Class Notebook 2-18

February 18, 2021

Broadcasting example

[2]:
```
import numpy as np
a = np.arange(4).reshape((4,1))
b = np.arange(4)
```

[79]:
```
a.shape
```

[79]:
```
(4, 1)
```

[80]:
```
b.shape
```

[80]:
```
(4,)
```

[81]:
```
a*b
```

[81]:
```
array([[0, 0, 0, 0],
       [0, 1, 2, 3],
       [0, 2, 4, 6],
       [0, 3, 6, 9]])
```

Broadcasting example fail

[9]:
```
c = np.ones((3,5))
d = np.arange(4)
```

[82]:
```
c*d
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-82-9f681d1a24d9> in <module>
----> 1 c*d

ValueError: operands could not be broadcast together with shapes (3,5) (4,)
```

Series - populate with data or create from Python dictionary

[90]:
```
import pandas as pd
stats = pd.
 ↪Series([75,312,461,361,822],index=['Points','Punting','Rushing','Passing','Total␣
 ↪Offense'])
stats.sort_index()
```

[90]:
```
Passing          361
Points            75
Punting          312
Rushing          461
Total Offense    822
dtype: int64
```

We can also create a series from a Python dictionary

[21]:
```
stat_dict = {'Points':75,'Punting':312.0,'Rushing':461,'Passing':361.0,'Total␣
 ↪Offense':822}
stat_dict
```

[21]:
```
{'Points': 75,
 'Punting': 312.0,
 'Rushing': 461,
 'Passing': 361.0,
 'Total Offense': 822}
```

[22]:
```
stats2 = pd.Series(stat_dict)
stats2
```

[22]:
```
Points           75.0
Punting         312.0
Rushing         461.0
Passing         361.0
Total Offense   822.0
dtype: float64
```

Let's just grab Total Offense

[84]:
```
stats['Total Offense']
```

[84]:
```
822
```

Let's grab yards...punting and rushing and passing

[88]:
```
stats['Points':'Rushing']
```

[88]:
```
Points     75
Punting   312
Rushing   461
```

```
dtype: int64
```

[92]:
```
stats[['Points','Rushing']]
```

[92]:
```
Points      75
Rushing    461
dtype: int64
```

Let's create a DataFrame. First, we need to create a second series.

[95]:
```
avgPerGame = pd.Series([39,37.5,230.5,180.
↪5,411],index=['Punting','Points','Rushing','Passing','Total Offense'])
avgPerGame
```

[95]:
```
Punting           39.0
Points            37.5
Rushing          230.5
Passing          180.5
Total Offense    411.0
dtype: float64
```

[96]:
```
panthers = pd.DataFrame({'Total':stats,'Per Game':avgPerGame})
panthers
```

[96]:
```
               Total  Per Game
Passing          361     180.5
Points            75      37.5
Punting          312      39.0
Rushing          461     230.5
Total Offense    822     411.0
```

Let's look at the index and the columns

[97]:
```
panthers.index
```

[97]:
```
Index(['Passing', 'Points', 'Punting', 'Rushing', 'Total Offense'],
dtype='object')
```

[98]:
```
panthers.columns
```

[98]:
```
Index(['Total', 'Per Game'], dtype='object')
```

How could I access the points per game? Column then the row

[102]:
```
panthers['Per Game']['Points']
```

[102]:
```
40.0
```

Edit a particular cell - row,column

[100]:
```
panthers.at['Points','Per Game'] = 40
panthers
```

[100]:

	Total	Per Game
Passing	361	180.5
Points	75	40.0
Punting	312	39.0
Rushing	461	230.5
Total Offense	822	411.0

Be careful in making "copies" - they aren't really copies

[103]:
```
panthersCopy = panthers
panthersCopy.at['Points','Per Game'] = 38
panthersCopy
```

[103]:

	Total	Per Game
Passing	361	180.5
Points	75	38.0
Punting	312	39.0
Rushing	461	230.5
Total Offense	822	411.0

[104]:
```
panthers
```

[104]:

	Total	Per Game
Passing	361	180.5
Points	75	38.0
Punting	312	39.0
Rushing	461	230.5
Total Offense	822	411.0

Let's look at data - US Baby Names, data on Moodle First we want to read the data

[105]:
```
baby_names = pd.read_csv('US_Baby_Names.csv')
baby_names.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1016395 entries, 0 to 1016394
Data columns (total 7 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   Unnamed: 0  1016395 non-null  int64
 1   Id          1016395 non-null  int64
 2   Name        1016395 non-null  object
 3   Year        1016395 non-null  int64
 4   Gender      1016395 non-null  object
 5   State       1016395 non-null  object
```

```
 6   Count       1016395 non-null  int64
dtypes: int64(4), object(3)
memory usage: 54.3+ MB
```

Top five lines of data

[107]:
```
baby_names.head()
```

[107]:

	Unnamed: 0	Id	Name	Year	Gender	State	Count
0	11349	11350	Emma	2004	F	AK	62
1	11350	11351	Madison	2004	F	AK	48
2	11351	11352	Hannah	2004	F	AK	46
3	11352	11353	Grace	2004	F	AK	44
4	11353	11354	Emily	2004	F	AK	41

Last five lines of data

[108]:
```
baby_names.tail()
```

[108]:

	Unnamed: 0	Id	Name	Year	Gender	State	Count
1016390	5647421	5647422	Seth	2014	M	WY	5
1016391	5647422	5647423	Spencer	2014	M	WY	5
1016392	5647423	5647424	Tyce	2014	M	WY	5
1016393	5647424	5647425	Victor	2014	M	WY	5
1016394	5647425	5647426	Waylon	2014	M	WY	5

[109]:
```
baby_names.head(10)
```

[109]:

	Unnamed: 0	Id	Name	Year	Gender	State	Count
0	11349	11350	Emma	2004	F	AK	62
1	11350	11351	Madison	2004	F	AK	48
2	11351	11352	Hannah	2004	F	AK	46
3	11352	11353	Grace	2004	F	AK	44
4	11353	11354	Emily	2004	F	AK	41
5	11354	11355	Abigail	2004	F	AK	37
6	11355	11356	Olivia	2004	F	AK	33
7	11356	11357	Isabella	2004	F	AK	30
8	11357	11358	Alyssa	2004	F	AK	29
9	11358	11359	Sophia	2004	F	AK	28

Remove (clean) the data by removing unnecessary columns

[110]:
```
del baby_names['Unnamed: 0']
del baby_names['Id']
baby_names.head()
```

[110]:

	Name	Year	Gender	State	Count
0	Emma	2004	F	AK	62
1	Madison	2004	F	AK	48

```
2   Hannah  2004      F    AK     46
3    Grace  2004      F    AK     44
4    Emily  2004      F    AK     41
```

More female or male names?

[116]:
```
baby_names['Gender'].value_counts()
```

[116]:
```
F    558846
M    457549
Name: Gender, dtype: int64
```

Count of each name

[118]:
```
names = baby_names.groupby('Name').sum()
names
```

[118]:
```
            Year  Count
Name
Aaban       4027     12
Aadan       8039     23
Aadarsh     2009      5
Aaden     393963   3426
Aadhav      2014      6
…            …    …
Zyra       14085     42
Zyrah       4024     11
Zyren       2013      6
Zyria      20089     59
Zyriah     18087     58

[17632 rows x 2 columns]
```

Don't want year…it provides meaningless data in this context 1. Delete year 2. Copy and delete from copy

[119]:
```
names = baby_names.copy()
del names['Year']
names.head()
```

[119]:
```
       Name Gender State  Count
0     Emma      F    AK     62
1  Madison      F    AK     48
2   Hannah      F    AK     46
3    Grace      F    AK     44
4    Emily      F    AK     41
```

[120]:
```
baby_names.head()
```

[120]:

	Name	Year	Gender	State	Count
0	Emma	2004	F	AK	62
1	Madison	2004	F	AK	48
2	Hannah	2004	F	AK	46
3	Grace	2004	F	AK	44
4	Emily	2004	F	AK	41

[121]:
```
names = names.groupby('Name').sum()
```

[122]:
```
names
```

[122]:

Name	Count
Aaban	12
Aadan	23
Aadarsh	5
Aaden	3426
Aadhav	6
…	…
Zyra	42
Zyrah	11
Zyren	6
Zyria	59
Zyriah	58

[17632 rows x 1 columns]

[123]:
```
names.sort_values('Count',ascending=0).head()
```

[123]:

Name	Count
Jacob	242874
Emma	214852
Michael	214405
Ethan	209277
Isabella	204798

[137]:
```
emmaNames = baby_names.loc[baby_names['Name']=='Cameron']
emmaNames
```

[137]:

	Name	Year	Gender	State	Count
2465	Cameron	2004	M	AK	17
2692	Cameron	2005	M	AK	14
2901	Cameron	2006	M	AK	15
3097	Cameron	2007	M	AK	20
3334	Cameron	2008	M	AK	18
…	…	…	…	…	…

```
1015587  Cameron  2010      M    WY     12
1015742  Cameron  2011      M    WY     15
1015983  Cameron  2012      M    WY      7
1016142  Cameron  2013      M    WY      7
1016367  Cameron  2014      M    WY      5

[944 rows x 5 columns]
```

[138]:
```
emmaNames['Count'].sum()
```

[138]:
```
96424
```

[143]:
```
names.Count.idxmax()
```

[143]:
```
'Jacob'
```

[144]:
```
names.Count.max()
```

[144]:
```
242874
```

[145]:
```
names.describe()
```

[145]:
```
              Count
count   17632.000000
mean     2008.932169
std     11006.069468
min        5.000000
25%       11.000000
50%       49.000000
75%      337.000000
max    242874.000000
```

[ ]: