Ques 01 : How do I read "Tabular Data" file into pandas?

Tabular Data means data that looks like a table (rows & columns)

In [1]:
```
import pandas as pd
```

In [2]:
```
# URL of the dataset

orders_ds = pd.read_table('http://bit.ly/chiporders')
orders_ds.head()
```

Out[2]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98

By default, **read_table( )** assumes that the data is **'tab'** seperated, and first column is 'header'. These are some default parameters of read_table( ) function. Here, the Dataset is in default format of read_Table( ) function, so it is loaded perfectly.

We can put the cursor in the bracket of read_table( ) function and press "shift + tab" to see all the parameters of this function.

In [3]:
```
movie_users_ds = pd.read_table('http://bit.ly/movieusers')
movie_users_ds.head()
```

Out[3]:

	1|24|M|technician|85711
0	2|53|F|other|94043
1	3|23|M|writer|32067
2	4|24|M|technician|43537
3	5|33|F|other|15213
4	6|42|M|executive|98101

The movie dataset is not in default format. Here, seperator is **' | '** (pipe) and 1st column is not header.

In [4]:
```
column_list = ['User_id', 'Age', 'Gender', 'Occupation', 'Zip-code']    # For column header

movie_users_ds = pd.read_table('http://bit.ly/movieusers', sep='|', header = None, names = column_list )
# Here, header = None, as there is no header in the dataset.
# If a header exists and we want to replace that, then header = 0
movie_users_ds.head()
```

Out[4]:

	User_id	Age	Gender	Occupation	Zip-code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

Ques 02: How do I select a "Series" / "Column" from a "Dataframe"?

'DataFrames' and 'Series' are the two main data structures in pandas for data storage: a **DataFrame** is like a table, and each column of the table is called a **'Series'**. We will often select a Series in order to analyze or manipulate it.

In [5]:
```
# UFO dataset
# read_csv( ) is used for 'comma' seperated file

ufo_ds = pd.read_csv('http://bit.ly/uforeports')  # ufo = pd.read_table('http://bit.ly/uforeports', sep =
ufo_ds.head()
```

Out[5]:

	City	Colors Reported	Shape Reported	State	Time
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00
2	Holyoke	NaN	OVAL	CO	2/15/1931 14:00
3	Abilene	NaN	DISK	KS	6/1/1931 13:00
4	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00

In [6]:
```
type(ufo_ds)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
# Selecting 'City' column (a series)
# Column name is case-sensitive

ufo_ds['City'].head()
```

Out[7]:
```
0                 Ithaca
1            Willingboro
2                Holyoke
3                Abilene
4    New York Worlds Fair
Name: City, dtype: object
```

In [8]:
```
type(ufo_ds['City'])
```

Out[8]:
```
pandas.core.series.Series
```

In [9]:
```
# Each name becomes an attribute of the dataframe.
# So, we can use dot notation which is more easy but there is a limitation.
# The column name must not be a "keyword", and there should be "no space" in column name.

ufo_ds.City.head()
```

Out[9]:
```
0                 Ithaca
1            Willingboro
2                Holyoke
3                Abilene
4    New York Worlds Fair
Name: City, dtype: object
```

Dot notation does not always work but **Bracket** notation always works. If we want to stick to 'Dot' notation, then we have to rename all the columns so that there is no space in the columns name and no name is a keyword (built-in method or attribute) of python.

We can add two series (columns) if they are strings.

Creating a new column by combining two columns.

In [10]:
```
# We can create a new series/colum

ufo_ds['Location'] = ufo_ds.City + ' , ' + ufo_ds.State   # Need Bracket notation for the new column

ufo_ds.head()
```

Out[10]:

	City	Colors Reported	Shape Reported	State	Time	Location
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00	Ithaca , NY
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00	Willingboro , NJ
2	Holyoke	NaN	OVAL	CO	2/15/1931 14:00	Holyoke , CO
3	Abilene	NaN	DISK	KS	6/1/1931 13:00	Abilene , KS
4	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00	New York Worlds Fair , NY

Ques 03: Why do some pandas commands end with parentheses, and other commands not?

Method ends with **parenthesis** but **attribute** doesn't.

In [11]:
```
# IMDB - Internet Movie Database (Dataset)

movie_ratings_ds = pd.read_csv('http://bit.ly/imdbratings')

movie_ratings_ds.head(4)
```

Out[11]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...

