Pandas Series (ds) ::

Required Action	Command
Creating Pandas Series	pd.Series([1,2,3,4,5])
Datatype in the Series	ds.dtypes

Pandas Dictionaries (dict) ::

Required Action	Command
To iterate a dictionary And use the key, value pairs	for key in booking_dict: print(key) print(booking_dict[key])

Pandas DataFrames (df) ::

Required Action	Command
DataFrames	
Creating df from dictionary	pd.DataFrame(data = {'x': ['a', 'b'], 'y': [1, 2]}, index = ['i','ii'])
Creating df from a Pandas Series (ds)	pd.DataFrame(ds)
Dataframe Summary	df.info()
Dataframe Type	type(df)
Columnwise Datatypes in the df	df.dtypes
Reindexing df	df.reindex(index = ['A', 'B'])
To get Top 'n' rows of a df	df.head(n)
To get Last 'n' rows of a df	df.tail (n)
To know the shape of the df	df.shape
To get 1 column	df['col_name_1']
To get multiple columns	df[['col_1','col_ 2']]
To select Specified Rows(1,3,5,6)/Columns	df.iloc[[1,3,5,6],[1,3]]
To select rows with condition on col data	df[df.col_1>2]
To count the no. of rows in a df	len(df.axes[0])
To get the rows in a df	df.axes[0] **OR** df.index
To count the no. of columns in a df	len(df.axes[1])
To get the column names in a df	df.axes[1] **OR** df.columns
To change the column names of a df to any row values	df.columns=df.iloc[<row_index_value>] e.g. df.columns=df.iloc[0]
To get rows where the data is missing in a column i.e. is NaN	df[df.col_1.isnull()] **OR** df[df['col_1'].isnull()]
To get rows with column data b/w 2 numbers (inclusive)	df[x.col_1.between(15,20)] **OR** df[df['col_1'].between(15,20)]
To get rows with condition on 2 col. data	df[(df['col_1']<3) & (df['col_1']>15)]

Sum of a column values	df['col_1'].sum() **OR** sum(col_1.attempts)
To remove a row from a df	df.drop(k) where 'k' is the index value/label e.g.df.drop(0)
To drop / remove a col from a df	df = df.drop('col_name', axis = 1)
Sorting a column in df in desc order	df.sort_values(by = ['col_1'],ascending = False)
Sorting a column in df in asc order	df.sort_values(by = ['col_1'],ascending = True)
Sorting multiple columns in a df	df.sort_values(by = ['col_1','col_2'],ascending =[False,True]) **OR** df.sort_values(['route_id','position'])
Replace certain values in a column of a df with new values	df['col_1'] = df[' col_1'].map({'yes': True, 'no': False}) **OR** df[' col_1']=df[' col_1'].replace(to_replace=['yes' ,'no'],value=['True','False'])
Replace a particular value in a col in a df	df['col_1'] = df[' col_1'].replace('James','Suresh')
To delete a column from a df	df.pop('col_1')
To insert a new column in a df with the name 'new_col_name'	df['new_col_name']=['data_1', 'data_2']
Lambda Function to make String Replacement in Series	df['new_col_1'] = df.col_1.apply(lambda x: x.replace('+91',''))
To get 'List of Route IDs'	','.join(map(str,(routes.routes_id.drop_duplicates())))
Filtering rows	df[df.trip_id=135536542] df[df.route_id.isin([242,243,482])]
Ignoring rows with driver id 1255	df[(~(df.driver_id==1255))]
To get rows with a specific route_id and columns from 3rd to 5th columns	df[(df.route_id==242)& (df.columns[2:5])]
Rows from index 10 to 29 with only three specified columns	df[['trip_id, driver_id, booking_id]][10:30]
To append 'df2' on 'df1'	df1.append(df2)
Group by	df.groupby(['zone_id']).trip_id.nunique().reset_index()
To drop duplicates from multiple columns	df[['zone_id','trip_id']].drop_duplicates()
To loop through a range of numbers	for i in range(0,5): print(i)
To define functions	def random_function(): print('I am a random func')
To rename columns	df.rename(columns={'created_time':'booking_time'})
To read 'CSV' file	pd.read_csv('<file name>')
To remotely assign a value in a string	print('The numbers are %s and %s'%(x, y)) where x = 4 and y = 5 Output: *The numbers are 4 and 5*
.loc	
To change the data in a df at a location	df.loc['row_1','column_1'] = 11.5
To add a new row in a df	df.loc['k']=['v1',v2,v3,'v4']
To assign rows basis certain condition	df.loc[df.boarding_time<'14:30:30' , 'session']= 'Morning'
To assign a col while putting a condition on rows	df.loc[user_data['route_name'] == 'xx', 'non_ac'] = 1
To check missing values of children in the table	df.loc[df['who'] == 'child'].isnull().sum()
To iterate a dataframe on index and put conditions on rows in a particular col	for i in df.index: if (float(df.loc[i, 'col_name']) < float(0.2)):

and assign values in a new column basis that condition	df.loc[i, 'new_col_name'] = 'xx' else: df.loc[i, 'new_col_name'] = 'yy'
To access a single value for a row/column pair by integer position	df.iat['row number','col_number']
Datetime	
To change the date time format of a column	df['date']=pd.to_datetime(df['string_date'])
Date of today (post importing dt lib)	pd.to_datetime(dt.date.today())
Date of Yesterday (post importing dt lib)	pd.to_datetime(dt.date.today()) - dt.timedelta(days=1)
To get 'List of Dates'	','.join(map(str,df.dates.unique()))
To add a column with today's date	df['date_today']=dt.datetime.today()
To covert the date col to new format	df['date_today']=pd.to_datetime(df['date_today'])
To get diff. in 'days' between two date col	df['dff']=(df['date_today']-df['date_time']).dt.days
To get diff. in 'time' between two date col	df['dff']=(df['date_today']-df['date_time']).dt.time
To get the date and time of earlier date	df(['a_day_before']) =df(['date_time']).dt.timedelta(days=60)
To convert a date column to datetime to put conditions on rows basis 'datetime' *(Note: Use '\' for new line)*	df.loc[(pd.to_datetime(user_data['trip_date']) >= '2019-08-27 00:00:00')&\ (pd.to_datetime(user_data['trip_date']) <= '2019-09-20 23:59:59')][['booking_id','booking_date']]
To convert datatime to the required format	df['trip_date'][0].date().strftime('%d-%m-%y')
To apply the reformatted date format to a col	df['trip_date_reformat'] = df.trip_date.apply(lambda x : x.date().strftime('%y-%m-%d'))
To convert a string to datetime	dt.datetime.strptime(df['xxx'][0], "%Y-%m-%d")
To convert timedelta to int (i.e. days, seconds etc.)	dt.timedelta(days=24).days ➔ Output: 24 dt.timedelta(days=24).total_seconds() ➔ Output: 2073600.0
For more info	https://www.journaldev.com/23365/python-string-to-datetime-strptime
Merge	
To merge df_1 and df_2 on a 'col_1'	new_df=df_1.merge(df_2,how='left',on=['col_1'])
To merge basis multiple columns and column names are different in 2 dfs	new_df=df_1.merge(df_2, how='left', left_on=['col_1', 'col_2'], right_on=['col_3','col_4'])
To merge only a few columns of a df *(Note: columns within 'right_on' conditions have to be there in the few columns to be merged)*	new_df=df_1.merge(df_2[['col_3', 'col_4', 'col_a', 'col_b']], how='left', left_on=['col_1', 'col_2'], right_on=['col_3','col_4'])
To put specific suffixes if certain column names in the 2 dfs are the same *(default is '_x' and '_y')*	new_df=df_1.merge(df_2[['col_3', 'col_4', 'col_a', 'col_b']], how='left', left_on=['col_1', 'col_2'], right_on=['col_3','col_4'], suffixes = ('_pickup','_drop'))
Pivot Tables & Group By	

To create a 'Pivot Table' with col_1 & col_2 as index and col_3 & col_4 as values	pd.pivot_table(df, index = ['col_1', 'col_2'], values = ['col_3', 'col_4'], aggfunc = np.sum)
Another way to create a 'Pivot Table'	df.pivot_table(index = ['col_1', 'col_2'], values = ['col_3', 'col_4'], aggfunc = np.sum)
Another way to create a 'Pivot Table' *(default aggregation function is 'average')*	df.pivot_table('col_5', index = ['col_1', 'col_2'], columns = ['col_3', 'col_4'])
Another way to create a 'Pivot Table'	df.pivot_table(index = ['col_1', 'col_2'], columns = ['col_3', 'col_4'], aggfunc = {'col_5':'sum', 'col_6':'mean'})
To add the 'Total' of each 'entire column' at the end of the 'Pivot Table'	pd.pivot_table(df, index = ['col_1', 'col_2'], values = ['col_3', 'col_4'], aggfunc = np.sum, fill_value = 0, margins = True)
To find data in the table with a specific condition **OR** To query the output pivot	table = pd.pivot_table(df, ……..) table.query('Manager == ["Douglas"]')
To find the 'maximum' sales value of the items	pd.pivot_table(df, index = 'Item', values = 'Sale_amt', aggfunc = np.max)
To find the 'minimum' sales value of the items	pd.pivot_table(df, index = 'Item', values = 'Sale_amt', aggfunc = np.min)
To find the 'maximum' and 'minimum' sales value of the items	pd.pivot_table(df, index = 'Item', values = 'Sale_amt', aggfunc = [np.max, np.min])
To create a pivot table mentioning 'rows' and 'columns'	df.pivot_table('survived', index = 'sex', columns = 'class')
To get averages in a column by grouping rows basis values in another column	df.pivot_table('survived', index = 'sex') **OR** df.groupby('sex')[['survived']].mean()
To partition data into various groups e.g. (0, 10), (10, 30), (30, 60), (60, 80)	pd.cut(df['age'],[0,10,30,60,80])
Quantile-based discretization function. Discretize variable into equal-sized buckets based on rank or based on sample quantiles	df['fare_cat'] = pd.qcut(x['fare'],2)
Aggregation on 2 levels using 'Groupby' and 'Unstack' to change 'class' to 'cols' from 'rows'	df.groupby(['sex','class'])['survived'].aggregate ('mean').unstack() **OR** df.pivot_table('survived', index = 'sex', columns = 'class')
To only 'count' the values under aggregated variables in a column	df.pivot_table(index=['sex'], columns=['pclass'], values='survived', aggfunc='count')
To aggregate different columns in different ways	df.pivot_table(index = 'sex', columns = 'class', aggfunc = {'survived':'sum', 'fare':'mean'})
Zipfiles	
To use zipfile library	Import zipfile
To access a zip file in 'read only' mode	df = zipfile.ZipFile('file_location/file_name.zip', 'r')
To get entire directory details of the zip file	df.printdir()
To access 'csv' files in a zip file	df1 = pd.read_csv(df.open('df/file_name.csv'))
Html Parser	
To run a html document through 'BeautifulSoup'	from bs4 import BeautifulSoup soup = BeautifulSoup(html_doc_name, 'html.parser')
To improve the visible html doc structure	Print(soup.prettify())

