Welcome to the notes

This is a Jupyter Notebook of my notes on using Jupyter Notebooks. It contains my notes on using Markdown and Latex to create math equations that look nice. Here are some key tips to know ASAP:

- comment and uncomment lines of code with the # which will enable and disable the line
- ipynb files are just plain text. Modern envinroments (VSCode, Colab, Jupyter Lab) apply syntax formatting to make things visually readable. This is a powerful learning aid.
- the second amazing learning aids are the new AIs (chatGPT, etc) which can answer questions, explain code, fix code, modify code, etc.

For a more complete and authoritative guide, see the official Jupyter notebook documentation here.

Introduction

Cells

A notebook is a list of cells. Cells contain either explanatory **text** or executable **code** and its output. Click a cell to select it.

Markdown vs code cells

Execution cells are divided into two **types**: **markdown** cells and **code** cells.

Markdown cells, like this one, allow you to type formatted text, equations, bulleted lists, and so on. To edit or view the raw content of a markdown cell, either **[double-click]** on the cell or just hit **ENTER**. Try that now. Take a look at the raw markdown, and then execute the cell and advance to the next cell (again by pressing SHIFT+ENTER).

The other type of cells are code cells. The notebooks in this series will be based on Python 3, which means that the code cells will be interpreted as Python3 code. You can type any commands that you would normally type in a Python script or iPython command line. Go ahead and advance to the next cell, which is an example of a code cell, and execute it.

In [ ]:
```
from time import sleep
mynumber = 1 + (2 / 3) - (4 * 5)
sleep(2)
mynumber
```

Out[ ]:
```
-18.333333333333332
```

Notice that after executing the code cell, it is assigned an input number and its output (if any) is assigned an output number. If you re-execute the code cell, it is assigned a new input/output number. If a code cell is still executing, say if you're performing an intensive calculation, the input number appears as [∗]. (Notice, here I included the sleep command only to slow down the execution of the cell to illustrate this.) You either need to wait until a cell has finished before executing a new cell, or you can interrupt the execution using the stop button in the toolbar above.

Differences between Colab Markdown and other Markdown dialects

Most LaTeX commands can be used in markdown cells. For example, you can express symbols like α, β, γ in the usual way, just as if you were working in a TeX file. You can also add equations, either using the usual LaTeX commands like this:

```
\begin{equation}
  \cos(a + b) = \cos{a}\cos{b} - \sin{a}\sin{b}
\end{equation}
```

or in more abbreviated form using the double dollar sign "

$$" likethis : ‘‘‘$$

\cos(a + b) = \cos{a}\cos{b} - \sin{a}\sin{b}

$$‘‘‘andbothwilllooklikethis :$$

\cos(a + b) = \cos{a}\cos{b} - \sin{a}\sin{b} $$ To see the raw markdown in thie nicely formatted cell with the equation above - to show the commands used to create the above expressions, double-click on this cell or just hit ENTER.

Although very similar to LaTeX, markdown and LaTeX are not identical. For example, at the time of writing this, LaTeX tables do not work well in markdown. You only need to be aware of these limitations if you are going to start creating your own content in Jupyter notebooks. As a user, these limitations will not affect you.

At any given time when working in a Jupyter notebook, you are either in **edit mode** or **command mode**.

When in command mode, the current cell has a blue highlighted border. In this mode you can navigate up and down between cells using the arrow keys on your keyboard. When you execute a cell and advance to the next cell, you always start that new cell in command mode. This allows you to rapidly execute the cells in a notebook by simply holding down SHIFT and repeatedly hitting the ENTER key.

To edit a cell, you need to enter edit mode. You do this by either double-clicking the cell, or simply hitting ENTER. (You should have done this already above, but if you haven't, try it now.) When in edit mode, the current cell has a green highlighted border.

To switch out of edit mode, execute the cell. You can execute the cell and advance to the next cell with SHIFT+ENTER. Alternatively, you can execute the cell without advancing to the next cell with CTRL+ENTER.

Take a minute to practice entering/exiting edit mode and command mode.

There is a lot of useful information under the "Help" menu. There is a list of keyboard shortcuts as well as links to tutorials and documentation for the various common tools and languages used in Jupyter notebooks. If you're wondering how to do something, or getting hung-up on something, browse the contents of the help menu.

Basic Usage of Jupyter Notebooks

split cell at this line: `ctrl` + `m` + `-`

Adding and Moving Cells

You can add new cells by using the **+ CODE** and **+ TEXT** buttons that show when you hover between cells. These buttons are also in the toolbar above the notebook where they can be used to add a cell below the currently selected cell.

You can move a cell by selecting it and clicking **Cell Up** or **Cell Down** in the top toolbar.

Consecutive cells can be selected by "lasso selection" by dragging from outside one cell and through the group. Non-adjacent cells can be selected concurrently by clicking one and then holding down Ctrl while clicking another. Similarly, using Shift instead of Ctrl will select all intermediate cells.

Text cells

This is a **text cell**. You can **double-click** to edit this cell. Text cells use markdown syntax. To learn more, see our markdown guide.

You can also add math to text cells using LaTeX to be rendered by MathJax. Just place the statement within a pair of **$** signs. For example `$\sqrt{3x-1}+(1+x)^2$` becomes $\sqrt{3x-1}+(1+x)^2$.

Code cells

Below is a **code cell**. Once the toolbar button indicates CONNECTED, click in the cell to select it and execute the contents in the following ways:

- Click the **Play icon** in the left gutter of the cell;
- Type **Cmd/Ctrl+Enter** to run the cell in place;
- Type **Shift+Enter** to run the cell and move focus to the next cell (adding one if none exists); or
- Type **Alt+Enter** to run the cell and insert a new code cell immediately below it.

There are additional options for running some or all cells in the **Runtime** menu.

Keyboard Navigation

1. Show Help `CMD/CTRL M + h` Perhaps the most useful shortcut you should know. This shows you all the currently set keyboard shortcuts in Colab, so if you ever forget anything, this is the place to go 😉
2. Mount Drive `Customise Your Own!` Executing this shortcut brings up this dialog box This shortcut allows you to mount your Google Drive onto Colab so that you can access your files with ease. There is no default keyboard shortcut for this, but you can set your own. For me, I use CMD + D (D for 'Drive).
3. Interrupt Execution `CMD/CTRL M + i` Interrupts cell execution if you ever get caught in an infinite loop!
4. Adding Cursors. There are 3 main ways to add cursors to make multiple edits at the same time, in increasing level of control: `CMD / CTRL + Shift + L` : Get a cursor for every word instance `CMD / CTRL + D` : Select each word one at a time `OPTN / ALT + Clic` k: Specify cursor position Great for moving lines up and down without all that copy-pasting. Hold the `Shift` key as well to copy the lines up or down.
5. Deleting Code

Delete line `CMD / CTRL + Shift + K` :

Delete all before cursor `CMD / CTRL + Delete`

1. Indenting Lines `CMD/CTRL M + Square brackets` Yes, typically one might use the TAB key to help with indenting, but if say you're in the middle of a line, that's not going to work so well: Tabbing in the middle of a line, Use the square brackets `[` (to indent left) and `]` (to indent right) 👍
2. Move To Start and End of Line `CTRL a/e` `CTRL-a` will bring you to the start of a line, and `CTRL-e` will take you to the end of a line. Simple, but super useful!

Formatting

```
>One level of indentation
```

> One level of indentation

```
>>Two levels of indentation
```

> > Two levels of indentation

italicized text Code blocks

```
    ```python
    print("a")
    ```
```

```python
print("a")
```

Markdown	Preview
`**bold text**`	**bold text**
`*italicized text*` or `_italicized text_`	*italicized text*
`` `Monospace` ``	`Monospace`
`~~strikethrough~~`	~~strikethrough~~
`[A link](https://www.google.com)`	A link
`![An image](https://www.google.com/images/rss.png)`	

Headings are rendered as titles.

```
# Section 1 Juan
# Section 2
## Sub-section under Section 2
### Sub-section under the sub-section under Section 2
# Section 3
```

The table of contents, available on the left side of Colab, is populated using at most one section title from each text cell.

the text in below is preceeded by ```markdown which makes it display in this lovely format

```
this is the text precedded by ```markdown. This is displayed within Colab and other jupyter viewers as markdown formatted text now.
```

In [ ]:
```
a = 10
a
```

Out[ ]:
```
10
```

Horizontal rules (like above):

Embedded SVGs

In [ ]:
```
##@title Vanilla, inline SVG tag
%%html
<svg xmlns="http://www.w3.org/2000/svg" height="40" width="40"><circle cx="20" cy="20" r="20" fill="red" /></svg>
```

In [ ]:
```
##@title Vanilla, URL-encoded, and Base64-encoded SVG in <img> tag
%%html
<!---->

<img src='data:image/svg+xml;ascii,<svg xmlns="http://www.w3.org/2000/svg" height="40" width="40"><circle cx="20" cy="20" r="20" f

<img src="data:image/svg+xml;ascii,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20height%3D%2240%22%20width%3D%2240

<img src="

<!--URL-encoded and Base64-encoded CSS Background Image-->

<div style="width: 40px; height: 40px; background-image: url('data:image/svg+xml;ascii,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org

<div style="width: 40px; height: 40px; background-image: url('
```

In [ ]:
```
##@title URL-encoded and Base64-encoded CSS Background Image
%%html

<div style="width: 40px; height: 40px; background-image: url('data:image/svg+xml;ascii,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org
<div style="width: 40px; height: 40px; background-image: url('
```

Lists

Ordered lists:

```
1. One
1. Two
1. Three
check out how it auto numerates even though 1 was entered on all
```

1. One
2. Two
3. Three

Unordered lists:

```
* One
* Two
* Three
```

- One
- Two
- Three

Tables

```
First column name  | Second column name
-------------------|------------------
Row 1, Col 1       | Row 1, Col 2
Row 2, Col 1       | Row 2, Col 2
```

First column name	Second column name
Row 1, Col 1	Row 1, Col 2
Row 2, Col 1	Row 2, Col 2

Checkpointing and Autosave

As a final note, by default Jupyter notebooks are autosaved periodically. So if you accidentally close a notebook or your machine crashes, any recent changes will have been saved. However when you manually click save (or CTRL+S), you also create a checkpoint. Jupyter always stores at least one checkpoint file, which you can revert to using the "File" menu. While a notebook is still running, the most up-to-date autosaved version of the file usually differs from the checkpoint file. When a notebook is gracefully saved and closed, the checkpoint file provides a backup to the saved file.

[NOTE: there is only one checkpoint file at any given time. So the built-in checkpoint file does not provide a fool-proof version control system.]

Colab uses marked.js and so is similar but not quite identical to the Markdown used by Jupyter and Github.

Colab supports (MathJax) $\LaTeX$ equations like Jupyter, but does not allow HTML tags in the Markdown. Colab does not support some GitHub additions like emojis and to-do checkboxes.

If HTML must be included in a Colab notebook, see the %%html magic.

The Power of Coding

In [ ]:
```

```

Magics

Part of the power of Jupyter notebooks is that provide an environment that supports multiple languages and tools simultaneously. Some of these tools are wrapped in so-called "magic commands" or simply "magics". Here we illustrate how to use just two of these: the %%writefile and %load magics.

The %%writefile magic is an example of a "cell magic", so-called because it operates on the contents of an entire cell (i.e., takes the entire cell as its argument). This cell magic will write the contents the cell to a file in your working directory.

The %load magic is an example of a "line magic", so-called because it operates on a single line (i.e., its arguments must be in the same line as the command). This magic will load the contents of the specified file into the current cell.

In [ ]:
```
%lsmagic #lists the availble magics
```

Out[ ]:
```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark  %cat  %cd  %clear  %colors  %conda  %config  %conn
ect_info  %cp  %debug  %dhist  %dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %history  %killbgscripts  %ldir  %less  %lf
%lk  %ll  %load  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %logstop  %ls  %lsmagic  %lx  %macro  %magic  %man  %
matplotlib  %mkdir  %more  %mv  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc  %pfile  %pinfo  %pinfo2  %pip  %popd  %pprint  %
precision  %prun  %psearch  %psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %rehashx  %reload_ext  %rep
%rerun  %reset  %reset_selective  %rm  %rmdir  %run  %save  %sc  %set_env  %shell  %store  %sx  %system  %tb  %tensorflow_version
%time  %timeit  %unalias  %unload_ext  %who  %who_ls  %whos  %xdel  %xmode

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

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

Example:

In [ ]:
```
%%writefile hello_world.txt
Hello world!!!
1 + 1 = 2
```

```
Writing hello_world.txt
```

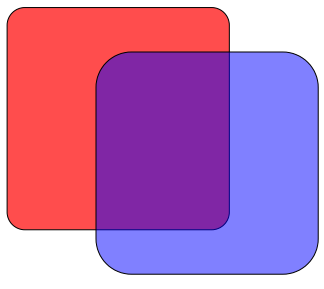
Magics in Colab

Colaboratory shares the notion of magics from Jupyter. There are shorthand annotations that change how a cell's text is executed. To learn more, see Jupyter's magics page.

In [ ]:
```
%%html
<marquee style='width: 30%; color: blue;'><b>Whee!</b></marquee>
```

Whee!

In [ ]:
```
%%html
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 450 400" width="200" height="200">
  <rect x="80" y="60" width="250" height="250" rx="20" style="fill:red; stroke:black; fill-opacity:0.7" />
  <rect x="180" y="110" width="250" height="250" rx="40" style="fill:blue; stroke:black; fill-opacity:0.5;" />
</svg>
```



Interupting Python

Long running python processes can be interrupted. Run the following cell and select **Runtime -> Interrupt execution** (*hotkey: Cmd/Ctrl-M I*) to stop execution.

In [ ]:
```
import time
print("Sleeping")
time.sleep(30) # sleep for a while; interrupt me!
print("Done Sleeping")
```

```
Sleeping
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-15-fb6d98dc021a> in <cell line: 3>()
      1 import time
      2 print("Sleeping")
----> 3 time.sleep(30) # sleep for a while; interrupt me!
      4 print("Done Sleeping")

KeyboardInterrupt:
```

System aliases

Jupyter includes shortcuts for common operations, such as ls:

In [ ]:
```
!ls /bin
```

```
'['
 7z
 7za
 7zr
 aclocal
 aclocal-1.16
 acyclic
 add-apt-repository
 addpart
 addr2line
 aggregate_profile
 apropos
 apt
 apt-add-repository
 apt-cache
 apt-cdrom
 apt-config
 apt-extracttemplates
 apt-ftparchive
 apt-get
 apt-key
 apt-mark
 apt-sortpkgs
 ar
 arch
 as
 asan_symbolize
 asan_symbolize-14
 autoconf
 autoheader
 autom4te
 automake
 automake-1.16
 autoreconf
 autoscan
 autoupdate
 awk
 b2
 b2sum
 base32
 base64
 basename
 basenc
 bash
 bashbug
 bcomps
 bcp
 bjam
 bootctl
 browse
 bunzip2
 busctl
 bzcat
 bzcmp
 bzdiff
 bzegrep
 bzexe
 bzfgrep
 bzgrep
 bzip2
 bzip2recover
 bzless
 bzmore
 c++
 c89
 c89-gcc
 c99
 c99-gcc
 captoinfo
 cat
 catman
 cc
 ccomps
 c++filt
 chage
 chattr
 chcon
 chfn
 chgrp
 chmod
 choom
 chown
 chronic
 chrt
 chsh
 circo
 cksum
 clang
 clang++
 clang++-14
 clang-14
 clang-cpp-14
 clear
 clear_console
 clinfo
 cluster
 mknod
 mktemp
 mm2gv
 more
 mount
 mountpoint
 mpexpand
 mpic++
 mpicc
 mpiCC
 mpicc.openmpi
 mpiCC.openmpi
 mpic++.openmpi
 mpicxx
 mpicxx.openmpi
 mpiexec
 mpiexec.openmpi
 mpif77
 mpif77.openmpi
 mpif90
 mpif90.openmpi
 mpifort
 mpifort.openmpi
 mpijavac
 mpijavac.pl
 mpirun
 mpirun.openmpi
 mv
 mysql_config
 namei
 nawk
 nc-config
 ncurses5-config
 ncurses6-config
 ncursesw5-config
 ncursesw6-config
 neato
 neqn
 networkctl
 newgrp
 nice
 nisdomainname
 nl
 nm
 nns
 nnsd
 nnslog
 nohup
 nop
 nproc
 nroff
 nsenter
 nstat
 numfmt
 objcopy
 objdump
 od
 ogdi-config
 ompi-clean
 ompi_info
 ompi-server
 opalc++
 opalcc
 opal_wrapper
 open
 opencv_annotation
 opencv_interactive-calibration
 opencv_model_diagnostics
 opencv_version
 opencv_visualisation
 opencv_waldboost_detector
 openssl
 ortecc
 orte-clean
 orted
 orte-info
 orterun
 orte-server
 osage
 oshc++
 oshcc
 oshCC
 oshcxx
 oshfort
 oshmem_info
 oshrun
 p7zip
 pack200
 page
 pager
 pandoc
 paperconf
 parallel
 partx
 passwd
 paste
```

```
cmake                                patch
cmp                                  patchwork
col                                  pathchk
colcrt                               pcre2-config
colrm                                pcre-config
column                               pdb3
combine                              pdb3.10
comm                                 pee
compile_et                           peekfd
corelist                             perl
cp                                   perl5.34.0
cpack                                perl5.34-x86_64-linux-gnu
cpan                                 perlbug
cpan5.34-x86_64-linux-gnu            perldoc
cpp                                  perlivp
cpp-11                               perlthanks
c_rehash                             pg_config
csplit                               pgrep
ctest                                pic
ctstat                               piconv
curl                                 pidof
curl-config                          pidwait
cut                                  pinentry
cvtsudoers                           pinentry-curses
dash                                 pinky
date                                 pkaction
dbus-cleanup-sockets                 pkcheck
dbus-daemon                          pkexec
dbus-monitor                         pkg-config
dbus-run-session                     pkgdata
dbus-send                            pkill
dbus-update-activation-environment   pkttyagent
dbus-uuidgen                         pl2pm
dd                                   pldd
debconf                              pmap
debconf-apt-progress                 pngfix
debconf-communicate                  png-fix-itxt
debconf-copydb                       pod2html
debconf-escape                       pod2man
debconf-set-selections               pod2text
debconf-show                         pod2usage
deb-systemd-helper                   podchecker
deb-systemd-invoke                   pprof-symbolize
delpart                              pr
derb                                 preconv
df                                   prename
dh_autotools-dev_restoreconfig       printenv
dh_autotools-dev_updateconfig        printf
diff                                 prlimit
diff3                                profile2mat
diffimg                              protoc
dijkstra                             prove
dir                                  prtstat
dircolors                            prune
dirmngr                              ps
dirmngr-client                       pslog
dirname                              pstree
dmesg                                pstree.x11
dnsdomainname                        pt
domainname                           ptar
dot                                  ptardiff
dot2gxl                              ptargrep
dot_builtins                         ptx
dotty                                pwd
dpkg                                 pwdx
dpkg-architecture                    py3clean
dpkg-buildflags                      py3compile
dpkg-buildpackage                    py3versions
dpkg-checkbuilddeps                  pydoc3
dpkg-deb                             pydoc3.10
dpkg-distaddfile                     pygettext3
dpkg-divert                          pygettext3.10
dpkg-genbuildinfo                    python3
dpkg-genchanges                      python3.10
dpkg-gencontrol                      python3.10-config
dpkg-gensymbols                      python3-config
dpkg-maintscript-helper              qt-faststart
dpkg-mergechangelogs                 quickbook
dpkg-name                            R
dpkg-parsechangelog                  ranlib
dpkg-query                           rbash
dpkg-realpath                        rcp
dpkg-scanpackages                    rdma
dpkg-scansources                     readelf
dpkg-shlibdeps                       readlink
dpkg-source                          realpath
dpkg-split                           rename
dpkg-statoverride                    renice
dpkg-trigger                         reset
dpkg-vendor                          resizepart
dtplite                              resolvectl
du                                   rev
dwp                                  rgrep
echo                                 rlogin
edgepaint                            rm
editor                               rmdir
```

