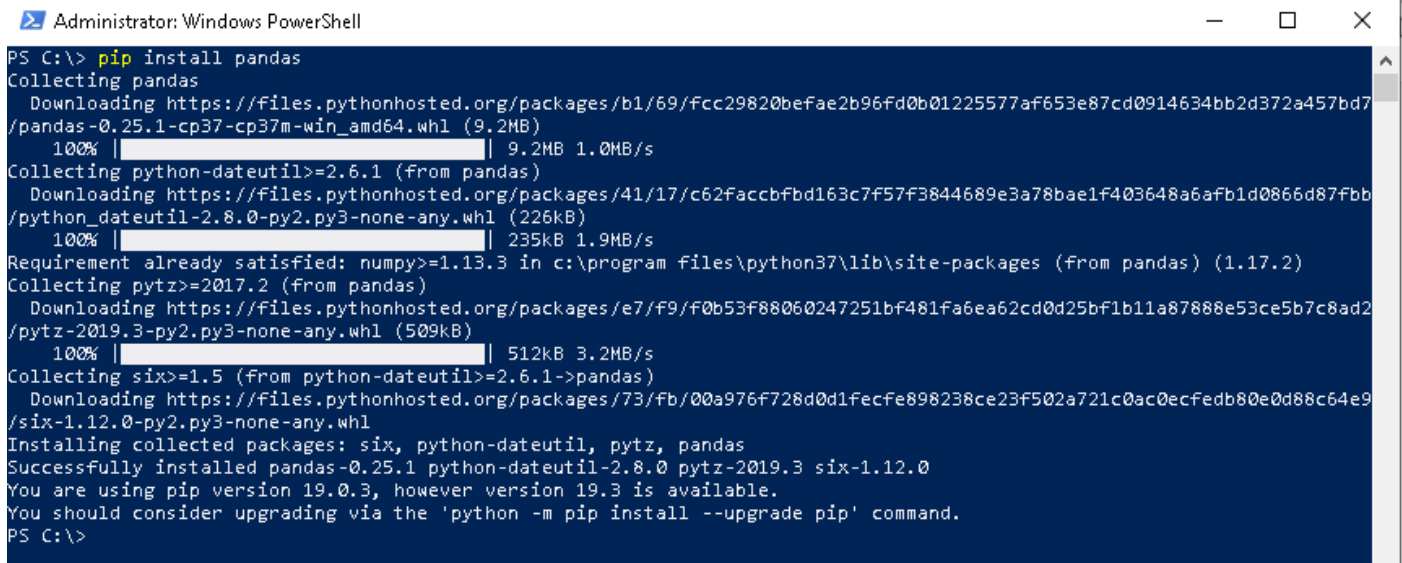


Python for Data Analysis – Pandas

Pandas is one of the most popular Python libraries for Data Science and Analytics. I like to say it's the "SQL of Python." Why? Because pandas helps you to manage two-dimensional data tables in Python. Of course, it has many more features. In this pandas tutorial series, I'll show you the most important (that is, the most often used) things that you have to know as an Analyst or a Data Scientist.

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Installation Pandas:



```
Administrator: Windows PowerShell
PS C:\> pip install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/b1/69/fcc29820befae2b96fd0b01225577af653e87cd0914634bb2d372a457bd7/pandas-0.25.1-cp37-cp37m-win_amd64.whl (9.2MB)
    100% |#####| 9.2MB 1.0MB/s
Collecting python-dateutil>=2.6.1 (from pandas)
  Downloading https://files.pythonhosted.org/packages/41/17/c62faccbfbd163c7f57f3844689e3a78bae1f403648a6afb1d0866d87fbb/python_dateutil-2.8.0-py2.py3-none-any.whl (226kB)
    100% |#####| 235kB 1.9MB/s
Requirement already satisfied: numpy>=1.13.3 in c:\program files\python37\lib\site-packages (from pandas) (1.17.2)
Collecting pytz>=2017.2 (from pandas)
  Downloading https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b11a87888e53ce5b7c8ad2/pytz-2019.3-py2.py3-none-any.whl (509kB)
    100% |#####| 512kB 3.2MB/s
Collecting six>=1.5 (from python-dateutil>=2.6.1->pandas)
  Downloading https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe898238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
Installing collected packages: six, python-dateutil, pytz, pandas
Successfully installed pandas-0.25.1 python-dateutil-2.8.0 pytz-2019.3 six-1.12.0
You are using pip version 19.0.3, however version 19.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\>
```

Let's start with Pandas:

#load library - pd is just an alias. I used pd because it's short and literally abbreviates pandas.