Updating HTML via Selenium Webdriver	
To send data to HTML	browser = webdriver.Chrome(executable_path= "/Users/admin/Documents/Webdriver/chromedriver") url = 'https://xxxx' browser.get(url) username = browser.find_element_by_id("navbar_username") password = browser.find_element_by_name("vb_login_password_hint") username.send_keys("MarioP") password.send_keys("codeswitching")
To change text in HTML	browser = webdriver.Chrome(executable_path= "/Users/admin/Documents/Webdriver/chromedriver") url = 'https://xxxx' browser.get(url) element = browser.find_elements_by_xpath("//div[@class='ncFU _3gSB']")[1] browser.execute_script('arguments[0].innerHTML = "11:30 PM";', element)
Script Run Time	
To get the run time of a Python Script *['number' represents the no. of iterations; if number = 100,* *run_time = timeit.timeit(script, number = 100)/100]*	import timeit script = """ xxx """ run_time = timeit.timeit(script, number = 1) print(run_time)
General Commands	
To upgrade pip in Jupyter Hub	!pip install --user --upgrade pip
To install a package (e.g. pandas) in Jupyter Hub	!pip install --user pandas, OR pip install --user unidecode
To get all the library methods available in a lib	dir(pandas)

```
len(df)          series + value     df[df.c == value]
df.head()        series + series2   df[(df.c >= value) & (df.d < value)]
df.tail()        series.notnull()   df[(df.c < value) | (df.d != value)]
df.COLUMN        series.isnull()    df.sort('column')
df['COLUMN']     series.order()     df.sort(['column1', 'column2'])

s.str.len()          s.value_counts()
s.str.contains()     s.sort_index()    df[['column1', 'column2']]
s.str.startswith()   s.plot(...)       df.plot(x='a', y='b', kind='bar')

df.set_index('a').sort_index()          df.loc['value']
df.set_index(['a', 'b']).sort_index()   df.loc[('v','u')]
df.groupby('column')                    .size() .mean() .min() .max()
df.groupby(['column1', 'column2'])      .agg(['min', 'max'])

df.unstack()
df.stack()
df.fillna(value)
s.fillna(value)
```

General:

- To install a new library, Go to Terminal/Jupyter Nb and run 'pip install <lib_name>'
- Useful libraries - pandas, numpy, dfply, matplotlib, datetime, pymysql
- You can use `*` and `**` when calling functions as well. This is a shortcut that allows you to pass multiple arguments to a function directly using either a list/tuple or a dict
- Use 'pd' shortcut/alias (as its typically used) for Dataframes library
- Use 'np' shortcut/alias (as its typically used) for 'Numpy' and use Numpy for arrays as it has arrays format
- Use 'dt' shortcut/alias (as its typically used) for 'Datatime' library

Spyder:

- #%% - Spyder Command to Isolate portions of the code
- Always 'import' relevant libraries via "import pandas" etc.

Jupyter Notebook ::

Required Action	Command
Cell Activity Shortcuts	
To split a cell	Cntrl + Shift + -
To add / merge multiple cells	Shift + M
To comment multiple lines	Cntrl + /
To indent multiple lines in one go	Select the lines; Press 'Tab'
To remove indent from multiple lines in one go	Select the lines; Press 'Shift + Tab'
To add the cell above the current cell	A (after pressing Esc)
To add the cell below the current cell	B (after pressing Esc)
To delete the current cell	X
To only run current cell	Cntrl + Enter
To run current cell and add an empty cell below	Alt + Enter