```
egrep
elfedit
enc2xs
encguess
env
eqn
errno
ex
expand
expiry
expr
f77
f95
factor
faillog
fallocate
false
fc-cache
fc-cat
fc-conflist
fc-list
fc-match
fc-pattern
fc-query
fc-scan
fc-validate
fdp
ffmpeg
ffplay
ffprobe
fgrep
file
file-rename
fincore
find
findmnt
fio
fio2gnuplot
fio-btrace2fio
fio-dedupe
fio_generate_plots
fio-genzipf
fio_jsonplus_clat2csv
fio-verify-state
flock
fmt
fold
free
funzip
fuser
fusermount
g++
g++-11
gapplication
gc
gcc
gcc-11
gcc-ar
gcc-ar-11
gcc-nm
gcc-nm-11
gcc-ranlib
gcc-ranlib-11
gcov
gcov-11
gcov-dump
gcov-dump-11
gcov-tool
gcov-tool-11
gdal-config
gdbus
genbrk
gencat
gencfu
gencnval
gendict
genfio
genrb
geos-config
geqn
getconf
getent
getopt
gfortran
gfortran-11
gio
gio-querymodules
git
git-lfs
git-receive-pack
git-shell
git-upload-archive
git-upload-pack
glib-compile-schemas
gmake
gml2gv
rmic
rmid
rmiregistry
routef
routel
rpcgen
rrsync
Rscript
rsh
rsync
rsync-ssl
rtstat
runcon
run-parts
rview
rvim
savelog
sccmap
scp
script
scriptlive
scriptreplay
sdiff
sed
select-editor
sensible-browser
sensible-editor
sensible-pager
seq
serialver
setarch
setpriv
setsid
setterm
sfdp
sftp
sg
sh
sha1sum
sha224sum
sha256sum
sha384sum
sha512sum
shasum
shmemc++
shmemcc
shmemCC
shmemcxx
shmemfort
shmemrun
shred
shuf
size
skill
slabtop
sleep
slogin
snice
soelim
sort
splain
split
sponge
ss
ssh
ssh-add
ssh-agent
ssh-argv0
ssh-copy-id
ssh-keygen
ssh-keyscan
stat
stdbuf
streamzip
strings
strip
stty
su
sudo
sudoedit
sudoreplay
sum
sync
systemctl
systemd
systemd-analyze
systemd-ask-password
systemd-cat
systemd-cgls
systemd-cgtop
systemd-cryptenroll
systemd-delta
systemd-detect-virt
systemd-escape
systemd-id128
systemd-inhibit
```

```
gold
google-pprof
gpasswd
gpg
gpg2
gpg-agent
gpgcompose
gpgconf
gpg-connect-agent
gpgparsemail
gpgsm
gpgsplit
gpgtar
gpgv
gpg-wks-server
gpg-zip
gpic
gprof
graphml2gv
grep
gresource
groff
grog
grops
grotty
groups
gsettings
gtbl
gtk-update-icon-cache
gunzip
gv2gml
gv2gxl
gvcolor
gvgen
gvmap
gvmap.sh
gvpack
gvpr
gxl2dot
gxl2gv
gzexe
gzip
h2ph
h2xs
h5c++
h5cc
h5fc
hardlink
hd
head
helpztags
hexdump
hostid
hostname
hostnamectl
i386
iconv
icuexportdata
icuinfo
id
ifdata
ifnames
ifne
infocmp
infotocap
inspect
install
instmodsh
ionice
ip
ipcmk
ipcrm
ipcs
ischroot
isutf8
jaotc
jar
jarsigner
java
javac
javadoc
javap
jcmd
jdb
jdeprscan
jdeps
jexec
jfr
jhsdb
jimage
jinfo
jjs
jlink
jmap
jmod
join
systemd-machine-id-setup
systemd-mount
systemd-notify
systemd-path
systemd-run
systemd-socket-activate
systemd-stdio-bridge
systemd-sysext
systemd-sysusers
systemd-tmpfiles
systemd-tty-ask-password-agent
systemd-umount
tabs
tac
tail
tar
taskset
tbl
tcldocstrip
tclsh
tclsh8.6
tcltk-depends
tee
tempfile
test
tic
timedatectl
timeout
tload
tmux
toe
top
touch
tput
tr
tred
troff
true
truncate
ts
tset
tsort
tty
twopi
tzselect
ucf
ucfq
ucfr
uclampset
uconv
ul
ulockmgr_server
umount
uname
uncompress
unexpand
unflatten
uniq
unlink
unlzma
unpack200
unrar
unrar-nonfree
unshare
unxz
unzip
unzipsfx
update-alternatives
update-mime-database
uptime
users
utmpdump
vdir
vi
vidir
view
vim
vim.basic
vimdiff
vimdot
vimtutor
vipe
vmstat
w
wall
watch
watchgnupg
wc
wdctl
wget
whatis
whereis
which
which.debianutils
who
whoami
```

```
journalctl                      wish
jps                             wish8.6
jrunscript                      write
jshell                          write.ul
json_pp                         X11
jstack                          x86_64
jstat                           x86_64-linux-gnu-addr2line
jstatd                          x86_64-linux-gnu-ar
kbxutil                         x86_64-linux-gnu-as
kernel-install                  x86_64-linux-gnu-c++filt
keyring                         x86_64-linux-gnu-cpp
keytool                         x86_64-linux-gnu-cpp-11
kill                            x86_64-linux-gnu-dwp
killall                         x86_64-linux-gnu-elfedit
kmod                            x86_64-linux-gnu-g++
krb5-config                     x86_64-linux-gnu-g++-11
krb5-config.mit                 x86_64-linux-gnu-gcc
last                            x86_64-linux-gnu-gcc-11
lastb                           x86_64-linux-gnu-gcc-ar
lastlog                         x86_64-linux-gnu-gcc-ar-11
lcf                             x86_64-linux-gnu-gcc-nm
lckdo                           x86_64-linux-gnu-gcc-nm-11
ld                              x86_64-linux-gnu-gcc-ranlib
ld.bfd                          x86_64-linux-gnu-gcc-ranlib-11
ldd                             x86_64-linux-gnu-gcov
ld.gold                         x86_64-linux-gnu-gcov-11
lefty                           x86_64-linux-gnu-gcov-dump
less                            x86_64-linux-gnu-gcov-dump-11
lessecho                        x86_64-linux-gnu-gcov-tool
lessfile                        x86_64-linux-gnu-gcov-tool-11
lesskey                         x86_64-linux-gnu-gfortran
lesspipe                        x86_64-linux-gnu-gfortran-11
lexgrog                         x86_64-linux-gnu-gold
libnetcfg                       x86_64-linux-gnu-gprof
libpng16-config                 x86_64-linux-gnu-ld
libpng-config                   x86_64-linux-gnu-ld.bfd
link                            x86_64-linux-gnu-ld.gold
linux32                         x86_64-linux-gnu-lto-dump-11
linux64                         x86_64-linux-gnu-nm
ln                              x86_64-linux-gnu-objcopy
lneato                          x86_64-linux-gnu-objdump
lnstat                          x86_64-linux-gnu-pkg-config
locale                          x86_64-linux-gnu-python3.10-config
locale-check                    x86_64-linux-gnu-python3-config
localectl                       x86_64-linux-gnu-ranlib
localedef                       x86_64-linux-gnu-readelf
logger                          x86_64-linux-gnu-size
login                           x86_64-linux-gnu-strings
loginctl                        x86_64-linux-gnu-strip
logname                         x86_64-pc-linux-gnu-pkg-config
look                            xargs
ls                              xauth
lsattr                          xdg-desktop-icon
lsblk                           xdg-desktop-menu
lsb_release                     xdg-email
lscpu                           xdg-icon-resource
lsipc                           xdg-mime
lslocks                         xdg-open
lslogins                        xdg-screensaver
lsmem                           xdg-settings
lsmod                           xml2-config
lsns                            xsel
lsof                            xsubpp
lspgpot                         xxd
lto-dump-11                     xz
lzcat                           xzcat
lzcmp                           xzcmp
lzdiff                          xzdiff
lzegrep                         xzegrep
lzfgrep                         xzfgrep
lzgrep                          xzgrep
lzless                          xzless
lzma                            xzmore
lzmainfo                        yes
lzmore                          ypdomainname
m4                              zcat
make                            zcmp
makeconv                        zdiff
make-first-existing-target      zdump
man                             zegrep
mandb                           zfgrep
manpath                         zforce
man.REAL                        zgrep
man-recode                      zip
mawk                            zipcloak
mcookie                         zipdetails
md5sum                          zipgrep
md5sum.textutils                zipinfo
mesg                            zipnote
migrate-pubring-from-classic-gpg zipsplit
mingle                          zless
mispipe                         zmore
mkdir                           znew
mkfifo                          zrun
```

That `!ls` probably generated a large output. You can select the cell and clear the output by either:

1. Clicking on the clear output button (x) in the toolbar above the cell; or
2. Right clicking the left gutter of the output area and selecting "Clear output" from the context menu.

Execute any other process using `!` with string interpolation from python variables, and note the result can be assigned to a variable:

In [ ]:
```
# In https://github.com/ipython/ipython/pull/10545, single quote strings are ignored
message = 'Colaboratory is great!'
foo = !unset message && echo -e '{message}\n{message}\n'$message"\n$message"
foo
```

Out[ ]:
```
['Colaboratory is great!',
 'Colaboratory is great!',
 'Colaboratory is great!',
 'Colaboratory is great!']
```

Automatic completions and exploring code

Colab provides automatic completions to explore attributes of Python objects, as well as to quickly view documentation strings. As an example, first run the following cell to import the `numpy` module.

In [ ]:
```
import numpy as np
```

If you now insert your cursor after `np` and press **Period**(`.`), you will see the list of available completions within the `np` module. Completions can be opened again by using **Ctrl+Space**.

In [ ]:
```
np
```

Out[ ]:
```
<module 'numpy' from '/usr/local/lib/python3.10/dist-packages/numpy/__init__.py'>
```

If you type an open parenthesis after any function or class in the module, you will see a pop-up of its documentation string:

In [ ]:
```
np.ndarray
```

Out[ ]:
```
numpy.ndarray
```

The documentation can be opened again using **Ctrl+Shift+Space** or you can view the documentation for method by mouse hovering over the method name.

When hovering over the method name the `Open in tab` link will open the documentation in a persistent pane. The `View source` link will navigate to the source code for the method.

Exception Formatting

Exceptions are formatted nicely in Colab outputs:

In [ ]:
```
x = 1
y = 4
z = y/(1-x)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-23-d93e730d8440> in <cell line: 3>()
      1 x = 1
      2 y = 4
----> 3 z = y/(1-x)

ZeroDivisionError: division by zero
```

Rich, interactive outputs

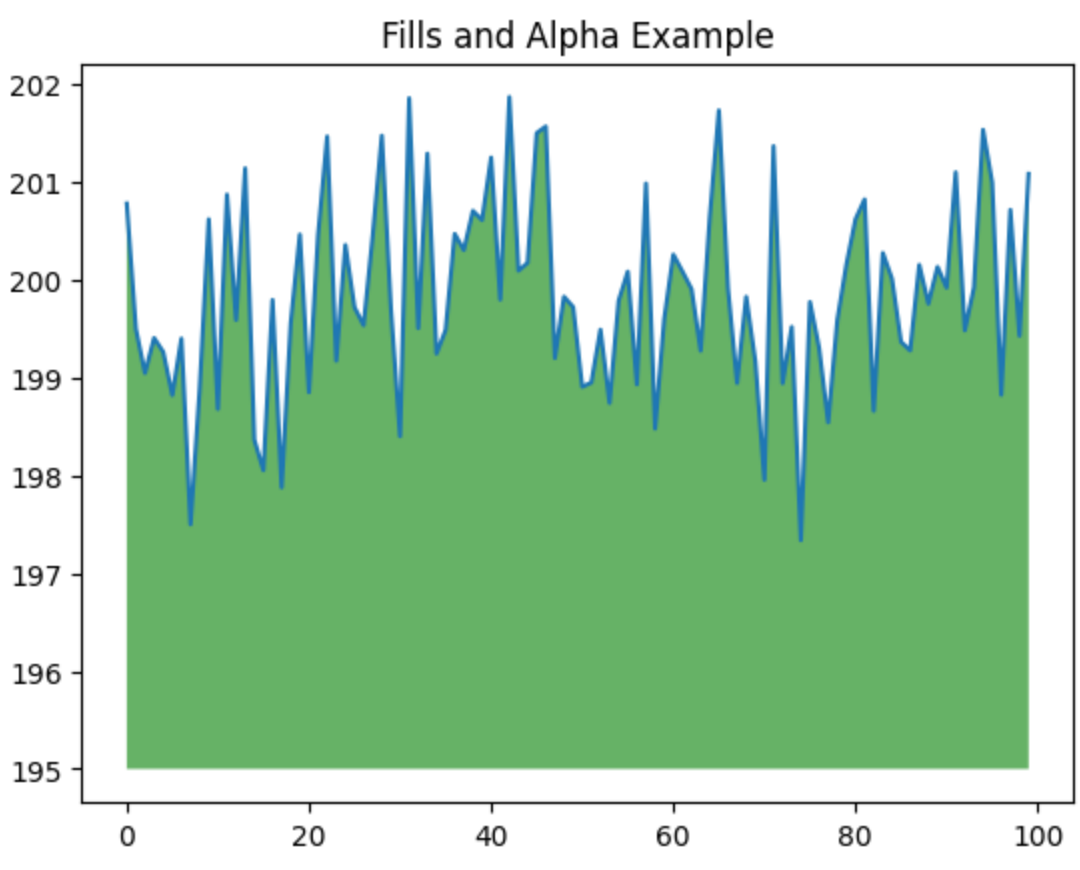
Until now all of the generated outputs have been text, but they can be more interesting, like the chart below.

In [ ]:
```
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Fills and Alpha Example")
plt.show()
```



Integration with Drive

Colaboratory is integrated with Google Drive. It allows you to share, comment, and collaborate on the same document with multiple people:

- The **SHARE** button (top-right of the toolbar) allows you to share the notebook and control permissions set on it.
- **File->Make a Copy** creates a copy of the notebook in Drive.
- **File->Save** saves the File to Drive. **File->Save and checkpoint** pins the version so it doesn't get deleted from the revision history.
- **File->Revision history** shows the notebook's revision history.

Commenting on a cell

You can comment on a Colaboratory notebook like you would on a Google Document. Comments are attached to cells, and are displayed next to the cell they refer to. If you have **comment-only** permissions, you will see a comment button on the top right of the cell when you hover over it.

If you have edit or comment permissions you can comment on a cell in one of three ways:

1. Select a cell and click the comment button in the toolbar above the top-right corner of the cell.
2. Right click a text cell and select **Add a comment** from the context menu.
3. Use the shortcut **Ctrl+Shift+M** to add a comment to the currently selected cell.

You can resolve and reply to comments, and you can target comments to specific collaborators by typing *+[email address]* (e.g., `+user@domain.com`). Addressed collaborators will be emailed.

The Comment button in the top-right corner of the page shows all comments attached to the notebook.

Handy Commands

In [ ]:
```
%matplotlib?
```

`%run` Runs an external script file as part of the cell being executed. For example, if %run myscript.py appears in a code cell, myscript.py will be executed by the kernel as part of that cell.

`%timeit` Counts loops, measures and reports how long a code cell takes to execute.

`%writefile` Save the contents of a cell to a file. For example, `%savefile myscript.py` would save the code cell as an external file called myscript.py.

`%store` Save a variable for use in a different notebook.

`%pwd` Print the directory path you're currently working in.

`%%javascript` Runs the cell as JavaScript code.

Writing Math Equations with Latex

en.wikibooks.org/wiki/LaTeX/Mathematics

The `\frac` , `\dfrac` , and `\tfrac` commands. The `\frac` command takes two arguments numerator and denominator and typesets them in normal fraction form. Use `\dfrac` or `\tfrac` to overrule LATEX's guess about the proper size to use for the fraction's contents `(t = text style, d = display style)` .

```
\begin{equation}
\frac{1}{k}\log_2c(f),\quad\dfrac{1}{k}\log_2 c(f),
```

```
\end{equation}
```

$$\frac{1}{k}\log_2 c(f), \quad \frac{1}{k}\log_2 c(f), \tag{1}$$

Text Alignment

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}} \tag{2}$$

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}} \tag{3}$$

This produces better-looking results than straightforward use of \frac. Left or right placement of any of the numerators is accomplished by using \cfrac[l] or \cfrac[r] instead of \cfrac.

Text Alignment

$$\begin{aligned} a+b+c+d+e+f \\ &+i+j+k+l+m+n \\ &&+o+p+q+r+s \end{aligned} \tag{4}$$

$$a_1 = b_1 + c_1 \tag{5}$$
$$a_2 = b_2 + c_2 - d_2 + e_2 \tag{6}$$

$$a_1 = b_1 + c_1 \tag{7}$$
$$a_2 = b_2 + c_2 - d_2 + e_2 \tag{8}$$

$$\begin{aligned} a_{11} &= b_{11} & a_{12} &= b_{12} \\ a_{21} &= b_{21} & a_{22} &= b_{22} + c_{22} \end{aligned} \tag{9}$$
$$\tag{10}$$

$$\begin{aligned} a_1 &= b_1 + c_1 & &+ e_1 - f_1 \\ a_2 &= b_2 + c_2 - d_2 + e_2 \end{aligned} \tag{11}$$
$$\tag{12}$$

$$\begin{aligned} a_{11} &= b_{11} & a_{12} &= b_{12} \\ a_{21} &= b_{21} & a_{22} &= b_{22} + c_{22} \end{aligned} \tag{13}$$
$$\tag{14}$$

```
$$
  \int_0^\infty \frac{x^3}{e^x-1}\,dx = \frac{\pi^4}{15}
$$
```

$$\int_0^\infty \frac{x^3}{e^x-1}\,dx = \frac{\pi^4}{15}$$

```
$y=x^2$

$e^{i\pi} + 1 = 0$
```

> $y = x^2$

> $e^{i\pi} + 1 = 0$

```
$e^x=\sum_{i=0}^\infty \frac{1}{i!}x^i$

$\frac{n!}{k!(n-k)!} = {n \choose k}$
```

> $e^x = \sum_{i=0}^{\infty} \frac{1}{i!}x^i$

> $\frac{n!}{k!(n-k)!} = \binom{n}{k}$

```
$A_{m,n} =
 \begin{pmatrix}
  a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
  a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
  \vdots  & \vdots  & \ddots & \vdots  \\
  a_{m,1} & a_{m,2} & \cdots & a_{m,n}
 \end{pmatrix}$
```

> $$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

```
\begin{equation}
\frac{1}{k}\log_2 c(f),\quad\dfrac{1}{k}\log_2 c(f),
\quad\tfrac{1}{k}\log_2 c(f)
\end{equation}
```

$$\frac{1}{k}\log_2 c(f), \quad \dfrac{1}{k}\log_2 c(f), \quad \tfrac{1}{k}\log_2 c(f) \tag{15}$$

```
\begin{equation}
V=\frac{dM}{dx}
\end{equation}
```

$$V = \frac{dM}{dx} \tag{16}$$

8.4. The **\sideset** command. There's also a command called \sideset, for a rather special purpose: putting symbols at the subscript and superscript corners of a symbol l

```
\begin{equation}\sideset{}{'}
\sum_{n<k,\;\text{$n$ odd}} nE_n\end{equation}
```

$$\sum_{n<k,\;\text{n odd}}{}' nE_n \tag{17}$$

```
$$
  \int_0^\infty \frac{x^3}{e^x-1}\,dx = \frac{\pi^4}{15}
$$
```

$$\int_0^\infty \frac{x^3}{e^x-1}\,dx = \frac{\pi^4}{15}$$

$\sum$ `\sum` $\prod$ `\prod` $\coprod$ `\coprod`

$\bigodot$ `\bigodot` $\bigoplus$ `\bigoplus` $\bigcup$ `\bigcup`

$\bigotimes$ `\bigotimes` $\bigcap$ `\bigcap` $\biguplus$ `\biguplus`

$\bigsqcup$ `\bigsqcup` $\bigvee$ `\bigvee` $\bigwedge$ `\bigwedge`

$\int$ `\int` $\oint$ `\oint` $\iint$ `\iint` $\iiint$ `\iiint` $\iiiint$ `\iiiint`

$\int\cdots\int$ `\idotsint` $\cdots$ `\cdots`

$\exists$ `\exists` $\nexists$ - `\nexists`

$\rightarrow$ `\rightarrow` $\Rightarrow$ `\Rightarrow` $\implies$ `\implies` $\mapsto$ `\mapsto`

$\leftarrow$ `\leftarrow` $\impliedby$ `\impliedby`

$\rightleftharpoons$ `\rightleftharpoons` $\Leftrightarrow$ `\Leftrightarrow` $\iff$ `\iff`

$\forall$ `\forall` $\lor$ `\lor` $\land$ `\land`

$\leq$ `\leq` $\geq$ `\geq` $\ll$ `\ll` $\gg$ `\gg`

$\neq$ `\neg` $\doteq$ `\doteq` $\approx$ `\approx` $\neq$ `\neq` $\sim$ `\sim`

$\top$ `\top` $\bot$ `\bot`

$\perp$ `\perp` $\vdash$ `\vdash` $\dashv$ `\dashv`

$\angle$ `\angle` $\emptyset$ `\emptyset` $\varnothing$ `\varnothing`

$\parallel$ `\parallel` $\nparallel$ `\nparallel`

$\asymp$ `\asymp` $\bowtie$ `\bowtie`

$\equiv$ `\equiv` $\propto$ `\propto`

$\cap$ `\cap` $\cup$ `\cup` $\supset$ `\supset` $\ni$ `\ni`

$\in$ `\in` $\notin$ `\notin` $\subset$ `\subset`

$\subset$ `\subset` $\supset$ `\supset`

$\in$ `\in` $\ni$ `\ni` $\subseteq$ `\subseteq` $\supseteq$ `\supseteq` $\cong$ `\cong`

$\smile$ `\smile` $\frown$ `\frown`

$\nsubseteq$ `\nsubseteq` $\nsupseteq$ `\nsupseteq` $\simeq$ `\simeq` $\models$ `\models` $\notin$ `\notin`

$\sphericalangle$ `\sphericalangle` $\measuredangle$ `\measuredangle` $\therefore$ `\therefore` $\because$ `\because`

∂ `\partial` $\imath$ `\imath` $\Re$ `\Re` ∇ `\nabla` $\jmath$ `\jmath` $\Im$ `\Im` ∞ `\infty`

$\mathbb{R}^2$ `$\mathbb{R^2}$`

- Github Markdown basics
- Github flavored Markdown
- Original Markdown spec: Syntax
- Original Markdown spec: Basics
- marked.js library used by Colab
- LaTex mathematics for equations

Introduction to LaTeX

In this tutorial, you will learn some of the basics on how to use $\LaTeX$ to display equations in Jupyter notebooks. For looking up symbols you may need, you can use any of the many cheat sheets you can find by asking Google. I have provided a few that will come up often in this course at the end of this lesson.

In this lesson, whenever a LaTeX expression is shown, the raw Markdown/LaTeX is shown beneath it. (The word LaTeX is generally stylized as $\LaTeX$, but I get tired of reading that, so going forward, I will just write "LaTeX.")

Basic inline LaTeX

To embed LaTeX within text, simply encapsulate the LaTeX portions in dollar signs (`$`). MathJax takes care of the rest. As an example, consider the sentence below and the markdown/LaTeX code to render it.

Einstein told us that $E = mc^2$.

```
Einstein told us that $E = mc^2$.
```

Notice how the equation is properly rendered, with mathematical variables in italics. Note also how `^2` was used to raise to a power. If the power has more than one character in it, it should be enclosed in braces (`{}`). In fact, braces are used to generally group symbols in LaTeX.

Euler told us that $\mathrm{e}^{i\pi} - 1 = 0$.

```
Euler told us that $\mathrm{e}^{i \pi} - 1 = 0$.
```

Aside from the grouping braces, there are several other syntactical items of note. First, notice that I made the special character π with `\pi` . In general, a backward slash precedes special symbols or commands in LaTeX. If we want another Greek letter, like θ, we use `\theta` . Now, also note that I used " `\mathrm{e}` " for the base of the natural logarithm. I was signaling to LaTeX that I wanted the character written in Roman font, and not italics, so I used `\mathrm` . Anything in the braces following the function `\mathrm` is rendered in Roman font. Note the difference.

This is e. This is e

```
This is $e$. This is $\mathrm{e}$.
```

Now, back to grouping things in braces. We can do similar groupings using braces with with subscripts.

The dot product of two n-vectors is $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$.

```
The dot product of two $n$-vectors is $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i$.
```

Here, I have used `$\mathbf{a}$` to make the character `a` boldface, denoting a vector. Note that we denote subscripts with an underscore. Notice also that the bounds of the sum use the same underscore and caret notation as for subscripts and superscripts.

Displaying equations on separate lines

The bounds on the summation in the above example may look a little funny to you because they are not above and below the summation symbol. This is because this particular equation is written inline. If we had separated it from the text, it renders differently.

We can make an equation appear centered on a new line, like

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i. \tag{18}$$

```
We can make an equation appear centered on a new line, like

\begin{align}
    \mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i.
\end{align}
```

The `align` environment in LaTeX specifies that you want centered equations, separated from the text. It is called `align` because it allows you to align the equations. You separate lines in the equations with a double backslash (`//`). Insert an ampersand (`&`) in each line at the alignment point. All equations will be aligned at the location of the ampersand symbols (and, of course, the ampersands will not appear in the rendered equations).

For a three-vector consisting of x, y, and z components,

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i \tag{19}$$

$$= a_x b_x + a_y b_y + a_z b_z. \tag{20}$$

```
For a three-vector consisting of $x$, $y$, and $z$ components,

\begin{align}
    \mathbf{a} \cdot \mathbf{b} &= \sum_{i=1}^n a_i b_i \\
    &= a_x b_x + a_y b_y + a_z b_z.
\end{align}
```

Note that I always put an extra blank line before the `\begin{align}` statement. This is not necessary, but I think things look better with the extra space.

Fractions (and an example of fine-tuning)

To display fractional quantities, we use the `\frac{}{}` command. `\frac` is always followed by two sets of braces; the numerator is contained in the first, and the denominator is contained in the second. As an example, we can write an equation you will become intimately familiar with if you take the second term of this course,

$$P(A \mid B) = \frac{P(B \mid A) \, P(A)}{P(B)} \tag{21}$$

```
\begin{align}
    P(A \mid B) = \frac{P(B \mid A) \, P(A)}{P(B)}
\end{align}
```

The right hand side has a nicely-formatted fraction. I did a little extra fine-tuning in this equation. I'll show the equation again without the fine-tuning, which used the `\mid` and `\,` commands.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{22}$$

```
\begin{align}
    P(A | B) = \frac{P(B | A) P(A)}{P(B)}.
\end{align}
```

First, the `\mid` command should be used in conditional probabilities. Just using a vertical bar (`|`) results in crowding. Similarly, I used the `\,` command to insert a little extra space between the two probabilities in the numerator. This makes the equation a bit easier to read. This `\,` operator is especially important when defining integrals. We can put a little space between the $\mathrm{d}x$ and the integrand.

$$\text{good: } \int_0^{2\pi} \mathrm{d}x \, \sin x. \tag{23}$$

$$\text{bad: } \int_0^{2\pi} \mathrm{d}x \sin x. \tag{24}$$

```
\begin{align}
    \text{good: } &\int_0^{2\pi} \mathrm{d}x \, \sin x. \\[1em]
    \text{bad: } &\int_0^{2\pi} \mathrm{d}x \sin x.
\end{align}
```

Note that I inserted extra space after the new line. Specifically, `\\[1em]` instructs LaTeX to insert a space equation to the width of an M character between the equations. I often do this to keep things clear.

It is also very important to note that I used $\sin$ and not sin. Mathematical functions should be in Roman font and are invoked with a backslash. Otherwise, the characters are interpreted as separate variables. To be clear:

$$\text{good: } \sin x. \tag{25}$$

$$\text{bad: } sinx. \tag{26}$$

```
\begin{align}
    \text{good: } &\sin x. \\[1em]
    \text{bad: } & sin x.
\end{align}
```

Finally, notice that I was able to put text in the equation like this: `\text{good: }`.

Grouping operators (and more fine-tuning)

Compare the following equations.

$$\text{good: } \sum_{i=1}^{n} i^3 = \left(\sum_{i=1}^{n} i \right)^2. \tag{27}$$

$$\text{bad: } \sum_{i=1}^{n} i^3 = (\sum_{i=1}^{n} i)^2. \tag{28}$$

```
\begin{align}
    \text{good: } &\sum_{i=1}^n i^3 = \left(\sum_{i=1}^n i\right)^2. \\[1em]
    \text{bad: }  &\sum_{i=1}^n i^3 = (\sum_{i=1}^n i)^2.
\end{align}
```

In the second equation, I did not use the `\left(` and `\right)` construction for parentheses and the result looks pretty awful. In LaTeX, the height of anything that is encapsulated by `\left(` and `\right)` scales the parentheses appropriately. You can use `\left` and `\right` with many symbols. An important example is `\left\{`. Note that to display braces in an equation, you have to use `\{` because just a plain brace (`{`) has a different meaning.

(By the way, that equation is true, and pretty amazing. It says that the sum of the first n *cubes* of integers is equal to the sum of the first n integers *squared*!)

Finally, if you use `\left.` or `\right.`, LaTeX will simply scale the opposite symbol to match the height of the text, but will suppress printing the other. For example,

$$\frac{1}{x+2}\Bigg|_0^2 = -\frac{1}{4}. \tag{29}$$

```
\begin{align}
    \left. \frac{1}{x + 2} \right|_0^2 = -\frac{1}{4}.
\end{align}
```

This is also useful if you are going to use `/` for a division operation. Compare the following.

$$\text{good: } x^2/y^2 \tag{30}$$

$$\text{bad: } x^2/y^2 \tag{31}$$

```
\begin{align}
    \text{good: } & \left. x^2 \middle/ y^2 \right. \\[1em]
    \text{bad: } & x^2 / y^2
\end{align}
```

Here, we used the `\middle` operator to scale the length of the division sign.

Matrices and arrays

On occasion, you'll need to express matrices. This is most easily done using the `pmatrix` environment. For example, a covariance matrix for two variables might be written as

$$\sigma^2 = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}. \tag{32}$$

```
\begin{align}
    \sigma^2 = \begin{pmatrix}
    \sigma_1^2 & \sigma_{12} \\
    \sigma_{12} & \sigma_2^2
    \end{pmatrix}.
\end{align}
```

Once in the `pmatrix` environment, each row has entries separated by an ampersand. The row ends with a `\\`. Each row must have the same number of entries.

You may also need to represent an values stacked on top of each other. For example, we might specify a piecewise linear function like this.

$$\text{rectifier}(x) = \begin{cases} 0 & x \le 0 \\ x & x > 0. \end{cases} \tag{33}$$

```
\begin{align}
    \text{rectifier}(x) = \left\{
    \begin{array}{cl}
    0 & x \le 0 \\
    x & x > 0.
    \end{array}
    \right.
\end{align}
```

The `array` environment allows arrays of text. The `{cl}` after `\begin{array}` indicates that two columns are wanted, with the first column being centered and the second being left-aligned. If we chose instead `{lr}`, the first column is left-aligned and the second column is right-aligned.

Useful LaTeX symbols for BE/Bi 103

Following is a list of some symbols you may find useful in this class.

LaTeX	symbol
`\approx`	$\approx$
`\sim`	$\sim$

LaTeX	symbol
`\propto`	$\propto$
`\le`	$\le$
`ge`	$\ge$
`\pm`	$\pm$
`\in`	$\in$
`\ln`	$\ln$
`\exp`	$\exp$
`\prod_{i\in D}`	$\prod_{i\in D}$
`\sum_{i\in D}`	$\sum_{i\in D}$
`\frac{\partial f}{\partial x}`	$\frac{\partial f}{\partial x}$
`\sqrt{x}`	$\sqrt{x}$
`\bar{x}`	$\bar{x}$
`\langle x \rangle`	$\langle x \rangle$
`\left\langle \frac{x}{y} \right\rangle`	$\left\langle \frac{x}{y} \right\rangle$