MemoryUsageDeep_CreateDataframe

November 19, 2016

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

In [2]: drinks = pd.read_csv('http://bit.ly/drinksbycountry')

In [3]: drinks.head(2)
```

Out[3]:

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

```
In [4]: drinks.info()
        # object usu means string is being stored

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 6 columns):
country                         193 non-null object
beer_servings                   193 non-null int64
spirit_servings                 193 non-null int64
wine_servings                   193 non-null int64
total_litres_of_pure_alcohol    193 non-null float64
continent                       193 non-null object
dtypes: float64(1), int64(3), object(2)
memory usage: 9.1+ KB


In [5]: drinks.info(memory_usage='deep') # will tell you exact memory usage

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 6 columns):
country                         193 non-null object
beer_servings                   193 non-null int64
spirit_servings                 193 non-null int64
```

```
wine_servings                   193 non-null int64
total_litres_of_pure_alcohol    193 non-null float64
continent                       193 non-null object
dtypes: float64(1), int64(3), object(2)
memory usage: 30.4 KB
```

```
In [6]: drinks.memory_usage(0) # tells you memory usage in bytes
        # doesn't tell inspect object columns by default unless you specify
        # drinks.memory_usage(deep = True)

Out[6]: country                         1544
        beer_servings                   1544
        spirit_servings                 1544
        wine_servings                   1544
        total_litres_of_pure_alcohol    1544
        continent                       1544
        dtype: int64

In [7]: drinks.memory_usage(deep=True).sum()
        # will total memory_usage => deep => around same as 30.4 KB

Out[7]: 31176

In [8]: sorted(drinks.continent.unique()) # shows six unique values of Continents

Out[8]: ['Africa', 'Asia', 'Europe', 'North America', 'Oceania', 'South America']

In [9]: drinks.continent.head()

Out[9]: 0      Asia
        1    Europe
        2    Africa
        3    Europe
        4    Africa
        Name: continent, dtype: object

In [10]: drinks['continent'] = drinks.continent.astype('category')

In [11]: drinks.dtypes

Out[11]: country                         object
         beer_servings                    int64
         spirit_servings                  int64
         wine_servings                    int64
         total_litres_of_pure_alcohol   float64
         continent                     category
         dtype: object

In [12]: drinks.head() # same
```

```
Out[12]:        country  beer_servings  spirit_servings  wine_servings  \
         0  Afghanistan              0                0              0
         1      Albania             89              132             54
         2      Algeria             25                0             14
         3      Andorra            245              138            312
         4       Angola            217               57             45

            total_litres_of_pure_alcohol continent
         0                           0.0      Asia
         1                           4.9    Europe
         2                           0.7    Africa
         3                          12.4    Europe
         4                           5.9    Africa
```

```
In [13]: drinks.continent.head() # storing strings as integers
```

```
Out[13]: 0      Asia
         1    Europe
         2    Africa
         3    Europe
         4    Africa
         Name: continent, dtype: category
         Categories (6, object): [Africa, Asia, Europe, North America, Oceania, Sou
```

```
In [14]: drinks.continent.cat.codes.head()
```

```
Out[14]: 0    1
         1    2
         2    0
         3    2
         4    0
         dtype: int8
```

```
In [15]: drinks.memory_usage(deep=True)
         # continent less memory usage => storing 193 integers that point to look u
```

```
Out[15]: Index                            80
         country                       12588
         beer_servings                  1544
         spirit_servings                1544
         wine_servings                  1544
         total_litres_of_pure_alcohol   1544
         continent                       584
         dtype: int64
```

```
In [16]: drinks['country'] = drinks.country.astype('category')
```

```
In [17]: drinks.memory_usage(deep=True)
```

```
Out[17]: Index                            80
         country                       12974
         beer_servings                  1544
         spirit_servings                1544
         wine_servings                  1544
         total_litres_of_pure_alcohol   1544
         continent                       584
         dtype: int64

In [18]: drinks.country.cat.categories # use cat datatype when object column of str

Out[18]: Index(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
                'Antigua & Barbuda', 'Argentina', 'Armenia', 'Australia', 'Austria'
                ...
                'United Arab Emirates', 'United Kingdom', 'Uruguay', 'Uzbekistan',
                'Vanuatu', 'Venezuela', 'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe'],
               dtype='object', length=193)

In [19]: # dataframe created with dictionary
         df = pd.DataFrame({'ID' :[100, 101, 102, 103], 'quality':['good', 'very go

In [20]: df

Out[20]:     ID    quality
         0  100       good
         1  101  very good
         2  102       good
         3  103  excellent

In [21]: df.sort_values('quality')

Out[21]:     ID    quality
         3  103  excellent
         0  100       good
         2  102       good
         1  101  very good

In [22]: df.sort_values('ID')

Out[22]:     ID    quality
         0  100       good
         1  101  very good
         2  102       good
         3  103  excellent

In [23]: # telling logical ordering
         df['quality'] = df.quality.astype('category', categories = ['good', 'very

In [24]: df.quality
         # ordered => good < very good < excellent
```

```
Out[24]: 0         good
         1    very good
         2         good
         3    excellent
         Name: quality, dtype: category
         Categories (3, object): [good < very good < excellent]

In [26]: df.sort_values('quality')
         # sorted in logical value

Out[26]:     ID    quality
         0  100       good
         2  102       good
         1  101  very good
         3  103  excellent

In [27]: df.loc[df.quality > 'good'] # will see columns very good and excellent bec

Out[27]:     ID    quality
         1  101  very good
         3  103  excellent

In [ ]:
```