In [12]:
```
# Give some descriptive statistical info about the numerical columns (default) of the dataframe
movie_ratings_ds.describe()
```

Out[12]:

	star_rating	duration
count	979.000000	979.000000
mean	7.889785	120.979571
std	0.336069	26.218010
min	7.400000	64.000000
25%	7.600000	102.000000
50%	7.800000	117.000000
75%	8.100000	134.000000
max	9.300000	242.000000

In [13]:
```
movie_ratings_ds.shape       # Returns no. of rows, no. of columns in the Dataframe
```

Out[13]:
```
(979, 6)
```

In [14]:
```
movie_ratings_ds.dtypes     # Show the datatype of all columns
```

Out[14]:
```
star_rating       float64
title              object
content_rating     object
genre              object
duration            int64
actors_list        object
dtype: object
```

In [15]:
```
movie_ratings_ds.describe(include=['object'])
# top --> Record having highest frequency , freq --> No. of frequecny of the top record
```

Out[15]:

	title	content_rating	genre	actors_list
count	979	976	979	979
unique	975	12	16	969
top	True Grit	R	Drama	[u'Daniel Radcliffe', u'Emma Watson', u'Rupert...
freq	2	460	278	6

Ques 04: How do I rename columns in pandas dataframe?

In [16]:
```
# See the column names

ufo_ds.columns
```

Out[16]:
```
Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time',
       'Location'],
      dtype='object')
```

In [17]:
```
ufo_ds.head(3)
```

Out[17]:

	City	Colors Reported	Shape Reported	State	Time	Location
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00	Ithaca , NY
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00	Willingboro , NJ
2	Holyoke	NaN	OVAL	CO	2/15/1931 14:00	Holyoke , CO

In [18]:
```
ufo_ds.rename(columns = {'Colors Reported':'Color_Reported', 'Shape Reported':'Shape_Reported'})
ufo_ds.columns
```

Out[18]:
```
Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time',
       'Location'],
      dtype='object')
```

In [19]:
```
# The column name is not changed permanently. We have to put one more argument : inplace = True.

ufo_ds.rename(columns = {'Colors Reported':'Color_Reported', 'Shape Reported':'Shape_Reported'}, inplace=
ufo_ds.columns
```

Out[19]:
```
Index(['City', 'Color_Reported', 'Shape_Reported', 'State', 'Time',
       'Location'],
      dtype='object')
```

In [20]:
```
ufo_ds.head(2)
```

Out[20]:

	City	Color_Reported	Shape_Reported	State	Time	Location
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00	Ithaca , NY
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00	Willingboro , NJ

In [21]:
```
# Another way of renaming      [ If we want to change all the column name ]

ufo_col = ['city', 'color', 'shape', 'state', 'time', 'location']
ufo_ds.columns = ufo_col
ufo_ds.columns
```

Out[21]:
```
Index(['city', 'color', 'shape', 'state', 'time', 'location'], dtype='object')
```

In [22]:
```
# Or we can rename the columns name when we load the dataset
# 0th row will be header

ufo_col = ['city', 'color', 'shape', 'state', 'time']
ufo_ds = pd.read_csv('http://bit.ly/uforeports', header=0, names = ufo_col)  # Replace 0th row (Given Hea
ufo_ds.head(3)
```

Out[22]:

	city	color	shape	state	time
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00
2	Holyoke	NaN	OVAL	CO	2/15/1931 14:00

If we have 100 columns having space in their names, and we want to replace these spaces. How we can do that ?

We can do that using **string methods**. Each column name is a string

In [23]:
```
ufo_ds = pd.read_csv('http://bit.ly/uforeports')
ufo_ds.head(2)
```

Out[23]:

	City	Colors Reported	Shape Reported	State	Time
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00

In [24]:
```
ufo_ds.columns = ufo_ds.columns.str.replace(' ','_')  # "ufo_ds.columns" is reassigned
```

In [25]:
```
ufo_ds.head(3)
```

Out[25]:

	City	Colors_Reported	Shape_Reported	State	Time
0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	Willingboro	NaN	OTHER	NJ	6/30/1930 20:00
2	Holyoke	NaN	OVAL	CO	2/15/1931 14:00

Ques 05: How do I remove columns / rows from a pandas dataframe?

In [26]:
```
ufo_ds.shape
```

Out[26]:
```
(18241, 5)
```

In [27]:
```
ufo_ds.columns
```

Out[27]:
```
Index(['City', 'Colors_Reported', 'Shape_Reported', 'State', 'Time'], dtype='object')
```

In [28]:
```
# We will drop the column : "Colors_Reported"
# axis = 0 means row
# axis = 1 means column

ufo_ds.drop('Colors_Reported', axis=1, inplace=True)
```

In [29]:
```
ufo_ds.columns
```

Out[29]:
```
Index(['City', 'Shape_Reported', 'State', 'Time'], dtype='object')
```

In [30]:
```
ufo_ds.head(3)
```

Out[30]:

	City	Shape_Reported	State	Time
0	Ithaca	TRIANGLE	NY	6/1/1930 22:00
1	Willingboro	OTHER	NJ	6/30/1930 20:00
2	Holyoke	OVAL	CO	2/15/1931 14:00

In [31]:
```
# Drop multiple columns
ufo_ds.drop(['City','Time'], axis=1, inplace=True)
```

In [32]:
```
ufo_ds.head()
```

Out[32]:

	Shape_Reported	State
0	TRIANGLE	NY
1	OTHER	NJ
2	OVAL	CO
3	DISK	KS
4	LIGHT	NY

Removing rows :

In [33]:
```
ufo_ds.drop([0, 1] , axis = 0, inplace = True)   # Index no of the rows

ufo_ds.head()
```

Out[33]:

	Shape_Reported	State
2	OVAL	CO
3	DISK	KS
4	LIGHT	NY
5	DISK	ND
6	CIRCLE	CA

Ques 06 : How do I sort pandas Dataframe or Series?

pandas allows us to sort a Dataframe by one of its columns (known as a "Series"), and also allows us to sort a **Series** alone.

