Cheatsheet

BASH

- ❑ **$ ls -Flag [location]** — list content of specified location, using specified flags
- ❑ **$ pwd** — print working directory → current location in filesystem
- ❑ **$ cd [location]** — change directory to specified location, relative paths work
  - ❑ ***. and ..*** — special characters denoting *here* and *directory above*
  - ❑ ***~ and -*** — special characters denoting *HOME* and *previous directory*
- ❑ **$ mkdir [name]** — make directory with specified name (can include paths)
- ❑ **$ nano [filename]** — open specified file using the *nano* text editor
  - ❑ **CTRL-O *then* <Enter>** — *nano* command saving content of file
  - ❑ **CTRL-X** — *nano* command to close file (asks for confirmation if file changed)
- ❑ **$ touch [filename]** — creates empty file with specified name if file does not exist

GIT - initialise and track

- ❑ **$ git init** — initiate the repository
- ❑ **$ git status** — check the current state of the repository
- ❑ **$ git add [filename]** — stage a file / changes made to a file
- ❑ **$ git commit -m "message"** — commit staged files / changes
- ❑ **$ git commit --amend** — amend last commit
- ❑ **$ git rm [filename]** — delete a file and stage the change
- ❑ **$ git rm --cached [filename]** — remove a file from tracking
- ❑ **$ .gitignore** — listed files and folders will not be tracked

GIT - explore history and revert

- ❑ **$ git log** — show a log of the commit history
- ❑ **$ git log --oneline** — show commit history with one line per commit
- ❑ **$ git diff [filename]** — compare current, unstaged file to latest commit
- ❑ **$ git diff --staged** — compare staged file(s) to the last commit
- ❑ **$ git diff [commit] [commit]** — comparing two commits using unique identifiers
- ❑ **$ git checkout [commit] [file]** — roll (specified file) back to specific commit
- ❑ **$ HEAD** — denotes the latest commit
- ❑ **$ HEAD~i** — denotes the *i*th commit before the last

GIT - remote repositories

- ❑ **$ git remote add origin [URL]** — link an empty remote repo to your local repo
- ❑ **$ git push origin master** — push your local changes to the remote repo
- ❑ **$ git pull origin master** — pull changes from the remote repo
- ❑ **$ git remote -v** — show nickname and URL of remote repo(s)
- ❑ **$ git clone [URL] [location]** — clone a remote repository to your computer
- ❑ **$ git push -u origin master** — push your local changes to the remote repo and set specified remote as your *upstream* → think of it as setting up the default so now you can update without specifying the remote nickname and branch using $ git push or $ git pull only

Jupyter Notebook Keyboard Shortcuts

The Jupyter Notebook has two different keyboard input modes. **Edit mode** allows you to type code or text into a cell and is indicated by a green cell border. **Command mode** binds the keyboard to notebook level commands and is indicated by a grey cell border with a blue left margin.

Mac OS X modifier keys:

- ⌘ : Command
- ^ : Control
- ⌥ : Option

Command Mode (press Esc to enable)

- F : find and replace
- ↩ : enter edit mode
- ⌘⇧F : open the command palette
- ⌘⇧P : open the command palette
- P : open the command palette
- ⇧↩ : run cell, select below
- ^↩ : run selected cells
- ⌥↩ : run cell and insert below

Edit Mode (press Enter to enable)

- ⌘] : indent
- ⌘[: dedent
- ⌘A : select all
- ⌘Z : undo
- ⌘/ : comment
- ⇧↩ : run cell, select below
- ^↩ : run selected cells
- ⌥↩ : run cell and insert below
- ^⇧Minus : split cell at cursor
- ⌘S : Save and Checkpoint

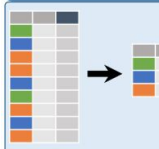
Python Pandas

http://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Handling Missing Data

df.dropna()
Drop rows with any column having NA/null data.
df.fillna(value)
Replace all NA/null data with value.

Group Data



df.groupby(by="col")
Return a GroupBy object, grouped by values in column named "col".

df.groupby(level="ind")
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group.
Additional GroupBy functions:

size()
Size of each group.

agg(*function*)
Aggregate group using function.

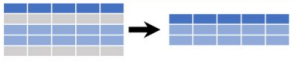
Summarize Data

df['w'].value_counts()
Count number of rows with each unique value of variable
len(df)
# of rows in DataFrame.
df['w'].nunique()
# of distinct values in a column.
df.describe()
Basic descriptive statistics for each column (or GroupBy)

pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

sum()
Sum values of each object.
count()
Count non-NA/null values of each object.
median()
Median value of each object.
quantile([0.25,0.75])
Quantiles of each object.
min()
Minimum value in each object.
max()
Maximum value in each object.
mean()
Mean value of each object.
var()
Variance of each object.
std()
Standard deviation of each object.

Subset Observations (Rows)



df[df.Length > 7]
Extract rows that meet logical criteria.
df.drop_duplicates()
Remove duplicate rows (only considers columns).
df.head(n)
Select first n rows.
df.tail(n)
Select last n rows.

df.sample(frac=0.5)
Randomly select fraction of rows.
df.sample(n=10)
Randomly select n rows.
df.iloc[10:20]
Select rows by position.
df.nlargest(n, 'value')
Select and order top n entries.
df.nsmallest(n, 'value')
Select and order bottom n entries.

Logic in Python (and pandas)			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

Subset Variables (Columns)

df[['width','length','species']]
Select multiple columns with specific names.
df['width'] *or* **df.width**
Select single column with specific name.
df.filter(regex='*regex*')
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples	
'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species$).*'	Matches strings except the string 'Species'

df.loc[:,'x2':'x4']
Select all columns between x2 and x4 (inclusive).
df.iloc[:,[1,2,5]]
Select columns in positions 1, 2 and 5 (first column is 0).
df.loc[df['a'] > 10, ['a','c']]
Select rows meeting logical condition, and only the specific columns .