▶ 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

```python
import ipyslides as isd
ls = isd.LiveSlides()
ls.set_animation(main='zoom')
```

Python

```python
%%title
# create a rich content title page
```

Python

```python
%%slide 1
# slide 1 content
```

Python

```python
%%slide 1 -m # new in 1.4.6
**Markdown here with extended options (see `ls.xmd_syntax` for info). Nested blocks are not supported**
 ```multicol 30 70 .Success
 less content
 +++
 more content
 ```
```

Python

```python
%%slide 2 -s # new in 1.7.7
var = some_long_computation() # This will run only first time or when cell code chnages.
```

```
Use `pprint` or `LiveSlides.capture_std` contextmanager
to see print output on slide in desired order!
--------------------------------------------------------------------------------
3
```

There is a python block above with header python run source. We can display that block by {{source}} as below:

Python

```python
x = 1 + 2
print(x) # will be printed on slide in somehwewere top as print appears first of all.
# Use `with ls.capture_std() as std: print();std.stdout` to see at exactly where it is printed.
```

variable x defined there is shown here x = 3. Only variables can be embeded in {{var}}, not expressions.

Python

```python
@ls.frames(1,*objs)
def func(obj):
```

```
    write(obj) #This will create as many slides after the slide number 1 as length(objs)
```

Python

```
ls # This displays slides if on the last line of cell, or use `ls.show()`.
```

- Use ls.from_markdown to create multiple slides from markdown file/text.
  - Slides are added in order of content.
  - Slides should be separated by --- (three dashes) in start of line.

Python

```
with open(2) as f:
    ls.from_markdown(2, f.read())
# Add other things to same file
```

- Use ls.demo to create example slides and start editing. Follow steps in first part.
- Use ls.docs to see upto date documentation.
- You can acess markdown content of an existing slide using ls[key or index].markdown if it has been created using ls.from_markdown or %%slide i -m.
- You can insert content usign with ls[key or index].insert(index) or ls[key or index].insert_markdown (1.7.7+).

New in 1.7.2

- Find special syntax to be used in markdown by LiveSlides.xmd_syntax.
- You can now show citations on bottom of each slide by setting citation_mode = 'footnote' in LiveSlides constructor.
- You can now access individual slides by indexing s_i = ls[i] where i is the slide index or by key as s_3_1 = ls['3.1'] will give you slide which shows 3.1 at bottom.
- Basides indexing, you can access current displayed slide by ls.current.
- You can add new content to existing slides by using with s_i.insert(where) context. All new changes can be reverted by s_i.reset().
- If a display is not complete, e.g. some widget missing on a slide, you can use (ls.current, ls[index], ls[key]).update_display() to update display.
- You can set overall animation by ls.set_overall_animation or per slide by s_i.set_animation
- You can now set CSS for each slide by s_i.set_css or ls.set_slide_css at current slide.

New in 1.7.5
Use LiveSlides.extender to add markdown extensions. Also look at PyMdown-Extensions.

New in 1.7.7
Use slides[i].insert_markdown({'index': 'markdown_string',...}) to insert markdown (pasrsed objects) at indices.
%%slide i -s can be used to execute code just once in current session. It will run again if code changes in cell.

👈 **Read more instructions in left panel**

Slide 1

> **This is inline markdown parsed by magic**

Version: 1.8.3 as executed from below code in markdown.

Python

```python
import ipyslides as isd
version = isd.__version__
%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

```python
slides.core.globals.extract #s1 was assigned as `s0, s1, s2 = slides.from_markdown...` in start.
    '#### 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.'
    f'I am {    .    "Alerted"} and I am *{    .    "colored and italic
text" "magenta" "whitesmoke"}*'
```

▶ Show Code

Slide 2

Created using %%slide 2 -m with markdown only

[1] Refrence 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

IPySlides Online Running Sources

ⓘ Launch as voila slides (may not work as expected [1]) 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

```
slides.write(obj)
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 inwrite 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 explictly 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 a function/class/module (without calling) and it will be displayed as a pretty printed code block.
- Give a registered object using @LiveSlides.serializer.registor decorator.

