



Creating Slides

Abdul Saboor¹, Unknown Author²

Mar 09, 2024

¹My University is somewhere in the middle of nowhere

²Their University is somewhere in the middle of nowhere

Read instructions by clicking **i** button in quick menu.



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: 3.7.2 as executed from below code in markdown.

Python

```
1 # get the slides instance under a python block in Markdown file, we will use it later to run a cell magic
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 [!\[\]\(633dd45d48d71eb51a85c6dd83ee51e9_img.jpg\) launch binder](#)
- Watch a Youtube Video

1. Add references like this per slide or 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

Markdown

```
1  ```toc Table of contents
2  Extra content for current section which is on right
3  ```
```

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

- Item 1
- Item 2

$$E = mc^2$$



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 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 ipywidgtes right here using textbox command

or libraries based on ipywidgtes such as bqplot,ipyvolume,plotly's FigureWidget¹(reference at end) can be included as well.

1. This is reference to FigureWidget using `cite`pf`` syntax



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.code` API.
- Use `Slides.alt(widget, func)` function to display widget on slides and alternative content in exported slides, function should return possible HTML representation of widget.
- `ipywidgets.HTML` and its subclasses will be displayed as `Slides.alt(widget, html_converter_func)`. The value of exported HTML will be most recent.
- 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:



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



Use ipyslides.Output instead of ipywidgets.Output

Python

```
1 import ipywidgets as ipw
2
3 with ipw.Output():
4     write('''
5         I was not supposed to be shown, but alert`ipywidgets.Output` could not
6         capture me and I went to main capture stream!''')
7
8 with (out := slides.Output()): # can by ipyslides.Output too
9     print('I am safely captured here and will only be displayed if user wants to!')
10
11 write([
12     '**User wanted to write me below!**',
13     slides.alt(out, lambda w: 'HTML export for Output widget is not supported!')])
```

I was not supposed to be shown, but `ipywidgets.Output` could not capture me and I went to main capture stream!

User wanted to write me below!

HTML export for Output widget is not supported!



Plotting with Matplotlib



Python

```
1 sl.set_css({'background':'linear-gradient(to right, #f0f0f0, #e0e0e0)}')
2
3 import numpy as np, matplotlib.pyplot as plt
4 plt.rcParams['svg.fonttype'] = 'none' # Global setting
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([0,3,4]))
```



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-data/master/iris.csv')  
4     df.describe() #Small for display  
5 except:  
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'
```

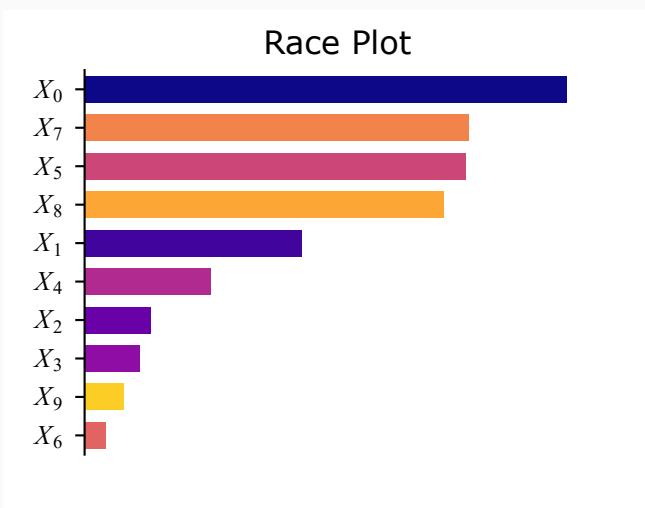
Interactive Apps with Widgets

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



Tip

Export to Slides to see what happens to this slide and next slide!



A Silly Plot

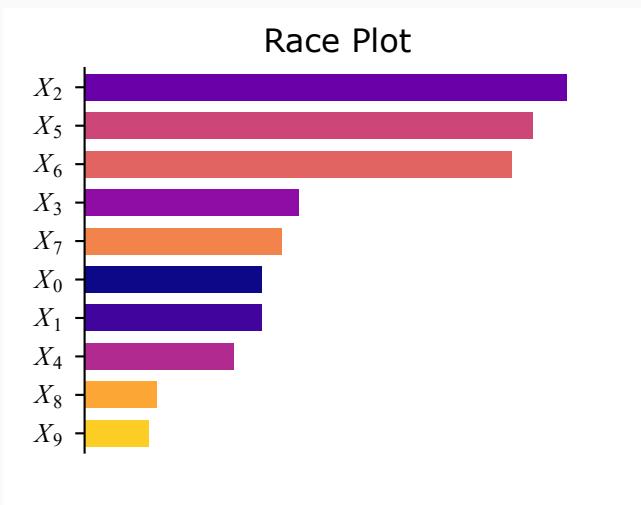
Python

```
1 import ipywidgets as ipw
2
3 write('''
4     ## Interactive Apps with Widgets section`Interactive Apps with Widgets` section
5     Use `ipywidgets`, `bqplot`, `ipyvolume`, `plotly` and `ipyvolume` sections
6     ::: note-tip
7         Export to Slides to see what happens to the sections
8 ''')
9 plot_html = ipw.HTML('Plot will be here')
10 button = ipw.Button(description='Click me to update the plot')
11
12 write([plot_html, button], src)
13
14 def update_plot(btn):
```



Dynamic Content without Widgets

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



Python

```
1 write('''
2     ## Dynamic Content without Widgets
3     Use refresh button below to update plot! Comp
4 ''')
5
6 def display_plot(): return race_plot().display()
7
8 write(lambda: slides.on_refresh(display_plot), rsl
9 slides.code.cast(race_plot).display()
```

Python

```
1 def race_plot():
2     import numpy as np
3     import matplotlib.pyplot as plt
4
```



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

[Skip All Next Frames]



This is Slide 17.0

and we are animating matplotlib

Python

```
1 fig, ax = plt.subplots()  
2 + 5 more lines ...
```

1

-
1. I was cited for no reason

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





This is Slide 17.1

and we are animating matplotlib

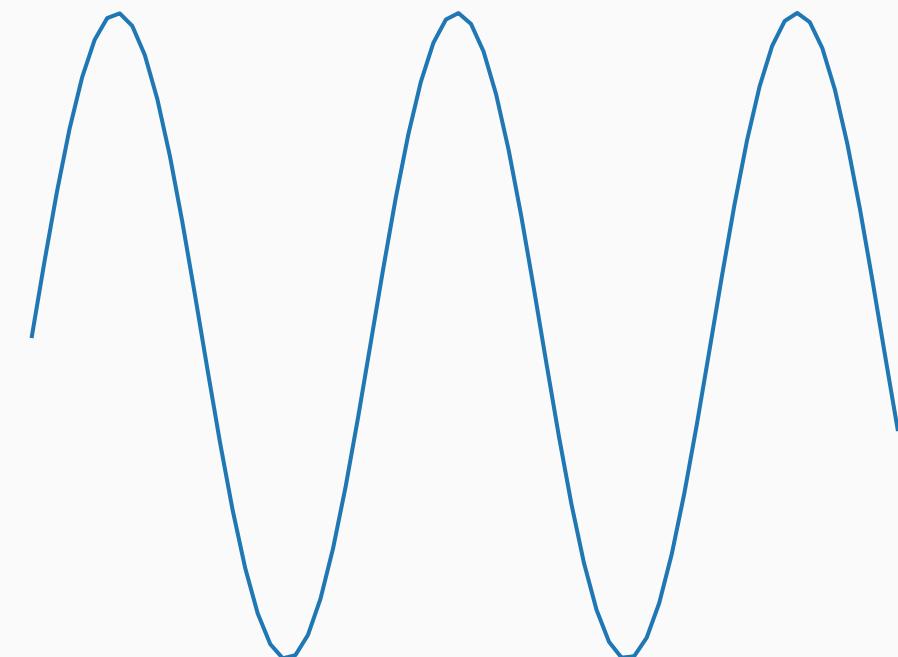
Python

```
1 + 1 more lines ...
2 x = np.linspace(0,obj+1,50+10*(idx+1))
3 + 4 more lines ...
```

1

1. I was cited for no reason

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





This is Slide 17.2

and we are animating matplotlib

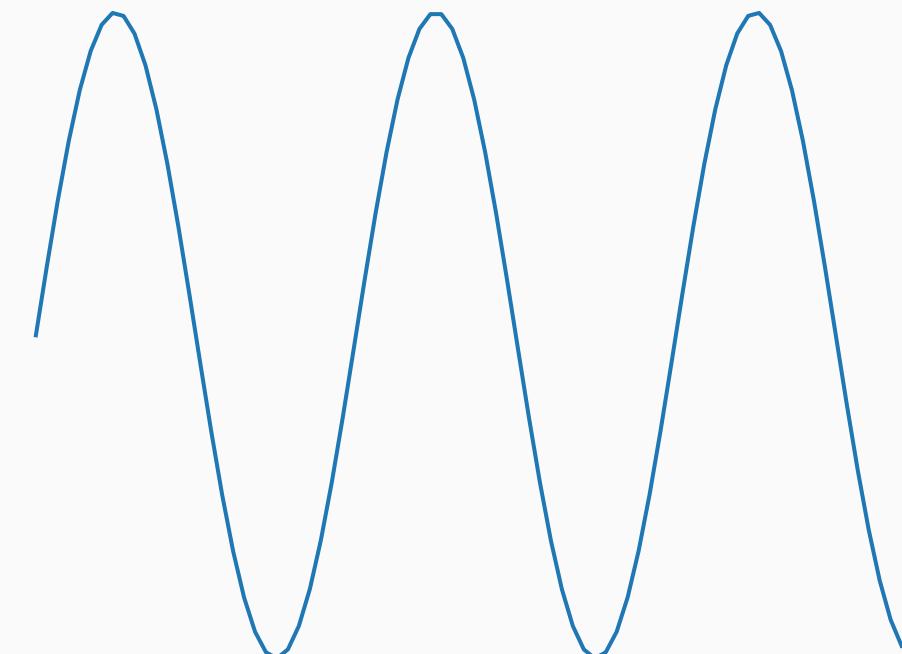
Python

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

1

1. I was cited for no reason

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





This is Slide 17.3

and we are animating matplotlib

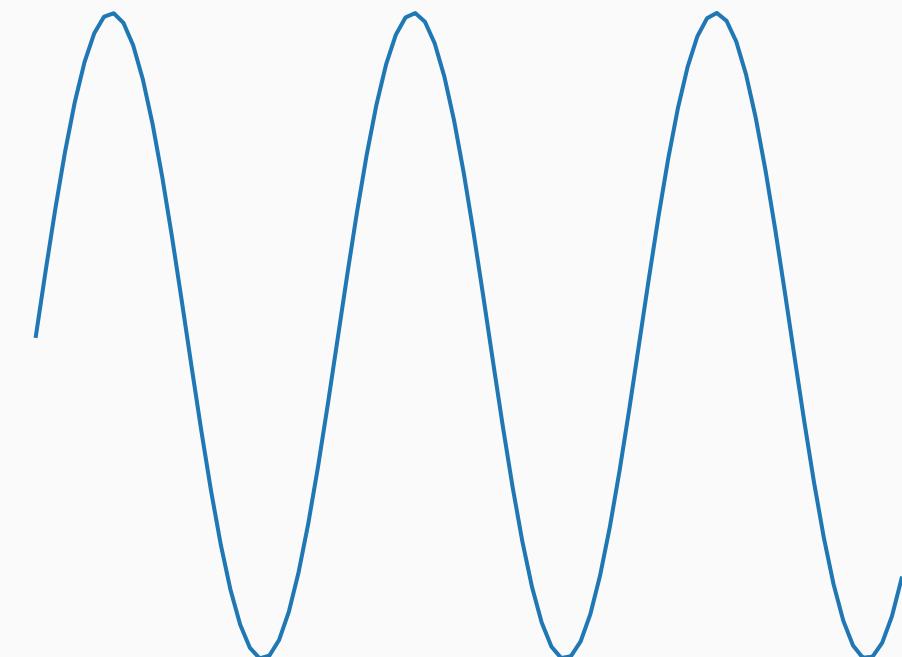
Python

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

1

1. I was cited for no reason

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





This is Slide 17.4

and we are animating matplotlib

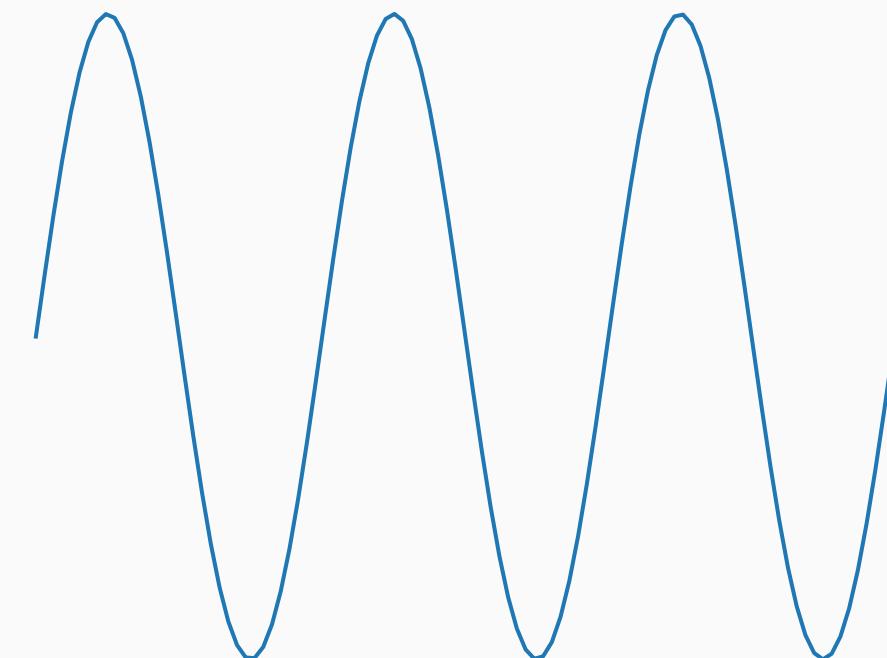
Python

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

1

1. I was cited for no reason

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





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



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)





Watching Youtube Video?

Want to do some drawing instead? Click on pencil icon 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 and draw something on [tldraw](https://
3
4 https://tldraw.app) video player below")
5
```



Block API

New block API is as robust as write command. On top of it, it makes single unit of related content.

Table

```
h1 h2 h3  
d1 d2 d3  
r1 r2 r3
```

Widgets



Click to do nothing

Python

```
1 write('## Block API\nNew `block` API is as robust as `write` command. On top of it, it makes single unit  
2 slides.block_red()  
3 [  
4     '### Table',  
5     ''',  
6     '|h1|h2|h3|',  
7     '|---|---|---|',  
8     '|d1|d2|d3|',  
9     '|r1|r2|r3|',  
10    ''',  
11    1,
```



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}`* →

KeyError: 'var' not found in given namespace. You may need to enclose input text in `Slides fmt` function!

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.



Tip

Varibale formatting alongwith *LATEX* `{'var}` →

KeyError: 'var' not found in given namespace. You may need to enclose input text in Slides(fmt function!

is seamless.

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



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.



Tip

Variable formatting alongwith *LATEX* `{'var}` →

KeyError: 'var' not found in given namespace. You may need to enclose input text in Slides_FMT function!

is seamless.

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

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



```
1 slides.write(fmt('## Built-in CSS styles\n`{slides.css_styles}`'))
```

Built-in CSS styles

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

className	Formatting Style
'text-[value]'	[value] should be one of tiny, small, big, large, huge.
'align-[value]'	[value] should be one of center, left, right. _____ اردو عربی
'rtl'	
'info'	Blue text. Icon i 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.
'export-only'	Hidden on main slides, but will appear in exported slides.
'jupyter-only'	Hidden on exported slides, but will appear on main slides.
'block'	Block of text/objects
'block-[color]'	Block of text/objects with specific background color from red, green, blue, yellow, cyan, magenta and gray.
'raw-text'	Text will not be formatted and will be shown as it is.
'zoom-self'	Zooms object on hover, when Zoom is enabled.
'zoom-child'	Zooms child object on hover, when Zoom is enabled.
'no-zoom'	Disables zoom on object when it is child of 'zoom-child'.



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 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 from ipyslides.formatters import libraries, supported_reprs  
5 from ipyslides.xmd import fmt  
6  
7 def demo_slides(slides):  
8     slides.close_view() # Close any previous view to speed up loading 10x faster on average  
9     slides.clear() # Clear previous content  
10    raw_source = slides.code.cast(__file__).raw  
11    N = raw_source.count('auto.') + raw_source.count('\n---') + 1 # Count number of slides, +1 for run_cell  
12    slides.create(*range(N)) # Create slides first, this is faster  
13
```



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`1, Unknown Authorsup`2
4     center`today```
5     :::: text-box
6         sup`1My University is somewhere in the middle of nowhere
7         sup`2Their University is somewhere in the middle of nowhere
8
9 Read instructions by clicking i button in quick menu.
```

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.
```