In [34]:
```
movie_ratings_ds.head()
```

Out[34]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....

In [35]:
```
# We can sort the dataframe by any column
# This don't sort the actual data. It just shows us in sorted order
# Ascending Order

movie_ratings_ds.sort_values('duration').head()
```

Out[35]:

	star_rating	title	content_rating	genre	duration	actors_list
389	8.0	Freaks	UNRATED	Drama	64	[u'Wallace Ford', u'Leila Hyams', u'Olga Bacla...
338	8.0	Battleship Potemkin	UNRATED	History	66	[u'Aleksandr Antonov', u'Vladimir Barsky', u'G...
258	8.1	The Cabinet of Dr. Caligari	UNRATED	Crime	67	[u'Werner Krauss', u'Conrad Veidt', u'Friedric...
293	8.1	Duck Soup	PASSED	Comedy	68	[u'Groucho Marx', u'Harpo Marx', u'Chico Marx']
88	8.4	The Kid	NOT RATED	Comedy	68	[u'Charles Chaplin', u'Edna Purviance', u'Jack...

In [36]:
```
# Changing the actual order of the rows in dataframe we need to use inplace=True.
# Descending order

# sort_values ( by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')

movie_ratings_ds.sort_values('duration', ascending=False).head()
```

Out[36]:

	star_rating	title	content_rating	genre	duration	actors_list
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...
157	8.2	Gone with the Wind	G	Drama	238	[u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
78	8.4	Once Upon a Time in America	R	Crime	229	[u'Robert De Niro', u'James Woods', u'Elizabet...
142	8.3	Lagaan: Once Upon a Time in India	PG	Adventure	224	[u'Aamir Khan', u'Gracy Singh', u'Rachel Shell...
445	7.9	The Ten Commandments	APPROVED	Adventure	220	[u'Charlton Heston', u'Yul Brynner', u'Anne Ba...

In [37]:
```
# See the sorted value of a column
# Sort the series

movie_ratings_ds.title.sort_values().head()
```

Out[37]:
```
542     (500) Days of Summer
5                12 Angry Men
201          12 Years a Slave
698                  127 Hours
110    2001: A Space Odyssey
Name: title, dtype: object
```

In [38]:
```
# Sort by multiple columns

movie_ratings_ds.sort_values(['star_rating', 'duration'], ascending=False).head(10)
```

Out[38]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure	201	[u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
8	8.9	Schindler's List	R	Biography	195	[u'Liam Neeson', u'Ralph Fiennes', u'Ben Kings...
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western	161	[u'Clint Eastwood', u'Eli Wallach', u'Lee Van ...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
9	8.9	Fight Club	R	Drama	139	[u'Brad Pitt', u'Edward Norton', u'Helena Bonh...
5	8.9	12 Angry Men	NOT RATED	Drama	96	[u'Henry Fonda', u'Lee J. Cobb', u'Martin Bals...

In [39]:
```
movie_ratings_ds.head()
```

Out[39]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....

Ques 07: How do I filter rows of a pandas dataframe by column value ?

In [40]:
```
movie_ratings_ds.head(3)
```

Out[40]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...

In [41]:
```
movie_ratings_ds.shape
```

Out[41]: (979, 6)

In [42]:
```
movie_ratings_ds.head()
```

Out[42]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....

We want to find the records having duration >= 200.

Long Process

In [43]:
```
# Creating a python list containing boolean value for each row
booleans = []
for length in movie_ratings_ds.duration:
    if length >= 200:
        booleans.append(True)
    else:
        booleans.append(False)

booleans[0:6]   # check first 5 values
```

Out[43]:
```
[False, False, True, False, False, False]
```

In [44]:
```
len(booleans)
```

Out[44]:
```
979
```

In [45]:
```
# convert the boolean list into pandas series

is_long = pd.Series(booleans)
is_long.head()
```

Out[45]:
```
0    False
1    False
2     True
3    False
4    False
dtype: bool
```

In [46]:
```
movie_ratings_ds[is_long]
# Pass the Series to dataframe. It will show only the rows having True value in the Series
```

Out[46]:

	star_rating	title	content_rating	genre	duration	actors_list
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure	201	[u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17	8.7	Seven Samurai	UNRATED	Drama	207	[u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78	8.4	Once Upon a Time in America	R	Crime	229	[u'Robert De Niro', u'James Woods', u'Elizabet...
85	8.4	Lawrence of Arabia	PG	Adventure	216	[u"Peter O'Toole", u'Alec Guinness', u'Anthony...
142	8.3	Lagaan: Once Upon a Time in India	PG	Adventure	224	[u'Aamir Khan', u'Gracy Singh', u'Rachel Shell...
157	8.2	Gone with the Wind	G	Drama	238	[u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
204	8.1	Ben-Hur	G	Adventure	212	[u'Charlton Heston', u'Jack Hawkins', u'Stephe...
445	7.9	The Ten Commandments	APPROVED	Adventure	220	[u'Charlton Heston', u'Yul Brynner', u'Anne Ba...
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...
630	7.7	Malcolm X	PG-13	Biography	202	[u'Denzel Washington', u'Angela Bassett', u'De...
767	7.6	It's a Mad, Mad, Mad, Mad World	APPROVED	Action	205	[u'Spencer Tracy', u'Milton Berle', u'Ethel Me...

Short Process

In [47]:
```
is_long = movie_ratings_ds.duration >= 200
movie_ratings_ds[is_long].head()
```

Out[47]:

	star_rating	title	content_rating	genre	duration	actors_list
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure	201	[u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17	8.7	Seven Samurai	UNRATED	Drama	207	[u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78	8.4	Once Upon a Time in America	R	Crime	229	[u'Robert De Niro', u'James Woods', u'Elizabet...
85	8.4	Lawrence of Arabia	PG	Adventure	216	[u"Peter O'Toole", u'Alec Guinness', u'Anthony...

In [48]:
```
# We can wrap it one line
movie_ratings_ds[ movie_ratings_ds.duration >= 200 ].head()
```

Out[48]:

	star_rating	title	content_rating	genre	duration	actors_list
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure	201	[u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17	8.7	Seven Samurai	UNRATED	Drama	207	[u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78	8.4	Once Upon a Time in America	R	Crime	229	[u'Robert De Niro', u'James Woods', u'Elizabet...
85	8.4	Lawrence of Arabia	PG	Adventure	216	[u"Peter O'Toole", u'Alec Guinness', u'Anthony...

In [49]:
```
# Showing only one column based on filter

movie_ratings_ds[ movie_ratings_ds.duration >= 200 ].title.head()

# It has some limitation. So, there is a better approach for this same task
```

Out[49]:
```
2                           The Godfather: Part II
7     The Lord of the Rings: The Return of the King
17                                  Seven Samurai
78                     Once Upon a Time in America
85                                Lawrence of Arabia
Name: title, dtype: object
```

In [50]:
```
# use of "loc" method
movie_ratings_ds.loc[ movie_ratings_ds.duration >= 200, 'genre']
```

Out[50]:
```
2          Crime
7      Adventure
17         Drama
78         Crime
85     Adventure
142    Adventure
157        Drama
204    Adventure
445    Adventure
476        Drama
630    Biography
767       Action
Name: genre, dtype: object
```

Ques 08: How do I apply multiple filter criteria to a pandas DataFrame?

In [51]:
```
movie_ratings_ds[(movie_ratings_ds.duration >= 200) & (movie_ratings_ds.genre == 'Drama')]
```

Out[51]:

	star_rating	title	content_rating	genre	duration	actors_list
17	8.7	Seven Samurai	UNRATED	Drama	207	[u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
157	8.2	Gone with the Wind	G	Drama	238	[u'Clark Gable', u'Vivien Leigh', u'Thomas Mit...
476	7.8	Hamlet	PG-13	Drama	242	[u'Kenneth Branagh', u'Julie Christie', u'Dere...

In [52]:
```
movie_ratings_ds[(movie_ratings_ds.duration >= 200) | (movie_ratings_ds.genre == 'Drama')].head()
```

Out[52]:

	star_rating	title	content_rating	genre	duration	actors_list
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
5	8.9	12 Angry Men	NOT RATED	Drama	96	[u'Henry Fonda', u'Lee J. Cobb', u'Martin Bals...
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure	201	[u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
9	8.9	Fight Club	R	Drama	139	[u'Brad Pitt', u'Edward Norton', u'Helena Bonh...
13	8.8	Forrest Gump	PG-13	Drama	142	[u'Tom Hanks', u'Robin Wright', u'Gary Sinise']

In [53]:
```
# Specify multiple-values for a column

movie_ratings_ds[ movie_ratings_ds.genre.isin(['Drama', 'Crime', 'Adventure']) ].head(10)
```

Out[53]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
5	8.9	12 Angry Men	NOT RATED	Drama	96	[u'Henry Fonda', u'Lee J. Cobb', u'Martin Bals...
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure	201	[u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
9	8.9	Fight Club	R	Drama	139	[u'Brad Pitt', u'Edward Norton', u'Helena Bonh...
10	8.8	The Lord of the Rings: The Fellowship of the Ring	PG-13	Adventure	178	[u'Elijah Wood', u'Ian McKellen', u'Orlando Bl...
13	8.8	Forrest Gump	PG-13	Drama	142	[u'Tom Hanks', u'Robin Wright', u'Gary Sinise']
14	8.8	The Lord of the Rings: The Two Towers	PG-13	Adventure	179	[u'Elijah Wood', u'Ian McKellen', u'Viggo Mort...

How to read only some specific columns & rows from a CSV file?

In [54]:
```
ufo_ds = pd.read_csv('http://bit.ly/uforeports', usecols = [0,3], nrows = 5)
# Picking 1st and 4th column [ City and State ]
# picking first 5 rows
ufo_ds
```

Out[54]:

	City	State
0	Ithaca	NY
1	Willingboro	NJ
2	Holyoke	CO
3	Abilene	KS
4	New York Worlds Fair	NY

How to iterate a Series and Dataframe ?

In [55]:
```
for c in ufo_ds.State:
    print(c)
```

```
NY
NJ
CO
KS
NY
```

In [56]:
```
for index, row in ufo_ds.iterrows():
    print(index, row.City, row.State)
```

```
0 Ithaca NY
1 Willingboro NJ
2 Holyoke CO
3 Abilene KS
4 New York Worlds Fair NY
```

Ques 09 : When should I use a "GROUP BY" pandas?

In [57]:
```
drinks_ds = pd.read_csv('http://bit.ly/drinksbycountry')
```

In [58]:
```
drinks_ds.head(4)
```

Out[58]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
0	Afghanistan	0	0	0	0.0	Asia
1	Albania	89	132	54	4.9	Europe
2	Algeria	25	0	14	0.7	Africa
3	Andorra	245	138	312	12.4	Europe

In [59]:
```
# What is the average beer_servings across all countries?
drinks_ds.beer_servings.mean()
```

Out[59]:
```
106.16062176165804
```

In [60]:
```
# What is the average beer_servings by continents? How beer_servings varied from continent to continent ?
drinks_ds.groupby('continent').beer_servings.mean()
```

Out[60]:
```
continent
Africa            61.471698
Asia              37.045455
Europe           193.777778
North America    145.434783
Oceania           89.687500
South America    175.083333
Name: beer_servings, dtype: float64
```

In [61]:
```
# Max beer_servings by continents
drinks_ds.groupby('continent').beer_servings.max()
```

Out[61]:
```
continent
Africa           376
Asia             247
Europe           361
North America    285
Oceania          306
South America    333
Name: beer_servings, dtype: int64
```

Specifying multiple aggregation functions at once

In [62]:
```
drinks_ds.groupby('continent').beer_servings.agg(['count','max','min','mean'])
```

Out[62]:

continent	count	max	min	mean
Africa	53	376	0	61.471698
Asia	44	247	0	37.045455
Europe	45	361	0	193.777778
North America	23	285	1	145.434783
Oceania	16	306	0	89.687500
South America	12	333	93	175.083333

Calculating mean/max/min for all numeric columns

In [63]:
```
drinks_ds.groupby('continent').mean()
```

Out[63]:

continent	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
Africa	61.471698	16.339623	16.264151	3.007547
Asia	37.045455	60.840909	9.068182	2.170455
Europe	193.777778	132.555556	142.222222	8.617778
North America	145.434783	165.739130	24.521739	5.995652
Oceania	89.687500	58.437500	35.625000	3.381250
South America	175.083333	114.750000	62.416667	6.308333

In [64]:
```
drinks_ds.groupby('continent').agg(['count','mean','max','min'])
```

Out[64]:

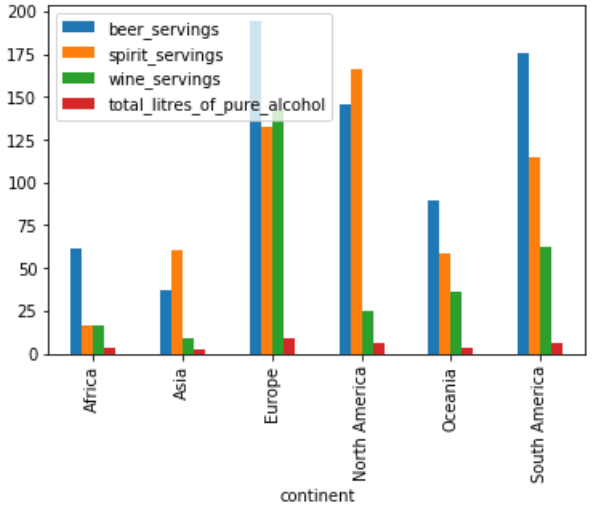
	beer_servings				spirit_servings				wine_servings				total_litres_of_pure_alcoho			
	count	mean	max	min	count	mean	max	min	count	mean	max	min	count	mean	max	mi
continent																
Africa	53	61.471698	376	0	53	16.339623	152	0	53	16.264151	233	0	53	3.007547	9.1	0
Asia	44	37.045455	247	0	44	60.840909	326	0	44	9.068182	123	0	44	2.170455	11.5	0
Europe	45	193.777778	361	0	45	132.555556	373	0	45	142.222222	370	0	45	8.617778	14.4	0
North America	23	145.434783	285	1	23	165.739130	438	68	23	24.521739	100	1	23	5.995652	11.9	2
Oceania	16	89.687500	306	0	16	58.437500	254	0	16	35.625000	212	0	16	3.381250	10.4	0
South America	12	175.083333	333	93	12	114.750000	302	25	12	62.416667	221	1	12	6.308333	8.3	3

Visually showing the result

In [65]:
```
%matplotlib inline

drinks_ds.groupby('continent').mean().plot(kind='bar')
```

Out[65]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xa021fbfc50>
```



Ques 10 : How to use the "axis" parameter in pandas?

In [66]:
```
drinks_ds = pd.read_csv('http://bit.ly/drinksbycountry')
```

In [67]:
```
drinks_ds.head(3)
```

Out[67]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
0	Afghanistan	0	0	0	0.0	Asia
1	Albania	89	132	54	4.9	Europe
2	Algeria	25	0	14	0.7	Africa

In [68]:
```
# Dropping the 'continent' column
drinks_ds.drop('continent', axis=1, inplace=True)
```

In [69]:
```
drinks_ds.head(2)
```

Out[69]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	Afghanistan	0	0	0	0.0
1	Albania	89	132	54	4.9

In [70]:
```
drinks_ds.drop(1, axis=0,inplace=True)  # Dropping 2nd row
drinks_ds.head(3)
```

Out[70]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	Afghanistan	0	0	0	0.0
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4

In [71]:
```
# We can use aggregation along row-wise (default) or column-wise.
# axis = 0 / axis = 'index' means row-wise
# axis = 1 / axis = 'columns' means column-wise
drinks_ds.mean(axis=0)
```

Out[71]:
```
beer_servings                   106.250000
spirit_servings                  80.729167
wine_servings                    49.427083
total_litres_of_pure_alcohol      4.716146
dtype: float64
```

In [72]:
```
drinks_ds.mean(axis='index')
```

Out[72]:
```
beer_servings                   106.250000
spirit_servings                  80.729167
wine_servings                    49.427083
total_litres_of_pure_alcohol      4.716146
dtype: float64
```

In [73]:
```
print(drinks_ds.mean(axis=0).shape)
print(drinks_ds.mean(axis=1).shape)
```

```
(4,)
(192,)
```

Ques 11 : How to use **String** methods in pandas?

In [74]:
```
'hi'.upper()
```

Out[74]:
```
'HI'
```

In [75]:
```
orders_ds.item_name.str.upper().head(3)
```

Out[75]:
```
0    CHIPS AND FRESH TOMATO SALSA
1                            IZZE
2                NANTUCKET NECTAR
Name: item_name, dtype: object
```

In [76]:
```
orders_ds.item_name.str.contains('Chicken').head()
```

Out[76]:
```
0    False
1    False
2    False
3    False
4     True
Name: item_name, dtype: bool
```

In [77]:
```
# Showing only the item related to 'chicken'
orders_ds[orders_ds.item_name.str.contains('Chicken')].head()
```

Out[77]:

	order_id	quantity	item_name	choice_description	item_price
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	$10.98
11	6	1	Chicken Crispy Tacos	[Roasted Chili Corn Salsa, [Fajita Vegetables,...	$8.75
12	6	1	Chicken Soft Tacos	[Roasted Chili Corn Salsa, [Rice, Black Beans,...	$8.75
13	7	1	Chicken Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	$11.25

[python API for string]

http://pandas.pydata.org/pandas-docs/stable/api.html#string-handling

In [78]:
```
orders_ds.head()
```

Out[78]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98

Removing Brackets from 'choice_description' column

In [79]:
```
orders_ds.choice_description.str.replace('[','').str.replace(']','').head()
```

Out[79]:
```
0                                                 NaN
1                                           Clementine
2                                               Apple
3                                                 NaN
4    Tomatillo-Red Chili Salsa (Hot), Black Beans, ...
Name: choice_description, dtype: object
```

In [80]:
```
orders_ds.head()
```

Out[80]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98

Using Regular expression

In [81]:
```
orders_ds.choice_description.str.replace('[\[\]]','').head()
```

Out[81]:
```
0                                                 NaN
1                                           Clementine
2                                               Apple
3                                                 NaN
4    Tomatillo-Red Chili Salsa (Hot), Black Beans, ...
Name: choice_description, dtype: object
```

Ques 12 : How do I change the data type of a pandas Series ?

This is useful when a column's data type is string but actually holds numbers. So, we can change the type from string to number for doing mathematical calculation.

In [82]:
```
drinks_ds = pd.read_csv('http://bit.ly/drinksbycountry')
```

In [83]:
```
drinks_ds.dtypes    # 'object' means string
```

Out[83]:
```
country                          object
beer_servings                     int64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
dtype: object
```

In [84]:
```
drinks_ds['beer_servings'] = drinks_ds.beer_servings.astype(float)
```

In [85]:
```
drinks_ds.dtypes
```

Out[85]:
```
country                          object
beer_servings                   float64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
dtype: object
```

In [86]:
```
drinks = pd.read_csv('http://bit.ly/drinksbycountry',dtype={'beer_servings':float})
drinks.dtypes
```

Out[86]:
```
country                          object
beer_servings                   float64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
dtype: object
```

In [87]:
```
orders_ds.head(3)
```

Out[87]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39

In [88]:
```
orders_ds.dtypes
```

Out[88]:
```
order_id               int64
quantity               int64
item_name             object
choice_description    object
item_price            object
dtype: object
```

```
Datatype of the column 'item_price' is 'object'. We will cast it as float and then find mean of the
column
```

In [89]:
```
orders_ds.item_price.str.replace('$','').head()
```

Out[89]:
```
0     2.39
1     3.39
2     3.39
3     2.39
4    16.98
Name: item_price, dtype: object
```

In [90]:
```
orders_ds.item_price.str.replace('$','').astype(float).mean()
```

Out[90]:
```
7.464335785374397
```

In [91]:
```
# converting True and False to 1 and 0 . This is important in ML
orders_ds.item_name.str.contains('Chicken').astype(int).head()
```

Out[91]:
```
0    0
1    0
2    0
3    0
4    1
Name: item_name, dtype: int32
```

Ques 13 : What is the best way of dropping every non-numeric columns from a dataframe?

In [92]:
```
drinks_ds.head()
```

Out[92]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
0	Afghanistan	0.0	0	0	0.0	Asia
1	Albania	89.0	132	54	4.9	Europe
2	Algeria	25.0	0	14	0.7	Africa
3	Andorra	245.0	138	312	12.4	Europe
4	Angola	217.0	57	45	5.9	Africa

In [93]:
```
drinks_ds.dtypes
```

Out[93]:
```
country                          object
beer_servings                   float64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
dtype: object
```

In [94]:
```
import numpy as np
drinks_ds_int = drinks_ds.select_dtypes(include=[np.number])
```

In [95]:
```
drinks_ds_int.head()
```

Out[95]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
0	0.0	0	0	0.0
1	89.0	132	54	4.9
2	25.0	0	14	0.7
3	245.0	138	312	12.4
4	217.0	57	45	5.9

Ques 14 : How do I explore a pandas Series?

In [96]:
```
movie_ratings_ds.head()
```

Out[96]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....

In [97]:
```
movie_ratings_ds.genre.describe()
```

Out[97]:
```
count       979
unique       16
top       Drama
freq        278
Name: genre, dtype: object
```

In [98]:
```
# showing the count for every distinct value of the 'genre' column
movie_ratings_ds.genre.value_counts()
```

Out[98]:
```
Drama        278
Comedy       156
Action       136
Crime        124
Biography     77
Adventure     75
Animation     62
Horror        29
Mystery       16
Western        9
Thriller       5
Sci-Fi         5
Film-Noir      3
Family         2
Fantasy        1
History        1
Name: genre, dtype: int64
```

In [99]:
```
# showing percentage
movie_ratings_ds.genre.value_counts(normalize=True)
```

Out[99]:
```
Drama        0.283963
Comedy       0.159346
Action       0.138917
Crime        0.126660
Biography    0.078652
Adventure    0.076609
Animation    0.063330
Horror       0.029622
Mystery      0.016343
Western      0.009193
Thriller     0.005107
Sci-Fi       0.005107
Film-Noir    0.003064
Family       0.002043
Fantasy      0.001021
History      0.001021
Name: genre, dtype: float64
```

In [100]:
```
movie_ratings_ds.genre.unique()  # Show all unique values
```

Out[100]:
```
array(['Crime', 'Action', 'Drama', 'Western', 'Adventure', 'Biography',
       'Comedy', 'Animation', 'Mystery', 'Horror', 'Film-Noir', 'Sci-Fi',
       'History', 'Thriller', 'Family', 'Fantasy'], dtype=object)
```

In [101]:
```
movie_ratings_ds.genre.nunique()  # Show the count of unique value
```

Out[101]:
```
16
```

In [102]:
```
# Show the the details content_rating for each genre type
pd.crosstab(movie_ratings_ds.genre, movie_ratings_ds.content_rating)
```

Out[102]:

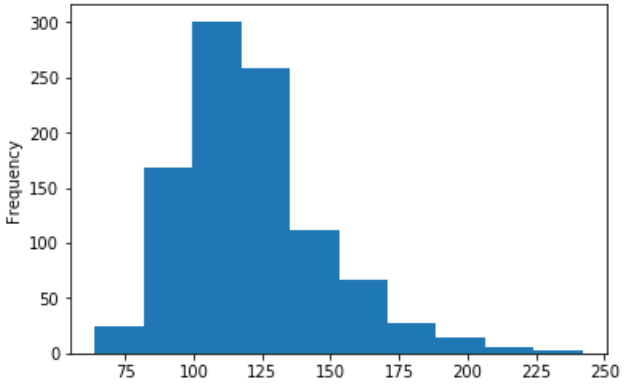
content_rating	APPROVED	G	GP	NC-17	NOT RATED	PASSED	PG	PG-13	R	TV-MA	UNRATED	X
genre												
Action	3	1	1	0	4	1	11	44	67	0	3	0
Adventure	3	2	0	0	5	1	21	23	17	0	2	0
Animation	3	20	0	0	3	0	25	5	5	0	1	0
Biography	1	2	1	0	1	0	6	29	36	0	0	0
Comedy	9	2	1	1	16	3	23	23	73	0	4	1
Crime	6	0	0	1	7	1	6	4	87	0	11	1
Drama	12	3	0	4	24	1	25	55	143	1	9	1
Family	0	1	0	0	0	0	1	0	0	0	0	0
Fantasy	0	0	0	0	0	0	0	0	1	0	0	0
Film-Noir	1	0	0	0	1	0	0	0	0	0	1	0
History	0	0	0	0	0	0	0	0	0	0	1	0
Horror	2	0	0	1	1	0	1	2	16	0	5	1
Mystery	4	1	0	0	1	0	1	2	6	0	1	0
Sci-Fi	1	0	0	0	0	0	0	1	3	0	0	0
Thriller	1	0	0	0	0	0	1	0	3	0	0	0
Western	1	0	0	0	2	0	2	1	3	0	0	0

In [103]:
```
movie_ratings_ds.duration.describe()
```

Out[103]:
```
count    979.000000
mean     120.979571
std       26.218010
min       64.000000
25%      102.000000
50%      117.000000
75%      134.000000
max      242.000000
Name: duration, dtype: float64
```

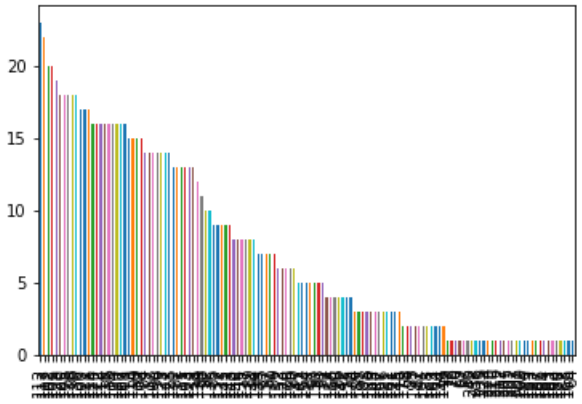
In [104]:
```
movie_ratings_ds.duration.plot(kind='hist')  # A histogram shows the distribution of a numerical variable
```

Out[104]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xa01f6b7b00>
```