#You can use any name as an alias.

#create a data frame - dictionary is used here where keys get converted to column names and values to row values.

```
>>> import pandas as pd
```

```
>>> data = pd.DataFrame({'Country': ['Russia','Colombia','Chile','Ecuador','Nigeria'],
                        'Rank':[121,40,100,130,11]})
```

```
>>> data
```

	Country	Rank
0	Russia	121
1	Colombia	40
2	Chile	100
3	Ecuador	130
4	Nigeria	11

#We can do a quick analysis of any data set using:

```
>>> data.describe()
```

	Rank
count	5.000000
mean	80.400000
std	52.300096
min	11.000000
25%	40.000000
50%	100.000000
75%	121.000000
max	130.000000

Remember, `describe()` method computes summary statistics of integer / double variables. To get the complete information about the data set, we can use `info()` function.

#Among other things, it shows the data set has 5 rows and 2 columns with their respective names.

```
>>> data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 5 entries, 0 to 4
```

```
Data columns (total 2 columns):
```

Designed by Abdur Rahman Joy - MCSD, MCPD, MCSE, MCTS, OCPJ, Sr. Technical Trainer for VFX at IDB BISW (Scholarship program), and C#.net, R, Scala, Kotlin, JAVA, Android/IOS/Windows Mobile Apps, SQL server, Azure, Oracle, SharePoint Development, AWS , CEH, KALI Linux, Python, Data Science, Machine Learning ,Software Testing, Graphics, Multimedia and Game Developer at Joy Infosys and other premises like BITM, SkillsJob, PNTL, Leads Training and New Horizon inc , Cell #: +880-1712587348, email: jspaonline@gmail.com. Web URL: <http://www.joyinfosys.com/me>.

Country 5 non-null object
Rank 5 non-null int64
dtypes: int64(1), object(1)
memory usage: 208.0+ bytes

#Let's create another data frame.

```
>>> data = pd.DataFrame({'group':['a', 'a', 'a', 'b','b', 'b', 'c', 'c','c'],'ounces':[4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

```
>>> data
```

	group	ounces
0	a	4.0
1	a	3.0
2	a	12.0
3	b	6.0
4	b	7.5
5	b	8.0
6	c	3.0
7	c	5.0
8	c	6.0

#Let's sort the data frame by ounces - inplace = True will make changes to the data

```
>>> data.sort_values(by=['ounces'],ascending=True,inplace=False)
```

	group	ounces
1	a	3.0
6	c	3.0
0	a	4.0
7	c	5.0
3	b	6.0
8	c	6.0
4	b	7.5
5	b	8.0
2	a	12.0

We can sort the data by not just one column but multiple columns as well.

```
>>> data.sort_values(by=['group','ounces'],ascending=[True,False],inplace=False)
```

	group	ounces
2	a	12.0
0	a	4.0
1	a	3.0
5	b	8.0
4	b	7.5
3	b	6.0
8	c	6.0
7	c	5.0
6	c	3.0

Often, we get data sets with duplicate rows, which is nothing but noise. Therefore, before training the model, we need to make sure we get rid of such inconsistencies in the data set. Let's see how we can remove duplicate rows.

#create another data with duplicated rows

```
>>> data = pd.DataFrame({'k1':['one']*3 + ['two']*4, 'k2':[3,2,1,3,3,4,4]})
```

```
>>> data
```

	k1	k2
0	one	3
1	one	2
2	one	1
3	two	3
4	two	3
5	two	4
6	two	4

#sort values

```
>>> data.sort_values(by='k2')
```

	k1	k2
2	one	1
1	one	2
0	one	3
3	two	3
4	two	3
5	two	4
6	two	4

#remove duplicates – data:

```
>>> data.drop_duplicates()
```

	k1	k2
0	one	3
1	one	2
2	one	1
3	two	3
5	two	4

Here, we removed duplicates based on matching row values across all columns. Alternatively, we can also remove duplicates based on a particular column. Let's remove duplicate values from the k1 column.

