How do I read a tabular data file into pandas?

In [1]:
```
import pandas as pd
# read a dataset of Chipotle orders directly from a URL and store the results
orders = pd.read_table('http://bit.ly/chiporders')
```

In [2]:
```
# examine the first 5 rows
orders.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

Documentation for **`read_table`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_table.html)

In [3]:
```
# read a dataset of movie reviewers (modifying the default parameter values f
user_cols = ['user_id', 'age', 'gender', 'occupation', 'zip_code']
user = pd.read_csv(r"user.txt",sep='|', header=None, names=user_cols)
```

In [4]:
```
# csv - '','', tsv - \t, txt -
```

In [5]:
```
# pd.read_excel('.xlsx',sheet_name='Sheet1')
```

In [6]:

```python
# examine the first 5 rows
user.head(10)
```

Out[6]:

	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
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
7	8	36	M	administrator	05201
8	9	29	M	student	01002
9	10	53	M	lawyer	90703

In [7]:

```python
user.tail(10)
```

Out[7]:

	user_id	age	gender	occupation	zip_code
933	934	61	M	engineer	22902
934	935	42	M	doctor	66221
935	936	24	M	other	32789
936	937	48	M	educator	98072
937	938	38	F	technician	55038
938	939	26	F	student	33319
939	940	32	M	administrator	02215
940	941	20	M	student	97229
941	942	48	F	librarian	78209
942	943	22	M	student	77841

How do I select a pandas Series from a DataFrame?

In [8]:

```python
# read a dataset of UFO reports into a DataFrame
ufo = pd.read_table('ufo.csv', sep=',')
```

In [9]:

```python
# read_csv is equivalent to read_table, except it assumes a comma separator
ufo = pd.read_csv('ufo.csv')
```

In [10]:
```
# examine the first 5 rows
ufo.head()
```

Out[10]:

	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 [11]:
```
# select the 'City' Series using bracket notation
ufo['City']

# or equivalently, use dot notation
ufo.State
```

Out[11]:
```
0        NY
1        NJ
2        CO
3        KS
4        NY
         ..
18236    IL
18237    IA
18238    WI
18239    WI
18240    FL
Name: State, Length: 18241, dtype: object
```

Bracket notation will always work, whereas **dot notation** has limitations:

- Dot notation doesn't work if there are **spaces** in the Series name
- Dot notation doesn't work if the Series has the same name as a **DataFrame method or attribute** (like 'head' or 'shape')
- Dot notation can't be used to define the name of a **new Series** (see below)

In [12]:
```
# create a new 'Location' Series (must use bracket notation to define the Ser
ufo['Location'] = ufo.City + ', ' + ufo.State
ufo.head()
```

Out[12]:

	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

In [13]:
```
ufo.shape
```

Out[13]:
```
(18241, 6)
```

Why do some pandas commands end with parentheses (and others don't)?

In [14]:
```
# read a dataset of top-rated IMDb movies into a DataFrame
import pandas as pd
movies = pd.read_csv('http://bit.ly/imdbratings')
```

Methods end with parentheses, while **attributes** don't:

In [15]:
```
# example method: show the first 5 rows
movies.head()
```

Out[15]:

	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 [16]:
```
movies.genre.nunique()
```

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

In [17]:
```
movies.genre.unique()
```

Out[17]:
```
array(['Crime', 'Action', 'Drama', 'Western', 'Adventure', 'Biography',
       'Comedy', 'Animation', 'Mystery', 'Horror', 'Film-Noir', 'Sci-Fi',
       'History', 'Thriller', 'Family', 'Fantasy'], dtype=object)
```

In [18]:
```
# example method: calculate summary statistics
movies.describe()
```

Out[18]:

	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 [19]:
```
movies.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 979 entries, 0 to 978
Data columns (total 6 columns):
star_rating       979 non-null float64
title             979 non-null object
content_rating    976 non-null object
genre             979 non-null object
duration          979 non-null int64
actors_list       979 non-null object
dtypes: float64(1), int64(1), object(4)
memory usage: 46.0+ KB
```

In [20]:
```
# example attribute: number of rows and columns
movies.shape
```
Out[20]:
```
(979, 6)
```

In [21]:
```
# example attribute: data type of each column
movies.dtypes
```
Out[21]:
```
star_rating       float64
title              object
content_rating     object
genre              object
duration            int64
actors_list        object
dtype: object
```

In [22]:
```
# use an optional parameter to the describe method to summarize only 'object
movies.describe(include='all')
```
Out[22]:

	star_rating	title	content_rating	genre	duration	actors_list
count	979.000000	979	976	979	979.000000	979
unique	NaN	975	12	16	NaN	969
top	NaN	Les Miserables	R	Drama	NaN	[u'Daniel Radcliffe', u'Emma Watson', u'Rupert...
freq	NaN	2	460	278	NaN	6
mean	7.889785	NaN	NaN	NaN	120.979571	NaN
std	0.336069	NaN	NaN	NaN	26.218010	NaN
min	7.400000	NaN	NaN	NaN	64.000000	NaN
25%	7.600000	NaN	NaN	NaN	102.000000	NaN
50%	7.800000	NaN	NaN	NaN	117.000000	NaN
75%	8.100000	NaN	NaN	NaN	134.000000	NaN
max	9.300000	NaN	NaN	NaN	242.000000	NaN

Documentation for **`describe`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.describe.html)

5. How do I rename columns in a pandas DataFrame?

In [23]:
```
ufo = pd.read_csv('ufo.csv')
```

In [24]:
```
# examine the column names
ufo.columns
```

Out[24]:
```
Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time'], dtype
='object')
```

In [25]:
```
# rename two of the columns by using the 'rename' method
ufo.rename(columns={'Colors Reported':'Colors_Reported', 'Shape Reported':'Sh
ufo.columns
```

Out[25]:
```
Index(['City', 'Colors_Reported', 'Shape_Reported', 'State', 'Time'], dtype
='object')
```

Documentation for **`rename`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.rename.html)

In [26]:
```
ufo.columns
```

Out[26]:
```
Index(['City', 'Colors_Reported', 'Shape_Reported', 'State', 'Time'], dtype
='object')
```

In [27]:
```
# replace all of the column names by overwriting the 'columns' attribute
ufo_cols = ['city', 'colors reported', 'shape reported', 'state', 'time']
ufo.columns = ufo_cols
ufo.columns
```

Out[27]:
```
Index(['city', 'colors reported', 'shape reported', 'state', 'time'], dtype
='object')
```

In [28]:
```
# replace the column names during the file reading process by using the 'name
import pandas as pd
ufo = pd.read_csv('ufo.csv', header=0, names=ufo_cols)
ufo.columns
```

Out[28]:
```
Index(['city', 'colors reported', 'shape reported', 'state', 'time'], dtype
='object')
```

Documentation for **`read_csv`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)

In [29]:

```
# replace all spaces with underscores in the column names by using the 'str.r
ufo.columns = ufo.columns.str.replace(' ', '_')
ufo.columns
```

Out[29]:

```
Index(['city', 'colors_reported', 'shape_reported', 'state', 'time'], dtype
='object')
```

Documentation for **`str.replace`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.str.replace.html)

How do I remove columns from a pandas DataFrame?

In [30]:

```
# read a dataset of UFO reports into a DataFrame
ufo = pd.read_csv('ufo.csv')
ufo.head()
```

Out[30]:

	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 [31]:

```
# remove a single column (axis=1 refers to columns)
ufo.drop('Colors Reported', axis=1, inplace=True)
ufo.head()
```

Out[31]:

	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
3	Abilene	DISK	KS	6/1/1931 13:00
4	New York Worlds Fair	LIGHT	NY	4/18/1933 19:00

Documentation for **`drop`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop.html)

In [32]:

```
# remove multiple columns at once
ufo.drop(['City', 'State'], axis=1)
ufo.head()
ufo.drop(['City', 'State'], axis=1).head()
```

Out[32]:

	Shape Reported	Time
0	TRIANGLE	6/1/1930 22:00
1	OTHER	6/30/1930 20:00
2	OVAL	2/15/1931 14:00
3	DISK	6/1/1931 13:00
4	LIGHT	4/18/1933 19:00

In [33]:

```
ufo.head(5)
```

Out[33]:

	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
3	Abilene	DISK	KS	6/1/1931 13:00
4	New York Worlds Fair	LIGHT	NY	4/18/1933 19:00

In [34]:

```
ufo.reset_index(inplace=True)
```

In [35]:

```
ufo.head()
```

Out[35]:

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

In [36]:

```
# remove multiple rows at once (axis=0 refers to rows)
ufo.drop([2,9], axis=0, inplace=True)
# ufo.reset_index()
```

How do I sort a pandas DataFrame or a Series?

In [37]:

```
# read a dataset of top-rated IMDb movies into a DataFrame
import pandas as pd
movies = pd.read_csv('http://bit.ly/imdbratings')
movies.head()
```

Out[37]:

	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....

Note: None of the sorting methods below affect the underlying data. (In other words, the sorting is temporary).

In [38]:

```
movies.actors_list.apply(pd.Series)
```

Out[38]:

	0
0	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...
974	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 1 columns

In [39]:
```
# sort the 'title' Series in ascending order (returns a Series)
movies.star_rating.sort_values().head()
```

Out[39]:
```
978    7.4
950    7.4
949    7.4
948    7.4
947    7.4
Name: star_rating, dtype: float64
```

In [40]:
```
# sort in descending order instead
movies.star_rating.sort_values(ascending=False).head()
```

Out[40]:
```
0    9.3
1    9.2
2    9.1
3    9.0
6    8.9
Name: star_rating, dtype: float64
```

Documentation for **`sort_values`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.sort_values.html) for a **Series**. (Prior to version 0.17, use **`order`** (http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.Series.order.html) instead.)

In [41]:
```
# sort the entire DataFrame by the 'title' Series (returns a DataFrame)
movies.sort_values('title').head()
```

Out[41]:

	star_rating	title	content_rating	genre	duration	actors_list
542	7.8	(500) Days of Summer	PG-13	Comedy	95	[u'Zooey Deschanel', u'Joseph Gordon-Levitt', ...
5	8.9	12 Angry Men	NOT RATED	Drama	96	[u'Henry Fonda', u'Lee J. Cobb', u'Martin Bals...
201	8.1	12 Years a Slave	R	Biography	134	[u'Chiwetel Ejiofor', u'Michael Kenneth Willia...
698	7.6	127 Hours	R	Adventure	94	[u'James Franco', u'Amber Tamblyn', u'Kate Mara']
110	8.3	2001: A Space Odyssey	G	Mystery	160	[u'Keir Dullea', u'Gary Lockwood', u'William S...

In [42]:
```
# sort in descending order instead
movies.sort_values('title', ascending=False).head()
```

Out[42]:

	star_rating	title	content_rating	genre	duration	actors_list
864	7.5	[Rec]	R	Horror	78	[u'Manuela Velasco', u'Ferran Terraza', u'Jorg...
526	7.8	Zulu	UNRATED	Drama	138	[u'Stanley Baker', u'Jack Hawkins', u'Ulla Jac...
615	7.7	Zombieland	R	Comedy	88	[u'Jesse Eisenberg', u'Emma Stone', u'Woody Ha...
677	7.7	Zodiac	R	Crime	157	[u'Jake Gyllenhaal', u'Robert Downey Jr.', u'M...
955	7.4	Zero Dark Thirty	R	Drama	157	[u'Jessica Chastain', u'Joel Edgerton', u'Chri...

Documentation for **`sort_values`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_values.html) for a **DataFrame**. (Prior to version 0.17, use **`sort`** (http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.sort.html) instead.)