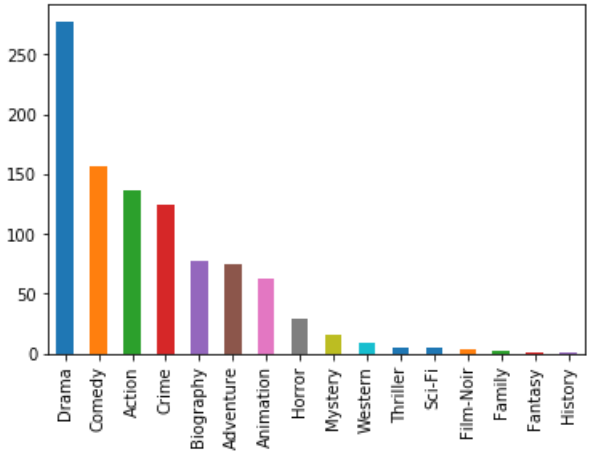
In [105]:
```
movie_ratings_ds.duration.value_counts().plot(kind='bar')   # Not feasibale if there is so many distinct
```

Out[105]: <matplotlib.axes._subplots.AxesSubplot at 0xa01f69a9b0>



In [106]:
```
movie_ratings_ds.genre.value_counts().plot(kind='bar')
```

Out[106]: <matplotlib.axes._subplots.AxesSubplot at 0xa0223da438>



In [ ]: