Cheatsheet

Filesystem

Relative path — specifies a location starting from the current location.
Absolute path — specifies a location from the root of the file system.
/ on its own — is the root directory of the whole file system.
/ or \ — separators for directory names in a path used on Unix and Windows, respectively.

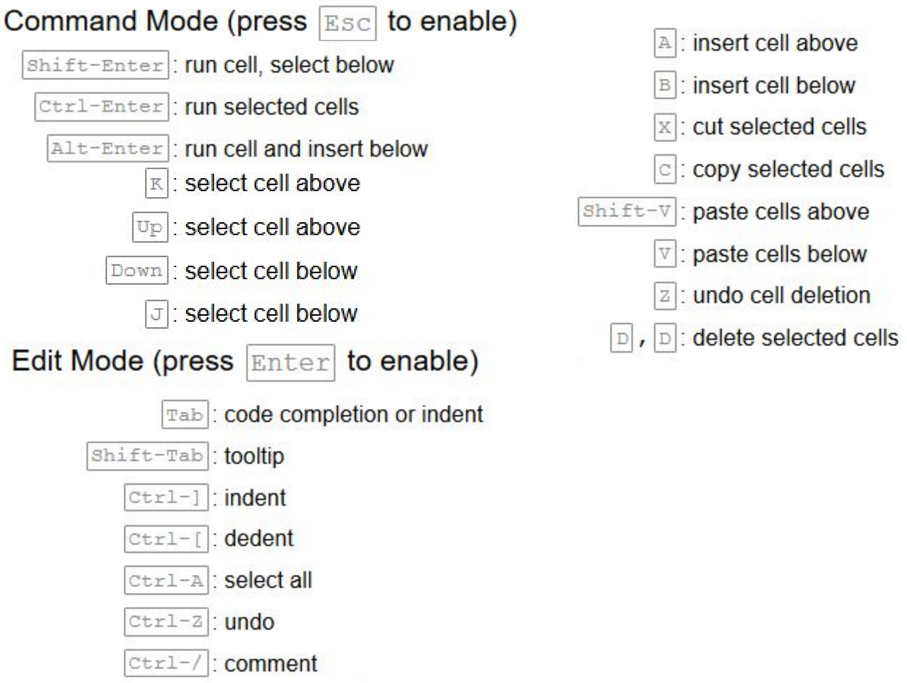
Special directory symbols:

. — this location
.. — the directory above
~ — the current user's home directory, has to be at the start of specified path
- — the previous directory I was in

BASH

- □ **$ ls -Flag [location]** — list content of specified location, using specified flags
- □ **$ dir** — *Windows prompt*: list content of current location
- □ **$ 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 an empty file with specified name if file does not exist

Jupyter Notebook



Command Mode (press Esc to enable)

Shift-Enter : run cell, select below
Ctrl-Enter : run selected cells
Alt-Enter : run cell and insert below
K : select cell above
Up : select cell above
Down : select cell below
J : select cell below

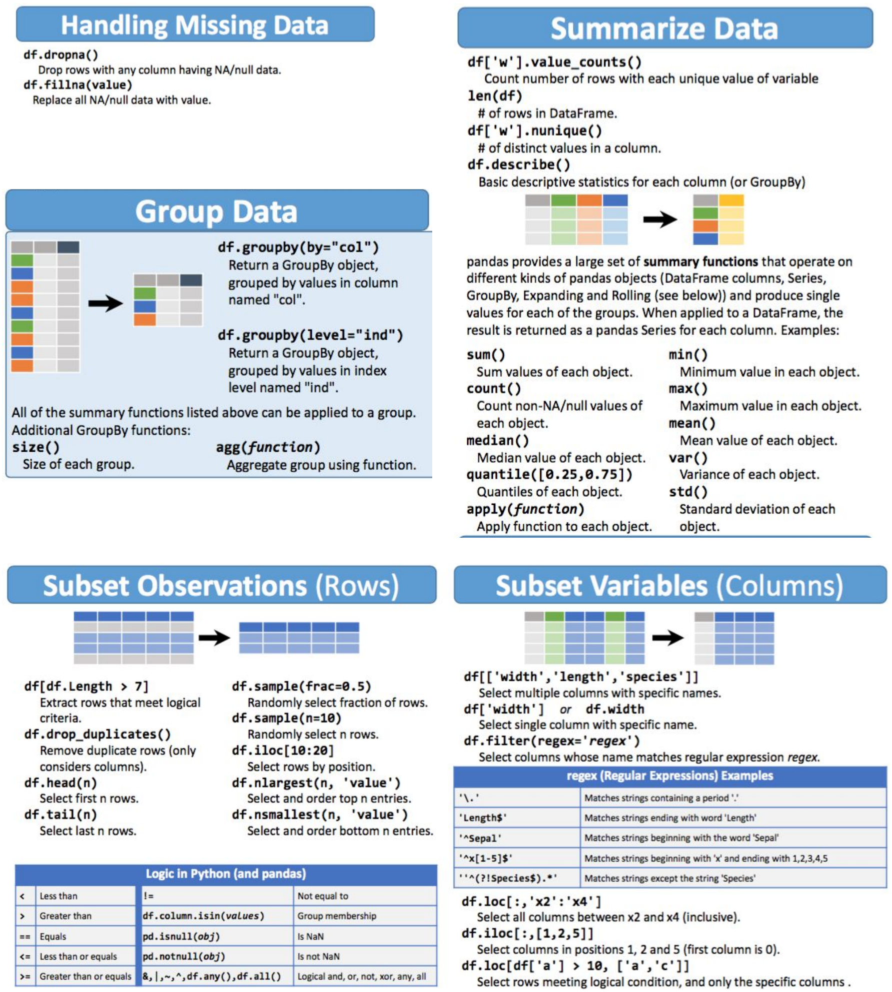
A : insert cell above
B : insert cell below
X : cut selected cells
C : copy selected cells
Shift-V : paste cells above
V : paste cells below
Z : undo cell deletion
D , D : delete selected cells

Edit Mode (press Enter to enable)

Tab : code completion or indent
Shift-Tab : tooltip
Ctrl-] : indent
Ctrl-[: dedent
Ctrl-A : select all
Ctrl-Z : undo
Ctrl-/ : comment

Python Pandas

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



Handling Missing Data

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

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.
apply(*function*)
Apply function to 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.

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.

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 .