Creating Slides

Abdul Saboor[1], Unknown Author[2]

Nov 20, 2025

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

Read instructions by clicking on

ⓘ

or same button in quick menu

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: `6.5.6`

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

$$E = mc^2$$

✎

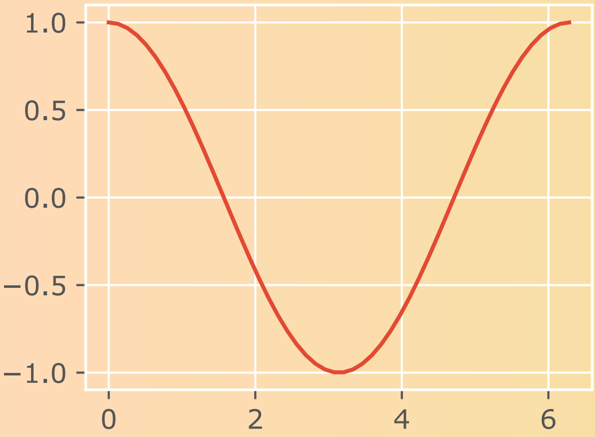
💡 Tip

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

Markdown

~~~
section`Adding informative TOC`
```multicol .block-blue
~~~

Plotting with Matplotlib



Python

```python
sl.set_css(bg1 = 'linear-gradient(to right, #FFDA

import numpy as np, matplotlib.pyplot as plt
plt.rcParams['svg.fonttype'] = 'none' # Global se
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))
slides.write(ax, s.focus([0,3,4]))
slides.write('Double click on the plot to focus o
```

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

```python
try:
    import pandas as pd
    df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
```

Writing Plotly Figure

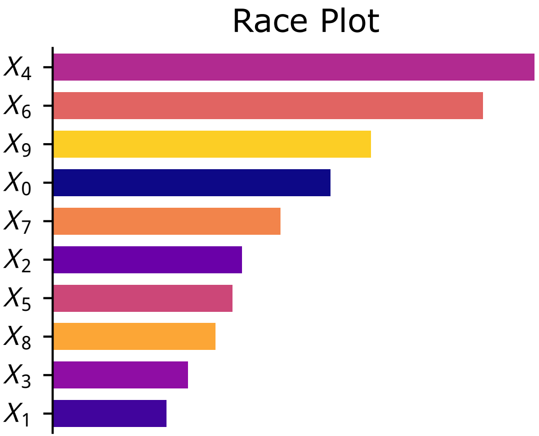
Install `plotly` to view output

Python

```python
try:
    import ipywidgets as ipw
    import plotly.graph_objects as go

    fig = slides.patched_plotly(go.FigureWidget()) # prefere Widget for interactivity and correct display
    fig.add_trace(go.Bar(y=[1,5,8,9], customdata=["A","B"]))

    # We have clicked and selected traits on patched plotly
    html = ipw.HTML()

    def observe_click(change):
        html.value = "<br/>".join(f"  {k} = {v}" for k, v in change['new'].items())

    fig.observe(observe_click, names='clicked')
```

Refreshable Content

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



A Silly Plot

Python

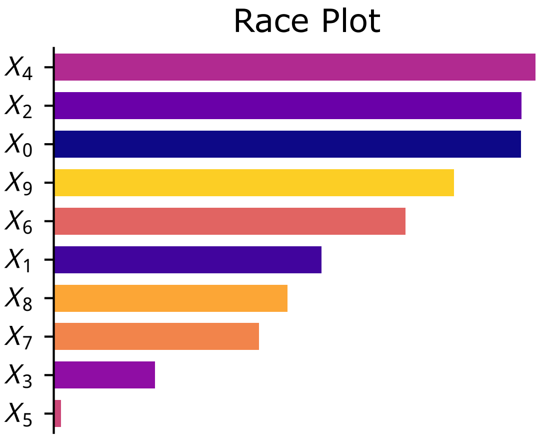
```python
slides.write('''
    ## Refreshable Content
    Use refresh button below to update plot!
    See alert`race_plot` function at end of slide
    ''')

def display_plot(btn): return race_plot().display

slides.write(
    slides.dl.interactive(display_plot, btn = sl
    rslide.get_source()
) # Only first columns will update
```

Refresh Plot

Animations with Widgets

anim ▶ ↺ 10

```
Animation Frame: 10
```



A Silly Plot

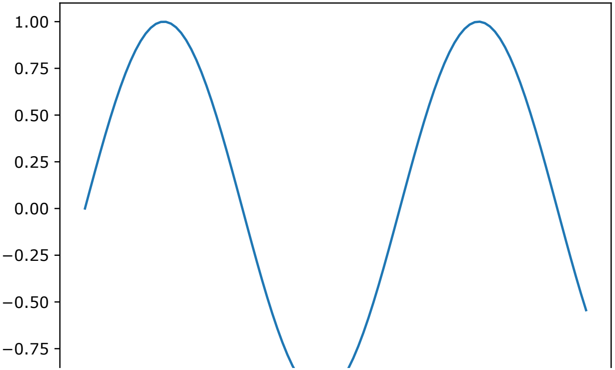
Rich Content `ListWidget`

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

```python
import numpy as np
import matplotlib.pyplot as plt

def plot_sine():
    plt.plot(np.sin(np.linspace(0,10,100)))

lw = slides.ListWidget(description='Execute a cod
    options = [
        lambda: print(np.random.random((10,2))),
        lambda: plt.plot(np.random.random((10,2)
        plot_sine,
    ], transform = lambda value: slides.code(val
)

def run(c):
    if callable(c): c() # avoid None value when 
    plt.show()

css = {'.out-main': {'height':'300px'}, 'grid':'a
```

Contents

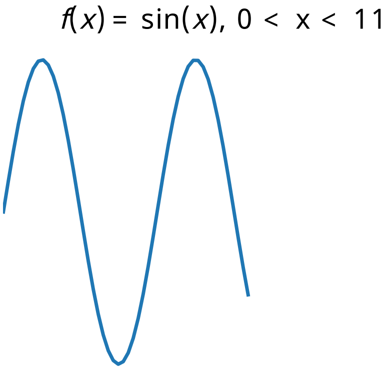
1. Introduction
2. Adding informative TOC
3. Plotting and DataFrame
4. **Simple Animations with Frames**
5. Controlling Content on Frames
6. Custom Objects Serilaization
7. Code to Generate Slides

Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```



💡 Tip

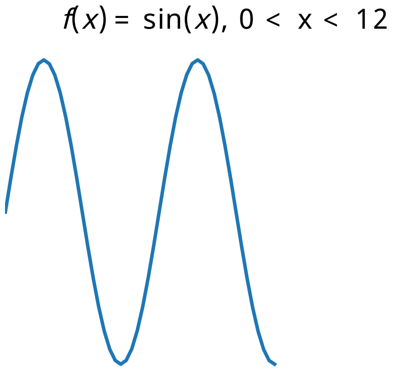
Unlike `interact/interactive` , this animation is based on slide frames, all of which are exported to HTML.

Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```



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

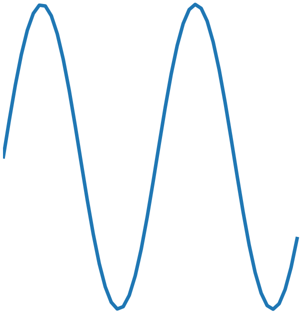
Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```

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

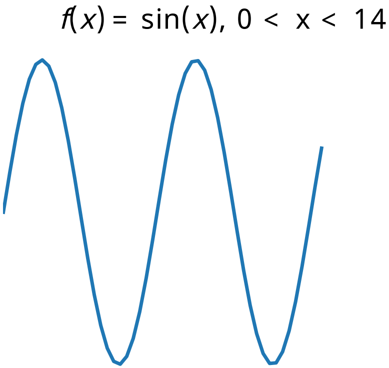


Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```

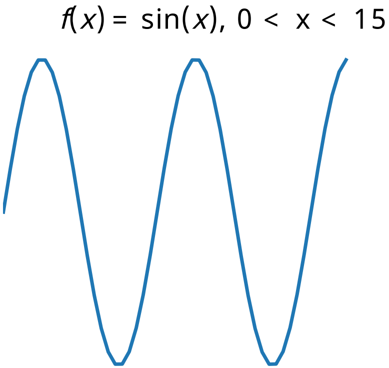


Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```



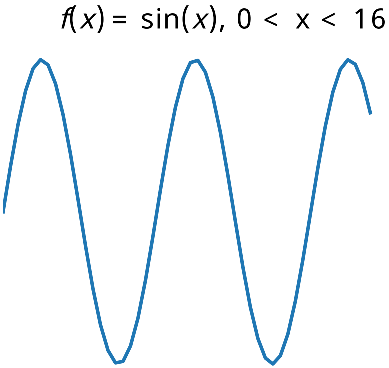
$f(x) = \sin(x)$, 0 < x < 15

Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```

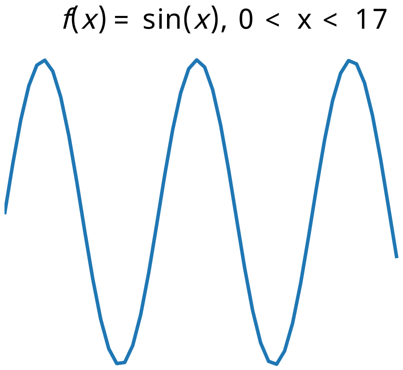


Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```

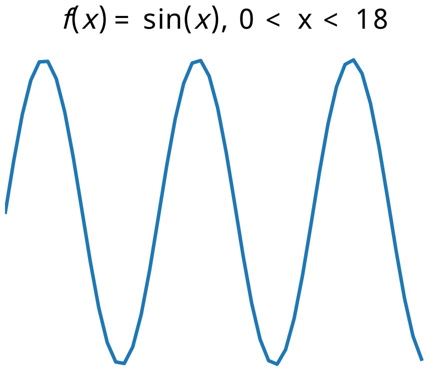


Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```

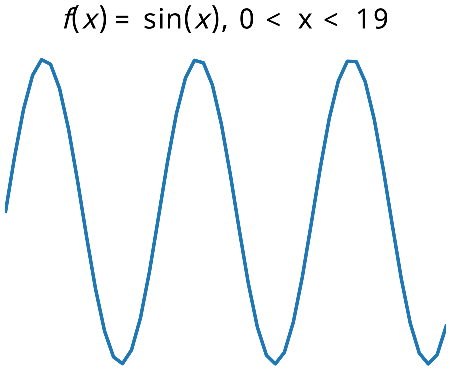


Animating Matplotlib!

→ Skip All Next Frames

Python

```python
for idx in slides.PAGE.iter(range(10,19)):
    fig, ax = plt.subplots(figsize=(3.4,2.6))
    x = np.linspace(0,idx,50)
    ax.plot(x,np.sin(x))
    ax.set_title(rf'$f(x)=\sin(x)$, 0 < x < {idx+1}')
    ax.set_xlim([0,18])
    ax.set_axis_off()
    slides.write(s.focus([idx - 10]),ax,widths=[60,40])

    if idx == 10:
        slides.write('Unlike `interact/interactive`, this a
```



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

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

Default Frames

Python

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

1

Default Frames

Python

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

2

Default Frames

Python

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

3

Default Frames

Python

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

4

Frames with

`PAGE.iter()` and Fancy Bullet List

Python

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

Frames with

`PAGE.iter()` and Fancy Bullet List

Python

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

💘 1

Frames with

`PAGE.iter()` and Fancy Bullet List

Python

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

💘 2

Frames with

`PAGE.iter()` and Fancy Bullet List

Python

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

- 💘 3

Frames with

`PAGE.iter()` and Fancy Bullet List

Python

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

💘 4

Frames with

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

Python

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

Frames with

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

Python

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

1

Frames with

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

Python

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

1	2

Frames with

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

Python

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

1

2

3

Frames with

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

Python

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

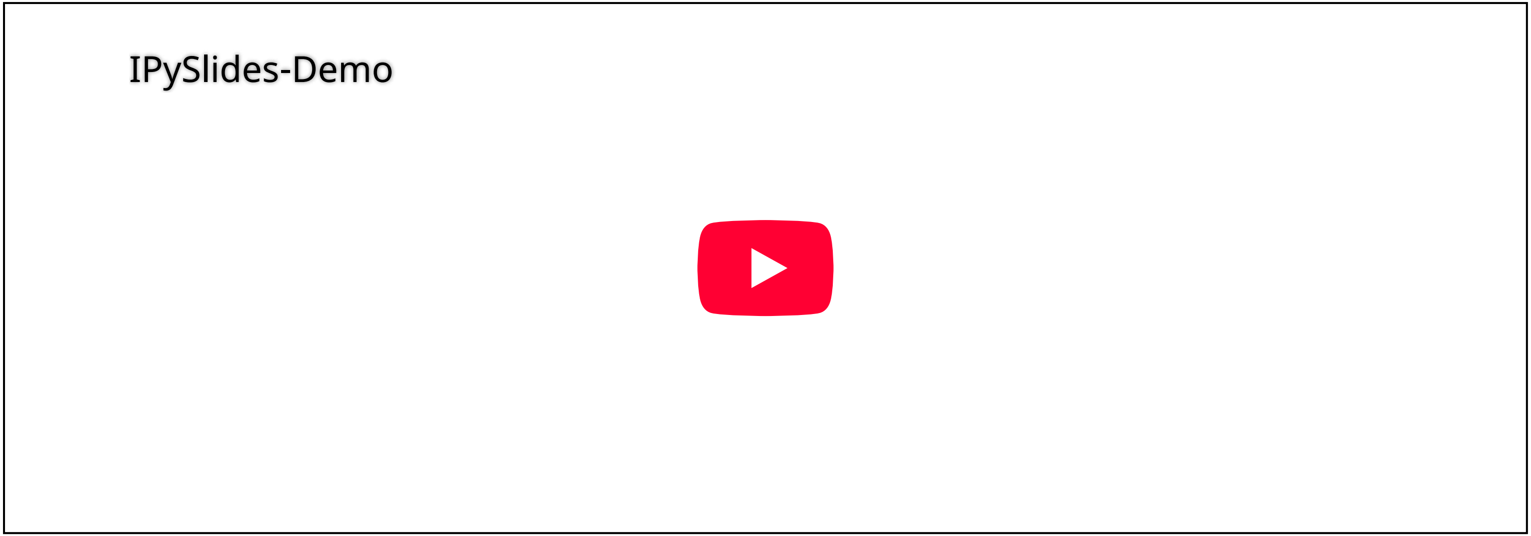
1 2

3 4

Skip Previous Frames

Watching Youtube Video?

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



Blocks with CSS classes

Table

h1	h2	h3
d1	d2	d3
r1	r2	r3

A rich content table

h1	h2	h3
	2	3
3	`import numpy as np`	5

Widgets

● 0

Click to do nothing

□ Select to do nothing

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.

> 💡 Tip
>
> Varibale formatting alongwith $\LaTeX$ %{var} → 'I was a variable' is seamless.

$\LaTeX$ 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.

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

$\LaTeX$ 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

```python
slides.write('## Serialize Custom Objects to HTML\nThis is useful for displaying user defined/third party
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."
    from .._demo import demo_slides
    return demo_slides(self)
```

e:\development\ipyslides\ipyslides_demo.py

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

def demo_slides(slides):
    slides.close_view() # Close any previous view to speed up loading 10x faster on average
    slides.clear() # Clear previous content
    raw_source = slides.code.cast(__file__).raw
    N = raw_source.count('.build') + raw_source.count('\n---')
    slides.create(range(N)) # Create slides first, this is faster

```

Source Code

Markdown: Slide 0

~~~
```md-src.collapsed
# Creating Slides
::: align-center width=50%
    alert`Abdul Saboor`^`1`, Unknown Author^`2`
    center`//today``//`
    ::: align-left text-box
        ^`1`My University is somewhere in the middle of nowhere
        ^`2`Their University is somewhere in the middle of nowhere

::: display align-center
    vspace`2`Read instructions by clicking on %{btn} or same button in quick menu
```
<md-src/>
~~~

Markdown: Slide 1

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