```
>>> data.drop_duplicates(subset='k1')
```

	k1	k2
0	one	3
3	two	3

Now, we will learn to categorize rows based on a predefined criteria. It happens a lot while data processing where you need to categorize a variable. For example, say we have got a column with country names and we want to create a new variable 'continent' based on these country names. In such situations, we will require the steps below:

```
>>> data = pd.DataFrame({'food': ['bacon', 'pulled pork', 'bacon', 'Pastrami', 'corned beef', 'Bacon', 'pastrami', 'honey ham', 'nova lox'],
```

```
    'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

```
>>> data
```

	food	ounces
0	bacon	4.0
1	pulled pork	3.0
2	bacon	12.0
3	Pastrami	6.0
4	corned beef	7.5
5	Bacon	8.0
6	pastrami	3.0
7	honey ham	5.0
8	nova lox	6.0

Now, we want to create a new variable which indicates the type of animal which acts as the source of the food. To do that, first we'll create a dictionary to map the food to the animals. Then, we'll use map function to map the dictionary's values to the keys. Let's see how is it done.

```
>>> meat_to_animal = {
'bacon': 'pig',
'pulled pork': 'pig',
'pastrami': 'cow',
'corned beef': 'cow',
'honey ham': 'pig',
'nova lox': 'salmon'
}
>>> def meat_2_animal(series):
    if series['food'] == 'bacon':
        return 'pig'
    elif series['food'] == 'pulled pork':
        return 'pig'
```



```

elif series['food'] == 'pastrami':

    return 'cow'

elif series['food'] == 'corned beef':

    return 'cow'

elif series['food'] == 'honey ham':

    return 'pig'

else:

    return 'salmon'

```

#create a new variable

```

>>> data['animal'] = data['food'].map(str.lower).map(meat_to_animal)
>>> data

```

	food	ounces	animal
0	bacon	4.0	pig
1	pulled pork	3.0	pig
2	bacon	12.0	pig
3	Pastrami	6.0	cow
4	corned beef	7.5	cow
5	Bacon	8.0	pig
6	pastrami	3.0	cow
7	honey ham	5.0	pig
8	nova lox	6.0	salmon

#another way of doing it is: convert the food values to the lower case and apply the function

```
>>> lower = lambda x: x.lower()
>>> data['food'] = data['food'].apply(lower)
>>> data['animal2'] = data.apply(meat_2_animal, axis='columns')
>>> data
```

	food	ounces	animal	animal2
0	bacon	4.0	pig	pig
1	pulled pork	3.0	pig	pig
2	bacon	12.0	pig	pig
3	pastrami	6.0	cow	cow
4	corned beef	7.5	cow	cow
5	bacon	8.0	pig	pig
6	pastrami	3.0	cow	cow
7	honey ham	5.0	pig	pig
8	nova lox	6.0	salmon	salmon

Another way to create a new variable is by using the assign function. With this tutorial, as you keep discovering the new functions, you'll realize how powerful pandas is.

```
>>> data.assign(new_variable = data['ounces']*10)
```

	food	ounces	animal	animal2	new_variable
0	bacon	4.0	pig	pig	40.0
1	pulled pork	3.0	pig	pig	30.0
2	bacon	12.0	pig	pig	120.0
3	pastrami	6.0	cow	cow	60.0
4	corned beef	7.5	cow	cow	75.0
5	bacon	8.0	pig	pig	80.0
6	pastrami	3.0	cow	cow	30.0
7	honey ham	5.0	pig	pig	50.0
8	nova lox	6.0	salmon	salmon	60.0

Let's remove the column animal2 from our data frame.

```
>>> data.drop('animal2',axis='columns',inplace=True)
```

```
>>> data
```

	food	ounces	animal
0	bacon	4.0	pig
1	pulled pork	3.0	pig
2	bacon	12.0	pig
3	Pastrami	6.0	cow
4	corned beef	7.5	cow
5	Bacon	8.0	pig
6	pastrami	3.0	cow
7	honey ham	5.0	pig
8	nova lox	6.0	salmon