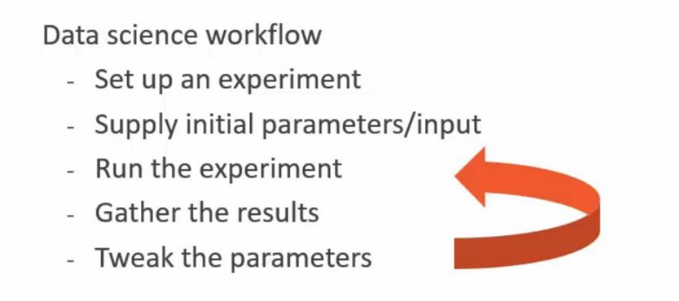
Jupyter

Getting Started with Jupyter Notebook and Python

Installing Jupyter Notebook

Jupyter is not an IDE and interactive environment. It is designed for an iterative development like below –



Python REPL requires more bare bone than chrome developer tool, due to which IPython interpreter was invented to support syntax highlighting, automatic indentation, execute shell commands. To improve this interpreter IPython Notebook is created for more collaborative, visualization and session features.

Jupyter notebook timeline –

REPL
IPython
IPython Notebook

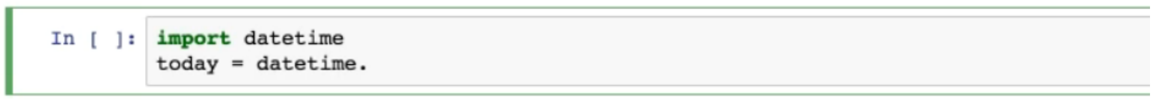
We can use jupyter notebook on the cloud using azure account. https://notebooks.azure.com/

Type jupyter notebook in console to make it started, use parameter –notebook_dir=notebook_path

Moving from the REPL to a Notebook

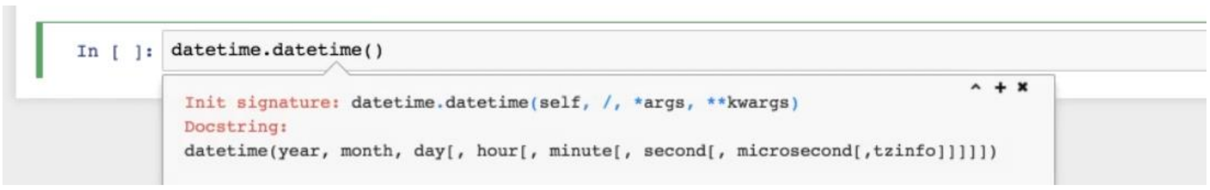
Use shortcuts – shift + enter

To get the code suggestion on 'tab' we need to first execute the library in that cell.



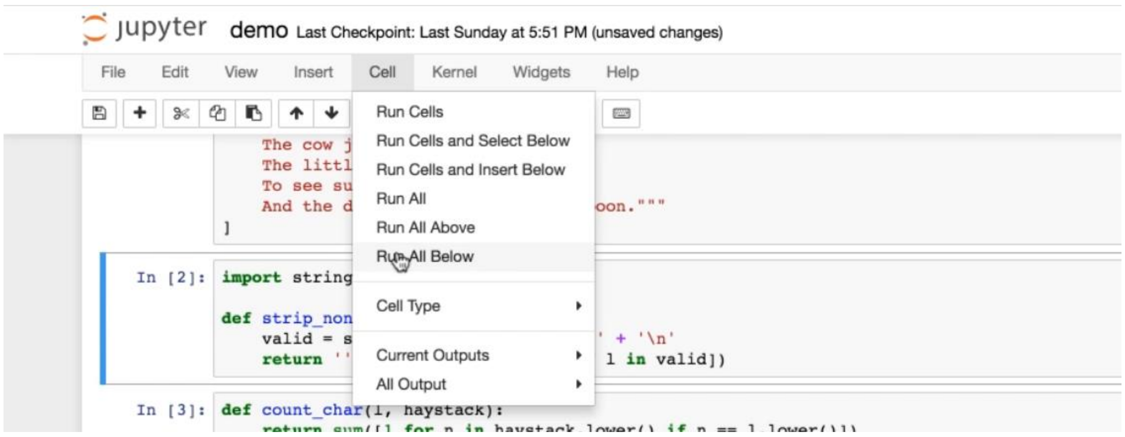
```
In [ ]: import datetime
        today = datetime.
```

To see the signature of a function use shift + tab



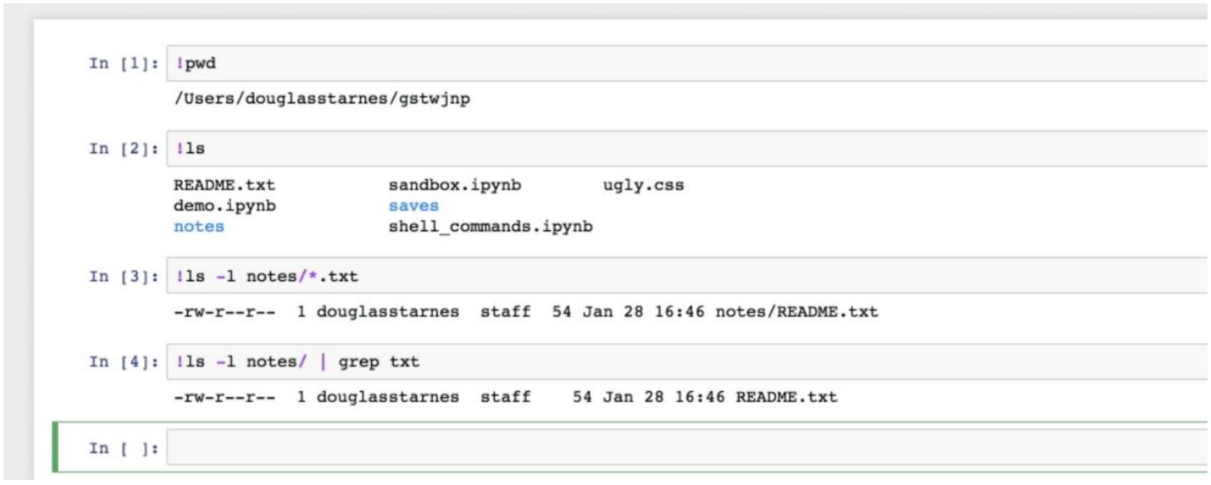
```
In [ ]: datetime.datetime()

Init signature: datetime.datetime(self, /, *args, **kwargs)
Docstring:
datetime(year, month, day[, hour[, minute[, second[, microsecond[,tzinfo]]]]])
```

Managing groups of cells – we can select a cell and then run all below cells



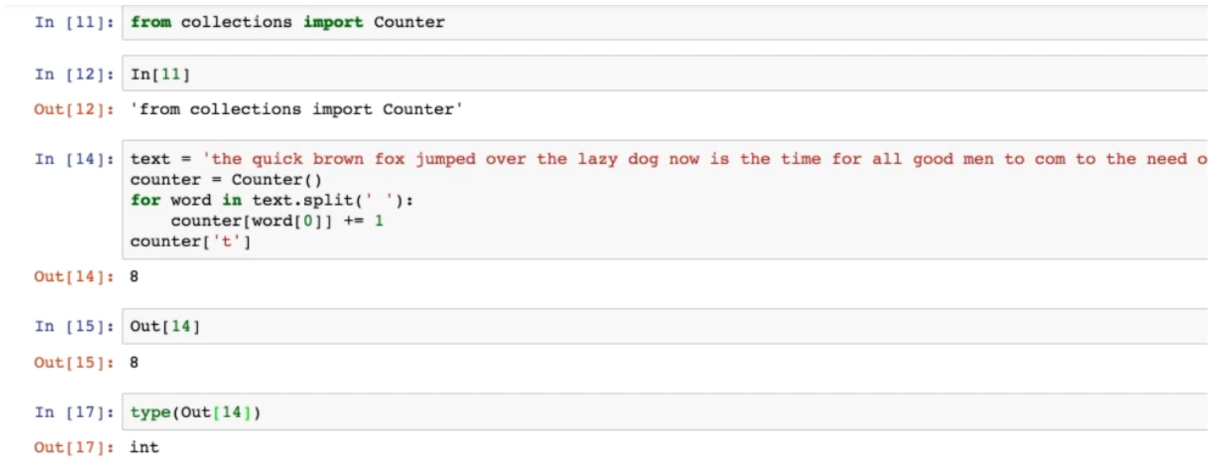
```
    The cow j
    The littl
    To see su
    And the d                                      oon."""
]

In [2]: import string

        def strip_non
            valid = s                              ' + '\n'
            return ''                              l in valid])

In [3]: def count_char(l, haystack):
```

Shell commands and special objects – we should prefix all shell commands with bang operation otherwise it can conflict with python local commands –



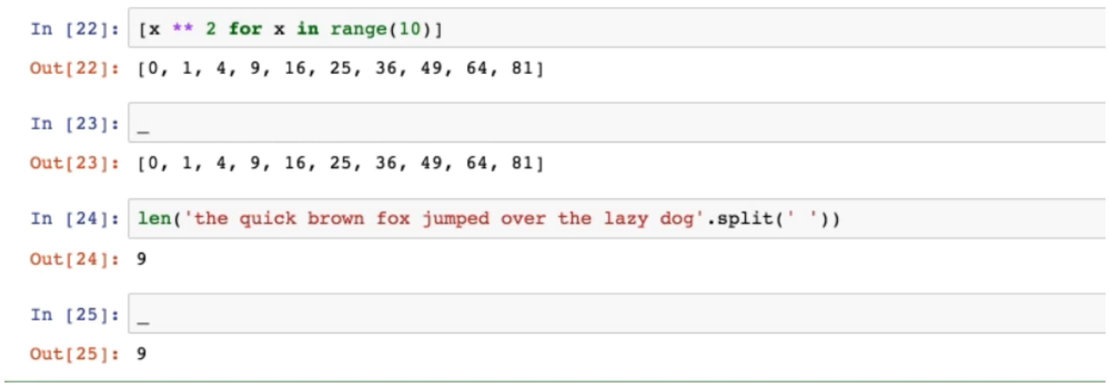
```
In [1]: !pwd

        /Users/douglasstarnes/gstwjnp

In [2]: !ls

        README.txt            sandbox.ipynb        ugly.css
        demo.ipynb            saves
        notes                 shell_commands.ipynb

In [3]: !ls -l notes/*.txt

        -rw-r--r--  1 douglasstarnes  staff  54 Jan 28 16:46 notes/README.txt

In [4]: !ls -l notes/ | grep txt

        -rw-r--r--  1 douglasstarnes  staff    54 Jan 28 16:46 README.txt

In [ ]:
```

The jupyter notebook stores the output of the cells into in and out object with numbered indices.



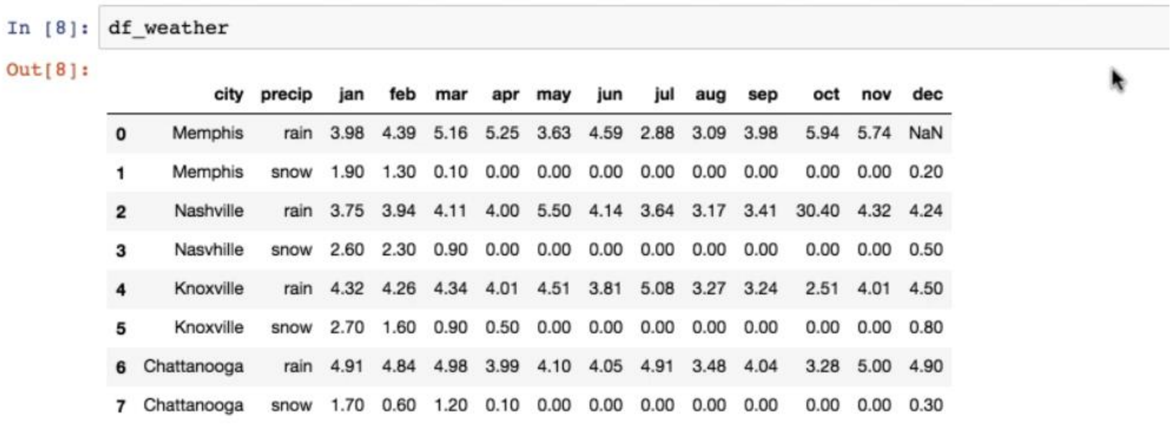
```
In [11]: from collections import Counter

In [12]: In[11]
Out[12]: 'from collections import Counter'

In [14]: text = 'the quick brown fox jumped over the lazy dog now is the time for all good men to com to the need o
         counter = Counter()
         for word in text.split(' '):
             counter[word[0]] += 1
         counter['t']
Out[14]: 8

In [15]: Out[14]
Out[15]: 8

In [17]: type(Out[14])
Out[17]: int
```

The _ variable with store the most recent output of a latest executed cell, we can also use this instead of using an temporary variable in some logic to hold the latest value –



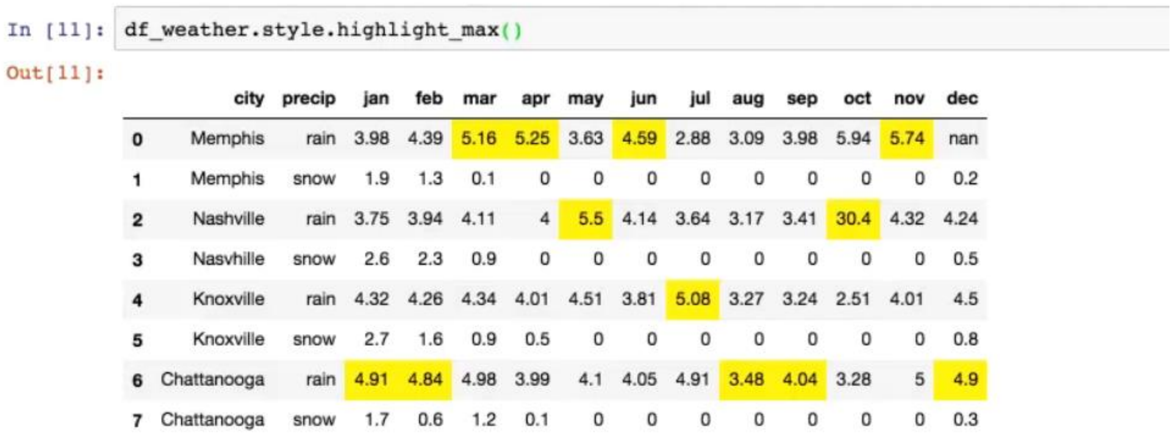
```
In [22]: [x ** 2 for x in range(10)]
Out[22]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

In [23]: _
Out[23]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

In [24]: len('the quick brown fox jumped over the lazy dog'.split(' '))
Out[24]: 9

In [25]: _
Out[25]: 9
```

Styling cell output – pandas told jupyter to render it as a rich text content



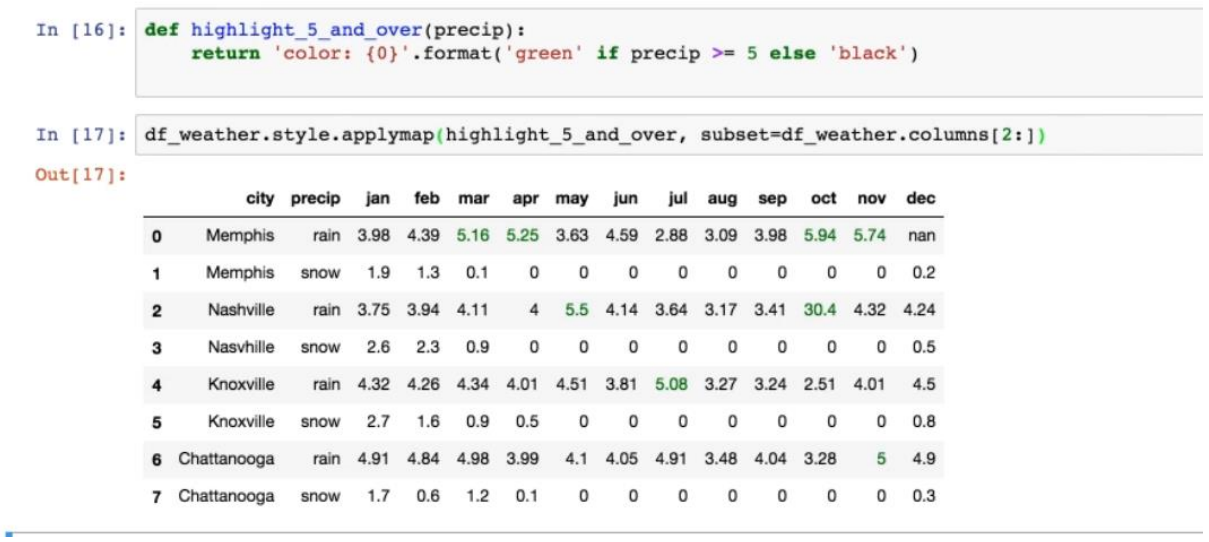
```
In [8]: df_weather
Out[8]:
```

	city	precip	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
0	Memphis	rain	3.98	4.39	5.16	5.25	3.63	4.59	2.88	3.09	3.98	5.94	5.74	NaN
1	Memphis	snow	1.90	1.30	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20
2	Nashville	rain	3.75	3.94	4.11	4.00	5.50	4.14	3.64	3.17	3.41	30.40	4.32	4.24
3	Nasvhille	snow	2.60	2.30	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50
4	Knoxville	rain	4.32	4.26	4.34	4.01	4.51	3.81	5.08	3.27	3.24	2.51	4.01	4.50
5	Knoxville	snow	2.70	1.60	0.90	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80
6	Chattanooga	rain	4.91	4.84	4.98	3.99	4.10	4.05	4.91	3.48	4.04	3.28	5.00	4.90
7	Chattanooga	snow	1.70	0.60	1.20	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30



```
In [11]: df_weather.style.highlight_max()
Out[11]:
```

	city	precip	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
0	Memphis	rain	3.98	4.39	5.16	5.25	3.63	4.59	2.88	3.09	3.98	5.94	5.74	nan
1	Memphis	snow	1.9	1.3	0.1	0	0	0	0	0	0	0	0	0.2
2	Nashville	rain	3.75	3.94	4.11	4	5.5	4.14	3.64	3.17	3.41	30.4	4.32	4.24
3	Nasvhille	snow	2.6	2.3	0.9	0	0	0	0	0	0	0	0	0.5
4	Knoxville	rain	4.32	4.26	4.34	4.01	4.51	3.81	5.08	3.27	3.24	2.51	4.01	4.5
5	Knoxville	snow	2.7	1.6	0.9	0.5	0	0	0	0	0	0	0	0.8
6	Chattanooga	rain	4.91	4.84	4.98	3.99	4.1	4.05	4.91	3.48	4.04	3.28	5	4.9
7	Chattanooga	snow	1.7	0.6	1.2	0.1	0	0	0	0	0	0	0	0.3

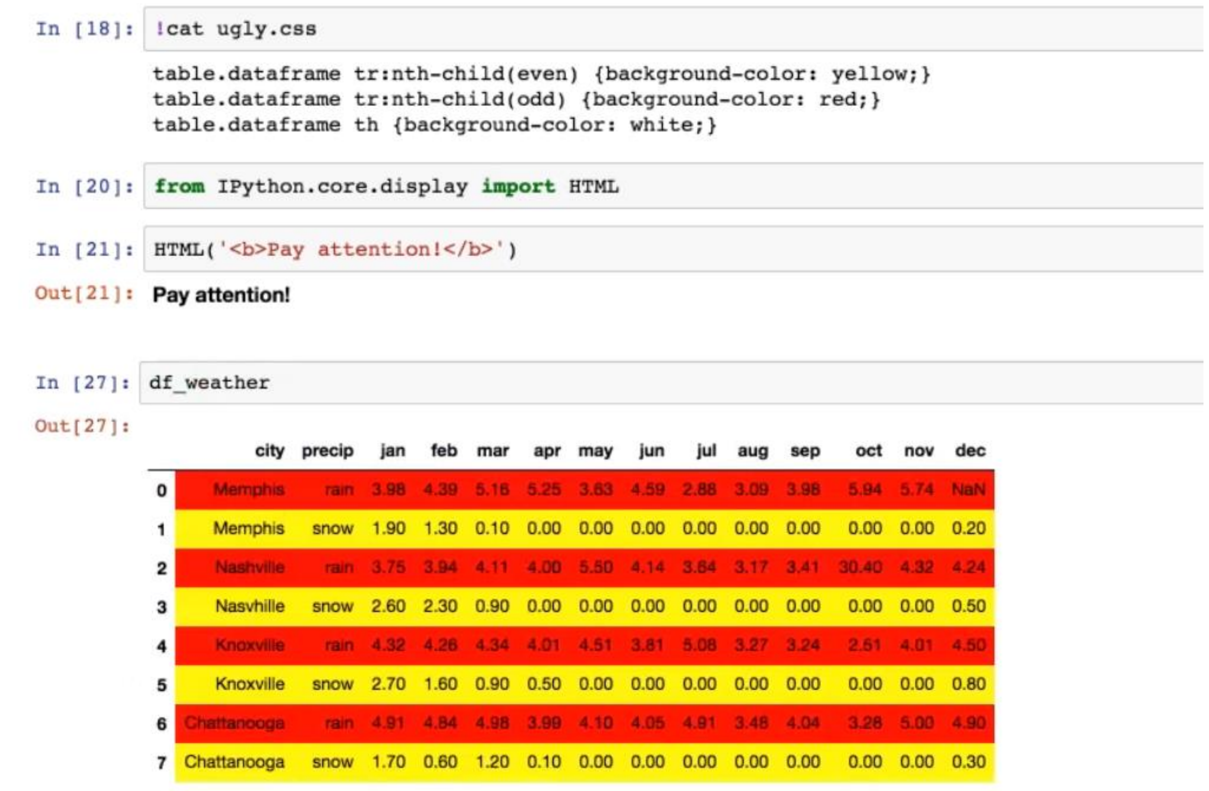


```
In [16]: def highlight_5_and_over(precip):
             return 'color: {0}'.format('green' if precip >= 5 else 'black')
```

```
In [17]: df_weather.style.applymap(highlight_5_and_over, subset=df_weather.columns[2:])
```

Out[17]:

	city	precip	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
0	Memphis	rain	3.98	4.39	5.16	5.25	3.63	4.59	2.88	3.09	3.98	5.94	5.74	nan
1	Memphis	snow	1.9	1.3	0.1	0	0	0	0	0	0	0	0	0.2
2	Nashville	rain	3.75	3.94	4.11	4	5.5	4.14	3.64	3.17	3.41	30.4	4.32	4.24
3	Nasvhille	snow	2.6	2.3	0.9	0	0	0	0	0	0	0	0	0.5
4	Knoxville	rain	4.32	4.26	4.34	4.01	4.51	3.81	5.08	3.27	3.24	2.51	4.01	4.5
5	Knoxville	snow	2.7	1.6	0.9	0.5	0	0	0	0	0	0	0	0.8
6	Chattanooga	rain	4.91	4.84	4.98	3.99	4.1	4.05	4.91	3.48	4.04	3.28	5	4.9
7	Chattanooga	snow	1.7	0.6	1.2	0.1	0	0	0	0	0	0	0	0.3



```
In [18]: !cat ugly.css

         table.dataframe tr:nth-child(even) {background-color: yellow;}
         table.dataframe tr:nth-child(odd) {background-color: red;}
         table.dataframe th {background-color: white;}
```

```
In [20]: from IPython.core.display import HTML
```

```
In [21]: HTML('<b>Pay attention!</b>')
```

Out[21]: **Pay attention!**

```
In [27]: df_weather
```

Out[27]:

	city	precip	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
0	Memphis	rain	3.98	4.39	5.16	5.25	3.63	4.59	2.88	3.09	3.98	5.94	5.74	NaN
1	Memphis	snow	1.90	1.30	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20
2	Nashville	rain	3.75	3.94	4.11	4.00	5.50	4.14	3.64	3.17	3.41	30.40	4.32	4.24
3	Nasvhille	snow	2.60	2.30	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50
4	Knoxville	rain	4.32	4.26	4.34	4.01	4.51	3.81	5.08	3.27	3.24	2.51	4.01	4.50
5	Knoxville	snow	2.70	1.60	0.90	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80
6	Chattanooga	rain	4.91	4.84	4.98	3.99	4.10	4.05	4.91	3.48	4.04	3.28	5.00	4.90
7	Chattanooga	snow	1.70	0.60	1.20	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30

Leveraging Special Notebook Features

Keyboard shortcuts – command mode and edit mode

X – cut selected cell

C – copy selected cell

V – paste **below**

Shift-V – paste **above**

D, D (press D twice) – delete selected cell

A – insert a cell above

B – insert a cell below

Shift Up/Down – Extend selection up/down

Ctrl Enter – Run selected cell

Shift Enter – Run selected, select cell below

- Insert new cell if at end of notebook

Alt Enter – Run selected, insert cell below

Shift M – Merge selected cells

Shift L – Toggle line numbers

Cut, Copy and Paste for text within a cell – use the browser shortcuts

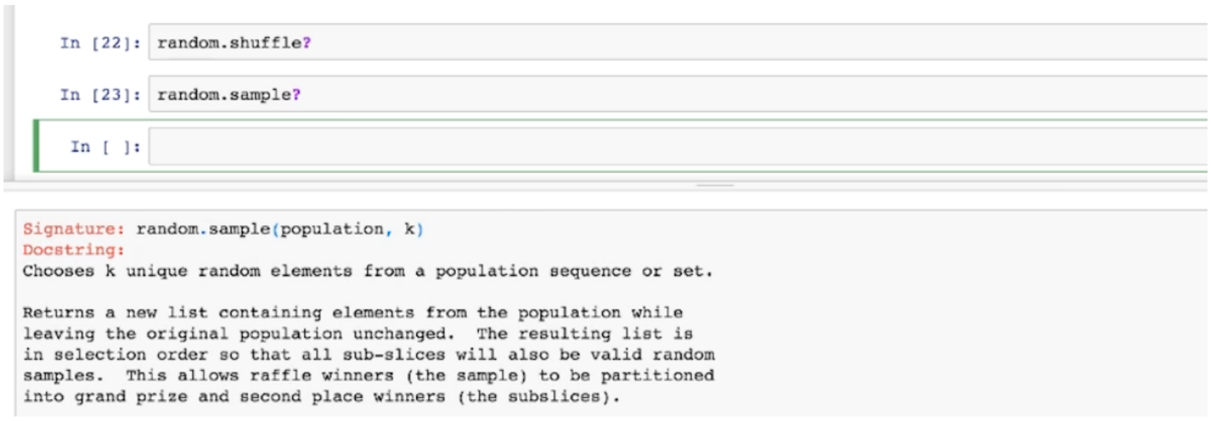
Ctrl] – indent

Ctrl [- dedent

Ctrl Shift Minus – split cell

Press H key while in command mode to see the full list of keyboard shortcuts.

Inline hep – append a question mark to any member and run the cell, docstring will appear at the bottom of the notebook window.



```
In [22]: random.shuffle?

In [23]: random.sample?

In [ ]:
```

```
Signature: random.sample(population, k)
Docstring:
Chooses k unique random elements from a population sequence or set.

Returns a new list containing elements from the population while
leaving the original population unchanged.  The resulting list is
in selection order so that all sub-slices will also be valid random
samples.  This allows raffle winners (the sample) to be partitioned
into grand prize and second place winners (the subslices).
```

Magic commands – prefixed with %.

Works with inline help

%lsmagic – list available magic commands

%history – manipulate session history

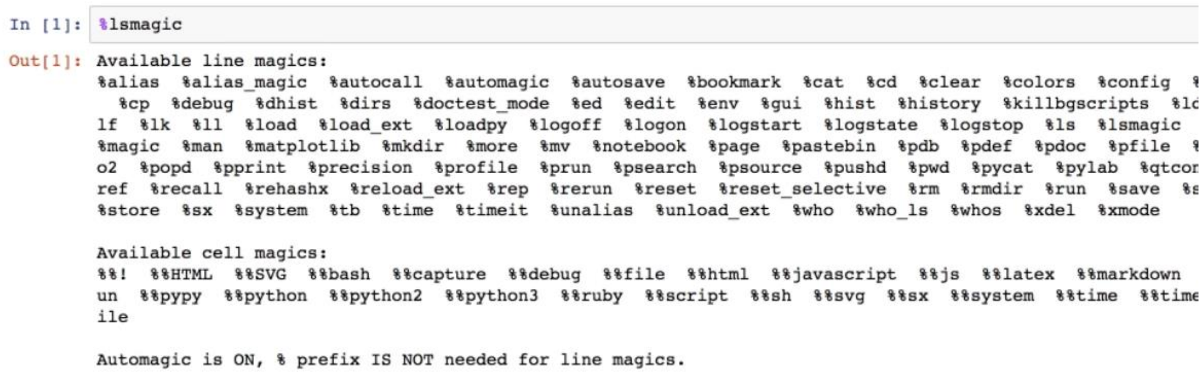
%load / %run – execute code in Python files

%alias – define a name for a system command

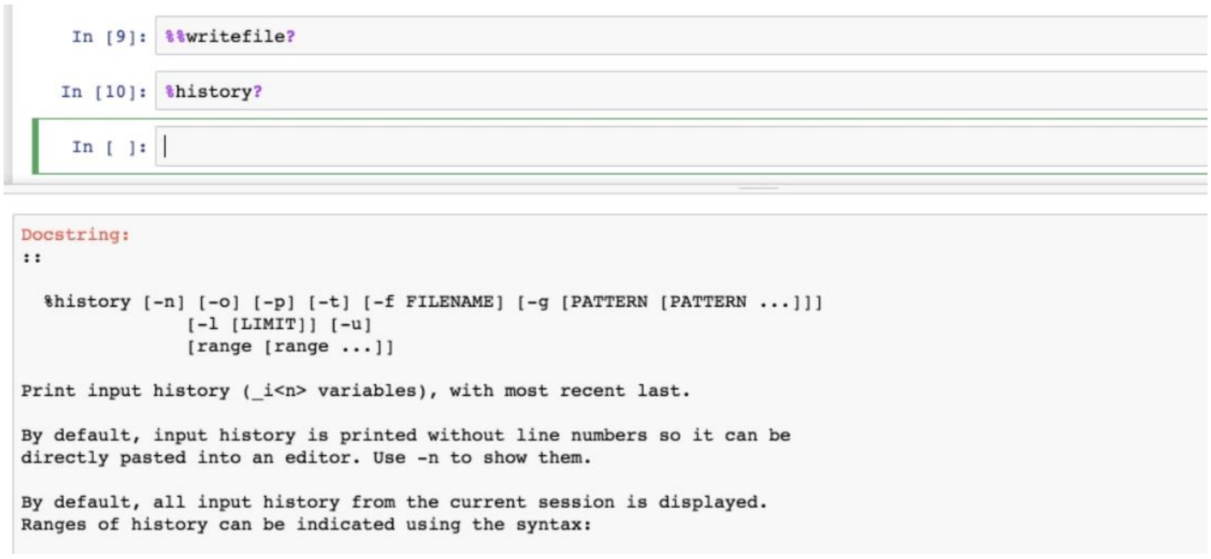
%reset – restores defaults

Security concerns - Jupyter notebook enables unrestricted access to system resources – shell commands in notebook, open a terminal window from the server.