If an object is not in above listed things, obj.__repr__() will be printed. If you need to show other than **repr**, use display(obj) outside write command or use methods specific to that library to show in jupyter notebook.

If you give a className, add CSS of it using format_css function and provide it to write function. Get a list of already available classes using slides.css_styles. For these you dont need to provide CSS.

Note: Use keep_format method to bypass markdown parser, for example keep_format(altair_chart.to_html()). Note: You can give your own type of data provided that it is converted to an HTML string. Note: _repr_<format>_ takes precedence to to_<format> methods. So in case you need specific output, use object.to_<format>.

LiveSlides.**iwrite**(*columns, width_percents=None, className=None)

Each obj in columns could be an IPython widget like ipywidgets,bqplots etc or list/tuple (or wrapped in rows function) of widgets to display as rows in a column. Other objects (those in write command) will be converted to HTML widgets if possible. Object containing javascript code may not work, use write command for that.

If you give a className, add CSS of it using format_css function and provide it to iwrite function. Get a list of already available classes using slides.css_styles. For these you dont need to provide CSS.

Returns: writer, columns as reference to use later and update. rows are packed in columns.

Examples:

```
writer, x = iwrite('X') # writer = iwrite('X'); x = writer.cols[0] # gives same result
writer, (x,y) = iwrite('X','Y')
writer, (x,y) = iwrite(['X','Y']) # One column with two rows
writer, (x,y),z = iwrite(['X','Y'],'Z')
#We unpacked such a way that we can replace objects with new one using `grid.update`
new_obj = writer.update(x, 'First column, first row with new data') #You can update same `new_obj` with it's own widget methods.
```

LiveSlides.**parse_xmd**(extended_markdown, display_inline=True, rich_outputs=False)

Parse extended markdown and display immediately. If you need output html, use display_inline = False but that won't execute python code blocks. Precedence of content return/display is rich_outputs = True > display_inline = True > parsed_html_string.

Example

````
```python run var_name
#If no var_name, code will be executed without assigning it to any variable
import numpy as np
```
# Normal Markdown {.report-only}
```multicol 40 60
# First column is 40% width
If 40 60 was not given, all columns will be of equal width, this paragraph will be inside info block due to class at bottom
{.Info}
+++
# Second column is 60% wide
This {{var_name}} is code from above and will be substituted with the value of var_name
```

```python
# This will not be executed, only shown
````
````

Each block can have class names (speparated with space or .) (in 1.4.7+) after all other options such as python .friendly or multicol .Sucess.Info. For example, python .friendly will be highlighted with friendly theme from pygments. Pygments themes, however, are not supported with multicol. You need to write and display CSS for a custom class. Aynthing with class name 'report-only' will not be displayed on slides, but appears in document when LiveSlides.export. <export_function> is called.

Note: Nested blocks are not supported.

This function was added in 1.4.6

New in 1.7.2
Find special syntax to be used in markdown by LiveSlides.xmd_syntax.

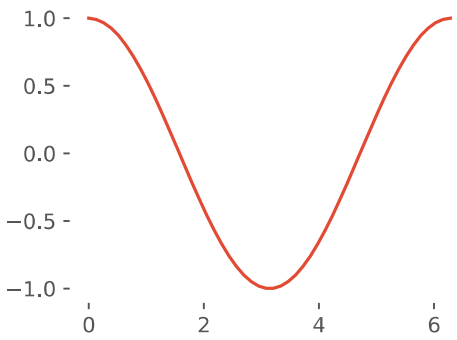
New in 1.7.5 Use LiveSlides.extender or ipyslides.extended_md.extender to add markdown extensions.

If an object does not render as you want, use display(object) or register it as you want using @LiveSlides.serializer.register decorator

Python

```python
write([slides.doc(write,'LiveSlides'), slides.doc(iwrite,'LiveSlides'),
 slides.doc(slides.parse_xmd,'LiveSlides')])
write("#### If an object does not render as you want, use `display(object)` or register it as you want using `@LiveSlides.serializer.register` decorator")
```

Plotting with Matplotlib



Python

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

Watching Youtube Video?

Python

```
write(f"### Watching Youtube Video?")
write(YouTubeVideo('Z3iR55lKgpI',width='100%',height='266px'))
@slides.notify_later()
def push():
  t = time.localtime()
  return f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm_min:02}'

s.display() # s = source.context(style='vs', className="Youtube")
```

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/>',
  textwrap.dedent('''
 |h1|h2|h3|
 |---|---|---|
 |d1|d2|d3|
 |r1|r2|r3|
 ''')))
s.focus_lines([3,4,5,6]).display()
```

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

Writing Altair Chart

ⓘ May not work everywhere, needs javascript

Python

```
+ 5 more lines ...
  chart = alt.Chart(df,width=300,height=260).mark_circle(size=60).encode(
    x='sepal_length',
    y='sepal_width',
    color='species',
    size = 'petal_width',
    tooltip=['species', 'sepal_length', 'sepal_width','petal_width','petal_length']
    ).interactive()
+ 5 more lines ...
```

Writing Plotly Figure

Interactive Apps on Slide

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

HBox(children=(VBox(children=('

Plot will be here! Click button below to activate it!

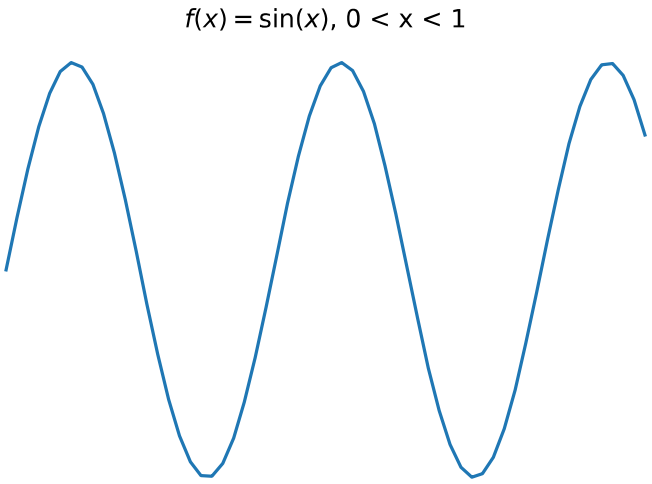
', Button(descrip...

This is Slide 14.1

and we are animating matplotlib

Python

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



Python

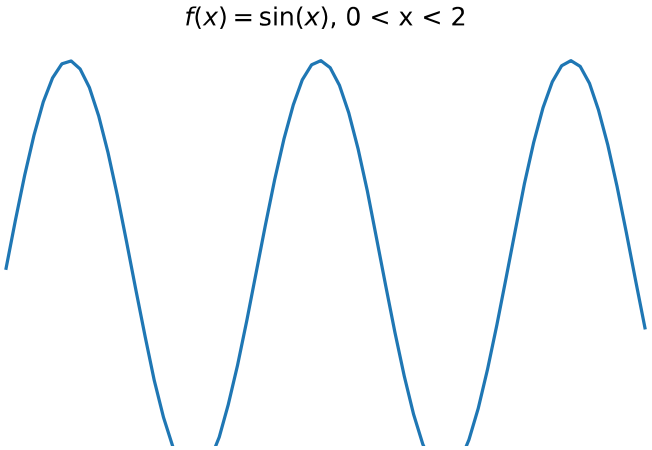
```
+ 5 more lines ...
slides.notes.insert(f'## This is under @frames decorator, so it will be shown only in first frame')
slides.notify_later()(lambda: f'This is under @frames decorator, so it will be shown only in first frame')
```

This is Slide 14.2

and we are animating matplotlib

Python

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

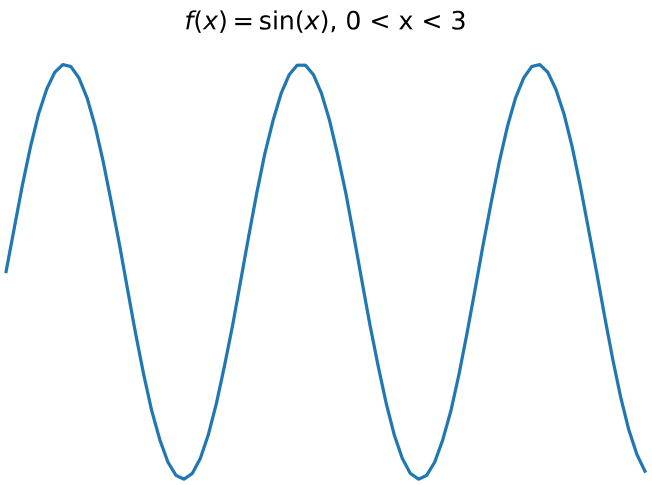


This is Slide 14.3

and we are animating matplotlib

Python

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

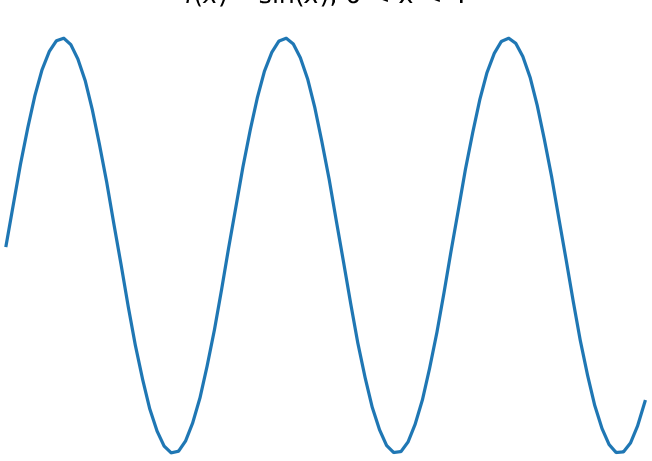


This is Slide 14.4

and we are animating matplotlib

Python

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

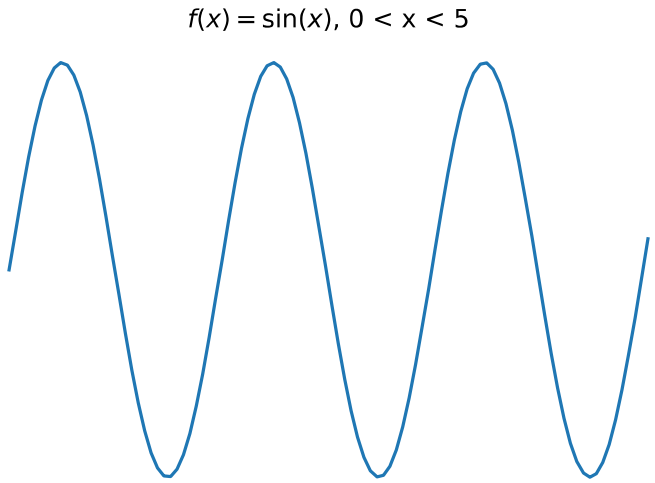


This is Slide 14.5

and we are animating matplotlib

Python

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



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

1

2

3

4

Frames with

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

1

2

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)]`')
```

Displaying image from url from somewhere in Kashmir (کشمیر)

Python

```
slides.write(['## 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'	------ اردو عربي
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.
className = 'Block'	Block of text/objects
className = 'Block-[color]'	Block of text/objects with specific background color from red, green, blue, yellow, cya

Info

```
slides.css_styles.display()
slides.write('Info',className='Info')
slides.write('Warning',className='Warning')
slides.write('سارے جہاں میں دھوم ہماری زبان کی ہے۔',className='Right RTL')
```

Can skip write commnad sometimes

Column A

Column B

Column C

Column D

Python

```
slides.rows(
  '## Can skip `write` commnad sometimes',
  slides.cols('### Column A','### Column B',className='Info'),
  '||### Column C {.Warning}||### Column D {.Success}||',
).display()
slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d61fe/master/pass/top-image%20(1).jpg').display()
```

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.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))
```

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 average
    self.clear() # Clear previous content

    import runpy
    file = os.path.join(os.path.dirname(os.path.dirname(__file__)), '_demo.py') # Relative path to this file
    slides = runpy.run_path(file, init_globals={'slides': self})['slides']

    N = len(slides)
    with slides.slide(N + 1):
      slides.write('## This is all code to generate slides')
      slides.write(self.demo)
      slides.source.from_file(file).display()

    with slides.slide(N + 2):
      slides.write('Slides made by using `from_markdown` or `%%slide` magic preserve their full code\n{.Note .Info}')
      slides.get_source().display()

    with slides.slide(N + 3, props_dict = {'': dict(background='#9ACD32')}):
      with slides.source.context() as s:
        slides.write_citations()
      s.display()

    # Just for func, set theme of all even slides to be fancy, and zoom animation
    fancy_even_slides_css = {
      '--heading-fg': '#105599',
        '--accent-color': '#955200',
      '--pointer-color': '#FF7722'
    }

    for s in slides[::2]:
```

```
    slides._slideindex = 0 # Go to title
    return slides
```

e:\research\ipyslides\ipyslides_demo.py

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

from ipyslides.writers import write, iwrite
from ipyslides.formatter import libraries, __reprs__
from ipyslides._base.intro import how_to_slide, logo_svg

slides = globals()['slides'] # gloabals are update from calling function demo()
slides.settings.set_footer('Author: Abdul Saboor عبدالصبور')
slides.settings.set_logo(logo_svg,width=60) # This is by defualt a logo of ipyslides
slides._citation_mode = 'global' # This could be changed by other functions
slides.set_citations({'pf': 'This is refernce to FigureWidget using `slides.cite` command'})

#Demo for loading slides from a file or text block
s0, s1, s2 = slides.from_markdown(0,'\n'.join(how_to_slide) + '''\n---
# Slide 1 {.Success}
```python run source
import ipyslides as isd
version = isd.__version__
%xmd #### This is inline markdown parsed by magic {.Note .Warning}
```
Version: {{version}} as executed from below code in markdown.
{{source}}

Slide 2 {.Success}
Created using `%%slide 2 -m` with markdown only
[slide2]:`This is reference created using markdown`
cite`slide2` Refrence to this will show at end
```multicol
# Column A
||### Sub column A {.Success}||### Sub column B ||
+++
# Column B
```
notes`This is note created using markdown`
That version from last slide is still in memory. See it is there {{version}}
''',trusted=True)

with s0.insert(0):
  s0.source.display(collapsed = True)

with slides.slide(1): #slide 1 will be modified with old and new content
  with slides.source.context(style='monokai', color='white', className='Mono') as s:
    slides.parse_xmd(s1.markdown) #s1 was assigned as `s0, s1, s2 = slides.from_markdown...` in start.
    write('#### 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.')
    write(f'I am {slides.alert("Alerted")} and I am *{slides.colored("colored and italic
```

```
slides.shell.user_ns['write'] = write #Inject variable in IPython shell

# Insert source of slide 2
with s2.insert(0):
  s2.source.display(collapsed = True)

s2.insert_markdown({-1: f'alert`I was added at end using {slides.backtick}s2.insert_markdown{slides.backtick}`'})

#slide 3
online_sources = '''# IPySlides Online Running Sources
Launch as voila slides (may not work as expected [^1])[![Binder](https://mybinder.org/badge_logo.svg)](https://mybinder.org/v2/gh/massgh/ipyslides-voila/HEAD?urlpath=voila%2Frender%2Fnotebooks%2Fipyslides.ipynb)
{.Note .Error}

[Edit on Kaggle](https://www.kaggle.com/massgh/ipyslides)\n{.Note .Warning}

Launch example Notebook [![Binder](https://mybinder.org/badge_logo.svg)](https://mybinder.org/v2/gh/massgh/ipyslides-voila/HEAD?urlpath=lab%2Ftree%2Fnotebooks%2Fipyslides.ipynb)
{.Note .Success}

[^1]: Add references like this per slide. Use slides.cite() or in markdown cite\`key\` to add citations generally.
'''
@slides.frames(3,'## I am created using `@slides.frames`',online_sources)
def func(obj):
  with slides.source.context() as s:
    slides.write(obj)
    slides.notify_later()(lambda: 'That is a notification which shows you can use decorator this way as well')
  s.display()

#Now generate many slides in a loop
__contents = [f"""## IPython Display Objects
#### Any object with following methods could be in`write` command:
{', '.join([f'`_repr_{rep}_`' for rep in __reprs__])}
Such as `IPython.display.<HTML,SVG,Markdown,Code>` etc. or third party such as `plotly.graph_objects.Figure`{{.Warning}}.
""",
f"""## Plots and Other **Data**{{style='color:var(--accent-color);'}} Types
#### These objects are implemented to be writable in `write` command:
{', '.join([f"`{lib['name']}.{lib['obj']}`" for lib in libraries])}
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`.
""",
f"""## Interactive Widgets
### Any object in `ipywidgets`{textbox('<a href="https://ipywidgets.readthedocs.io/en/latest/">Link to ipywidgtes right here using textbox command</a>')}
or libraries based on ipywidgtes such as `bqplot`,`ipyvolume`,plotly's `FigureWidget`{slides.cite('pf')} (reference at end)
can be included in `iwrite` command as well as other objects that can be passed to `write` with caveat of Javascript.
{{.Warning}}
```

```
  with slides.slide(i, props_dict = {'':dict(background = 'skyblue')}):
    write(__contents[i-4])
    if i == 7:
      with slides.source.context() as s:
        write([slides.doc(write,'LiveSlides'), slides.doc(iwrite,'LiveSlides'),
slides.doc(slides.parse_xmd,'LiveSlides')])
        write("#### If an object does not render as you want, use `display(object)` or register it as you
want using `@LiveSlides.serializer.register` decorator")

      s.show_lines([0,1]).display()

# Matplotlib
with slides.slide(8,props_dict = {'': dict(background='linear-gradient(to right, #FFDAB9 0%, #F0E68C
100%)')}):
  write('## Plotting with Matplotlib')
  with slides.source.context() as s:
    import numpy as np, matplotlib.pyplot as plt
    plt.rcParams['svg.fonttype'] = 'none' # Global setting, enforce same fonts as presentation
    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])])


# Youtube
from IPython.display import YouTubeVideo
with slides.slide(9):
  with slides.source.context(style='vs',className="Youtube") as s:
    write(f"### Watching Youtube Video?")
    write(YouTubeVideo('Z3iR551KgpI',width='100%',height='266px'))
    @slides.notify_later()
    def push():
      t = time.localtime()
      return f'You are watching Youtube at Time-{t.tm_hour:02}:{t.tm_min:02}'

    s.display() # s = source.context(style='vs', className="Youtube")

# Data Table
slides.shell.user_ns['slides'] = slides #Inject variable in IPython shell to use in below cell magic
slides.shell.run_cell_magic('slide','10',"""import textwrap
with slides.source.context() as s:
 write('## Data Tables')
 write(slides.block_r('Here is Table','<hr/>',
 textwrap.dedent('''
 |h1|h2|h3|
 |---|---|---|
 |d1|d2|d3|
 |r1|r2|r3|
 '''')))
 s.focus_lines([3,4,5,6]).display()
""")

# Plotly and Pandas DataFrame only show if you have installed
```

```
      import altair as alt
      alt.themes.enable('dark')
      df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
      chart = alt.Chart(df,width=300,height=260).mark_circle(size=60).encode(
        x='sepal_length',
        y='sepal_width',
        color='species',
        size='petal_width',
        tooltip=['species', 'sepal_length', 'sepal_width','petal_width','petal_length']
        ).interactive()
      df = df.describe() #Small for display
    except:
      df = '### Install `pandas` to view output'
      chart = '### Install Altair to see chart'
    write(('## Writing Pandas DataFrame',df),
      ('## Writing Altair Chart\nMay not work everywhere, needs javascript\n{.Note .Warning}',chart)
      )
  slides.source.current.show_lines(range(5,12)).display() #Show source code of above block even without assignning to variable explicitly

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'
with slides.slide(12):
  write(('## Writing Plotly Figure',fig))

# Interactive widgets can't be used in write command, but still they are displayed.

with slides.slide(13):
  with slides.source.context() as src:
    import ipywidgets as ipw
    import numpy as np, matplotlib.pyplot as plt

    write('## Interactive Apps on Slide\n Use `ipywidgets`, `bqplot`,`ipyvolume` , `plotly Figurewidget` etc.  to show live apps like this!')
    writer, (plot,button, _), code = slides.iwrite([
      '## Plot will be here! Click button below to activate it!',
      ipw.Button(description='Click me to update race plot',layout=ipw.Layout(width='max-content')),
      "[Check out this app](https://massgh.github.io/pivotpy/Widgets.html#VasprunApp)"],src.focus_lines([4,5,6,7,*range(24,30)]))

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

      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]))

      for s in ['right','top','bottom']:
        ax.spines[s].set_visible(False)
```

```
        global plot # Only need if you want to change something on it inside function
        plot = writer.update(plot, fig) #Update plot each time

    def onclick(btn):
        plot_theme = 'dark_background' if 'Dark' in slides.settings.theme_dd.value else 'default'
        with plt.style.context(plot_theme):
            update_plot()

    button.on_click(onclick)
    update_plot() #Initialize plot

  slides.notes.insert('## Something to hide from viewers!')


# Animat plot in slides
@slides.frames(14,*range(14,19))
def func(obj):
  with slides.source.context() as s:
    fig, ax = plt.subplots()
    x = np.linspace(0,obj+1,50+10*(obj-13))
    ax.plot(x,np.sin(x));
    ax.set_title(f'$f(x)=\sin(x)$, 0 < x < {obj-13}')
    ax.set_axis_off()
    slides.notes.insert(f'## This is under @frames decorator, so it will be shown only in first frame')
    slides.notify_later()(lambda: f'This is under @frames decorator, so it will be shown only in first frame')

  slides.write([f'### This is Slide {14}.{obj-13}\n and we are animating matplotlib',
          s.show_lines([obj-14])
          ],ax,width_percents=[40,60])
  if obj == 14:
    s.show_lines([5,6]).display()

# Frames structure
boxes = [f'<div style="background:var(--tr-hover-bg);width:auto;height:auto;padding:8px;margin:8px;border-radius:4px;"><h1>{i}</h1></div>' for i in range(1,5)]
@slides.frames(15,*boxes, repeat=False)
def f(obj):
  slides.write('# Frames with \n#### `repeat = False`')
  slides.write(obj)

@slides.frames(16,*boxes, repeat=True)
def f(obj):
  slides.write('# Frames with \n#### `repeat = True`')
  slides.write(*obj,className='Warning')

@slides.frames(17,*boxes, repeat=[(0,1),(2,3)])
def f(obj):
  with slides.source.context() as s:
    slides.write('# Frames with \n#### `repeat = [(0,1),(2,3)]`')
    slides.write(*obj)
  s.display()

with slides.slide(18):
  with slides.source.context() as s:
```

```
  s.display()

slides.from_markdown(19, '''## $\LaTeX$ 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.
{.Note .Warning}

\$\$\int_0^1\\frac{1}{1-x^2}dx\$\$
$$\int_0^1\\frac{1}{1-x^2}dx$$
''', trusted=True)

with slides.slide(20):
  slides.write('## Built-in CSS styles')
  with slides.source.context() as s:
    slides.css_styles.display()
    slides.write('Info',className='Info')
    slides.write('Warning',className='Warning')
    slides.write('سارے جہاں میں دھوم ہماری زباں کی ہے۔',className='Right RTL')
  s.display()

with slides.slide(21):
  with slides.source.context() as s:
    slides.rows(
      '## Can skip `write` commnad sometimes',
      slides.cols('### Column A','### Column B',className='Info'),
      '||### Column C {.Warning}||### Column D {.Success}||',
    ).display()
    slides.image(r'https://assets.gqindia.com/photos/616d2712c93aeaf2a32d61fe/master/pass/top-image%20(1).jpg').display()
  s.display()

with slides.slide(22):
  slides.write('## Serialize Custom Objects to HTML\nThis is useful for displaying user defined/third party objects in slides')
  with slides.capture_std():
    with slides.source.context() as s:
      @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))

    s.display()
```

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

Source Code

Markdown: Slide 0

```
# Creating Slides
**Assuming you have `ls = ipyslides.LiveSlides()`**
```

````
  - `%%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
import ipyslides as isd
ls = isd.LiveSlides()
ls.set_animation(main='zoom')
```
```python
%%title
# create a rich content title page
```
```python
%%slide 1
# slide 1 content
```
```python
%%slide 1 -m # new in 1.4.6
**Markdown here with extended options (see `ls.xmd_syntax` for info). Nested blocks are not supported**
```multicol 30 70 .Success
 less content
 +++
 more content
 ```
```

```python
%%slide 2 -s # new in 1.7.7
var = some_long_computation() # This will run only first time or when cell code chnages.
```

```python run source
x = 1 + 2
print(x) # will be printed on slide in somehwewere top as print appears first of all.
# Use `with ls.capture_std() as std: print();std.stdout` to see at exactly where it is printed.
```
There is a python block above with header `python run source`. We can display that block by &lcub;&lcub;source&rcub;&rcub; as below:
{{source}}

variable `x` defined there is shown here x = {{x}}. Only variables can be embeded in &lcub;&lcub;var&rcub;&rcub;, not expressions.
```python
@ls.frames(1,*objs)
def func(obj):
 write(obj) #This will create as many slides after the slide number 1 as length(objs)
```
```python
ls # This displays slides if on the last line of cell, or use `ls.show()`.
```

- Use `ls.from_markdown` to create multiple slides from markdown file/text.
  - Slides are added in order of content.
  - Slides should be separated by `---` (three dashes) in start of line
````

```
    write(ls[2].markdown) # write content of slide 2 from file
    plot_something() # Add other things to same file
    write_something()
```
- Use `ls.demo` to create example slides and start editing. Follow steps in first part.
- Use `ls.docs` to see upto date documentation.
- You can acess markdown content of an existing slide using `ls[key or index].markdown` if it has been created using `ls.from_markdown` or `%%slide i -m`.
- You can insert content usign `with ls[key or index].insert(index)` or `ls[key or index].insert_markdown` (1.7.7+).

New in 1.7.2

- Find special syntax to be used in markdown by `LiveSlides.xmd_syntax`.
- You can now show citations on bottom of each slide by setting `citation_mode = 'footnote'` in `LiveSlides` constructor.
- You can now access individual slides by indexing `s_i = ls[i]` where `i` is the slide index or by key as `s_3_1 = ls['3.1']` will give you slide which shows 3.1 at bottom.
- Basides indexing, you can access current displayed slide by `ls.current`.
- You can add new content to existing slides by using `with s_i.insert(where)` context. All new changes can be reverted by `s_i.reset()`.
- If a display is not complete, e.g. some widget missing on a slide, you can use `(ls.current, ls[index], ls[key]).update_display()` to update display.
- You can set overall animation by `ls.set_overall_animation` or per slide by `s_i.set_animation`
- You can now set CSS for each slide by `s_i.set_css` or `ls.set_slide_css` at current slide.

New in 1.7.5
Use `LiveSlides.extender` to add [markdown extensions](https://python-markdown.github.io/extensions/).
Also look at [PyMdown-Extensions](https://facelessuser.github.io/pymdown-extensions/).

New in 1.7.7
Use `slides[i].insert_markdown({'index': 'markdown_string',...})` to insert markdown (pasrsed objects) at indices.
`%%slide i -s` can be used to execute code just once in current session. It will run again if code changes in cell.

<h4 style=""color:green;"> 👈 Read more instructions in left panel</h4>

Markdown: Slide 2

```
# Slide 2 {.Success}
Created using `%%slide 2 -m` with markdown only

<a href="#slide2"><sup id ="slide2-back" style="color:var(--accent-color);">1</sup></a> Refrence to this will sh
```multicol
# Column A
||### Sub column A {.Success}||### Sub column B ||
+++
# Column B
```

That version from last slide is still in memory. See it is there {{version}}
```

Python: Slide 10

```
import textwrap
with slides.source.context() as s:
```

```
|---|---|---|
|d1|d2|d3|
|r1|r2|r3|
'''))
 s.focus_lines([3,4,5,6]).display()
```

Markdown: Slide 19

```
## $\LaTeX$ 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.
{.Note .Warning}

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

References

[1] This is reference created using markdown

[2] This is refernce to FigureWidget using slides.cite command

Python

```
slides.write_citations()
```