

Creating Slides

Abdul Saboor¹, Unknown Author²

Nov 25, 2025

¹My University is somewhere in the middle of nowhere

²Their University is somewhere in the middle of nowhere

Right click (or click on footer) to open context menu and click on ⓘ icon for instructions.

 Show Code

Contents

- 1. Introduction**
2. Adding informative TOC
3. Plotting and DataFrame
4. Simple Animations with Frames
5. Controlling Content on Frames
6. Custom Objects Serilaization
7. Code to Generate Slides

Introduction

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

Info

This slide was built purely from markdown, so you can create a variable `test` to overwite this →

Exception: Could not resolve '%{test}':

NameError: name 'test' is not defined

You can update this variable by '`Slides[int,|list|slice].vars.update`` or by defining it in notebook if 'Auto Rebuild' is enabled.

Version: 6.6.1

Contents

1. Introduction
2. Adding informative TOC
3. Plotting and DataFrame
4. Simple Animations with Frames
5. Controlling Content on Frames
6. Custom Objects Serilaization
7. Code to Generate Slides

This is summary for current section created using block syntax of toc. See `Slides.xmd.syntax` for details.

- Item 1
- Item 2

$$E = mc^2$$



Tip

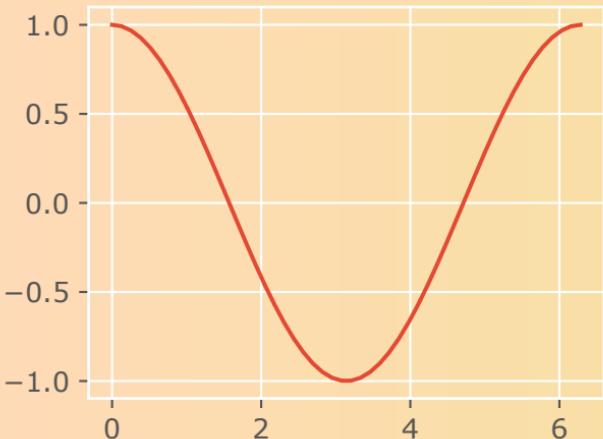
Above `btn` variable can be updated later via
`Slides[number,].vars.update` method.

Markdown

```
1 section`Adding informative TOC`  
2 ````multicols.block-blue  
3 toc[True]`### Contents`  
4 +++
```

Plotting with Matplotlib

Python



```
1 sl.set_css(bg1 = 'linear-gradient(to right, #FFDA  
2  
3 import numpy as np, matplotlib.pyplot as plt  
4 plt.rcParams['svg.fonttype'] = 'none' # Global se  
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     slides.write(ax, s.focus([0,3,4]))  
10    slides.write('Double click on the plot to focus on it!')
```

Double click on the plot to focus on it!

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-data/master/iris.csv')  
4     df.describe() #Small for display  
5 except:  
6     df = '### To see all `pandas` to view output!'
```

Writing Plotly Figure

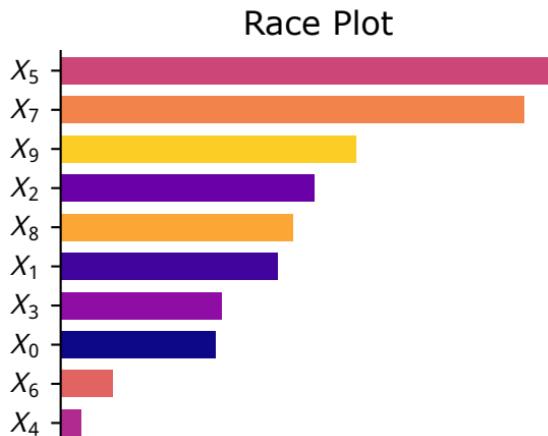
Install `plotly` to view output

Python

```
1 try:
2     import ipywidgets as ipw
3     import plotly.graph_objects as go
4
5     fig = slides.patched_plotly(go.FigureWidget()) # prefer Widget for interactivity and correct display
6     fig.add_trace(go.Bar(y=[1,5,8,9], customdata=["A", "B"]))
7
8     # We have clicked and selected traits on patched plotly
9     html = ipw.HTML()
10
11    def observe_click(change):
12        html.value = "<br/>".join(f" {k} = {v}" for k, v in change['new'].items())
13
14    fig.observe(observe_click, names='clicked')
15    box = ipw.HBox([fig, html])
16
17 except:
```

Refreshable Content

Use refresh button below to update plot! See `race_plot` function at end of slides.



A Silly Plot

Python

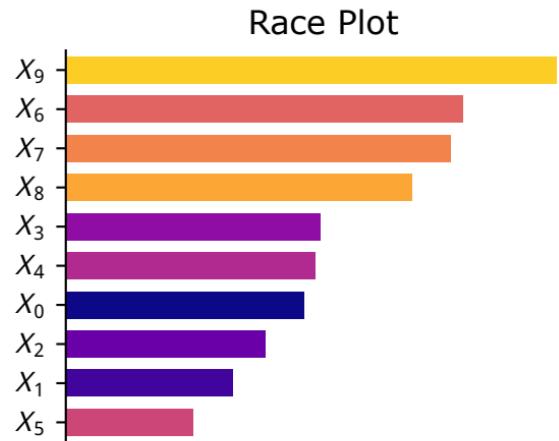
```
1  slides.write(''  
2      ## Refreshable Content  
3      Use refresh button below to update plot!  
4      See alert('race_plot' function at end of slide)  
5      '')  
6  
7  def display_plot(btn): return race_plot().display  
8  
9  slides.write(  
10     slides.dl.interactive(display_plot, btn = sl:  
11     rslide.get_source()  
12 ) # Only first columns will update
```

Refresh Plot

Animations with Widgets

anim 8

Animation Frame: 8

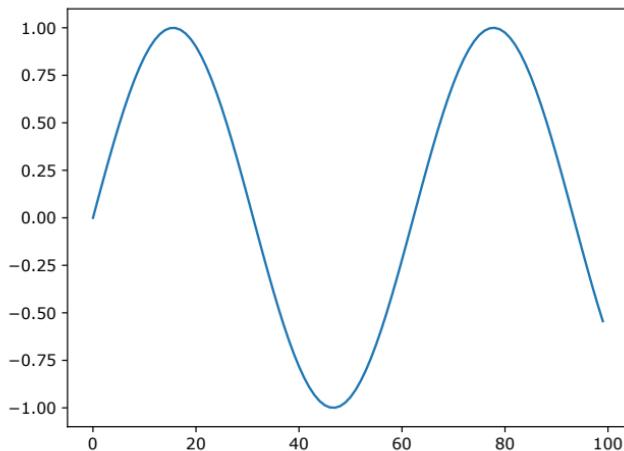


A Silly Plot

Rich Content ListWidget

Execute a code block

```
lambda: print(np.random.random((10,2))),  
  
lambda: plt.plot(np.random.random((10,2))),  
  
def plot_sine():  
    plt.plot(np.sin(np.linspace(0,10,100)))
```



Python

```
1 import numpy as np  
2 import matplotlib.pyplot as plt  
3  
4 def plot_sine():  
5     plt.plot(np.sin(np.linspace(0,10,100)))  
6  
7 lw = slides.ListWidget(description='Execute a cod  
8 options = [  
9     lambda: print(np.random.random((10,2))),  
10    lambda: plt.plot(np.random.random((10,2)),  
11    plot_sine,  
12    ], transform = lambda value: slides.code(val  
13 )  
14  
15 def run(c):  
16     if callable(c): c() # avoid None value when i  
17     plt.show()  
18  
19 css = {'.out-main': {'height':'300px'}, 'grid':'a  
20 it = slides.dl.interactive(run, c = lw, post_init
```

Contents

1. Introduction
2. Adding informative TOC
3. Plotting and DataFrame
- 4. Simple Animations with Frames**
5. Controlling Content on Frames
6. Custom Objects Serilaization
7. Code to Generate Slides

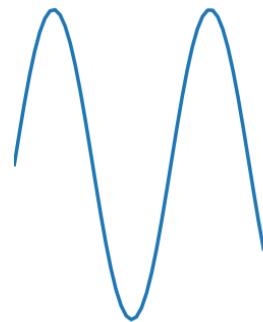
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1 for idx in slides.PAGE.iter(range(10,19)):  
2     fig, ax = plt.subplots(figsize=(3.4,2.6))  
3     x = np.linspace(0, idx, 50)  
4     ax.plot(x, np.sin(x))  
5     ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6     ax.set_xlim([0,18])  
7     ax.set_axis_off()  
8     slides.write(s.focus([idx - 10]), ax, widths=[60,40])  
9  
10    if idx == 10:  
11        slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 11$$



Unlike `interact/interactive`, this animation is based on slide frames, all of which are exported to HTML.

Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 12$$



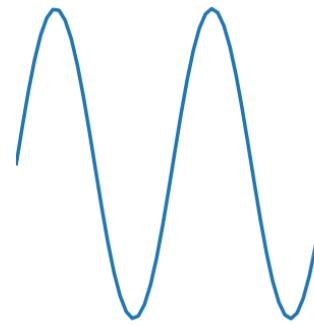
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 13$$



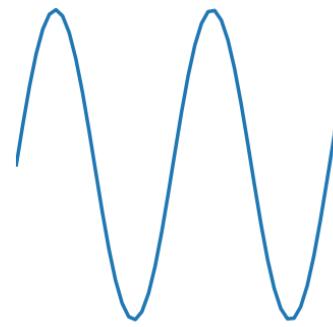
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 14$$



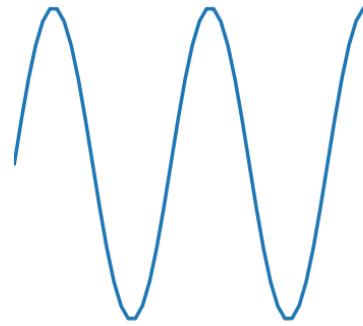
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 15$$



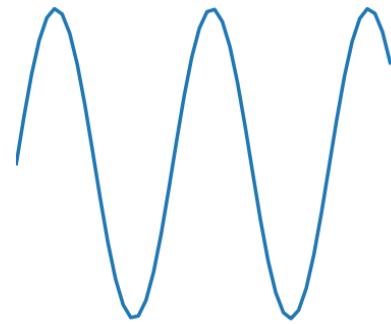
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 16$$



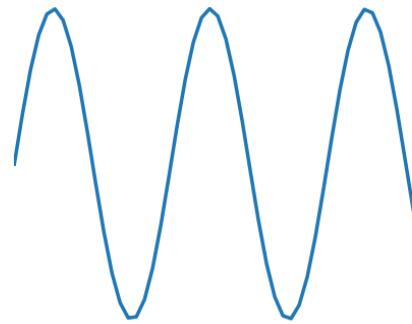
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 17$$



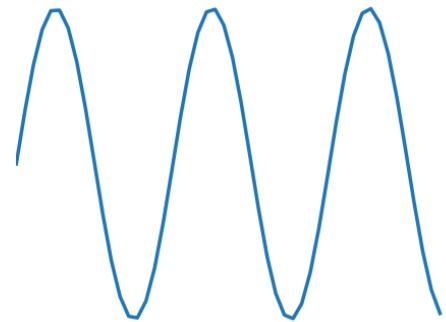
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 18$$



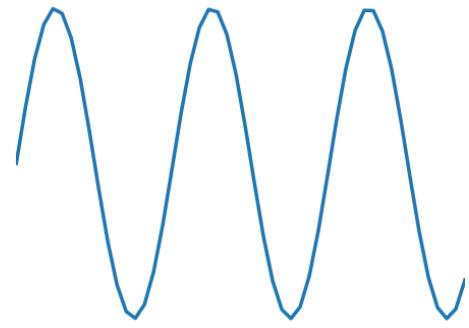
Animating Matplotlib!

→ Skip All Next Frames

Python

```
1  for idx in slides.PAGE.iter(range(10,19)):  
2      fig, ax = plt.subplots(figsize=(3.4,2.6))  
3      x = np.linspace(0,idx,50)  
4      ax.plot(x,np.sin(x))  
5      ax.set_title(rf'$f(x)=\sin(x)$, $0 < x < {idx+1}$')  
6      ax.set_xlim([0,18])  
7      ax.set_axis_off()  
8      slides.write(s.focus([idx - 10]),ax,widths=[60,40])  
9  
10     if idx == 10:  
11         slides.write('Unlike `interact/interactive`, this a
```

$$f(x) = \sin(x), 0 < x < 19$$



Contents

1. Introduction
2. Adding informative TOC
3. Plotting and DataFrame
4. Simple Animations with Frames
- 5. Controlling Content on Frames**
6. Custom Objects Serilaization
7. Code to Generate Slides

Default Frames

Python

```
1 slides.write('# Default Frames')
2 s.get_source().focus([2,3]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.write(item)
```

Default Frames

Python

```
1 slides.write('# Default Frames')
2 s.get_source().focus([2,3]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.write(item)
```

1

Default Frames

Python

```
1 slides.write('# Default Frames')
2 s.get_source().focus([2,3]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.write(item)
```

2

Default Frames

Python

```
1 slides.write('# Default Frames')
2 s.get_source().focus([2,3]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.write(item)
```

3

Default Frames

Python

```
1 slides.write('# Default Frames')
2 s.get_source().focus([2,3]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.write(item)
```

4

Frames with

PAGE.iter() and Fancy Bullet List

Python

```
1 slides.write('# Frames with \n#### code`PAGE.iter()` and Fancy Bullet List yoffset`0`')
2 s.get_source().focus([2,3,4]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.bullets([item], marker='❤').display()
```

Frames with

PAGE.iter() and Fancy Bullet List

Python

```
1 slides.write('# Frames with \n#### code`PAGE.iter()` and Fancy Bullet List yoffset`0`')
2 s.get_source().focus([2,3,4]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.bullets([item], marker='❤').display()
```



1

Frames with

PAGE.iter() and Fancy Bullet List

Python

```
1 slides.write('# Frames with \n#### code`PAGE.iter()` and Fancy Bullet List yoffset`0`')
2 s.get_source().focus([2,3,4]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.bullets([item], marker='❤').display()
```



2

Frames with

`PAGE.iter()` and Fancy Bullet List

Python

```
1 slides.write('# Frames with \n#### code`PAGE.iter()` and Fancy Bullet List yoffset`0`')
2 s.get_source().focus([2,3,4]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.bullets([item], marker='❤').display()
```



3

Frames with

PAGE.iter() and Fancy Bullet List

Python

```
1 slides.write('# Frames with \n#### code`PAGE.iter()` and Fancy Bullet List yoffset`0`')
2 s.get_source().focus([2,3,4]).display()
3 slides.PAGE() # want to show source alone first
4 for item in slides.PAGE.iter(boxes):
5     slides.bullets([item], marker='❤').display()
```



4

Frames with

PART.iter() and 2x2 grid of boxes

Python

```
1 slides.write('# Frames with \n#### code`PART.iter()` and 2x2 grid of boxes yoffset`0`')
2 s.get_source().focus(range(2,7)).display()
3 objs = [boxes[:2],boxes[2:]]
4 widths = [(1,3),(3,2)]
5 for ws, cols in slides.PART.iter(zip(widths,objs)):
6     slides.write(*cols, widths=ws)
```

Frames with

PART.iter() and 2x2 grid of boxes

Python

```
1 slides.write('# Frames with \n#### code`PART.iter()` and 2x2 grid of boxes yoffset`0`')
2 s.get_source().focus(range(2,7)).display()
3 objs = [boxes[:2],boxes[2:]]
4 widths = [(1,3),(3,2)]
5 for ws, cols in slides.PART.iter(zip(widths,objs)):
6     slides.write(*cols, widths=ws)
```

1

Frames with

PART.iter() and 2x2 grid of boxes

Python

```
1 slides.write('# Frames with \n#### code`PART.iter()` and 2x2 grid of boxes yoffset`0`')
2 s.get_source().focus(range(2,7)).display()
3 objs = [boxes[:2],boxes[2:]]
4 widths = [(1,3),(3,2)]
5 for ws, cols in slides.PART.iter(zip(widths,objs)):
6     slides.write(*cols, widths=ws)
```

1

2

Frames with

PART.**iter()** and 2x2 grid of boxes

Python

```
1 slides.write('# Frames with \n#### code`PART.iter()` and 2x2 grid of boxes yoffset`0`')  
2 s.get_source().focus(range(2,7)).display()  
3 objs = [boxes[:2],boxes[2:]]  
4 widths = [(1,3),(3,2)]  
5 for ws, cols in slides.PART.iter(zip(widths,objs)):  
6     slides.write(*cols, widths=ws)
```

1

2

3

Frames with

PART.**iter()** and 2x2 grid of boxes

Python

```
1 slides.write('# Frames with \n#### code`PART.iter()` and 2x2 grid of boxes yoffset`0`')
2 s.get_source().focus(range(2,7)).display()
3 objs = [boxes[:2],boxes[2:]]
4 widths = [(1,3),(3,2)]
5 for ws, cols in slides.PART.iter(zip(widths,objs)):
6     slides.write(*cols, widths=ws)
```

1

2

3

4

[← Skip Previous Frames](#)

Watching Youtube Video?

Want to do some drawing instead? Click on button on the right!



IPySlides-Demo



IPySlides | From a Hobby to a Sol...



[>Show Code](#)

Blocks with CSS classes

Table

h1	h2	h3
d1	d2	d3
r1	r2	r3

Widgets

0

Select to do nothing

A rich content table

h1	h2	h3
C	2	3
3	<code>import numpy as np</code>	5

[Show Code](#)

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.



Varibale formatting alongwith *LATEX* `%{var}` → 'I was a variable' is seamless.

*LAT*_EX in Slides

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

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

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.

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

$$ax^2 + bx + c = 0$$

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.

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

$$ax^2 + bx + c = 0$$

 Show Code

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 displaying user defined/third party')
2  with slides.suppress_stdout(): # suppress stdout from register function below
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     from .._demo import demo_slides
4     return demo_slides(self)
```

e:\development\ipyslides\ipyslides_demo.py

```
1 # Author: Abdul Saboor
2 # This demonstrates that you can generate slides from a .py file too, which you can import in notebook.
3
4 def demo_slides(slides):
5     slides.close_view() # Close any previous view to speed up loading 10x faster on average
6     slides.clear() # Clear previous content
7     raw_source = slides.code.cast(__file__).raw
8     N = raw_source.count('.build') + raw_source.count('\n---')
9     slides.create(range(N)) # Create slides first, this is faster
10
11    slides.settings.footer.text = slides.get_logo("1em") + 'Author: Abdul Saboor ' عبدالصبور'
12    slides.set_citations(r'''
13        @pf: This is reference to FigureWidget using alert`cite`\`pf\`` syntax
```



Source Code

Markdown: Slide 0

```
1  ````md-src.collapsed
2  # Creating Slides
3  :::: align-center width=50%
4      alert`Abdul Saboor``^1, Unknown Author``^2`
5      center`//today``//` 
6  :::: align-left text-box
7      ^`^1`My University is somewhere in the middle of nowhere
8      ^`^2`Their University is somewhere in the middle of nowhere
9
10 :::: display align-center
11     vspace`2`Right click (or click on footer) to open context menu and click on fa`info` icon f
12 ``
13 <md-src/>
```

Markdown: Slide 1

1 section`Introduction` toc`## Contents`

Markdown: Slide 2

1 # Introduction