List of magic commands –



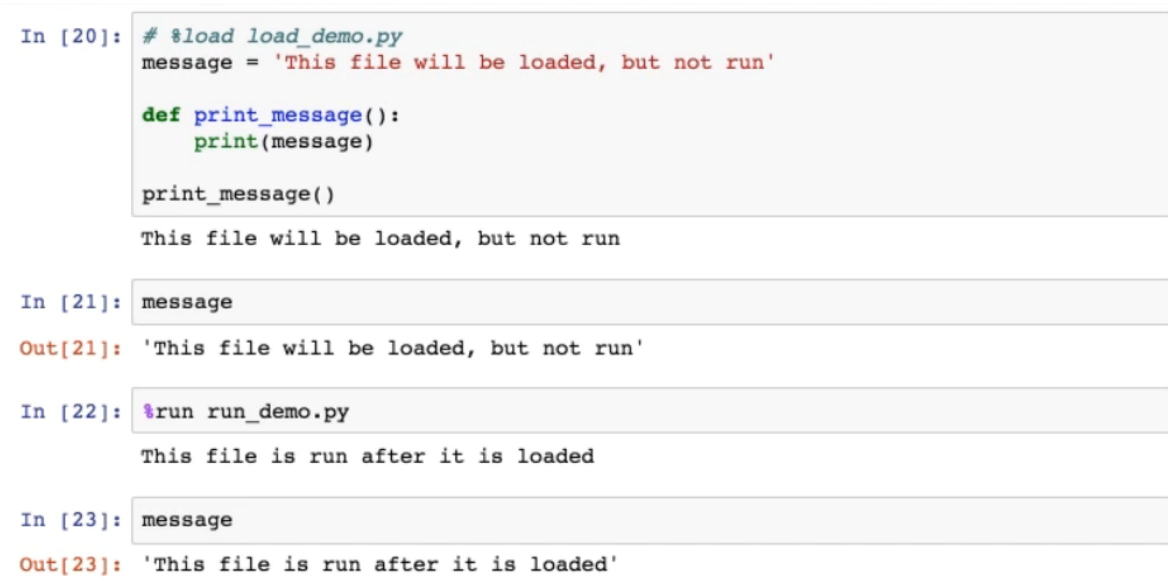
```
In [1]: %lsmagic

Out[1]: Available line magics:
        %alias  %alias_magic  %autocall  %automagic  %autosave  %bookmark  %cat  %cd  %clear  %colors  %config  %
          %cp  %debug  %dhist  %dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %history  %killbgscripts  %lc
        lf  %lk  %ll  %load  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %logstop  %ls  %lsmagic
        %magic  %man  %matplotlib  %mkdir  %more  %mv  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc  %pfile  %
        o2  %popd  %pprint  %precision  %profile  %prun  %psearch  %psource  %pushd  %pwd  %pycat  %pylab  %qtcor
        ref  %recall  %rehashx  %reload_ext  %rep  %rerun  %reset  %reset_selective  %rm  %rmdir  %run  %save  %s
        %store  %sx  %system  %tb  %time  %timeit  %unalias  %unload_ext  %who  %who_ls  %whos  %xdel  %xmode

        Available cell magics:
        %%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%javascript  %%js  %%latex  %%markdown
        un  %%pypy  %%python  %%python2  %%python3  %%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%time
        ile

        Automagic is ON, % prefix IS NOT needed for line magics.
```



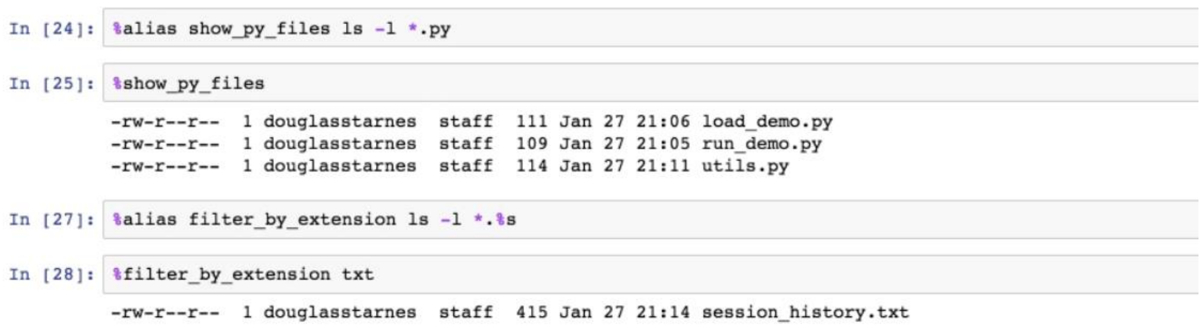
```
In [9]: %%writefile?

In [10]: %history?

In [ ]: |
```

```
Docstring:
::

  %history [-n] [-o] [-p] [-t] [-f FILENAME] [-g [PATTERN [PATTERN ...]]]
               [-l [LIMIT]] [-u]
               [range [range ...]]

Print input history (_i<n> variables), with most recent last.

By default, input history is printed without line numbers so it can be
directly pasted into an editor. Use -n to show them.

By default, all input history from the current session is displayed.
Ranges of history can be indicated using the syntax:
```

Load, run and alias magic commands –

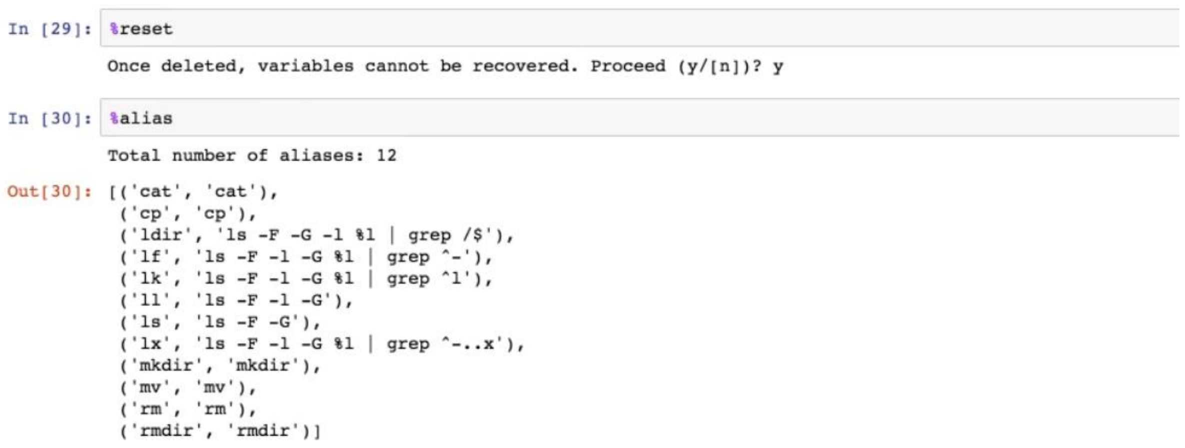


```
In [20]: # %load load_demo.py
         message = 'This file will be loaded, but not run'

         def print_message():
             print(message)

         print_message()

         This file will be loaded, but not run

In [21]: message

Out[21]: 'This file will be loaded, but not run'

In [22]: %run run_demo.py

         This file is run after it is loaded

In [23]: message

Out[23]: 'This file is run after it is loaded'
```



```
In [24]: %alias show_py_files ls -l *.py

In [25]: %show_py_files

         -rw-r--r--  1 douglasstarnes  staff  111 Jan 27 21:06 load_demo.py
         -rw-r--r--  1 douglasstarnes  staff  109 Jan 27 21:05 run_demo.py
         -rw-r--r--  1 douglasstarnes  staff  114 Jan 27 21:11 utils.py

In [27]: %alias filter_by_extension ls -l *.%s

In [28]: %filter_by_extension txt

         -rw-r--r--  1 douglasstarnes  staff  415 Jan 27 21:14 session_history.txt
```

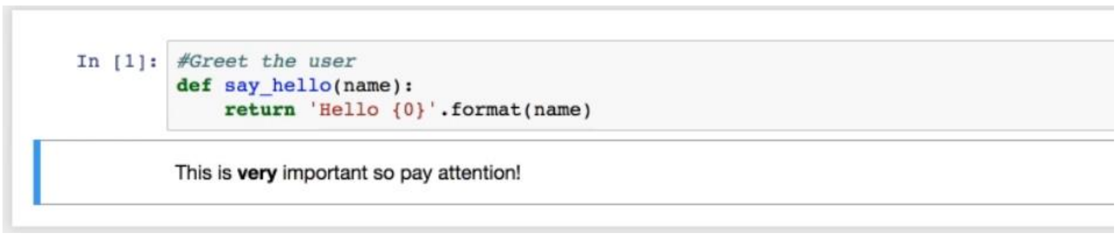
Resetting the magic commands –



```
In [29]: %reset

         Once deleted, variables cannot be recovered. Proceed (y/[n])? y

In [30]: %alias

         Total number of aliases: 12

Out[30]: [('cat', 'cat'),
          ('cp', 'cp'),
          ('ldir', 'ls -F -G -l %l | grep /$'),
          ('lf', 'ls -F -l -G %l | grep ^-'),
          ('lk', 'ls -F -l -G %l | grep ^l'),
          ('ll', 'ls -F -l -G'),
          ('ls', 'ls -F -G'),
          ('lx', 'ls -F -l -G %l | grep ^-..x'),
          ('mkdir', 'mkdir'),
          ('mv', 'mv'),
          ('rm', 'rm'),
          ('rmdir', 'rmdir')]
```

