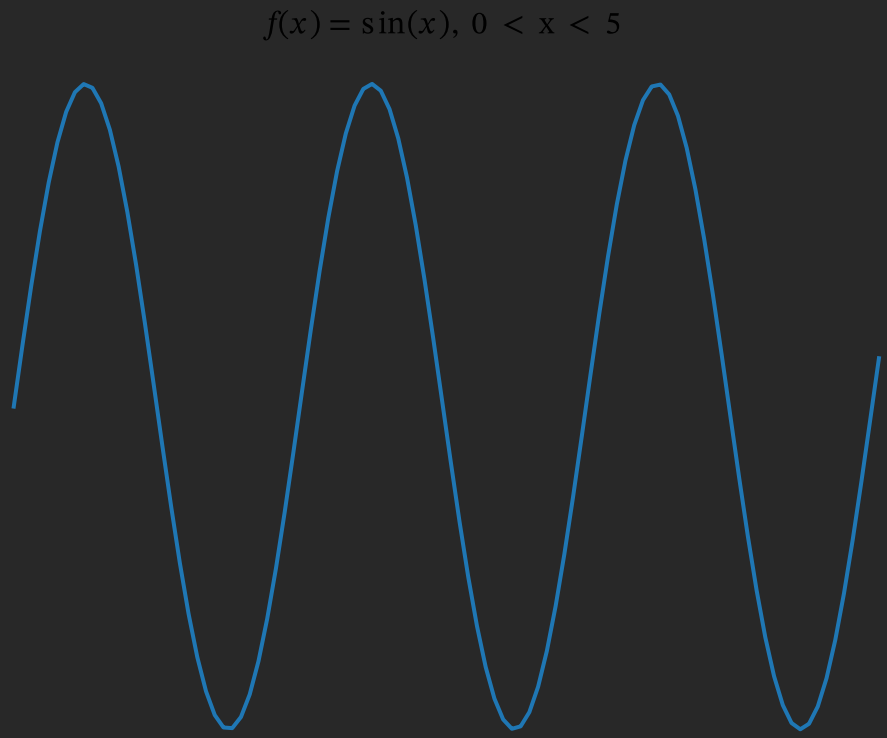
2

This is Slide 16.4

and we are animating matplotlib

Python

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



2

Contents

Frames with

`repeat = False`

```
1
```

Frames with

`repeat = False`

```
2
```

Frames with

`repeat = False`

```
3
```

Frames with

`repeat = False`

```
4
```

Frames with

`repeat = True` and Fancy Bullet List

- 1

Frames with

`repeat = True` and Fancy Bullet List

- 1
- 2

Frames with

`repeat = True` and Fancy Bullet List

- 1
- 2
- 3

Frames with

`repeat = True` and Fancy Bullet List

- 1
- 2
- 3
- 4

Frames with

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

1

2

Python

```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)
```

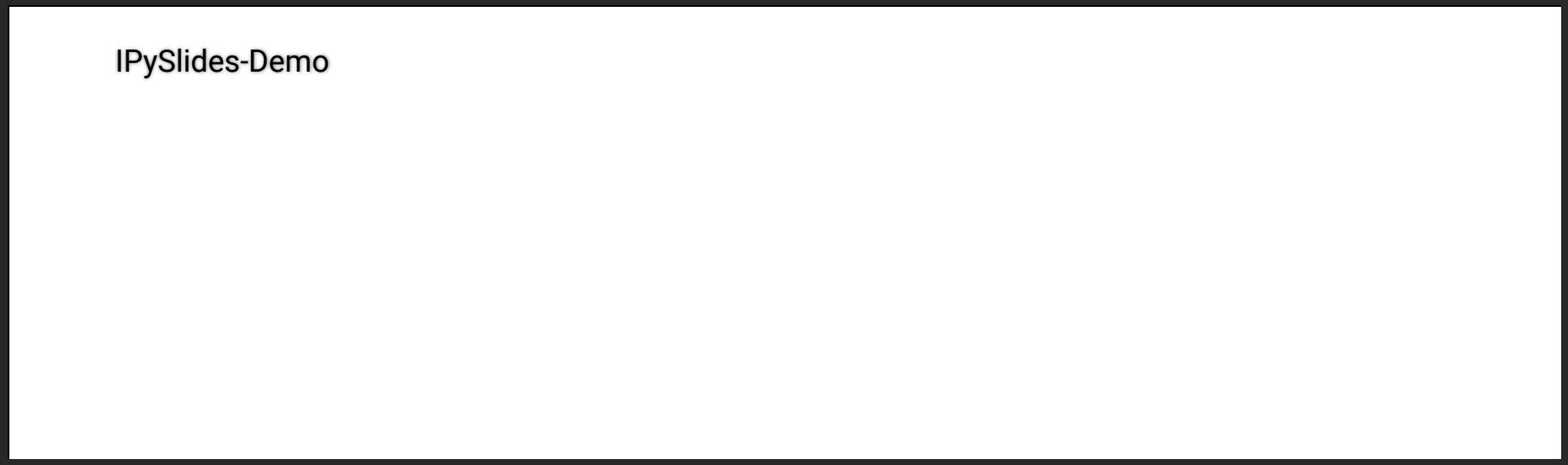
Displaying image from url from somewhere in Kashmir (کشمیر)

Python

```
slides.write('## Displaying image from url from somewhere in Kashmir color[crimson
try:
    slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d61fe/mast
except:
    slides.write('Could not retrieve image from url. Check internt connection!\n{.
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!



Python

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

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

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

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

Data Tables

Here is Table

h1	h2	h3
d1	d2	d3
r1	r2	r3

Python

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

$\LaTeX$ in Slides

⚠ Alert

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

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

Python

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

Built-in CSS styles

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

className	Formatting Style
'align-center'	------Text------
'align-left'	Text----------
'align-right'	----------Text
'rtl'	------ اردو عربی
'info'	Blue text. Icon ℹ️ for note-info class.
'tip'	Blue Text. Icon 💡 for note-tip class.
'warning'	Orange Text. Icon ⚠️ for note-warning class.
'success'	Green text. Icon ✅ for note-success class.
'error'	Red Text. Icon ⚡ for note-error class.
'note'	📝 Text with note icon.
'slides-only'	Text will not appear in exported html report.
'report-only'	Text will not appear on slides. Use to fill content in report.
'export-only'	Hidden on main slides, but will appear in exported slides/repor
'page-break'	Report will break page in print after object with this class.

Contents

Serialize Custom Objects to HTML

This is useful for displaying user defined/third party objects in slides

0 1 2 3 4 5 6 7 8 9

Python

```python
slides.write('## Serialize Custom Objects to HTML\nThis is useful for displaying u
with slides.suppress_stdout(): # suppress stdout from register fuction below
    @slides.serializer.register(int)
    def colorize(obj):
        color = 'red' if obj % 2 == 0 else 'green'
        return f'<span style="color:{color};">{obj}</span>'
    slides.write(*range(10))

some_slide.get_source().display()
```

This is all code to generate slides

Python

```python
def demo(self):
    "Demo slides with a variety of content."
    self.close_view() # Close any previous view to speed up loading 10x faster on
    self.clear() # Clear previous content

    with self.set_dir(os.path.split(__file__)[0]):
        file = '../_demo.py'
        raw_source = self.source.from_file(file).raw
        N = raw_source.count('auto.') + raw_source.count('\n---') + 1 # Count num
        self.create(*range(N)) # Create slides first, this is faster
        self.shell.run_line_magic('run', file) # Run demo in same namespace

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

e:\research\ipyslides\ipyslides_demo.py

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

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


slides = Slides() # It returns running slides instance or creates a new one
```

📝 Note

Slides keep their full code if they are not made by @frames decorator!

Source Code

Markdown: Slide 0

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

Markdown: Slide 1

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

Markdown: Slide 2

```
proxy`something will be here in start`
# Introduction
To see how commands work, use `Slides.docs()` to see the documentation.
Here we will focus on using all that functionality to create slides.
```python run source
# get the slides instance under a python block in Markdown file, we will use it la
myslides = get_slides_instance()
import ipyslides as isd
version = myslides.version
%xmd #### This is inline markdown parsed by magic { note  warning}
```
```

Reference via Markdown

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

Python

```
slides.write('citations`## Reference via Markdown\n———`')
bib_slide.get_source().display()
```