```
In [43]:  # sort the DataFrame first by 'content_rating', then by 'duration'
          movies.sort_values(['star_rating', 'duration','content_rating'],ascending=[Fa
```

Out[43]:

	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...
5	8.9	12 Angry Men	NOT RATED	Drama	96	[u'Henry Fonda', u'Lee J. Cobb', u'Martin Bals...
9	8.9	Fight Club	R	Drama	139	[u'Brad Pitt', u'Edward Norton', u'Helena Bonh...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western	161	[u'Clint Eastwood', u'Eli Wallach', u'Lee Van ...
8	8.9	Schindler's List	R	Biography	195	[u'Liam Neeson', u'Ralph Fiennes', u'Ben Kings...
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...

How do I filter rows of a pandas DataFrame by column value?

In [44]:

```
# read a dataset of top-rated IMDb movies into a DataFrame
movies = pd.read_csv('http://bit.ly/imdbratings')
movies.head()
```

Out[44]:

	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 [45]:

```
# examine the number of rows and columns
movies.shape
```

Out[45]:

```
(979, 6)
```

Goal: Filter the DataFrame rows to only show movies with a 'duration' of at least 200 minutes.

In [46]:

```
movies.duration
```

Out[46]:

```
0      142
1      175
2      200
3      152
4      154
      ...
974    116
975    118
976    138
977    114
978    126
Name: duration, Length: 979, dtype: int64
```

In [47]:

```
# create a list in which each element refers to a DataFrame row: True if the
booleans = []
for length in movies.duration:
    if length >= 200:
        booleans.append(True)
    else:
        booleans.append(False)
```

In [48]:
```
# confirm that the list has the same length as the DataFrame
len(booleans)
```

Out[48]:
```
979
```

In [49]:
```
# examine the first five list elements
booleans[0:5]
```

Out[49]:
```
[False, False, True, False, False]
```

In [50]:
```
# convert the list to a Series
is_long = pd.Series(booleans)
is_long.head(10)
```

Out[50]:
```
0    False
1    False
2     True
3    False
4    False
5    False
6    False
7     True
8    False
9    False
dtype: bool
```

In [51]:
```
# use bracket notation with the boolean Series to tell the DataFrame which r
movies[is_long]
```

Out[51]:

	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...

	star_rating	title	content_rating	genre	duration	actors_list
767	7.6	It's a Mad, Mad, Mad, Mad World	APPROVED	Action	205	[u'Spencer Tracy', u'Milton Berle', u'Ethel Me...

In [52]:
```
movies.duration >= 200
```

Out[52]:
```
0      False
1      False
2       True
3      False
4      False
       ...  
974    False
975    False
976    False
977    False
978    False
Name: duration, Length: 979, dtype: bool
```

In [53]:

```
movies[movies.duration >= 200]
```

Out[53]:

	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...

	star_rating	title	content_rating	genre	duration	actors_list
767	7.6	It's a Mad, Mad, Mad, Mad World	APPROVED	Action	205	[u'Spencer Tracy', u'Milton Berle', u'Ethel Me...

In [54]:
```
# simplify the steps above: no need to write a for loop to create 'is_long' s
is_long = movies.duration >= 200
movies[is_long]

# or equivalently, write it in one line (no need to create the 'is_long' obje
movies[movies.duration >= 200]
```

Out[54]:

	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...

	star_rating	title	content_rating	genre	duration	actors_list
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...

In [55]:

```
# select the 'genre' Series from the filtered DataFrame
movies[movies.duration >= 200].genre

# or equivalently, use the 'loc' method
movies.loc[0:2,'star_rating']  # t OR f  OR INDEX , COLUMN NAME
```

Out[55]:

```
0    9.3
1    9.2
2    9.1
Name: star_rating, dtype: float64
```

Documentation for `loc` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html)

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

In [56]:

```
# read a dataset of top-rated IMDb movies into a DataFrame
movies = pd.read_csv('http://bit.ly/imdbratings')
movies.head()
```

Out[56]:

	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 [57]:

```
# filter the DataFrame to only show movies with a 'duration' of at least 200
movies[movies.duration >= 200]
```

Out[57]:

	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...

	star_rating	title	content_rating	genre	duration	actors_list
767	7.6	It's a Mad, Mad, Mad, Mad World	APPROVED	Action	205	[u'Spencer Tracy', u'Milton Berle', u'Ethel Me...

Understanding **logical operators:**

- **and** : True only if **both sides** of the operator are True
- **or** : True if **either side** of the operator is True

In [58]:
```
# demonstration of the 'and' operator
print(True and True)
print(True and False)
print(False and False)
```

```
True
False
False
```

In [59]:
```
# demonstration of the 'or' operator
print(True or True)
print(True or False)
print(False or False)
```

```
True
True
False
```

Rules for specifying **multiple filter criteria** in pandas:

- use **&** instead of **and**
- use **|** instead of **or**
- add **parentheses** around each condition to specify evaluation order

Goal: Further filter the DataFrame of long movies (duration >= 200) to only show movies which also have a 'genre' of 'Drama'

In [60]:
```
# CORRECT: use the '&' operator to specify that both conditions are required
movies[(movies.duration >=200) & (movies.genre == 'Drama')]
```

Out[60]:

	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 [61]:
```
# INCORRECT: using the '|' operator would have shown movies that are either l
movies[(movies.duration >=200) | (movies.genre == 'Drama')].head()
```

Out[61]:

	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']

Goal: Filter the original DataFrame to show movies with a 'genre' of 'Crime' or 'Drama' or 'Action'

```
In [62]:  # use the '|' operator to specify that a row can match any of the three crite
          movies[(movies.genre == 'Crime') | (movies.genre == 'Drama') | (movies.genre

          # or equivalently, use the 'isin' method
          movies[movies.genre.isin(['Crime', 'Drama', 'Action'])].head(10)

          movies[movies.genre.isin(['Crime','Action','Drama'])]
```

Out[62]:

	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....
...	...	...	...	...	...	...
970	7.4	Wonder Boys	R	Drama	107	[u'Michael Douglas', u'Tobey Maguire', u'Franc...
972	7.4	Blue Valentine	NC-17	Drama	112	[u'Ryan Gosling', u'Michelle Williams', u'John...
973	7.4	The Cider House Rules	PG-13	Drama	126	[u'Tobey Maguire', u'Charlize Theron', u'Micha...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

538 rows × 6 columns

Documentation for **`isin`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.isin.html)

When reading from a file, how do I read in only a subset of the columns?

In [63]:
```
# read a dataset of UFO reports into a DataFrame, and check the columns
import pandas as pd
ufo = pd.read_csv('ufo.csv')
ufo.columns
```

Out[63]:
```
Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time'], dtype
='object')
```

In [64]:
```
# specify which columns to include by name
ufo = pd.read_csv('ufo.csv', usecols=['City', 'State'])
print(ufo.columns)
# or equivalently, specify columns by position
ufo = pd.read_csv('ufo.csv', usecols=[0, 4])
ufo.columns
```

```
Index(['City', 'State'], dtype='object')
```

Out[64]:
```
Index(['City', 'Time'], dtype='object')
```

Question: When reading from a file, how do I read in only a subset of the rows?

In [65]:
```
# specify how many rows to read
ufo = pd.read_csv('ufo.csv', nrows=3)
ufo
```

Out[65]:

	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

Documentation for **`read_csv`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)

Documentation for **`select_dtypes`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.select_dtypes.html)

How do I use string methods in pandas?

In [66]:
```
# read a dataset of Chipotle orders into a DataFrame
orders = pd.read_table('http://bit.ly/chiporders')
orders.head()
```

Out[66]:

	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

In [67]:
```
# normal way to access string methods in Python
'hello'.upper()
```

Out[67]:
```
'HELLO'
```

In [68]:
```
# string methods for pandas Series are accessed via 'str'
orders.item_name.str.upper().head()
```

Out[68]:
```
0             CHIPS AND FRESH TOMATO SALSA
1                                     IZZE
2                         NANTUCKET NECTAR
3    CHIPS AND TOMATILLO-GREEN CHILI SALSA
4                             CHICKEN BOWL
Name: item_name, dtype: object
```

In [69]:
```
# string method 'contains' checks for a substring and returns a boolean Serie
orders.item_name.str.contains('Chicken').head()
```

Out[69]:
```
0    False
1    False
2    False
3    False
4     True
Name: item_name, dtype: bool
```

```
In [70]:  # use the boolean Series to filter the DataFrame
          orders[orders.item_name.str.contains('Chicken')].head()
```

Out[70]:

	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

```
In [71]:  # string methods can be chained together
          orders.choice_description.str.replace('[', '').str.replace(']', '').str.repla
```

```
Out[71]: 0                                                  NaN
         1                                           Clementine
         2                                                Apple
         3                                                  NaN
         4     Tomatillo-Red Chili Salsa (Hot) Black Beans Ri...
         Name: choice_description, dtype: object
```

```
In [72]:  # many pandas string methods support regular expressions (regex)
          orders.choice_description.str.replace('[\[\]]', '').head()
```

```
Out[72]: 0                                                  NaN
         1                                           Clementine
         2                                                Apple
         3                                                  NaN
         4     Tomatillo-Red Chili Salsa (Hot), Black Beans, ...
         Name: choice_description, dtype: object
```

String handling section (http://pandas.pydata.org/pandas-docs/stable/api.html#string-handling) of the pandas API reference

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

In [73]:
```python
# read a dataset of alcohol consumption into a DataFrame
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
drinks.head()
```

Out[73]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0	0	0	0.0	
1	Albania	89	132	54	4.9	Eu
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	Eu
4	Angola	217	57	45	5.9	A

In [74]:
```python
# examine the data type of each Series
drinks.dtypes
```

Out[74]:
```
country                          object
beer_servings                     int64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
dtype: object
```

In [75]:
```python
# change the data type of an existing Series
drinks['beer_servings'] = drinks.beer_servings.astype(float)
drinks.dtypes
```

Out[75]:
```
country                          object
beer_servings                   float64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
dtype: object
```

Documentation for **`astype`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.astype.html)

In [76]:
```
# alternatively, change the data type of a Series while reading in a file
drinks = pd.read_csv('http://bit.ly/drinksbycountry', dtype={'beer_servings':
drinks.dtypes
```

Out[76]:
```
country                          object
beer_servings                   float64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
dtype: object
```

In [77]:
```
# read a dataset of Chipotle orders into a DataFrame
orders = pd.read_table('http://bit.ly/chiporders')
orders.head()
```

Out[77]:

	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

In [78]:
```
# examine the data type of each Series
orders.dtypes
```

Out[78]:
```
order_id               int64
quantity               int64
item_name             object
choice_description    object
item_price            object
dtype: object
```

In [79]:
```
# convert a string to a number in order to do math
orders.item_price.str.replace('$', '').astype(float).mean()
```

Out[79]:
```
7.464335785374397
```

In [80]:
```
# string method 'contains' checks for a substring and returns a boolean Serie
orders.item_name.str.contains('Chicken').head()
```

Out[80]:
```
0    False
1    False
2    False
3    False
4     True
Name: item_name, dtype: bool
```

In [81]:
```
# convert a boolean Series to an integer (False = 0, True = 1)
orders.item_name.str.contains('Chicken').astype(int).head()
```

Out[81]:
```
0    0
1    0
2    0
3    0
4    1
Name: item_name, dtype: int32
```

When should I use a "groupby" in pandas?

In [82]:
```
# read a dataset of alcohol consumption into a DataFrame
import pandas as pd
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
drinks.head()
```

Out[82]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0	0	0	0.0	
1	Albania	89	132	54	4.9	Eu
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	Eu
4	Angola	217	57	45	5.9	A

In [83]:
```
drinks.describe()
```

Out[83]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
count	193.000000	193.000000	193.000000	193.000000
mean	106.160622	80.994819	49.450777	4.717098
std	101.143103	88.284312	79.697598	3.773298
min	0.000000	0.000000	0.000000	0.000000
25%	20.000000	4.000000	1.000000	1.300000
50%	76.000000	56.000000	8.000000	4.200000
75%	188.000000	128.000000	59.000000	7.200000
max	376.000000	438.000000	370.000000	14.400000

In [84]:
```
# calculate the mean beer servings across the entire dataset
drinks.beer_servings.mean()
```

Out[84]:
```
106.16062176165804
```

In [85]:
```
# calculate the mean beer servings just for countries in Africa
drinks[drinks.continent=='Africa'].beer_servings.mean()
```

Out[85]:
```
61.471698113207545
```

In [86]:
```
drinks.continent.unique()
```

Out[86]:
```
array(['Asia', 'Europe', 'Africa', 'North America', 'South America',
       'Oceania'], dtype=object)
```

In [87]:
```
# calculate the mean beer servings for each continent
drinks.groupby('continent').beer_servings.mean()
```

Out[87]:
```
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
```

Documentation for **groupby** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html)

In [88]:
```
# other aggregation functions (such as 'max') can also be used with groupby
drinks.groupby('continent').beer_servings.max()
```

Out[88]:
```
continent
Africa           376
Asia             247
Europe           361
North America    285
Oceania          306
South America    333
Name: beer_servings, dtype: int64
```

In [89]:
```python
# multiple aggregation functions can be applied simultaneously
drinks.groupby('continent').agg(['count', 'mean', 'min', 'max'])
```

Out[89]:

	beer_servings				spirit_servings				wine_servings		
	count	mean	min	max	count	mean	min	max	count	mean	mi
continent											
Africa	53	61.471698	0	376	53	16.339623	0	152	53	16.264151	
Asia	44	37.045455	0	247	44	60.840909	0	326	44	9.068182	
Europe	45	193.777778	0	361	45	132.555556	0	373	45	142.222222	
North America	23	145.434783	1	285	23	165.739130	68	438	23	24.521739	
Oceania	16	89.687500	0	306	16	58.437500	0	254	16	35.625000	
South America	12	175.083333	93	333	12	114.750000	25	302	12	62.416667	

Documentation for **agg** [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.core.groupby.DataFrameGroupBy.agg.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.core.groupby.DataFrameGroupBy.agg.html)

In [90]:
```python
# specifying a column to which the aggregation function should be applied is
drinks.groupby('continent').mean()
```

Out[90]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
continent				
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 [91]:
```
drinks.describe()
```

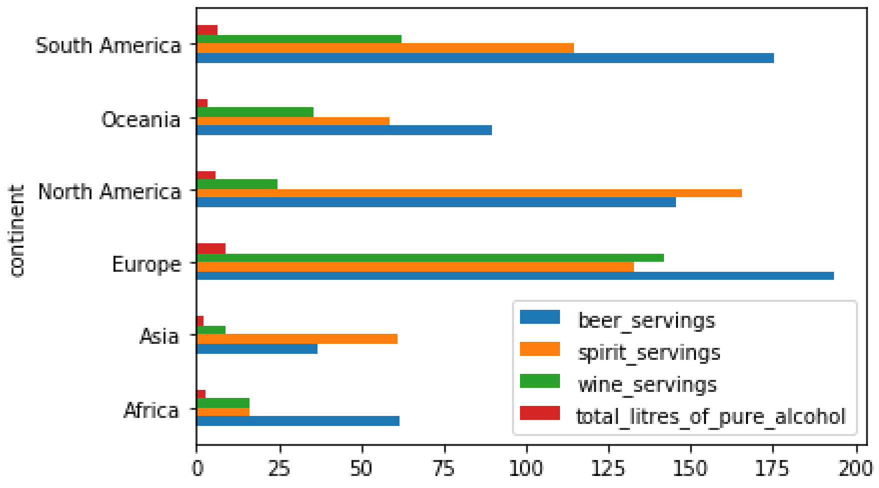
Out[91]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
count	193.000000	193.000000	193.000000	193.000000
mean	106.160622	80.994819	49.450777	4.717098
std	101.143103	88.284312	79.697598	3.773298
min	0.000000	0.000000	0.000000	0.000000
25%	20.000000	4.000000	1.000000	1.300000
50%	76.000000	56.000000	8.000000	4.200000
75%	188.000000	128.000000	59.000000	7.200000
max	376.000000	438.000000	370.000000	14.400000

In [92]:
```
# allow plots to appear in the notebook
import matplotlib.pyplot as plt
%matplotlib inline
```

In [93]:
```
# side-by-side bar plot of the DataFrame directly above
drinks.groupby('continent').mean().plot(kind='barh')
```

Out[93]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11ff729a3c8>
```



Documentation for **`plot`** [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html)

[Back to top]

In [94]:
```
# count how many times each value in the Series occurs
movies.genre.value_counts()
```

Out[94]:
```
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
```

Documentation for **`value_counts`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.value_counts.html)

In [95]:
```
movies.head(10)
```
Out[95]:

	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....
5	8.9	12 Angry Men	NOT RATED	Drama	96	[u'Henry Fonda', u'Lee J. Cobb', u'Martin Bals...
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western	161	[u'Clint Eastwood', u'Eli Wallach', u'Lee Van ...
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...
9	8.9	Fight Club	R	Drama	139	[u'Brad Pitt', u'Edward Norton', u'Helena Bonh...

In [96]:
```
# display percentages instead of raw counts
movies.genre.value_counts(normalize=True)
```

Out[96]:
```
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 [97]:
```
# 'value_counts' (like many pandas methods) outputs a Series
type(movies.genre.value_counts())
```

Out[97]:
```
pandas.core.series.Series
```

In [98]:
```
# display the unique values in the Series
movies.genre.unique()
```

Out[98]:
```
array(['Crime', 'Action', 'Drama', 'Western', 'Adventure', 'Biography',
       'Comedy', 'Animation', 'Mystery', 'Horror', 'Film-Noir', 'Sci-Fi',
       'History', 'Thriller', 'Family', 'Fantasy'], dtype=object)
```

In [99]:
```
# count the number of unique values in the Series
movies.genre.nunique()
```

Out[99]:
```
16
```

Documentation for **`unique`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.unique.html) and **`nunique`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.nunique.html)

In [100]:
```
print(movies.content_rating.nunique())
movies.genre.nunique()
```

```
12
```

Out[100]:
```
16
```

```
In [101]:  # compute a cross-tabulation of two Series
            pd.crosstab(movies.genre, movies.content_rating,margins=True)
```

Out[101]:

content_rating	APPROVED	G	GP	NC-17	NOT RATED	PASSED	PG	PG-13	R	TV-MA	UNRATED
genre											
Action	3	1	1	0	4	1	11	44	67	0	3
Adventure	3	2	0	0	5	1	21	23	17	0	2
Animation	3	20	0	0	3	0	25	5	5	0	1
Biography	1	2	1	0	1	0	6	29	36	0	0
Comedy	9	2	1	1	16	3	23	23	73	0	4
Crime	6	0	0	1	7	1	6	4	87	0	11
Drama	12	3	0	4	24	1	25	55	143	1	9
Family	0	1	0	0	0	0	1	0	0	0	0
Fantasy	0	0	0	0	0	0	0	0	1	0	0
Film-Noir	1	0	0	0	1	0	0	0	0	0	1
History	0	0	0	0	0	0	0	0	0	0	1
Horror	2	0	0	1	1	0	1	2	16	0	5
Mystery	4	1	0	0	1	0	1	2	6	0	1
Sci-Fi	1	0	0	0	0	0	0	1	3	0	0
Thriller	1	0	0	0	0	0	1	0	3	0	0
Western	1	0	0	0	2	0	2	1	3	0	0
All	47	32	3	7	65	7	123	189	460	1	38

Documentation for **`crosstab`** [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.crosstab.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.crosstab.html)

Exploring a numeric Series:

```
In [102]:  # calculate various summary statistics
            movies.duration.describe()
```

```
Out[102]: 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
```

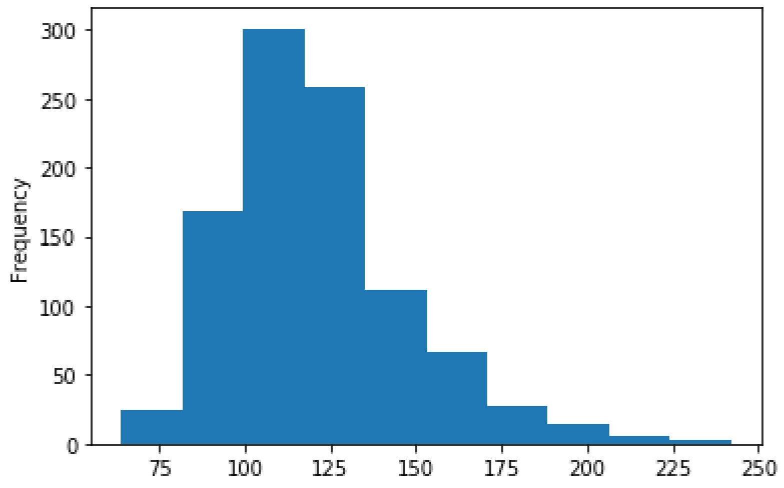
In [103]:
```
# many statistics are implemented as Series methods
movies.duration.mean()
```
Out[103]:
```
120.97957099080695
```

Documentation for **`mean`** [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.mean.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.mean.html)

In [104]:
```
# 'value_counts' is primarily useful for categorical data, not numerical datc
movies.duration.value_counts().head()
```
Out[104]:
```
112    23
113    22
102    20
101    20
129    19
Name: duration, dtype: int64
```

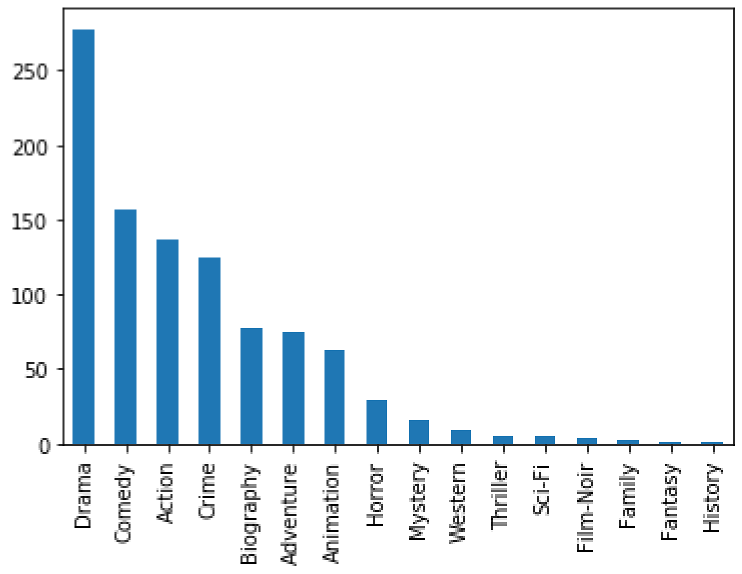
In [105]:
```
# allow plots to appear in the notebook
%matplotlib inline
```

In [106]:
```
# histogram of the 'duration' Series (shows the distribution of a numerical v
movies.duration.plot(kind='hist')
```
Out[106]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11ff8beabc8>
```



```
In [107]:  # bar plot of the 'value_counts' for the 'genre' Series
           movies.genre.value_counts().plot(kind='bar')
```

```
Out[107]: <matplotlib.axes._subplots.AxesSubplot at 0x11ff8c87488>
```



Documentation for **`plot`** [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.plot.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.plot.html)

How do I handle missing values in pandas?

In [108]:
```
# read a dataset of UFO reports into a DataFrame
ufo = pd.read_csv('http://bit.ly/uforeports')
ufo.tail()
```

Out[108]:

	City	Colors Reported	Shape Reported	State	Time
18236	Grant Park	NaN	TRIANGLE	IL	12/31/2000 23:00
18237	Spirit Lake	NaN	DISK	IA	12/31/2000 23:00
18238	Eagle River	NaN	NaN	WI	12/31/2000 23:45
18239	Eagle River	RED	LIGHT	WI	12/31/2000 23:45
18240	Ybor	NaN	OVAL	FL	12/31/2000 23:59

What does "NaN" mean?

- "NaN" is not a string, rather it's a special value: `numpy.nan` .
- It stands for "Not a Number" and indicates a **missing value**.
- `read_csv` detects missing values (by default) when reading the file, and replaces them with this special value.

Documentation for `read_csv` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)

In [109]:
```
# 'isnull' returns a DataFrame of booleans (True if missing, False if not mis
ufo.isnull().tail()
```

Out[109]:

	City	Colors Reported	Shape Reported	State	Time
18236	False	True	False	False	False
18237	False	True	False	False	False
18238	False	True	True	False	False
18239	False	False	False	False	False
18240	False	True	False	False	False

In [110]:
```
# 'nonnull' returns the opposite of 'isnull' (True if not missing, False if n
ufo.notnull().tail()
```

Out[110]:

	City	Colors Reported	Shape Reported	State	Time
18236	True	False	True	True	True
18237	True	False	True	True	True
18238	True	False	False	True	True
18239	True	True	True	True	True
18240	True	False	True	True	True

Documentation for `isnull` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.isnull.html) and **`notnull`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.notnull.html)

In [111]:
```
# count the number of missing values in each Series
ufo.isnull().sum()
```

Out[111]:
```
City                  25
Colors Reported    15359
Shape Reported      2644
State                  0
Time                   0
dtype: int64
```

This calculation works because:

1. The **`sum`** method for a DataFrame operates on **`axis=0`** by default (and thus produces column sums).
2. In order to add boolean values, pandas converts **`True`** to **1** and **`False`** to **0**.

In [112]:
```
# use the 'isnull' Series method to filter the DataFrame rows
ufo[ufo.City.isnull()].head()
```

Out[112]:

	City	Colors Reported	Shape Reported	State	Time
21	NaN	NaN	NaN	LA	8/15/1943 0:00
22	NaN	NaN	LIGHT	LA	8/15/1943 0:00
204	NaN	NaN	DISK	CA	7/15/1952 12:30
241	NaN	BLUE	DISK	MT	7/4/1953 14:00
613	NaN	NaN	DISK	NV	7/1/1960 12:00

How to handle missing values depends on the dataset as well as the nature of your analysis. Here are some options:

In [113]:
```
# examine the number of rows and columns
ufo.shape
```

Out[113]:
```
(18241, 5)
```

In [114]:
```
# if 'any' values are missing in a row, then drop that row
ufo.dropna(how='any').shape
```

Out[114]:
```
(2486, 5)
```

Documentation for **`dropna`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html)

In [115]:

```
# 'inplace' parameter for 'dropna' is False by default, thus rows were only d
ufo.shape
```

Out[115]:

```
(18241, 5)
```

In [116]:

```
# if 'all' values are missing in a row, then drop that row (none are dropped
ufo.dropna(how='all').shape
```

Out[116]:

```
(18241, 5)
```

In [117]:

```
# if 'any' values are missing in a row (considering only 'City' and 'Shape Re
ufo.dropna(subset=['City', 'Shape Reported'], how='any').shape
```

Out[117]:

```
(15576, 5)
```

In [118]:

```
# if 'all' values are missing in a row (considering only 'City' and 'Shape Re
ufo.dropna(subset=['City', 'Shape Reported'], how='all').shape
```

Out[118]:

```
(18237, 5)
```

In [119]:

```
# 'value_counts' does not include missing values by default
ufo['Shape Reported'].value_counts().head()
```

Out[119]:

```
LIGHT       2803
DISK        2122
TRIANGLE    1889
OTHER       1402
CIRCLE      1365
Name: Shape Reported, dtype: int64
```

In [120]:

```
# explicitly include missing values
ufo['City'].value_counts(dropna=True).head()
```

Out[120]:

```
Seattle          187
New York City    161
Phoenix          137
Houston          108
Las Vegas        105
Name: City, dtype: int64
```

Documentation for **`value_counts`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.value_counts.html)

In [121]:

```
# fill in missing values with a specified value
ufo['Shape Reported'].fillna(value='VARIOUS', inplace=True)
```

Documentation for **`fillna`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.fillna.html)

In [122]:
```
# confirm that the missing values were filled in
ufo['Shape Reported'].value_counts().head()
```

Out[122]:
```
VARIOUS     2977
LIGHT       2803
DISK        2122
TRIANGLE    1889
OTHER       1402
Name: Shape Reported, dtype: int64
```

[Working with missing data in pandas (http://pandas.pydata.org/pandas-docs/stable/missing_data.html)](http://pandas.pydata.org/pandas-docs/stable/missing_data.html)

What do I need to know about the pandas index?

In [123]:
```
# read a dataset of alcohol consumption into a DataFrame
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
drinks.head()
```

Out[123]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0	0	0	0.0	
1	Albania	89	132	54	4.9	Eu
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	Eu
4	Angola	217	57	45	5.9	A

In [124]:
```
# every DataFrame has an index (sometimes called the "row labels")
drinks.index
```

Out[124]:
```
RangeIndex(start=0, stop=193, step=1)
```

In [125]:
```
# column names are also stored in a special "index" object
drinks.columns
```

Out[125]:
```
Index(['country', 'beer_servings', 'spirit_servings', 'wine_servings',
       'total_litres_of_pure_alcohol', 'continent'],
      dtype='object')
```

In [126]:
```
# neither the index nor the columns are included in the shape
drinks.shape
```

Out[126]:
```
(193, 6)
```

In [127]:

```python
# index and columns both default to integers if you don't define them
pd.read_table('http://bit.ly/movieusers', header=None, sep='|').head()
```

Out[127]:

	0	1	2	3	4
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

What is the index used for?

1. identification
2. selection
3. alignment

In [128]:
```
# identification: index remains with each row when filtering the DataFrame
drinks[drinks.continent=='South America']
```

Out[128]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	cont
6	Argentina	193	25	221	8.3	An
20	Bolivia	167	41	8	3.8	An
23	Brazil	245	145	16	7.2	An
35	Chile	130	124	172	7.6	An
37	Colombia	159	76	3	4.2	An
52	Ecuador	162	74	3	4.2	An
72	Guyana	93	302	1	7.1	An
132	Paraguay	213	117	74	7.3	An
133	Peru	163	160	21	6.1	An
163	Suriname	128	178	7	5.6	An
185	Uruguay	115	35	220	6.6	An
188	Venezuela	333	100	3	7.7	An

In [129]:
```
# selection: select a portion of the DataFrame using the index
drinks.loc[0:10, 'beer_servings']
```

Out[129]:
```
0       0
1      89
2      25
3     245
4     217
5     102
6     193
7      21
8     261
9     279
10     21
Name: beer_servings, dtype: int64
```

Documentation for **loc** [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html)

In [130]:
```
# set an existing column as the index
drinks.set_index('country', inplace=True)
drinks.head()
```

Out[130]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continen
country					
Afghanistan	0	0	0	0.0	Asi
Albania	89	132	54	4.9	Europ
Algeria	25	0	14	0.7	Afric
Andorra	245	138	312	12.4	Europ
Angola	217	57	45	5.9	Afric

Documentation for **set_index** [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.set_index.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.set_index.html)

In [131]:
```
# 'country' is now the index
drinks.index
```

Out[131]:
```
Index(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antigua & Barbuda', 'Argentina', 'Armenia', 'Australia', 'Austria',
       ...
       'Tanzania', 'USA', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela',
       'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype='object', name='country', length=193)
```

In [132]:
```
# 'country' is no longer a column
drinks.columns
```

Out[132]:
```
Index(['beer_servings', 'spirit_servings', 'wine_servings',
       'total_litres_of_pure_alcohol', 'continent'],
      dtype='object')
```

In [133]:
```
# 'country' data is no longer part of the DataFrame contents
drinks.shape
```

Out[133]:
```
(193, 5)
```

In [134]:
```
# country name can now be used for selection
drinks.loc['Brazil', 'beer_servings']
```

Out[134]:
```
245
```

In [135]:
```python
# index name is optional
drinks.index.name = None
drinks.head()
```

Out[135]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continen
Afghanistan	0	0	0	0.0	Asi
Albania	89	132	54	4.9	Europ
Algeria	25	0	14	0.7	Afric
Andorra	245	138	312	12.4	Europ
Angola	217	57	45	5.9	Afric

In [136]:
```python
# restore the index name, and move the index back to a column
drinks.index.name = 'country'
drinks.reset_index(inplace=True)
drinks.head()
```

Out[136]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0	0	0	0.0	
1	Albania	89	132	54	4.9	Eu
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	Eu
4	Angola	217	57	45	5.9	A

Documentation for **`reset_index`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.reset_index.html)

In [137]:
```python
# many DataFrame methods output a DataFrame
drinks.describe()
```

Out[137]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol
count	193.000000	193.000000	193.000000	193.000000
mean	106.160622	80.994819	49.450777	4.717098
std	101.143103	88.284312	79.697598	3.773298
min	0.000000	0.000000	0.000000	0.000000
25%	20.000000	4.000000	1.000000	1.300000
50%	76.000000	56.000000	8.000000	4.200000
75%	188.000000	128.000000	59.000000	7.200000
max	376.000000	438.000000	370.000000	14.400000

In [138]:
```
# you can interact with any DataFrame using its index and columns
drinks.describe().loc['25%', 'beer_servings']
```

Out[138]:
```
20.0
```

[Indexing and selecting data (http://pandas.pydata.org/pandas-docs/stable/indexing.html)](http://pandas.pydata.org/pandas-docs/stable/indexing.html)

[[Back to top]]

How do I select multiple rows and columns from a pandas DataFrame?

In [139]:
```
# read a dataset of UFO reports into a DataFrame
ufo = pd.read_csv('http://bit.ly/uforeports')
ufo.head(3)
```

Out[139]:

	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

The `loc` [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html) method is used to select rows and columns by **label**. You can pass it:

- A single label
- A list of labels
- A slice of labels
- A boolean Series
- A colon (which indicates "all labels")

In [140]:
```
# row 0, all columns
ufo.loc[0, :]
```

Out[140]:
```
City                           Ithaca
Colors Reported                   NaN
Shape Reported               TRIANGLE
State                              NY
Time                   6/1/1930 22:00
Name: 0, dtype: object
```

In [141]:
```
# rows 0 and 1 and 2, all columns
ufo.loc[[0, 1, 2], :]
```

Out[141]:

	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

In [142]:
```
# rows 0 through 2 (inclusive), all columns
ufo.loc[0:2, :]
```

Out[142]:

	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

In [143]:
```
# this implies "all columns", but explicitly stating "all columns" is better
ufo.loc[0:2]
```

Out[143]:

	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

In [144]:
```
# rows 0 through 2 (inclusive), column 'City'
ufo.loc[0:2, 'City']
```

Out[144]:
```
0         Ithaca
1    Willingboro
2        Holyoke
Name: City, dtype: object
```

In [145]:
```
# rows 0 through 2 (inclusive), columns 'City' and 'State'
ufo.loc[0:2, ['City', 'State']]
```

Out[145]:

	City	State
0	Ithaca	NY
1	Willingboro	NJ
2	Holyoke	CO

In [146]:
```
# accomplish the same thing using double brackets - but using 'loc' is prefer
ufo[['City', 'State']].head(3)
```

Out[146]:

	City	State
0	Ithaca	NY
1	Willingboro	NJ
2	Holyoke	CO

In [147]:
```
# rows 0 through 2 (inclusive), columns 'City' through 'State' (inclusive)
ufo.loc[0:2, 'City':'State']
```

Out[147]:

	City	Colors Reported	Shape Reported	State
0	Ithaca	NaN	TRIANGLE	NY
1	Willingboro	NaN	OTHER	NJ
2	Holyoke	NaN	OVAL	CO

In [148]:
```
# accomplish the same thing using 'head' and 'drop'
ufo.head(3).drop('Time', axis=1)
```

Out[148]:

	City	Colors Reported	Shape Reported	State
0	Ithaca	NaN	TRIANGLE	NY
1	Willingboro	NaN	OTHER	NJ
2	Holyoke	NaN	OVAL	CO

In [149]:
```
# rows in which the 'City' is 'Oakland', column 'State'
ufo.loc[ufo.City=='Oakland', 'State']
```

Out[149]:
```
1694     CA
2144     CA
4686     MD
7293     CA
8488     CA
8768     CA
10816    OR
10948    CA
11045    CA
12322    CA
12941    CA
16803    MD
17322    CA
Name: State, dtype: object
```

In [150]:

```
# accomplish the same thing using "chained indexing" - but using 'loc' is pre
ufo[ufo.City=='Oakland'].State
```

Out[150]:

```
1694     CA
2144     CA
4686     MD
7293     CA
8488     CA
8768     CA
10816    OR
10948    CA
11045    CA
12322    CA
12941    CA
16803    MD
17322    CA
Name: State, dtype: object
```

```
The [**`iloc`**](http://pandas.pydata.org/pandas-
docs/stable/generated/pandas.DataFrame.iloc.html) method is used to select
rows and columns by **integer position**. You can pass it:

- A single integer position
- A list of integer positions
- A slice of integer positions
- A colon (which indicates "all integer positions")
```

In [151]:

```
ufo.head()
```

Out[151]:

	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 [152]:

```
# rows in positions 0 and 1, columns in positions 0 and 3
ufo.iloc[[0, 1], [0, 3]]
```

Out[152]:

	City	State
0	Ithaca	NY
1	Willingboro	NJ

In [153]:
```
# rows in positions 0 through 2 (exclusive), columns in positions 0 through 4
ufo.iloc[0:2, 0:4]
```

Out[153]:

	City	Colors Reported	Shape Reported	State
0	Ithaca	NaN	TRIANGLE	NY
1	Willingboro	NaN	OTHER	NJ

In [154]:
```
# rows in positions 0 through 2 (exclusive), all columns
ufo.iloc[0:2, :]
```

Out[154]:

	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 [155]:
```
# accomplish the same thing - but using 'iloc' is preferred since it's more e
ufo[0:2]
```

Out[155]:

	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

```
The [**`ix`**](http://pandas.pydata.org/pandas-
docs/stable/generated/pandas.DataFrame.ix.html) method is used to select rows
and columns by **label or integer position**, and should only be used when you
need to mix label-based and integer-based selection in the same call.
```

In [305]:
```
# read a dataset of alcohol consumption into a DataFrame and set 'country' as
import pandas as pd
drinks = pd.read_csv('http://bit.ly/drinksbycountry', index_col='country')
drinks.head()
```

Out[305]:

	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continen
country					
Afghanistan	0	0	0	0.0	Asi
Albania	89	132	54	4.9	Europ
Algeria	25	0	14	0.7	Afric
Andorra	245	138	312	12.4	Europ
Angola	217	57	45	5.9	Afric

In [282]:

```
# row with label 'Albania', column in position 0
#drinks.ix['Andorra', 0]--->Since IX is now not in use
```

In [284]:

```
# row in position 1, column with label 'beer_servings'
drinks.ix[1, 'beer_servings']
```

```
C:\Users\Meghaa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Future
Warning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-ind
exer-is-deprecated (http://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#ix-indexer-is-deprecated)
```

Out[284]:

```
89.0
```

```
**Rules for using numbers with `ix`:**

- If the index is **strings**, numbers are treated as **integer positions**,
and thus slices are **exclusive** on the right.
- If the index is **integers**, numbers are treated as **labels**, and thus
slices are **inclusive**.
```

In [298]:

```
# rows 'Albania' through 'Andorra' (inclusive), columns in positions 0 throug
drinks.ix['Albania':'Andorra', 0:2]
```

```
C:\Users\Meghaa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Future
Warning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-ind
exer-is-deprecated (http://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#ix-indexer-is-deprecated)
```

Out[298]:

country	beer_servings

In [299]:
```
# rows 0 through 2 (inclusive), columns in positions 0 through 2 (exclusive)
ufo.ix[0:2, 0:2]
```

```
C:\Users\Meghaa\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Future
Warning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-ind
exer-is-deprecated (http://pandas.pydata.org/pandas-docs/stable/user_guide/
indexing.html#ix-indexer-is-deprecated)
```

Out[299]:

	City	Colors Reported
0	Ithaca	NaN
1	Willingboro	NaN
2	Holyoke	NaN

In [290]:
```
# use the 'category' data type (new in pandas 0.15) to store the 'continent'
drinks['continent'] = drinks.continent.astype('category')
drinks.dtypes
```

Out[290]:
```
country                          object
beer_servings                   float64
spirit_servings                 float64
wine_servings                   float64
total_litres_of_pure_alcohol    float64
continent                      category
dtype: object
```

In [291]:
```
# 'continent' Series appears to be unchanged
drinks.continent.head()
```

Out[291]:
```
0      Asia
1    Europe
2    Africa
3    Europe
4    Africa
Name: continent, dtype: category
Categories (6, object): [Africa, Asia, Europe, North America, Oceania, Sout
h America]
```

In [163]:
```
# strings are now encoded (0 means 'Africa', 1 means 'Asia', 2 means 'Europe
drinks.continent.cat.codes.head()
```

Out[163]:
```
country
Afghanistan    1
Albania        2
Algeria        0
Andorra        2
Angola         0
dtype: int8
```

In [164]:
```
# memory usage has been drastically reduced
drinks.memory_usage(deep=True)
```

Out[164]:
```
Index                           17708
beer_servings                    1544
spirit_servings                  1544
wine_servings                    1544
total_litres_of_pure_alcohol     1544
continent                         744
dtype: int64
```

In [315]:
```
# repeat this process for the 'country' Series
drinks['country'] = drinks.country.astype('category')
drinks.memory_usage(deep=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-315-6edf4187ca71> in <module>
      1 # repeat this process for the 'country' Series
----> 2 drinks['country'] = drinks.country.astype('category')
      3 drinks.memory_usage(deep=True)

~\Anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, n
ame)
   5177             if self._info_axis._can_hold_identifiers_and_holds_name
(name):
   5178                 return self[name]
-> 5179             return object.__getattribute__(self, name)
   5180 
   5181     def __setattr__(self, name, value):

AttributeError: 'DataFrame' object has no attribute 'country'
```

In [304]:
```
# memory usage increased because we created 193 categories
drinks.country.cat.categories
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-304-1af6b0e83ad5> in <module>
      1 # memory usage increased because we created 193 categories
----> 2 drinks.country.cat.categories

~\Anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, n
ame)
   5177             if self._info_axis._can_hold_identifiers_and_holds_name
(name):
   5178                 return self[name]
-> 5179             return object.__getattribute__(self, name)
   5180 
   5181     def __setattr__(self, name, value):

AttributeError: 'DataFrame' object has no attribute 'country'
```

The **category** data type should only be used with a string Series that has a **small number of possible values**.

In [300]:
```
# create a small DataFrame from a dictionary
df = pd.DataFrame({'ID':[100, 101, 102, 103], 'quality':['good', 'very good',
df
```

Out[300]:

	ID	quality
0	100	good
1	101	very good
2	102	good
3	103	excellent

In [301]:
```
# sort the DataFrame by the 'quality' Series (alphabetical order)
df.sort_values('quality')
```

Out[301]:

	ID	quality
3	103	excellent
0	100	good
2	102	good
1	101	very good

```
In [317]:  # define a logical ordering for the categories
           df['quality'] = df.quality.astype('category', categories=['good', 'very good'
           df.quality
```

```
---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
<ipython-input-317-9cdaae1356ea> in <module>
      1 # define a logical ordering for the categories
----> 2 df['quality'] = df.quality.astype('category', categories=['good',
'very good', 'excellent'], ordered=True)
      3 df.quality

~\Anaconda3\lib\site-packages\pandas\core\generic.py in astype(self, dtyp
e, copy, errors, **kwargs)
   5880             # else, only a single dtype is given
   5881             new_data = self._data.astype(
-> 5882                 dtype=dtype, copy=copy, errors=errors, **kwargs
   5883             )
   5884             return self._constructor(new_data).__finalize__(self)

~\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in astype
(self, dtype, **kwargs)
    579
    580     def astype(self, dtype, **kwargs):
--> 581         return self.apply("astype", dtype=dtype, **kwargs)
    582
    583     def convert(self, **kwargs):

~\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in apply
(self, f, axes, filter, do_integrity_check, consolidate, **kwargs)
    436                     kwargs[k] = obj.reindex(b_items, axis=axis, c
opy=align_copy)
    437
--> 438             applied = getattr(b, f)(**kwargs)
    439             result_blocks = _extend_blocks(applied, result_blocks
)
    440

~\Anaconda3\lib\site-packages\pandas\core\internals\blocks.py in astype(s
elf, dtype, copy, errors, values, **kwargs)
    557
    558     def astype(self, dtype, copy=False, errors="raise", values=No
ne, **kwargs):
--> 559         return self._astype(dtype, copy=copy, errors=errors, valu
es=values, **kwargs)
    560
    561     def _astype(self, dtype, copy=False, errors="raise", values=N
one, **kwargs):

~\Anaconda3\lib\site-packages\pandas\core\internals\blocks.py in _astype
(self, dtype, copy, errors, values, **kwargs)
    598                 if deprecated_arg in kwargs:
    599                     raise ValueError(
--> 600                         "Got an unexpected argument: {}".format(d
eprecated_arg)
    601                          )
    602

ValueError: Got an unexpected argument: categories
```

In [310]:

```
# sort the DataFrame by the 'quality' Series (logical order)
df.sort_values('quality')
```

Out[310]:

	ID	quality
3	103	excellent
0	100	good
2	102	good
1	101	very good

In [316]:

```
# comparison operators work with ordered categories
df.loc[df.quality > 'good', :]
```

Out[316]:

	ID	quality
1	101	very good

[Overview of categorical data in pandas (http://pandas.pydata.org/pandas-docs/stable/categorical.html)](http://pandas.pydata.org/pandas-docs/stable/categorical.html)

[API reference for categorical methods (http://pandas.pydata.org/pandas-docs/stable/api.html#categorical)](http://pandas.pydata.org/pandas-docs/stable/api.html#categorical)

How do I create dummy variables in pandas?

In [170]:

```
# read the training dataset from Kaggle's Titanic competition
train = pd.read_csv('http://bit.ly/kaggletrain')
train.head()
```

Out[170]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [171]:
```
# create the 'Sex_male' dummy variable using the 'map' method
train['Sex_male'] = train.Sex.map({'female':0, 'male':1})
train.head()
```

Out[171]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Documentation for `map` [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.map.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.map.html)

In [172]:
```
# alternative: use 'get_dummies' to create one column for every possible valu
pd.get_dummies(train.Sex,prefix = 'Sex',prefix_sep='_',drop_first=True).head(
```

Out[172]:

	Sex_male
0	1
1	0
2	0
3	0
4	1

Generally speaking:

- If you have **"K" possible values** for a categorical feature, you only need **"K-1" dummy variables** to capture all of the information about that feature.
- One convention is to **drop the first dummy variable**, which defines that level as the "baseline".

```
In [173]:
# drop the first dummy variable ('female') using the 'iloc' method
pd.get_dummies(train.Sex).iloc[:, 1:].head()
```

Out[173]:

	male
0	1
1	0
2	0
3	0
4	1

```
In [174]:
# add a prefix to identify the source of the dummy variables
pd.get_dummies(train.Sex, prefix='Sex').iloc[:, 1:].head()
```

Out[174]:

	Sex_male
0	1
1	0
2	0
3	0
4	1

```
In [175]:
train.Embarked.unique()
```

```
Out[175]: array(['S', 'C', 'Q', nan], dtype=object)
```

In [176]:

```
# use 'get_dummies' with a feature that has 3 possible values
pd.get_dummies(train.Embarked, prefix='Embarked').head(10)
```

Out[176]:

	Embarked_C	Embarked_Q	Embarked_S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1
5	0	1	0
6	0	0	1
7	0	0	1
8	0	0	1
9	1	0	0

In [177]:

```
# drop the first dummy variable ('C')
pd.get_dummies(train.Embarked, prefix='Embarked').iloc[:, 1:].head(10)
```

Out[177]:

	Embarked_Q	Embarked_S
0	0	1
1	0	0
2	0	1
3	0	1
4	0	1
5	1	0
6	0	1
7	0	1
8	0	1
9	0	0

How to translate these values back to the original 'Embarked' value:

- **0, 0** means **C**
- **1, 0** means **Q**
- **0, 1** means **S**

In [178]:

```
# save the DataFrame of dummy variables and concatenate them to the original
embarked_dummies = pd.get_dummies(train.Embarked, prefix='Embarked').iloc[:,
train = pd.concat([train, embarked_dummies], axis=1)
train.head()
```

Out[178]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Documentation for `concat` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.concat.html)

In [179]:

```
# reset the DataFrame
train = pd.read_csv('http://bit.ly/kaggletrain')
train.head()
```

Out[179]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

```
In [180]:  # pass the DataFrame to 'get_dummies' and specify which columns to dummy (it
           pd.get_dummies(train, columns=['Sex', 'Embarked']).head()
```

Out[180]:

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	35.0	0	0	373450	8.0500	NaN

In [181]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
movies['title']= le.fit_transform(movies['title'])
movies
```

Out[181]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	866	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	756	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	757	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	730	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	560	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	914	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	81	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	459	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	550	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	939	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

979 rows × 6 columns

In [182]:

```
# use the 'drop_first' parameter (new in pandas 0.18) to drop the first dummy
pd.get_dummies(train, columns=['Sex', 'Embarked'], drop_first=True).head()
```

Out[182]:

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	35.0	0	0	373450	8.0500	NaN

Documentation for **`get_dummies`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)

How do I work with dates and times in pandas?

In [183]:

```
# read a dataset of UFO reports into a DataFrame
ufo = pd.read_csv('http://bit.ly/uforeports')
ufo.head()
```

Out[183]:

	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 [184]:
```
# 'Time' is currently stored as a string
ufo.dtypes
```

Out[184]:
```
City               object
Colors Reported    object
Shape Reported     object
State              object
Time               object
dtype: object
```

In [185]:
```
# hour could be accessed using string slicing, but this approach breaks too e
ufo.Time.str.slice(-5, -3).astype(int).head()
```

Out[185]:
```
0    22
1    20
2    14
3    13
4    19
Name: Time, dtype: int32
```

In [186]:
```
# convert 'Time' to datetime format
ufo['Time'] = pd.to_datetime(ufo.Time)
ufo.head()
```

Out[186]:

	City	Colors Reported	Shape Reported	State	Time
0	Ithaca	NaN	TRIANGLE	NY	1930-06-01 22:00:00
1	Willingboro	NaN	OTHER	NJ	1930-06-30 20:00:00
2	Holyoke	NaN	OVAL	CO	1931-02-15 14:00:00
3	Abilene	NaN	DISK	KS	1931-06-01 13:00:00
4	New York Worlds Fair	NaN	LIGHT	NY	1933-04-18 19:00:00

In [187]:
```
ufo.dtypes
```

Out[187]:
```
City                       object
Colors Reported            object
Shape Reported             object
State                      object
Time               datetime64[ns]
dtype: object
```

Documentation for **`to_datetime`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.to_datetime.html)

```
In [188]:  # convenient Series attributes are now available
           ufo.Time.dt.hour.head()

Out[188]:  0    22
           1    20
           2    14
           3    13
           4    19
           Name: Time, dtype: int64
```

```
In [189]:  ufo.Time.dt.weekday_name.head()

Out[189]:  0     Sunday
           1     Monday
           2     Sunday
           3     Monday
           4    Tuesday
           Name: Time, dtype: object
```

```
In [190]:  ufo.Time.dt.dayofyear.head()

Out[190]:  0    152
           1    181
           2     46
           3    152
           4    108
           Name: Time, dtype: int64
```

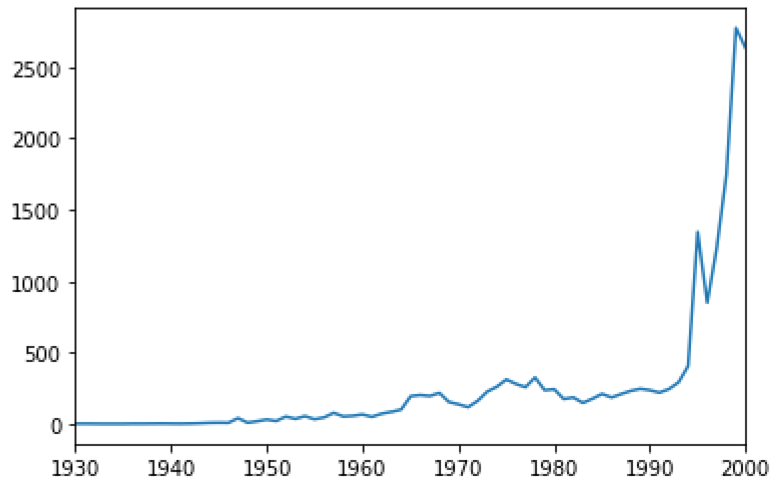
API reference for [datetime properties and methods (http://pandas.pydata.org/pandas-docs/stable/api.html#datetimelike-properties)](http://pandas.pydata.org/pandas-docs/stable/api.html#datetimelike-properties)

```
In [191]:  # convert a single string to datetime format (outputs a timestamp object)
           ts = pd.to_datetime('1/1/1999')
           ts

Out[191]:  Timestamp('1999-01-01 00:00:00')
```

```
In [192]:  # compare a datetime Series with a timestamp
           ufo.loc[ufo.Time >= ts, :].head()

Out[192]:
```

	City	Colors Reported	Shape Reported	State	Time
12832	Loma Rica	NaN	LIGHT	CA	1999-01-01 02:30:00
12833	Bauxite	NaN	NaN	AR	1999-01-01 03:00:00
12834	Florence	NaN	CYLINDER	SC	1999-01-01 14:00:00
12835	Lake Henshaw	NaN	CIGAR	CA	1999-01-01 15:00:00
12836	Wilmington Island	NaN	LIGHT	GA	1999-01-01 17:15:00

```
In [193]:  # perform mathematical operations with timestamps (outputs a timedelta object
           ufo.Time.max() - ufo.Time.min()

Out[193]:  Timedelta('25781 days 01:59:00')

In [194]:  # timedelta objects also have attributes you can access
           (ufo.Time.max() - ufo.Time.min()).days

Out[194]:  25781

In [195]:  # allow plots to appear in the notebook
           %matplotlib inline

In [196]:  # count the number of UFO reports per year
           ufo['Year'] = ufo.Time.dt.year
           ufo.Year.value_counts().sort_index().head()

Out[196]:  1930    2
           1931    2
           1933    1
           1934    1
           1935    1
           Name: Year, dtype: int64

In [197]:  # plot the number of UFO reports per year (line plot is the default)
           ufo.Year.value_counts().sort_index().plot()

Out[197]:  <matplotlib.axes._subplots.AxesSubplot at 0x11ffb903b48>
```



How do I find and remove duplicate rows in pandas?

In [198]:

```
# read a dataset of movie reviewers into a DataFrame
user_cols = ['user_id', 'age', 'gender', 'occupation', 'zip_code']
users = pd.read_table('http://bit.ly/movieusers', sep='|', header=None, names
users.head()
```

Out[198]:

	age	gender	occupation	zip_code
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213

In [199]:

```
users.shape
```

Out[199]:

```
(943, 4)
```

In [200]:

```
# detect duplicate zip codes: True if an item is identical to a previous iten
users.zip_code.duplicated().tail()
```

Out[200]:

```
user_id
939    False
940     True
941    False
942    False
943    False
Name: zip_code, dtype: bool
```

In [201]:

```
# count the duplicate items (True becomes 1, False becomes 0)
users.zip_code.duplicated().sum()
```

Out[201]:

```
148
```

In [202]:

```
# detect duplicate DataFrame rows: True if an entire row is identical to a pr
users.duplicated().tail()
```

Out[202]:

```
user_id
939    False
940    False
941    False
942    False
943    False
dtype: bool
```

In [203]:
```
# count the duplicate rows
users.duplicated().sum()
```

Out[203]:
```
7
```

Logic for **`duplicated`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.duplicated.html):

- **`keep='first'`** (default): Mark duplicates as True except for the first occurrence.
- **`keep='last'`** : Mark duplicates as True except for the last occurrence.
- **`keep=False`** : Mark all duplicates as True.

In [204]:
```
# examine the duplicate rows (ignoring the first occurrence)
users.loc[users.duplicated(keep='first'), :]
```

Out[204]:

	age	gender	occupation	zip_code
user_id				
496	21	F	student	55414
572	51	M	educator	20003
621	17	M	student	60402
684	28	M	student	55414
733	44	F	other	60630
805	27	F	other	20009
890	32	M	student	97301

In [205]:
```
# examine the duplicate rows (ignoring the last occurrence)
users.loc[users.duplicated(keep='last'), :]
```

Out[205]:

	age	gender	occupation	zip_code
user_id				
67	17	M	student	60402
85	51	M	educator	20003
198	21	F	student	55414
350	32	M	student	97301
428	28	M	student	55414
437	27	F	other	20009
460	44	F	other	60630

In [206]:
```
# examine the duplicate rows (including all duplicates)
users.loc[users.duplicated(keep=False), :]
```

Out[206]:

	age	gender	occupation	zip_code
user_id				
67	17	M	student	60402
85	51	M	educator	20003
198	21	F	student	55414
350	32	M	student	97301
428	28	M	student	55414
437	27	F	other	20009
460	44	F	other	60630
496	21	F	student	55414
572	51	M	educator	20003
621	17	M	student	60402
684	28	M	student	55414
733	44	F	other	60630
805	27	F	other	20009
890	32	M	student	97301

In [207]:
```
# drop the duplicate rows (inplace=False by default)
users.drop_duplicates(keep='first').shape
```

Out[207]:
```
(936, 4)
```

In [208]:
```
users.drop_duplicates(keep='last').shape
```

Out[208]:
```
(936, 4)
```

In [209]:
```
users.drop_duplicates(keep=False).shape
```

Out[209]:
```
(929, 4)
```

Documentation for **`drop_duplicates`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop_duplicates.html)

In [210]:
```
# only consider a subset of columns when identifying duplicates
users.duplicated(subset=['age', 'zip_code']).sum()
```
Out[210]:
```
16
```
In [211]:
```
users.drop_duplicates(subset=['age', 'zip_code']).shape
```
Out[211]:
```
(927, 4)
```

How do I avoid a SettingWithCopyWarning in pandas?

In [212]:
```
# read a dataset of top-rated IMDb movies into a DataFrame
movies = pd.read_csv('http://bit.ly/imdbratings')
movies.head()
```
Out[212]:

	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 [213]:
```
# count the missing values in the 'content_rating' Series
movies.content_rating.isnull().sum()
```
Out[213]:
```
3
```
In [214]:
```
# examine the DataFrame rows that contain those missing values
movies[movies.content_rating.isnull()]
```
Out[214]:

	star_rating	title	content_rating	genre	duration	actors_list
187	8.2	Butch Cassidy and the Sundance Kid	NaN	Biography	110	[u'Paul Newman', u'Robert Redford', u'Katharin...
649	7.7	Where Eagles Dare	NaN	Action	158	[u'Richard Burton', u'Clint Eastwood', u'Mary ...
936	7.4	True Grit	NaN	Adventure	128	[u'John Wayne', u'Kim Darby', u'Glen Campbell']

In [215]:

```
# examine the unique values in the 'content_rating' Series
movies.content_rating.value_counts()
```

Out[215]:

```
R            460
PG-13        189
PG           123
NOT RATED     65
APPROVED      47
UNRATED       38
G             32
NC-17          7
PASSED         7
X              4
GP             3
TV-MA          1
Name: content_rating, dtype: int64
```

Goal: Mark the 'NOT RATED' values as missing values, represented by 'NaN'.

In [216]:

```
# first, locate the relevant rows
movies[movies.content_rating=='NOT RATED'].head()
```

Out[216]:

	star_rating	title	content_rating	genre	duration	actors_list
5	8.9	12 Angry Men	NOT RATED	Drama	96	[u'Henry Fonda', u'Lee J. Cobb', u'Martin Bals...
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western	161	[u'Clint Eastwood', u'Eli Wallach', u'Lee Van ...
41	8.5	Sunset Blvd.	NOT RATED	Drama	110	[u'William Holden', u'Gloria Swanson', u'Erich...
63	8.4	M	NOT RATED	Crime	99	[u'Peter Lorre', u'Ellen Widmann', u'Inge Land...
66	8.4	Munna Bhai M.B.B.S.	NOT RATED	Comedy	156	[u'Sunil Dutt', u'Sanjay Dutt', u'Arshad Warsi']

In [217]:

```
# then, select the 'content_rating' Series from those rows
movies[movies.content_rating=='NOT RATED'].content_rating.head()
```

Out[217]:

```
5     NOT RATED
6     NOT RATED
41    NOT RATED
63    NOT RATED
66    NOT RATED
Name: content_rating, dtype: object
```

In [279]:
```
# finally, replace the 'NOT RATED' values with 'NaN' (imported from NumPy)
import numpy as np
movies[movies.content_rating=='NOT RATED'].content_rating = np.nan
```

Problem: That statement involves two operations, a `__getitem__` and a `__setitem__` . pandas can't guarantee whether the `__getitem__` operation returns a view or a copy of the data.

- If `__getitem__` returns a view of the data, `__setitem__` will affect the 'movies' DataFrame.
- But if `__getitem__` returns a copy of the data, `__setitem__` will not affect the 'movies' DataFrame.

In [219]:
```
# the 'content_rating' Series has not changed
movies.content_rating.isnull().sum()
```

Out[219]:
```
3
```

Solution: Use the `loc` method, which replaces the 'NOT RATED' values in a single `__setitem__` operation.

In [220]:
```
# replace the 'NOT RATED' values with 'NaN' (does not cause a SettingWithCopy
movies.loc[movies.content_rating=='NOT RATED', 'content_rating'] = np.nan
```

In [221]:
```
# this time, the 'content_rating' Series has changed
movies.content_rating.isnull().sum()
```

Out[221]:
```
68
```

Summary: Use the `loc` method any time you are selecting rows and columns in the same statement.

More information: [Modern Pandas (Part 1) (http://tomaugspurger.github.io/modern-1.html)](http://tomaugspurger.github.io/modern-1.html)

In [222]:
```
# create a DataFrame only containing movies with a high 'star_rating'
top_movies = movies.loc[movies.star_rating >= 9, :]
top_movies
```

Out[222]:

	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...

Goal: Fix the 'duration' for 'The Shawshank Redemption'.

Goal: Fix the duration for 'The Shawshank Redemption'.

In [276]:

```python
# overwrite the relevant cell with the correct duration
top_movies.loc[0, 'duration'] = 150
```

Problem: pandas isn't sure whether 'top_movies' is a view or a copy of 'movies'.

In [275]:

```python
# 'top_movies' DataFrame has been updated
top_movies
```

Out[275]:

	star_rating	title	content_rating	genre	duration	actors_list	4
0	9.3	The Shawshank Redemption	R	Crime	150	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...	150.0
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']	NaN
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...	NaN
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...	NaN

In [225]:

```python
# 'movies' DataFrame has not been updated
movies.head(1)
```

Out[225]:

	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...

Solution: Any time you are attempting to create a DataFrame copy, use the **copy** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.copy.html) method.

In [226]:

```python
# explicitly create a copy of 'movies'
top_movies = movies.loc[movies.star_rating >= 9, :].copy()
```

In [227]:

```python
# pandas now knows that you are updating a copy instead of a view (does not c
top_movies.loc[0, 'duration'] = 150
```

In [228]:
```
# 'top_movies' DataFrame has been updated
top_movies
```

Out[228]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	150	[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...

Documentation on indexing and selection: Returning a view versus a copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)

Stack Overflow: What is the point of views in pandas if it is undefined whether an indexing operation returns a view or a copy? (http://stackoverflow.com/questions/34884536/what-is-the-point-of-views-in-pandas-if-it-is-undefined-whether-an-indexing-oper)

How do I change display options in pandas?

In [229]:
```
# read a dataset of alcohol consumption into a DataFrame
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
```

In [230]:
```
# only 60 rows will be displayed when printing
drinks
```

Out[230]:

	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
2	Algeria	25	0	14	0.7
3	Andorra	245	138	312	12.4
4	Angola	217	57	45	5.9
...	...	...	...	...	...
188	Venezuela	333	100	3	7.7
189	Vietnam	111	2	1	2.0
190	Yemen	6	0	0	0.1
191	Zambia	32	19	4	2.5
192	Zimbabwe	64	18	4	4.7

193 rows × 6 columns

In [231]:
```
# check the current setting for the 'max_rows' option
pd.get_option('display.max_rows')
```

Out[231]:
```
60
```

Documentation for [`get_option` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_option.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_option.html)

```
In [232]:  # overwrite the current setting so that all rows will be displayed
           pd.set_option('display.max_rows', None)
           drinks
```

Out[232]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcoho
0	Afghanistan	0	0	0	0.
1	Albania	89	132	54	4.
2	Algeria	25	0	14	0.
3	Andorra	245	138	312	12.
4	Angola	217	57	45	5.
5	Antigua & Barbuda	102	128	45	4.
6	Argentina	193	25	221	8.
7	Armenia	21	179	11	3.
8	Australia	261	72	212	10.

```
In [233]:  # reset the 'max_rows' option to its default
           pd.reset_option('display.max_rows')
```

Documentation for **`set_option`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.set_option.html) and **`reset_option`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.reset_option.html)

```
In [234]:  # the 'max_columns' option is similar to 'max_rows'
           pd.get_option('display.max_columns')
```

```
Out[234]:  20
```

```
In [235]:  # read the training dataset from Kaggle's Titanic competition into a DataFram
           train = pd.read_csv('http://bit.ly/kaggletrain')
           train.head()
```

Out[235]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

```
In [236]:  # an ellipsis is displayed in the 'Name' cell of row 1 because of the 'max_co
           pd.get_option('display.max_colwidth')

Out[236]:  50
```

In [237]:
```
# overwrite the current setting so that more characters will be displayed
pd.set_option('display.max_colwidth', 1000)
train.head()
```

Out[237]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [238]:
```
# overwrite the 'precision' setting to display 2 digits after the decimal poi
pd.set_option('display.precision', 2)
train.head()
```

Out[238]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.0	1	0	PC 17599	71.28	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10	(
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	

In [239]:
```
# add two meaningless columns to the drinks DataFrame
drinks['x'] = drinks.wine_servings * 1000
drinks['y'] = drinks.total_litres_of_pure_alcohol * 1000
drinks.head()
```

Out[239]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0	0	0	0.0	
1	Albania	89	132	54	4.9	Eu
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	Eu
4	Angola	217	57	45	5.9	A

In [240]:
```
# use a Python format string to specify a comma as the thousands separator
pd.set_option('display.float_format', '{:,}'.format)
drinks.head()
```

Out[240]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0	0	0	0.0	
1	Albania	89	132	54	4.9	Eu
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	Eu
4	Angola	217	57	45	5.9	A

In [241]:
```
# 'y' was affected (but not 'x') because the 'float_format' option only affec
drinks.dtypes
```

Out[241]:
```
country                          object
beer_servings                     int64
spirit_servings                   int64
wine_servings                     int64
total_litres_of_pure_alcohol    float64
continent                        object
x                                 int64
y                               float64
dtype: object
```

In [242]:
```
# view the option descriptions (including the default and current values)
pd.describe_option()
```

```
    Whether to use the Unicode East Asian Width to calculate the display
text
    width.
    Enabling this may affect to the performance (default: False)
    [default: False] [currently: False]display.unicode.east_asian_width :
boolean
    Whether to use the Unicode East Asian Width to calculate the display
text
    width.
    Enabling this may affect to the performance (default: False)
    [default: False] [currently: False]display.width : int
    Width of the display in characters. In case python/IPython is running
in
    a terminal this can be set to None and pandas will correctly auto-det
ect
    the width.
    Note that the IPython notebook, IPython qtconsole, or IDLE do not run
in a
    terminal and hence it is not possible to correctly detect the width.
    [default: 80] [currently: 80]io.excel.ods.reader : string
```

In [243]:
```
# search for specific options by name
pd.describe_option('rows')
```

```
display.max_info_rows : int or None
    df.info() will usually show null-counts for each column.
    For large frames this can be quite slow. max_info_rows and max_info_col
s
    limit this null check only to frames with smaller dimensions than
    specified.
    [default: 1690785] [currently: 1690785]display.max_rows : int
    If max_rows is exceeded, switch to truncate view. Depending on
    `large_repr`, objects are either centrally truncated or printed as
    a summary view. 'None' value means unlimited.

    In case python/IPython is running in a terminal and `large_repr`
    equals 'truncate' this can be set to 0 and pandas will auto-detect
    the height of the terminal and print a truncated object which fits
    the screen height. The IPython notebook, IPython qtconsole, or
    IDLE do not run in a terminal and hence it is not possible to do
    correct auto-detection.
    [default: 60] [currently: 60]display.min_rows : int
    The numbers of rows to show in a truncated view (when `max_rows` is
    exceeded). Ignored when `max_rows` is set to None or 0. When set to
    None, follows the value of `max_rows`.
    [default: 10] [currently: 10]
```

Documentation for **`describe_option`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.describe_option.html)

In [277]:
```
# reset all of the options to their default values
pd.reset_option('all')
```

```
: boolean
    use_inf_as_null had been deprecated and will be removed in a future
    version. Use `use_inf_as_na` instead.
```

How do I create a pandas DataFrame from another object?

In [245]:
```
# create a DataFrame from a dictionary (keys become column names, values becc
pd.DataFrame({'id':[100, 101, 102], 'color':['red', 'blue', 'red']})
```

Out[245]:

	id	color
0	100	red
1	101	blue
2	102	red

In [246]:

```
# optionally specify the order of columns and define the index
df = pd.DataFrame({'id':[100, 101, 102], 'color':['red', 'blue', 'red']}, co
df
```

Out[246]:

	id	color
a	100	red
b	101	blue
c	102	red

Documentation for **`DataFrame`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html)

In [247]:

```
# create a DataFrame from a list of lists (each inner list becomes a row)
pd.DataFrame([[100, 'red'], [101, 'blue'], [102, 'red']], columns=['id', 'co
```

Out[247]:

	id	color
0	100	red
1	101	blue
2	102	red

In [248]:

```
# create a NumPy array (with shape 4 by 2) and fill it with random numbers be
import numpy as np
arr = np.random.rand(4, 2)
arr
```

Out[248]:

```
array([[0.76905988, 0.03304819],
       [0.0732601 , 0.32422705],
       [0.10660698, 0.16173659],
       [0.65393043, 0.02880436]])
```

In [249]:

```
# create a DataFrame from the NumPy array
pd.DataFrame(arr, columns=['one', 'two'])
```

Out[249]:

	one	two
0	0.769060	0.033048
1	0.073260	0.324227
2	0.106607	0.161737
3	0.653930	0.028804

In [250]:

```
# create a DataFrame of student IDs (100 through 109) and test scores (randon
pd.DataFrame({'student':np.arange(100, 110, 1), 'test':np.random.randint(60,
```

Out[250]:

```
   student  test
0      100    64
1      101    83
2      102    92
3      103   100
4      104    60
5      105    67
6      106    82
7      107    79
8      108    80
9      109    66
```

Documentation for **`np.arange`** (http://docs.scipy.org/doc/numpy/reference/generated/numpy.arange.html) and **`np.random`** (http://docs.scipy.org/doc/numpy/reference/routines.random.html)

In [251]:

```
# 'set_index' can be chained with the DataFrame constructor to select an inde
pd.DataFrame({'student':np.arange(100, 110, 1), 'test':np.random.randint(60,
```

Out[251]:

```
         test
student
    100    83
    101    82
    102    94
    103    96
    104    69
    105    72
    106    98
    107    73
    108    95
    109    90
```

Documentation for **`set_index`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.set_index.html)

In [252]:

```
# create a new Series using the Series constructor
s = pd.Series(['round', 'square'], index=['c', 'b'], name='shape')
s
```

Out[252]:

```
c     round
b    square
Name: shape, dtype: object
```

Documentation for **`Series`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html)

In [271]:

```
# concatenate the DataFrame and the Series (use axis=1 to concatenate columns
pd.concat([df, s], axis=1,sort=True)
```

Out[271]:

	id	color	shape
a	100	red	NaN
b	101	blue	square
c	102	red	round

Notes:

- The Series name became the column name in the DataFrame.
- The Series data was aligned to the DataFrame by its index.
- The 'shape' for row 'a' was marked as a missing value (NaN) because that index was not present in the Series.

Documentation for **`concat`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.concat.html)

How do I apply a function to a pandas Series or DataFrame?

In [254]:
```
# read the training dataset from Kaggle's Titanic competition into a DataFram
train = pd.read_csv('http://bit.ly/kaggletrain')
train.head()
```

Out[254]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Goal: Map the existing values of a Series to a different set of values

Method: **map** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.map.html) (Series method)

In [255]:
```
# map 'female' to 0 and 'male' to 1
train['Sex_num'] = train.Sex.map({'female':0, 'male':1})
train.loc[0:4, ['Sex', 'Sex_num']]
```

Out[255]:

	Sex	Sex_num
0	male	1
1	female	0
2	female	0
3	female	0
4	male	1

Goal: Apply a function to each element in a Series

Method: `apply` [(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.apply.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.apply.html) (Series method)

Note: `map` can be substituted for `apply` in many cases, but `apply` is more flexible and thus is recommended

In [256]:
```
# calculate the length of each string in the 'Name' Series
train['Name_length'] = train.Name.apply(len)
train.loc[0:4, ['Name', 'Name_length']]
```

Out[256]:

	Name	Name_length
0	Braund, Mr. Owen Harris	23
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	51
2	Heikkinen, Miss. Laina	22
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	44
4	Allen, Mr. William Henry	24

In [257]:
```
# round up each element in the 'Fare' Series to the next integer
import numpy as np
train['Fare_ceil'] = train.Fare.apply(np.ceil)
train.loc[0:4, ['Fare', 'Fare_ceil']]
```

Out[257]:

	Fare	Fare_ceil
0	7.2500	8.0
1	71.2833	72.0
2	7.9250	8.0
3	53.1000	54.0
4	8.0500	9.0

In [258]:
```
# we want to extract the last name of each person
train.Name.head()
```

Out[258]:
```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
Name: Name, dtype: object
```

In [259]:
```python
# use a string method to split the 'Name' Series at commas (returns a Series
train.Name.str.split(',').head()
```

Out[259]:
```
0                           [Braund,  Mr. Owen Harris]
1    [Cumings,  Mrs. John Bradley (Florence Briggs ...
2                            [Heikkinen,  Miss. Laina]
3      [Futrelle,  Mrs. Jacques Heath (Lily May Peel)]
4                          [Allen,  Mr. William Henry]
Name: Name, dtype: object
```

In [260]:
```python
# define a function that returns an element from a list based on position
def get_element(my_list, position):
    return my_list[position]
```

In [261]:
```python
# apply the 'get_element' function and pass 'position' as a keyword argument
train.Name.str.split(',').apply(get_element, position=0).head()
```

Out[261]:
```
0       Braund
1      Cumings
2    Heikkinen
3     Futrelle
4        Allen
Name: Name, dtype: object
```

In [262]:
```python
# alternatively, use a lambda function
train.Name.str.split(',').apply(lambda x: x[0]).head()
```

Out[262]:
```
0       Braund
1      Cumings
2    Heikkinen
3     Futrelle
4        Allen
Name: Name, dtype: object
```

Goal: Apply a function along either axis of a DataFrame

Method: `apply` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.apply.html) (DataFrame method)

In [263]:
```
# read a dataset of alcohol consumption into a DataFrame
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
drinks.head()
```

Out[263]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0	0	0	0.0	
1	Albania	89	132	54	4.9	Eu
2	Algeria	25	0	14	0.7	A
3	Andorra	245	138	312	12.4	Eu
4	Angola	217	57	45	5.9	A

In [264]:
```
# select a subset of the DataFrame to work with
drinks.loc[:, 'beer_servings':'wine_servings'].head()
```

Out[264]:

	beer_servings	spirit_servings	wine_servings
0	0	0	0
1	89	132	54
2	25	0	14
3	245	138	312
4	217	57	45

In [265]:
```
# apply the 'max' function along axis 0 to calculate the maximum value in eac
drinks.loc[:, 'beer_servings':'wine_servings'].apply(max, axis=0)
```

Out[265]:
```
beer_servings      376
spirit_servings    438
wine_servings      370
dtype: int64
```

In [266]:
```
# apply the 'max' function along axis 1 to calculate the maximum value in eac
drinks.loc[:, 'beer_servings':'wine_servings'].apply(max, axis=1).head()
```

Out[266]:
```
0      0
1    132
2     25
3    312
4    217
dtype: int64
```

In [278]:

```
# use 'np.argmax' to calculate which column has the maximum value for each ro
drinks.loc[:, 'beer_servings':'wine_servings'].apply(np.argmax, axis=1).head(
```

Out[278]:

```
0      beer_servings
1    spirit_servings
2      beer_servings
3      wine_servings
4      beer_servings
dtype: object
```

Goal: Apply a function to every element in a DataFrame

Method: **`applymap`** (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.applymap.html) (DataFrame method)

In [268]:

```
# convert every DataFrame element into a float
drinks.loc[:, 'beer_servings':'wine_servings'].applymap(float).head()
```

Out[268]:

	beer_servings	spirit_servings	wine_servings
0	0.0	0.0	0.0
1	89.0	132.0	54.0
2	25.0	0.0	14.0
3	245.0	138.0	312.0
4	217.0	57.0	45.0

In [269]:

```
# overwrite the existing DataFrame columns
drinks.loc[:, 'beer_servings':'wine_servings'] = drinks.loc[:, 'beer_servings
drinks.head()
```

Out[269]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	conti
0	Afghanistan	0.0	0.0	0.0	0.0	
1	Albania	89.0	132.0	54.0	4.9	Eu
2	Algeria	25.0	0.0	14.0	0.7	A
3	Andorra	245.0	138.0	312.0	12.4	Eu
4	Angola	217.0	57.0	45.0	5.9	A