Lect P1 – Pandas Series

Rob Capra

INLS 570

- We are going to talk about pandas today....

- But first, a small aside about floating point numbers...

Try this...

```
In [86]: a = 0

In [87]: for i in range(10):
    ...:     a += 0.1
    ...:

In [88]: a
Out[88]: 0.9999999999999999
```

What?!?!?!?

Try this...

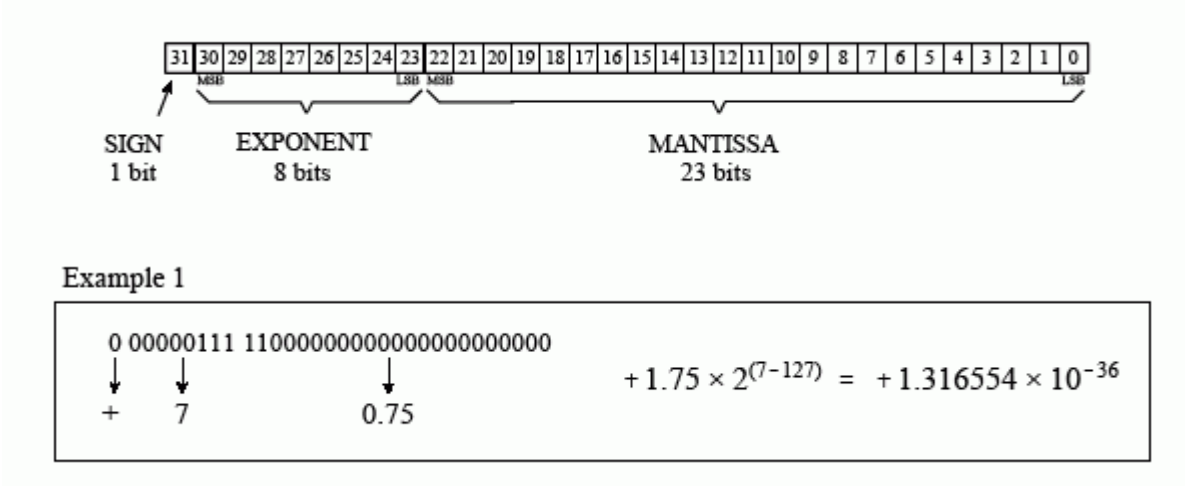
```
In [86]: a = 0

In [87]: for i in range(10):
    ...:     a += 0.1
    ...:
In [88]: a
Out[88]: 0.9999999999999999


In [91]: 0.1 + 0.2
Out[91]: 0.30000000000000004

In [92]: 0.1 + 0.1 + 0.1 == 0.3
Out[92]: False
```

What?!?!?!?

Uhm... is Python broken?

Should we go back to IDLE?

Floating Point Numbers



from: http://www.dspguide.com/ch4/3.htm

Decimal fraction: 1.834 = 1 + 8/10 + 3/100 + 4/1000

Binary fraction: 1.011 = 1 + 0/2 + 1/4 + 1/8

Exponent bias: subtract 127 from exponent

Python, along with most other high-level programming languages and the chips on your computer, represent floating-point numbers as base 2 (binary) fractions.

Similar to scientific notation, except in base 2 instead of base 10.

Floating Point Numbers

- Python, along with most other programming languages and the chips on your computer, represent floating-point numbers as base 2 (binary) fractions.

- Unfortunately, many decimal fractions cannot be represented *exactly* as binary fractions.

- Thus, the decimal FP numbers we enter are stored as binary FP numbers that are *very close approximations*.

from: https://docs.python.org/3.5/tutorial/floatingpoint.html

Floating Point Numbers

- Actually, we have this problem with decimal fractions also.

- Consider 1/3 in decimal:
    - 0.3, 0.333, 0.3333333333333333
    - all are approximations

- With binary fractions, the decimal value 0.1 has the same problem.

from: https://docs.python.org/3.5/tutorial/floatingpoint.html

format

```
In [93]: a = 0.1 + 0.2

In [94]: a
Out[94]: 0.30000000000000004

In [95]: print ("{0:.2f}".format(a))
0.30
```

https://docs.python.org/3.5/library/string.html#formatstring
http://stackoverflow.com/questions/455612/python-limiting-floats-to-two-decimal-points

Exercise P1.1

- Modify the code below to print the value of the variable **`a`** to four decimal places.

```
a = 0.1 + 0.2
print ("{0:.2f}".format(a))
```

- And now... pandas

pandas

- pandas is a powerful Python library for data analysis and analytics
- Series, DataFrame
- Indexing
- Aggregation and groupby operations

```
from pandas import Series, DataFrame
import pandas as pd
from numpy.random import randn
import numpy as np
```

Series

- One-dimensional data structure
  - Contains an array of data
  - And an associated array of data labels called its ***index***

```
In [40]: s = Series([31, 25, 18])

In [41]: s
Out[41]:
0    31
1    25
2    18
dtype: int64

In [42]: s.values
Out[42]: array([31, 25, 18], dtype=int64)

In [43]: s.index
Out[43]: RangeIndex(start=0, stop=3, step=1)
```

In this example, we did not specified an index, so pandas creates a default index of integers.

Series Index

- When creating a Series, you can specify an index

```
In [44]: s = Series([31, 25, 18],
                    index=['inls285', 'inls382', 'inls523'])

In [45]: s
Out[45]:
inls285    31
inls382    25
inls523    18
dtype: int64

In [46]: s.values
Out[46]: array([31, 25, 18], dtype=int64)

In [47]: s.index
Out[47]: Index(['inls285', 'inls382', 'inls523'],
dtype='object')
```

Series Index

- Series can be created from lists

```
In [46]: a = [31, 25, 18]

In [47]: b = ['inls285', 'inls382', 'inls523']

In [48]: s = Series(a, index=b)

In [49]: s
Out[49]:
inls285    31
inls382    25
inls523    18
dtype: int64
```

Selecting Elements from a Series

- We can select elements from a Series using either their position, or their corresponding index

```
In [34]: s = Series([31, 25, 18],
                    index=['inls285', 'inls382', 'inls523'])


In [35]: s[1]
Out[35]: 25

In [36]: s['inls382']
Out[36]: 25
```

Series Operations & Indexing

- Operations can be applied across the Series
- Elements can be selected using Boolean expressions

```
In [52]: s13 = Series([31, 25, 18], index=['inls285', 'inls382', 'inls523'])

In [53]: s14 = Series([29, 23, 14], index=['inls285', 'inls382', 'inls523'])
```

```
In [61]: s13 + 1
Out[61]:
inls285    32
inls382    26
inls523    19
dtype: int64

In [62]: s14 * 2
Out[62]:
inls285    58
inls382    46
inls523    28
dtype: int64

In [63]: s13 + s14
Out[63]:
inls285    60
inls382    48
inls523    32
dtype: int64
```

```
In [64]: s13[s13>20]
Out[64]:
inls285    31
inls382    25
dtype: int64

In [65]: s14[s14>20]
Out[65]:
inls285    29
inls382    23
dtype: int64

In [66]: s14[s13>20]
Out[66]:
inls285    29
inls382    23
dtype: int64
```

Iterating over Series

- Series can be used like collections

```
In [134]: s14
Out[134]:
inls285    29
inls382    23
inls523    14
dtype: int64

In [135]: for n in s14:
     ...:     print (n)
     ...:
29
23
14
```

Series from Dicts

- Series can be created from dicts
- Can be used like an ordered dict
- map index vals to data vals

```
In [68]: d = {'a': 5, 'b': 10, 'c': 15}

In [69]: s = Series(d)

In [70]: d
Out[70]: {'a': 5, 'b': 10, 'c': 15}

In [71]: s
Out[71]:
a     5
b    10
c    15
dtype: int64

In [72]: if 'a' in s:
    ...:     print (s['a'])
    ...:
5
```

Series from Dict

- When creating a Series from a dict
- If an index is provided, only items that match will be included.
- If the index has “extra” items, they will get **`NaN`**
  - NaN means not a number – treated as “missing” data

```
In [73]: d = {'a': 5, 'b': 10, 'c': 15}

In [74]: t = ['b', 'c', 'd']

In [75]: s = Series(d, index=t)

In [76]: s
Out[76]:
b    10
c    15
d   NaN
dtype: float64
```

Note that in this case, the Series used float64 even though we gave it integers.

This is because NaN is only supported for floats.

Series Name & Index Name

- Series are objects
- A Series has a name attribute.
- The index of a series also has a name attribute.

```
In [67]: s14
Out[67]:
inls285    29
inls382    23
inls523    14
dtype: int64

In [68]: s14.name = "Spring 2016"

In [69]: s14.index.name = "Course names"

In [70]: s14
Out[70]:
Course names
inls285    29
inls382    23
inls523    14
Name: Spring 2016, dtype: int64
```

Series – Index assignment

- The index of a series can be changed by assignment.

```
In [70]: s14
Out[70]:
Course names
inls285    29
inls382    23
inls523    14
Name: Spring 2016, dtype: int64

In [71]: s14.index = ['INLS 285', 'INLS 382', 'INLS 523']

In [72]: s14
Out[72]:
INLS 285    29
INLS 382    23
INLS 523    14
Name: Spring 2016, dtype: int64
```

Exercise P1.2 – Series Practice

- Create two Series objects:
  - aug_plays, sept_plays
  - Both should use the same index vals:
    - Britney Spears, Depeche Mode, Lady Gaga
  - Use the play counts shown on the right as values (just type them in)
- Use aug_plays and sept_plays to create avg_plays
  - Aug + Sept / 2

```
In [91]: aug_plays
Out[91]:
Britney Spears    190
Depeche Mode      274
Lady Gaga         344
dtype: int64

In [92]: sept_plays
Out[92]:
Britney Spears    123
Depeche Mode      497
Lady Gaga         273
dtype: int64

In [94]: avg_plays
Out[94]:
Britney Spears    156
Depeche Mode      385
Lady Gaga         308
dtype: int64
```