Displaying Rich Content

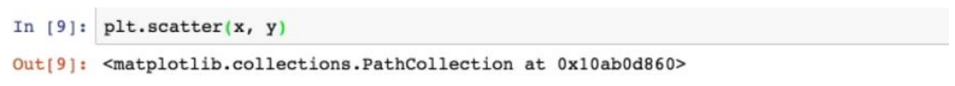
We can use markdown in jupyter it will render it in that cell itself, we can use M and Y key to convert the cell into markdown or code vice versa –



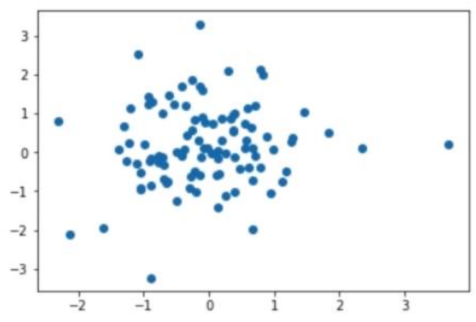
```
In [1]: #Greet the user
        def say_hello(name):
            return 'Hello {0}'.format(name)
```

This is **very** important so pay attention!

Including visualization – use %matplotlib inline to see visualization in same cell instead of new window.

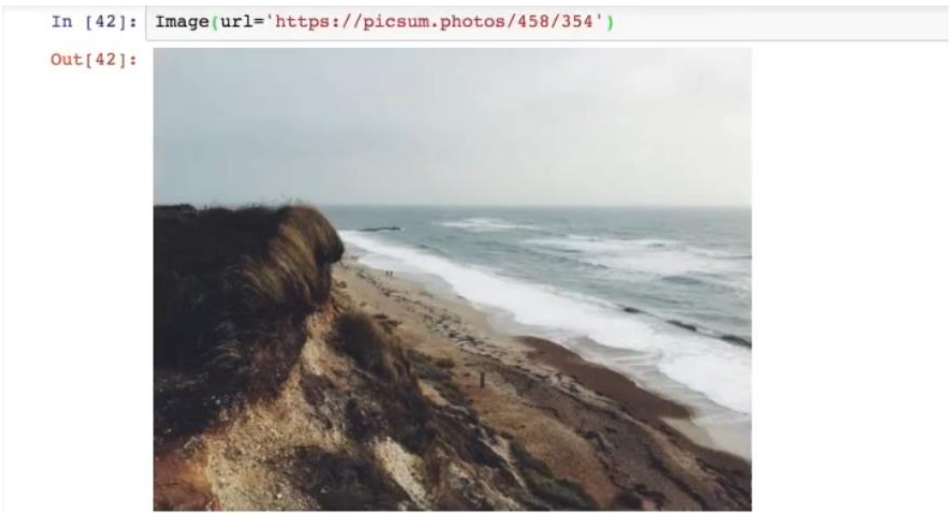


```
In [9]: plt.scatter(x, y)

Out[9]: <matplotlib.collections.PathCollection at 0x10ab0d860>
```



Instead of using mat plot library we can use seaborn which requires less code.

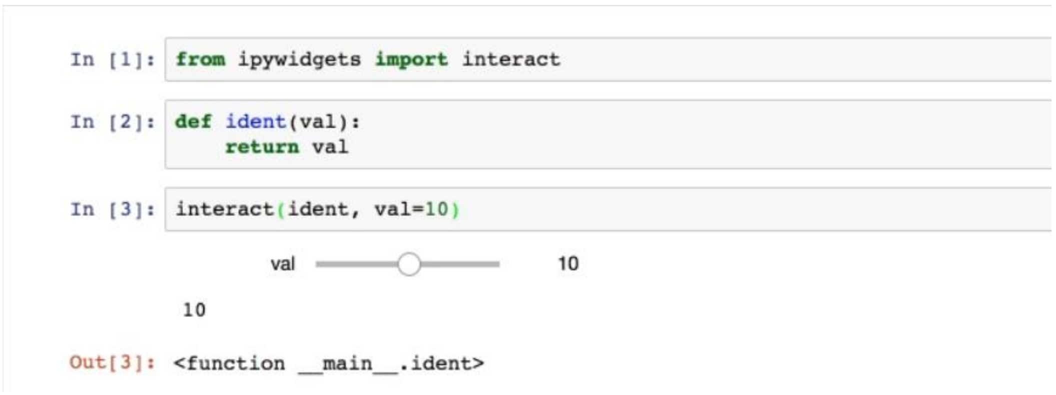
Displaying images



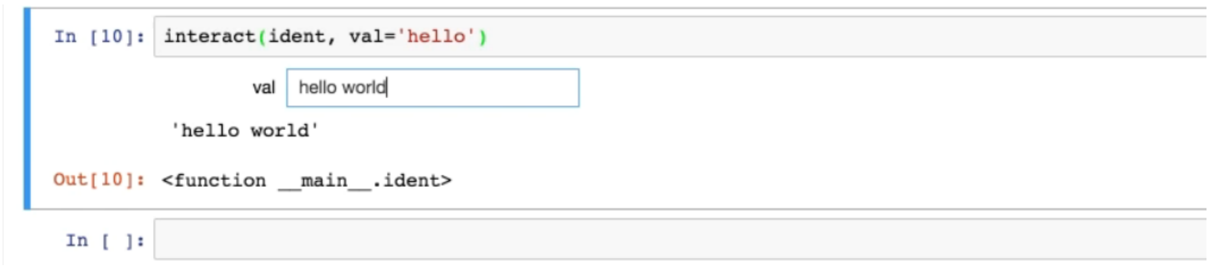
```
In [42]: Image(url='https://picsum.photos/458/354')
Out[42]:
```

Extending the Notebook User Interface

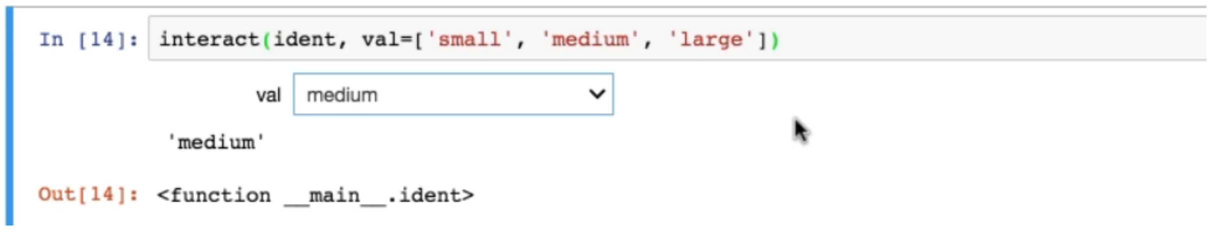
Embedding javascript controls in the notebook to make the user interface more user friendly – we need to install ipywidgets package



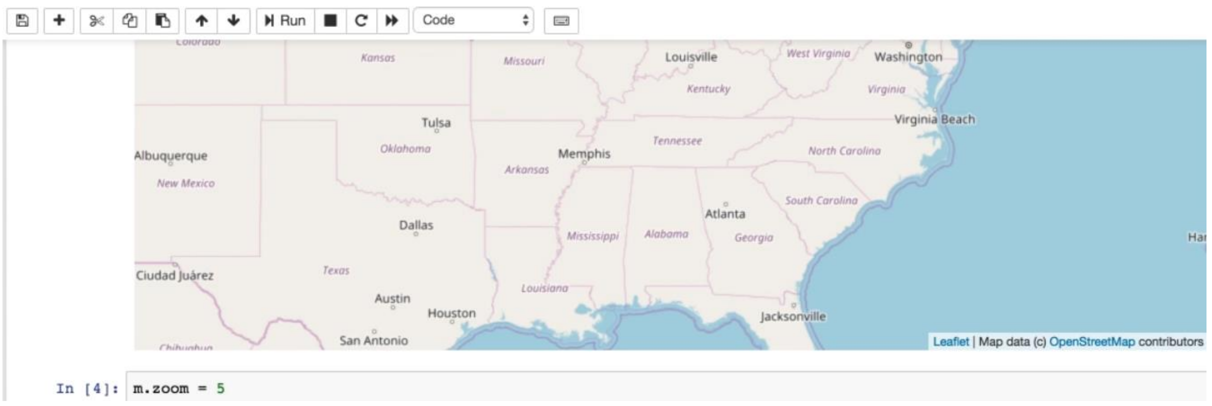
```
In [1]: from ipywidgets import interact

In [2]: def ident(val):
            return val

In [3]: interact(ident, val=10)

            val                10

        10

Out[3]: <function __main__.ident>
```



```
In [10]: interact(ident, val='hello')

             val  hello world

         'hello world'

Out[10]: <function __main__.ident>

In [ ]:
```

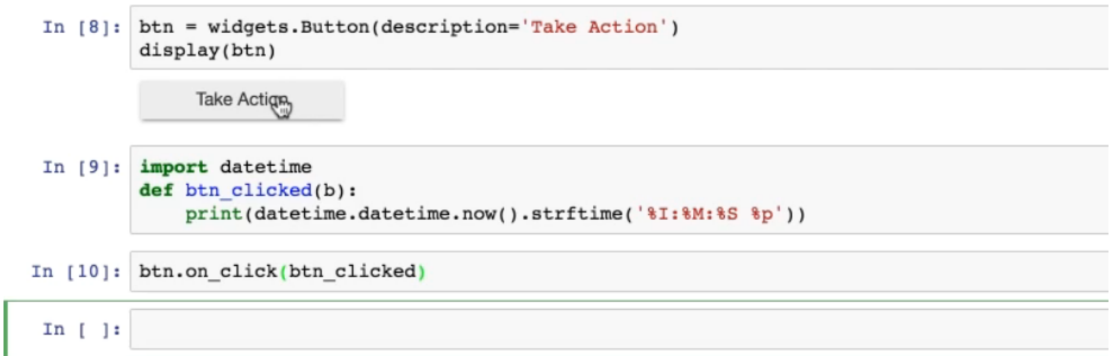


```
In [14]: interact(ident, val=['small', 'medium', 'large'])

             val  medium

         'medium'

Out[14]: <function __main__.ident>
```



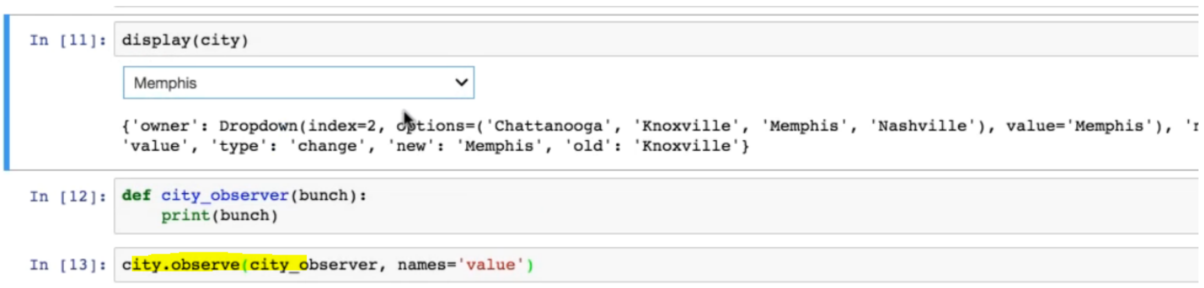
```
In [4]: m.zoom = 5
```

Handling widget events –



```
In [8]: btn = widgets.Button(description='Take Action')
        display(btn)
```

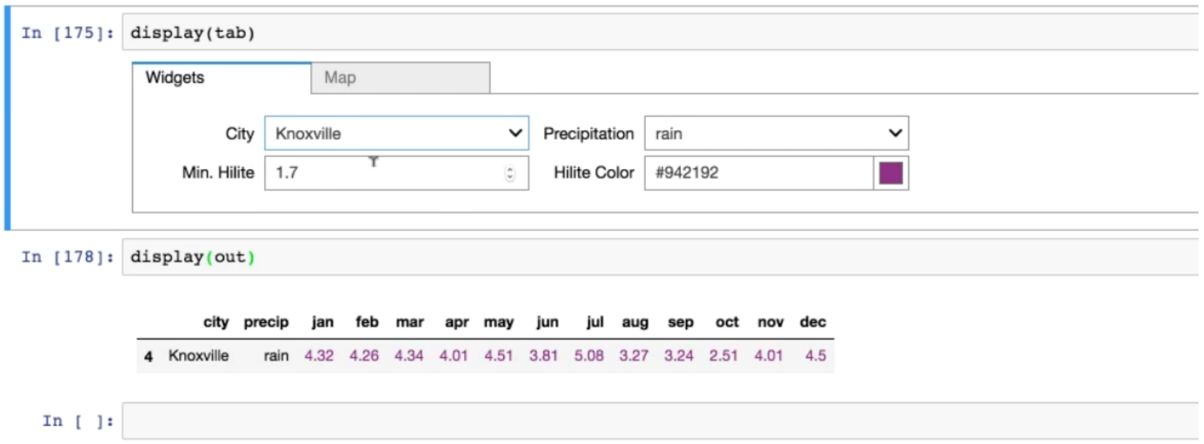
```
In [9]: import datetime
        def btn_clicked(b):
            print(datetime.datetime.now().strftime('%I:%M:%S %p'))

In [10]: btn.on_click(btn_clicked)

In [ ]:
```



```
In [11]: display(city)
```

```
{'owner': Dropdown(index=2, options=('Chattanooga', 'Knoxville', 'Memphis', 'Nashville'), value='Memphis'), 'r
'value', 'type': 'change', 'new': 'Memphis', 'old': 'Knoxville'}

In [12]: def city_observer(bunch):
             print(bunch)

In [13]: city.observe(city_observer, names='value')
```

Create an interactive dashboard like below by controller widget output –



```
In [175]: display(tab)
```

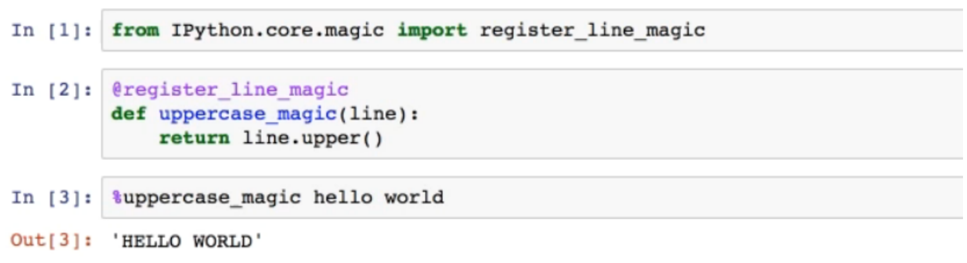
City: Knoxville
Precipitation: rain
Min. Hilite: 1.7
Hilite Color: #942192

```
In [178]: display(out)
```

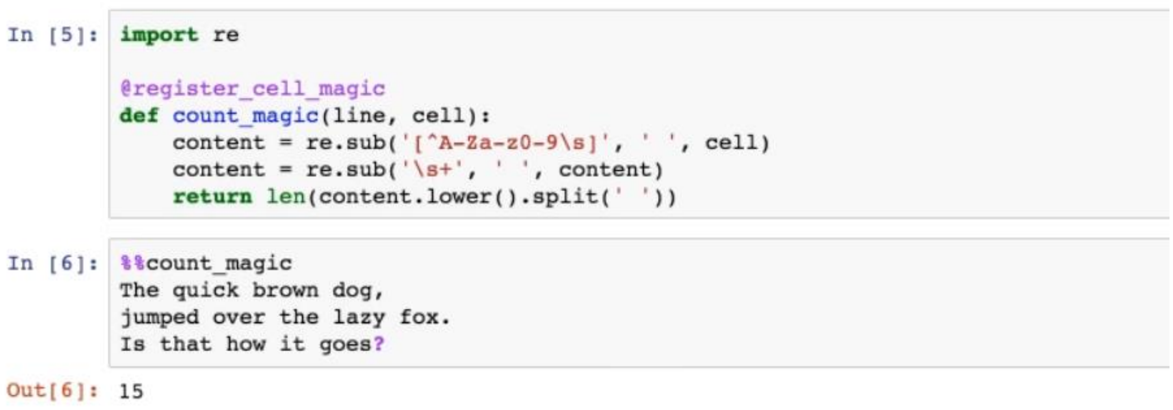
	city	precip	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
4	Knoxville	rain	4.32	4.26	4.34	4.01	4.51	3.81	5.08	3.27	3.24	2.51	4.01	4.5

```
In [ ]:
```

Custom magic commands – line magic and cell magic, these are regular old python functions and classes that have some decorators applied from the IPython.

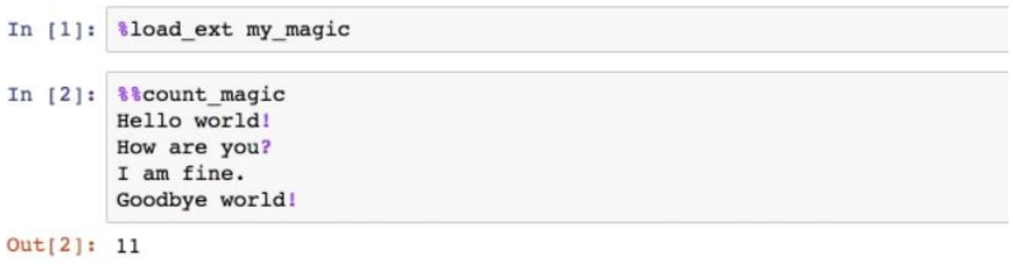


```
In [1]: from IPython.core.magic import register_line_magic

In [2]: @register_line_magic
        def uppercase_magic(line):
            return line.upper()

In [3]: %uppercase_magic hello world
Out[3]: 'HELLO WORLD'
```

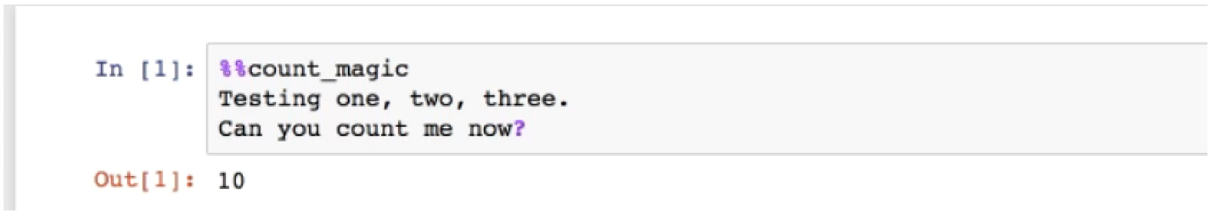


```
In [5]: import re

        @register_cell_magic
        def count_magic(line, cell):
            content = re.sub('[^A-Za-z0-9\s]', ' ', cell)
            content = re.sub('\s+', ' ', content)
            return len(content.lower().split(' '))

In [6]: %%count_magic
        The quick brown dog,
        jumped over the lazy fox.
        Is that how it goes?
Out[6]: 15
```

We can put them into some separate file and load it from there, or we can load them in user profile to load automatically when jupyter notebook server start and available all to use –

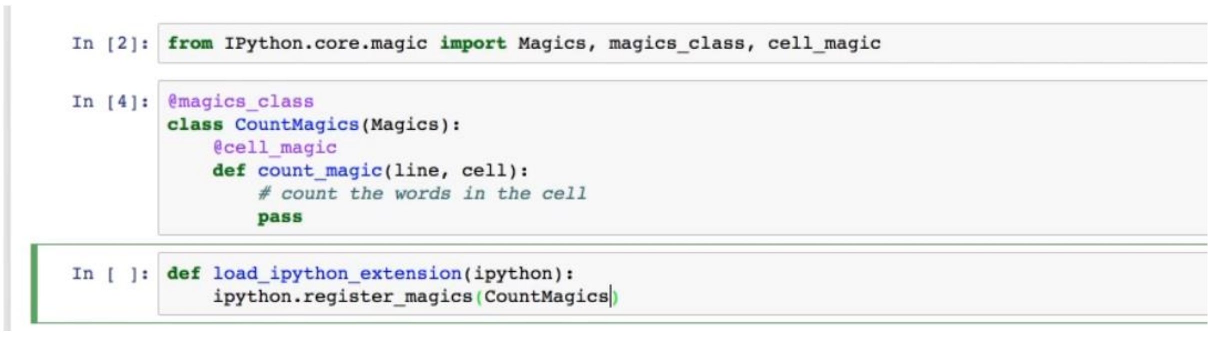


```
In [1]: %load_ext my_magic

In [2]: %%count_magic
        Hello world!
        How are you?
        I am fine.
        Goodbye world!
Out[2]: 11
```



```
In [1]: %%count_magic
        Testing one, two, three.
        Can you count me now?
Out[1]: 10
```

Class based magic commands –



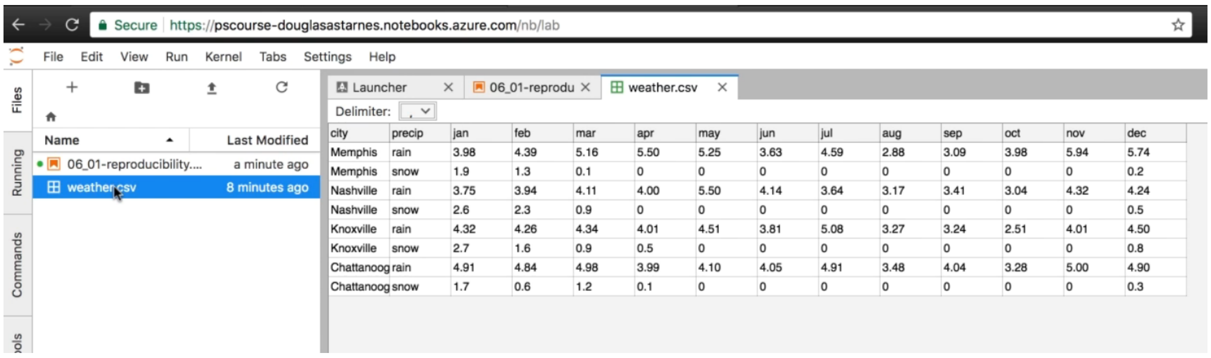
```
In [2]: from IPython.core.magic import Magics, magics_class, cell_magic

In [4]: @magics_class
        class CountMagics(Magics):
            @cell_magic
            def count_magic(line, cell):
                # count the words in the cell
                pass

In [ ]: def load_ipython_extension(ipython):
            ipython.register_magics(CountMagics)
```

Collaborating on Notebooks

We should store the dependencies in requirement.txt file.

Use jupyter ide on azure notebooks.

Jupyter Labs is new future of Jupyter notebook, try jupyter lab.


Secure https://pscourse-douglasastarnes.notebooks.azure.com/nb/lab
File Edit View Run Kernel Tabs Settings Help
Files
Running
Commands
Name
Last Modified
06_01-reproducibility.... a minute ago
weather.csv 8 minutes ago
Launcher
06_01-reprodu
weather.csv
Delimiter: ,

city	precip	jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
Memphis	rain	3.98	4.39	5.16	5.50	5.25	3.63	4.59	2.88	3.09	3.98	5.94	5.74
Memphis	snow	1.9	1.3	0.1	0	0	0	0	0	0	0	0	0.2
Nashville	rain	3.75	3.94	4.11	4.00	5.50	4.14	3.64	3.17	3.41	3.04	4.32	4.24
Nashville	snow	2.6	2.3	0.9	0	0	0	0	0	0	0	0	0.5
Knoxville	rain	4.32	4.26	4.34	4.01	4.51	3.81	5.08	3.27	3.24	2.51	4.01	4.50
Knoxville	snow	2.7	1.6	0.9	0.5	0	0	0	0	0	0	0	0.8
Chattanoog	rain	4.91	4.84	4.98	3.99	4.10	4.05	4.91	3.48	4.04	3.28	5.00	4.90
Chattanoog	snow	1.7	0.6	1.2	0.1	0	0	0	0	0	0	0	0.3