Please execute the code below and observe the output you get. Also, please learn

```
how to use each of these statements to get a similar task done.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
df =
pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wranglin
g/master/dat
a/chp3/data-text.csv')
df.head(2)
df1 =
pd.read_csv('https://raw.githubusercontent.com/kjam/data-wranglingpycon/m
aster/d
ata/berlin_weather_oldest.csv')
df1.head(2)
```

In [4]:
```
#fetching data from online CSV files
import pandas as pd
df=pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wrangling/mast
df.head(2)
```

Out[4]:

	Indicator	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Co
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	

In [3]:
```
df1=pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/mast
df1.head(2) #fetching 2 rows from df
```

Out[3]:

	STATION	STATION_NAME	DATE	PRCP	SNWD	SNOW	TMAX	TMIN	WDFG	I
0	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310101	46	-9999	-9999	-9999	-11	-9999	
1	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310102	107	-9999	-9999	50	11	-9999	

2 rows × 21 columns

1. Get the Metadata from the above files.

In [5]:
```
#fetching metadata from the 2 dataframes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4656 entries, 0 to 4655
Data columns (total 12 columns):
Indicator                    4656 non-null object
PUBLISH STATES               4656 non-null object
Year                         4656 non-null int64
WHO region                   4656 non-null object
World Bank income group      4656 non-null object
Country                      4656 non-null object
Sex                          4656 non-null object
Display Value                4656 non-null int64
Numeric                      4656 non-null float64
Low                          0 non-null float64
High                         0 non-null float64
Comments                     0 non-null float64
dtypes: float64(4), int64(2), object(6)
memory usage: 436.6+ KB
```

In [6]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117208 entries, 0 to 117207
Data columns (total 21 columns):
STATION         117208 non-null object
STATION_NAME    117208 non-null object
DATE            117208 non-null int64
PRCP            117208 non-null int64
SNWD            117208 non-null int64
SNOW            117208 non-null int64
TMAX            117208 non-null int64
TMIN            117208 non-null int64
WDFG            117208 non-null int64
PGTM            117208 non-null int64
WSFG            117208 non-null int64
WT09            117208 non-null int64
WT07            117208 non-null int64
WT01            117208 non-null int64
WT06            117208 non-null int64
WT05            117208 non-null int64
WT04            117208 non-null int64
WT16            117208 non-null int64
WT08            117208 non-null int64
WT18            117208 non-null int64
WT03            117208 non-null int64
dtypes: int64(19), object(2)
memory usage: 18.8+ MB
```

2. Get the row names from the above files.

In [7]:
```
#Getting the row names from the 2 dataframes
df.index.values
```

Out[7]:
```
array([   0,    1,    2, ..., 4653, 4654, 4655], dtype=int64)
```

In [8]:
```
df1.index.values
```

Out[8]:
```
array([     0,      1,      2, ..., 117205, 117206, 117207], dtype=int64)
```

3. Change the column name from any of the above file.

In [11]:
```
# Renaming Indicator to indicator_id
df.rename(columns={'Indicator':'indicator_id'})
```

Out[11]:

	indicator_id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	C
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	

4. Change the column name from any of the above file and store the changes made permanently.

In [12]:
```
# Renaming Indicator to indicator_id and making it permanent
df.rename(columns={'Indicator':'indicator_id'},inplace=True)
df.head(2)
```

Out[12]:

	indicator_id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	C
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	

5. Change the names of multiple columns.

In [13]:
```
# Changing colum names ('PUBLISH STATES','WHO region') to ('Publication Status','
df.rename(columns={'PUBLISH STATES':'Publication Status','WHO region':'WHO Region
df.head(2)
```

Out[13]:

	indicator_id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN

6. Arrange values of a particular column in ascending order

In [16]:
```
# Arranging columns based on Year column
df.sort_values('Year').head()
```

Out[16]:

	indicator_id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low	Hi
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	N
1270	Life expectancy at birth (years)	Published	1990	Europe	High-income	Germany	Male	72	72.0	NaN	N
3193	Life expectancy at birth (years)	Published	1990	Europe	Lower-middle-income	Republic of Moldova	Male	65	65.0	NaN	N
3194	Life expectancy at birth (years)	Published	1990	Europe	Lower-middle-income	Republic of Moldova	Both sexes	68	68.0	NaN	N
3197	Life expectancy at age 60 (years)	Published	1990	Europe	Lower-middle-income	Republic of Moldova	Male	15	15.0	NaN	N

7. Arrange multiple column values in ascending order.

In [25]:
```
# Unable to get the same results because there is no combination avaliable to get
df.sort_values(['Publication Status','indicator_id','Country','Year'],ascending=[
```

Out[25]:

	Country	indicator_id	Publication Status	Year	WHO Region	World Bank income group	Sex	Display Value	Numer
554	Afghanistan	Life expectancy at birth (years)	Published	1990	Eastern Mediterranean	Low-income	Both sexes	49	49
965	Afghanistan	Life expectancy at birth (years)	Published	1990	Eastern Mediterranean	Low-income	Male	49	49
1792	Afghanistan	Life expectancy at birth (years)	Published	1990	Eastern Mediterranean	Low-income	Female	50	50
146	Afghanistan	Life expectancy	Published	2000	Eastern	Low-	Both	55	55

8. Make countryas the first column of the dataframe.

In [22]:
```
# Takeing the column into a list, and repositioning the column by making changes
cols=df.columns.tolist()
cols.remove("Country")
cols = ["Country"] + cols
df=df[cols]
df.head(5)
```

Out[22]:

	Country	indicator_id	Publication Status	Year	WHO Region	World Bank income group	Sex	Display Value	Numeric	Low
0	Andorra	Life expectancy at birth (years)	Published	1990	Europe	High-income	Both sexes	77	77.0	NaN
1	Andorra	Life expectancy at birth (years)	Published	2000	Europe	High-income	Both sexes	80	80.0	NaN
2	Andorra	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Female	28	28.0	NaN
3	Andorra	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Both sexes	23	23.0	NaN
4	United Arab Emirates	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	Female	78	78.0	NaN

9. Get the column array using a variable

In [23]:
```
#fetching column name into varaiable
who_region=df["WHO Region"].as_matrix()
who_region
```

```
C:\Users\manoj\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarni
ng: Method .as_matrix will be removed in a future version. Use .values instead.
  """Entry point for launching an IPython kernel.
```

Out[23]:
```
array(['Europe', 'Europe', 'Europe', ..., 'Africa', 'Africa', 'Africa'],
      dtype=object)
```

10. Get the subset rows 11, 24, 37

In [26]:
```
df.iloc[[11,24,37]]
```

Out[26]:

	Country	indicator_id	Publication Status	Year	WHO Region	World Bank income group	Sex	Display Value	Numeric	Low
11	Austria	Life expectancy at birth (years)	Published	2012	Europe	High-income	Female	83	83.0	NaN
24	Brunei Darussalam	Life expectancy at age 60 (years)	Published	2012	Western Pacific	High-income	Female	21	21.0	NaN
37	Cyprus	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Female	26	26.0	NaN

11. Get the subset rows excluding 5, 12, 23, and 56

In [35]:
```
# using the ~ and isin function to exclude the specific rows
df[~df.index.isin([5,12,23,56])]
```

Out[35]:

	Country	indicator_id	Publication Status	Year	WHO Region	World Bank income group	Sex	Display Value	Numer
0	Andorra	Life expectancy at birth (years)	Published	1990	Europe	High-income	Both sexes	77	77
1	Andorra	Life expectancy at birth (years)	Published	2000	Europe	High-income	Both sexes	80	80
2	Andorra	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Female	28	28
3	Andorra	Life expectancy	Published	2000	Europe	High-	Both	23	23

Load datasets from CSV

```
users=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWranglin
g/master/Data/users.csv')
sessions=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangl
ing/master/Data/sessions.csv')
products=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangl
ing/master/Data/products.csv')
transactions=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWr
angling/master/Data/transactions.csv')
users.head()
sessions.head()
transactions.head()
```

In [80]:
```
#fetching CSV files from URLs into dataframes and getting first 2 rows
users=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/
sessions=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/mast
products=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/mast
transactions=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/
users.head(2)
```

Out[80]:

	UserID	User	Gender	Registered	Cancelled
0	1	Charles	male	2012-12-21	NaN
1	2	Pedro	male	2010-08-01	2010-08-08

In [37]:
```
sessions.head(2)
```

Out[37]:

	SessionID	SessionDate	UserID
0	1	2010-01-05	2
1	2	2010-08-01	2

In [38]:
```
products.head(2)
```

Out[38]:

	ProductID	Product	Price
0	1	A	14.16
1	2	B	33.04

In [39]:
```
transactions.head(2)
```

Out[39]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity
0	1	2010-08-21	7.0	2	1
1	2	2011-05-26	3.0	4	1

12. Join users to transactions, keeping all rows from transactions and only matching rows from users (left join)

In [42]:
```
# Using merge with parameter left to get all rows from transactions and only matc
transactions.merge(users, how='left', on='UserID')
```

Out[42]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Canc
0	1	2010-08-21	7.0	2	1	NaN	NaN	NaN	
1	2	2011-05-26	3.0	4	1	Caroline	female	2012-10-23	201
2	3	2011-06-16	3.0	3	1	Caroline	female	2012-10-23	201
3	4	2012-08-26	1.0	2	3	Charles	male	2012-12-21	
4	5	2013-06-06	2.0	4	1	Pedro	male	2010-08-01	201
5	6	2013-12-23	2.0	5	6	Pedro	male	2010-08-01	201
6	7	2013-12-30	3.0	4	1	Caroline	female	2012-10-23	201
7	8	2014-04-24	NaN	2	3	NaN	NaN	NaN	
8	9	2015-04-24	7.0	4	3	NaN	NaN	NaN	
9	10	2016-05-08	3.0	4	4	Caroline	female	2012-10-23	201

13. Which transactions have a UserID not in users?

In [46]:
```
# same way as question 11, using ~ and isin fucntion to exclude the rows in trans
transactions[~transactions['UserID'].isin(users['UserID'])]
```

Out[46]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity
0	1	2010-08-21	7.0	2	1
7	8	2014-04-24	NaN	2	3
8	9	2015-04-24	7.0	4	3

14. Join users to transactions, keeping only rows from transactions and users that match via UserID (inner join)

In [47]:
```
# Using merge with parameter inner to get only matching rows from both the datafr
transactions.merge(users, how='inner', on='UserID')
```

Out[47]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Canc
0	2	2011-05-26	3.0	4	1	Caroline	female	2012-10-23	201
1	3	2011-06-16	3.0	3	1	Caroline	female	2012-10-23	201
2	7	2013-12-30	3.0	4	1	Caroline	female	2012-10-23	201
3	10	2016-05-08	3.0	4	4	Caroline	female	2012-10-23	201
4	4	2012-08-26	1.0	2	3	Charles	male	2012-12-21	
5	5	2013-06-06	2.0	4	1	Pedro	male	2010-08-01	201
6	6	2013-12-23	2.0	5	6	Pedro	male	2010-08-01	201

15. Join users to transactions, displaying all matching rows AND all non-matching rows (full outer join)

In [48]:
```
# Using merge with parameter outer to get all rows from transactions and users
transactions.merge(users, how='outer', on='UserID')
```

Out[48]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Ca
0	1.0	2010-08-21	7.0	2.0	1.0	NaN	NaN	NaN	
1	9.0	2015-04-24	7.0	4.0	3.0	NaN	NaN	NaN	
2	2.0	2011-05-26	3.0	4.0	1.0	Caroline	female	2012-10-23	2
3	3.0	2011-06-16	3.0	3.0	1.0	Caroline	female	2012-10-23	2
4	7.0	2013-12-30	3.0	4.0	1.0	Caroline	female	2012-10-23	2
5	10.0	2016-05-08	3.0	4.0	4.0	Caroline	female	2012-10-23	2
6	4.0	2012-08-26	1.0	2.0	3.0	Charles	male	2012-12-21	
7	5.0	2013-06-06	2.0	4.0	1.0	Pedro	male	2010-08-01	2
8	6.0	2013-12-23	2.0	5.0	6.0	Pedro	male	2010-08-01	2
9	8.0	2014-04-24	NaN	2.0	3.0	NaN	NaN	NaN	
10	NaN	NaN	4.0	NaN	NaN	Brielle	female	2013-07-17	
11	NaN	NaN	5.0	NaN	NaN	Benjamin	male	2010-11-25	

16. Determine which sessions occurred on the same day each user registered

In [49]:
```
# Using merge with parameter innter to get all rows from transactions and users wi
pd.merge(left=users, right=sessions, how='inner', left_on=['UserID', 'Registered'
```

Out[49]:

	UserID	User	Gender	Registered	Cancelled	SessionID	SessionDate
0	2	Pedro	male	2010-08-01	2010-08-08	2	2010-08-01
1	4	Brielle	female	2013-07-17	NaN	9	2013-07-17

17. Build a dataset with every possible (UserID, ProductID) pair (cross join)

In [83]:
```
# Fetching the required columns into 2 dataframes
df_users=pd.DataFrame(users['UserID'],columns=['UserID'])
df_products=pd.DataFrame(products['ProductID'],columns=["ProductID"])

# adding a row called key with value as 1
df_users['key']=1
df_products['key']=1

# Using the merge with default how parameter to get the required output
df_U_P = pd.merge(df_users,df_products,on='key')[["UserID","ProductID"]]
df_U_P
```

Out[83]:

	UserID	ProductID
0	1	1
1	1	2
2	1	3
3	1	4
4	1	5
5	2	1
6	2	2
7	2	3
8	2	4
9	2	5
10	3	1
11	3	2
12	3	3
13	3	4
14	3	5
15	4	1
16	4	2
17	4	3
18	4	4
19	4	5
20	5	1
21	5	2
22	5	3
23	5	4
24	5	5

18. Determine how much quantity of each product was purchased by each user

In [88]:
```
# Using merge, groupby and sum to get the quantity of each product was purchased 
pd.merge(df_U_P,transactions,how='left').groupby(["UserID","ProductID"])["Quantit
```

Out[88]:
```
UserID  ProductID
1       1            0.0
        2            3.0
        3            0.0
        4            0.0
        5            0.0
2       1            0.0
        2            0.0
        3            0.0
        4            1.0
        5            6.0
3       1            0.0
        2            0.0
        3            1.0
        4            6.0
        5            0.0
4       1            0.0
        2            0.0
        3            0.0
        4            0.0
        5            0.0
5       1            0.0
        2            0.0
        3            0.0
        4            0.0
        5            0.0
Name: Quantity, dtype: float64
```

19. For each user, get each possible pair of pair transactions (TransactionID1,TransacationID2)

In [89]:
```python
# Using the merge on transactions twice based on "UserID" to get possible pair of
pd.merge(transactions, transactions, on='UserID')
```

Out[89]:

	TransactionID_x	TransactionDate_x	UserID	ProductID_x	Quantity_x	TransactionID_y	Transact
0	1	2010-08-21	7.0	2	1	1	2
1	1	2010-08-21	7.0	2	1	9	2
2	9	2015-04-24	7.0	4	3	1	2
3	9	2015-04-24	7.0	4	3	9	2
4	2	2011-05-26	3.0	4	1	2	2
5	2	2011-05-26	3.0	4	1	3	2
6	2	2011-05-26	3.0	4	1	7	2
7	2	2011-05-26	3.0	4	1	10	2
8	3	2011-06-16	3.0	3	1	2	2
9	3	2011-06-16	3.0	3	1	3	2
10	3	2011-06-16	3.0	3	1	7	2
11	3	2011-06-16	3.0	3	1	10	2
12	7	2013-12-30	3.0	4	1	2	2
13	7	2013-12-30	3.0	4	1	3	2
14	7	2013-12-30	3.0	4	1	7	2
15	7	2013-12-30	3.0	4	1	10	2
16	10	2016-05-08	3.0	4	4	2	2
17	10	2016-05-08	3.0	4	4	3	2
18	10	2016-05-08	3.0	4	4	7	2
19	10	2016-05-08	3.0	4	4	10	2
20	4	2012-08-26	1.0	2	3	4	2
21	5	2013-06-06	2.0	4	1	5	2
22	5	2013-06-06	2.0	4	1	6	2
23	6	2013-12-23	2.0	5	6	5	2
24	6	2013-12-23	2.0	5	6	6	2
25	8	2014-04-24	NaN	2	3	8	2

20. Join each user to his/her first occuring transaction in the transactions table

In [91]:
```
# Using merge, left and first method to get each user to his/her first occuring ti
pd.merge(users, transactions.groupby('UserID').first(), how='left', on='UserID')
```

Out[91]:

	UserID	User	Gender	Registered	Cancelled	TransactionID	TransactionDate	ProductID	Qu
0	1	Charles	male	2012-12-21	NaN	4.0	2012-08-26	2.0	
1	2	Pedro	male	2010-08-01	2010-08-08	5.0	2013-06-06	4.0	
2	3	Caroline	female	2012-10-23	2016-06-07	2.0	2011-05-26	4.0	
3	4	Brielle	female	2013-07-17	NaN	NaN	NaN	NaN	
4	5	Benjamin	male	2010-11-25	NaN	NaN	NaN	NaN	

This is for the bonus points

In [111]:
```
def removeYears(item):
    if type(item) == str:
        if '(years)' in item:
            return item.replace("(years)","")
        else:
            return item
    else:
        return item
df1 =  df.applymap(removeYears)
df1
```

6	Antigua and Barbuda	Life expectancy at age 60	Published	1990	Americas	High-income	Male	17	17
7	Antigua and Barbuda	Life expectancy at age 60	Published	2012	Americas	High-income	Both sexes	22	22
8	Australia	Life expectancy at birth	Published	2012	Western Pacific	High-income	Male	81	81
9	Australia	Life expectancy at birth	Published	2000	Western Pacific	High-income	Both sexes	80	80

In [ ]: