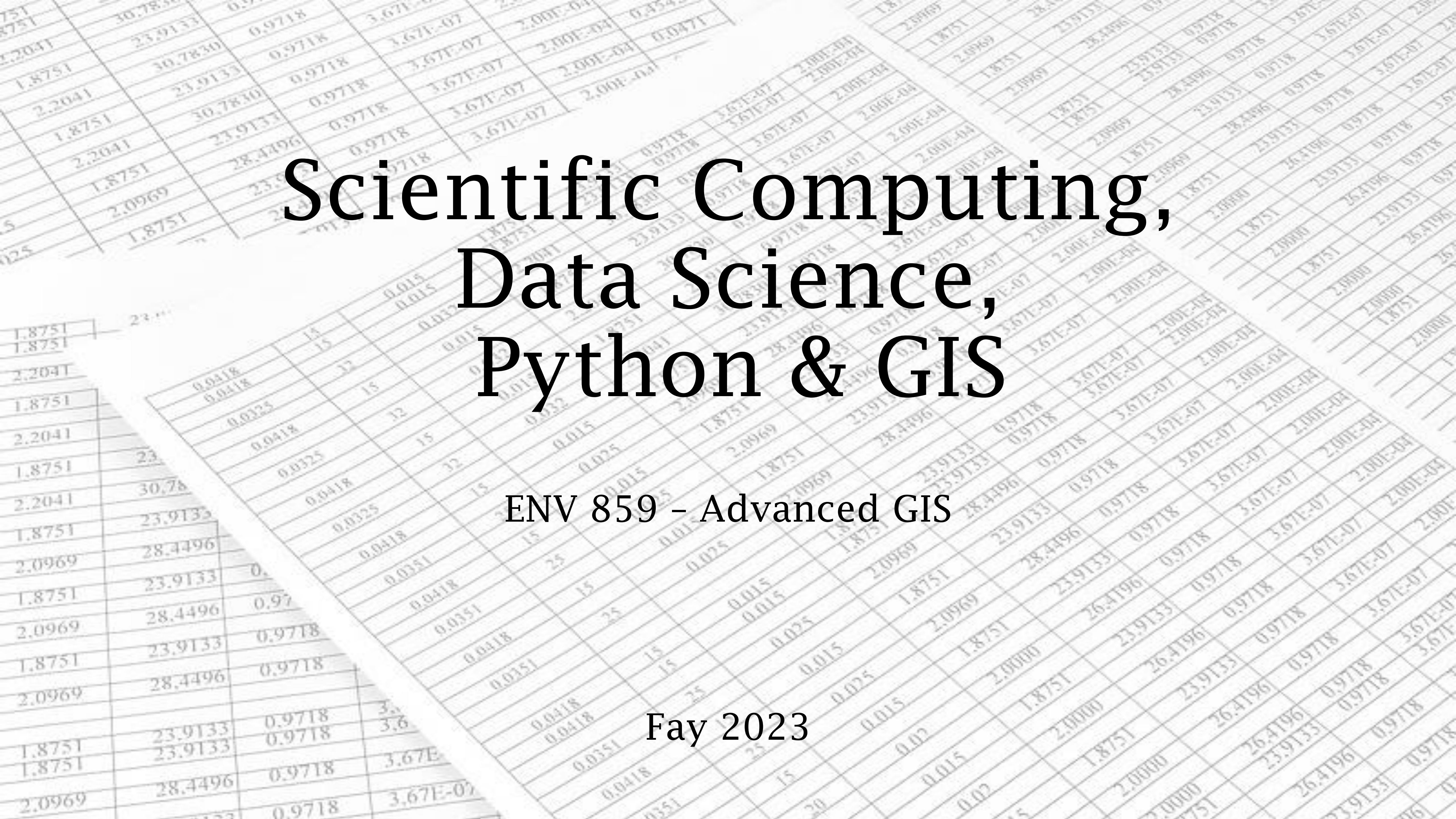
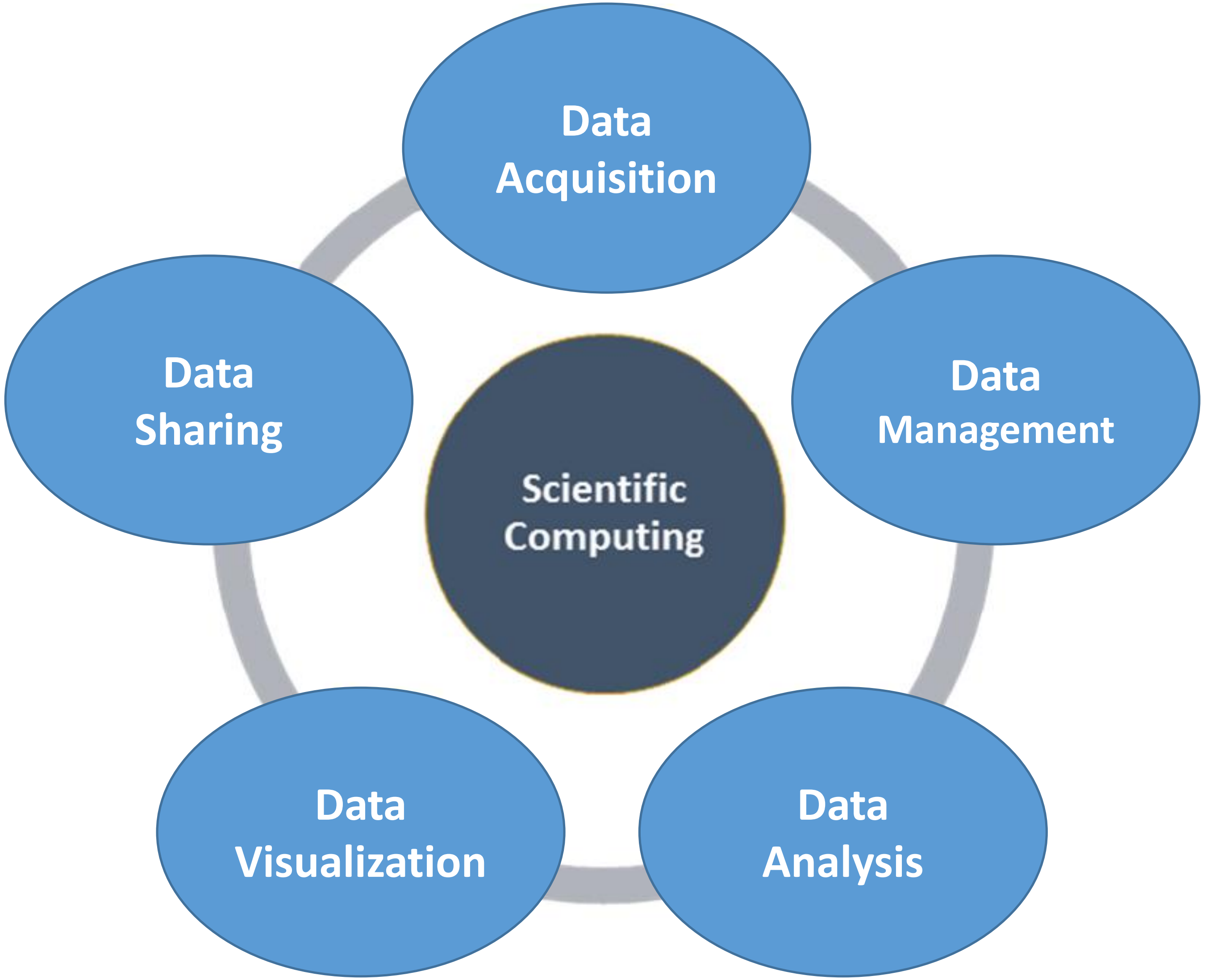


# Scientific Computing, Data Science, Python & GIS

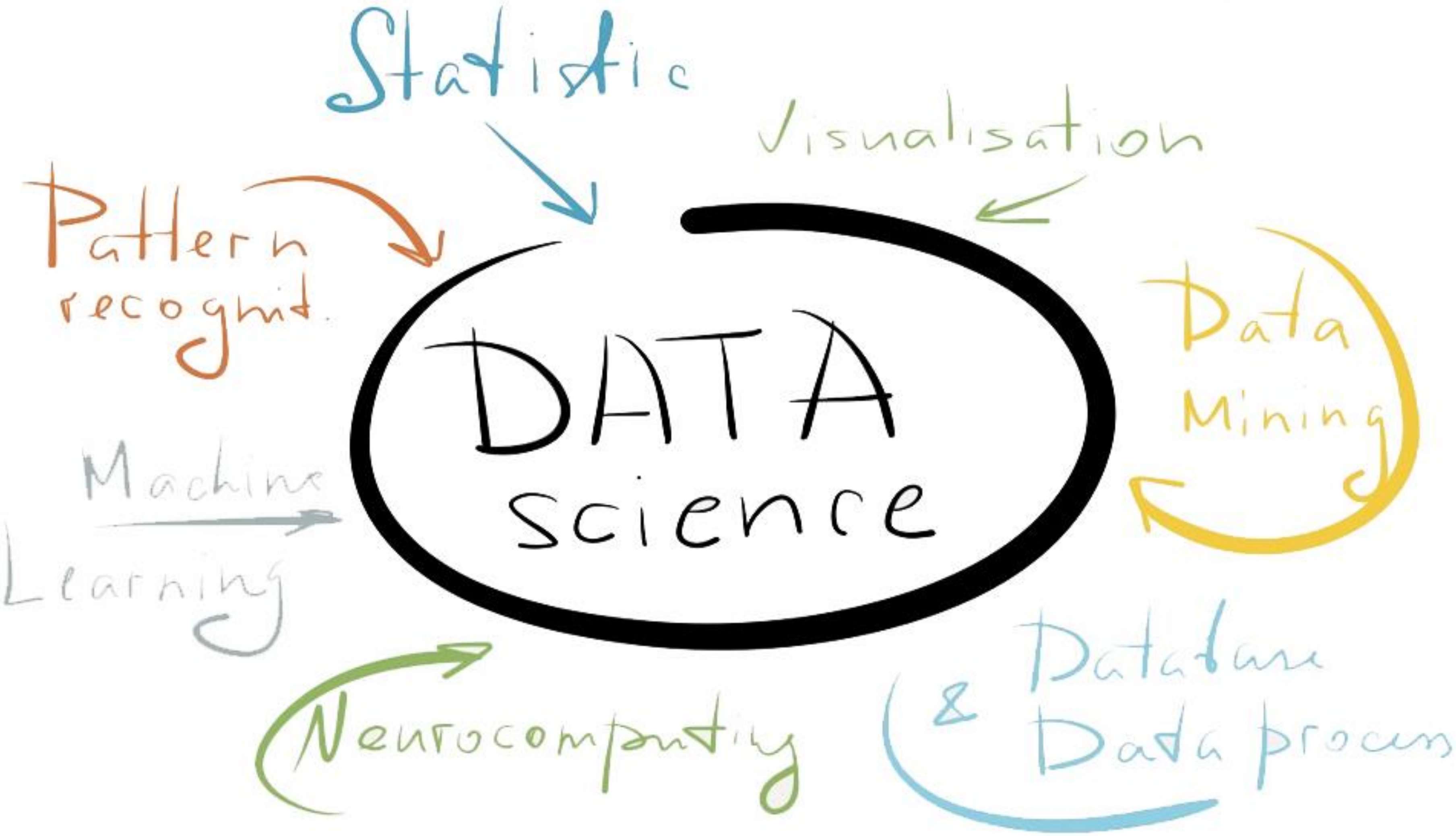
ENV 859 – Advanced GIS

Fay 2023

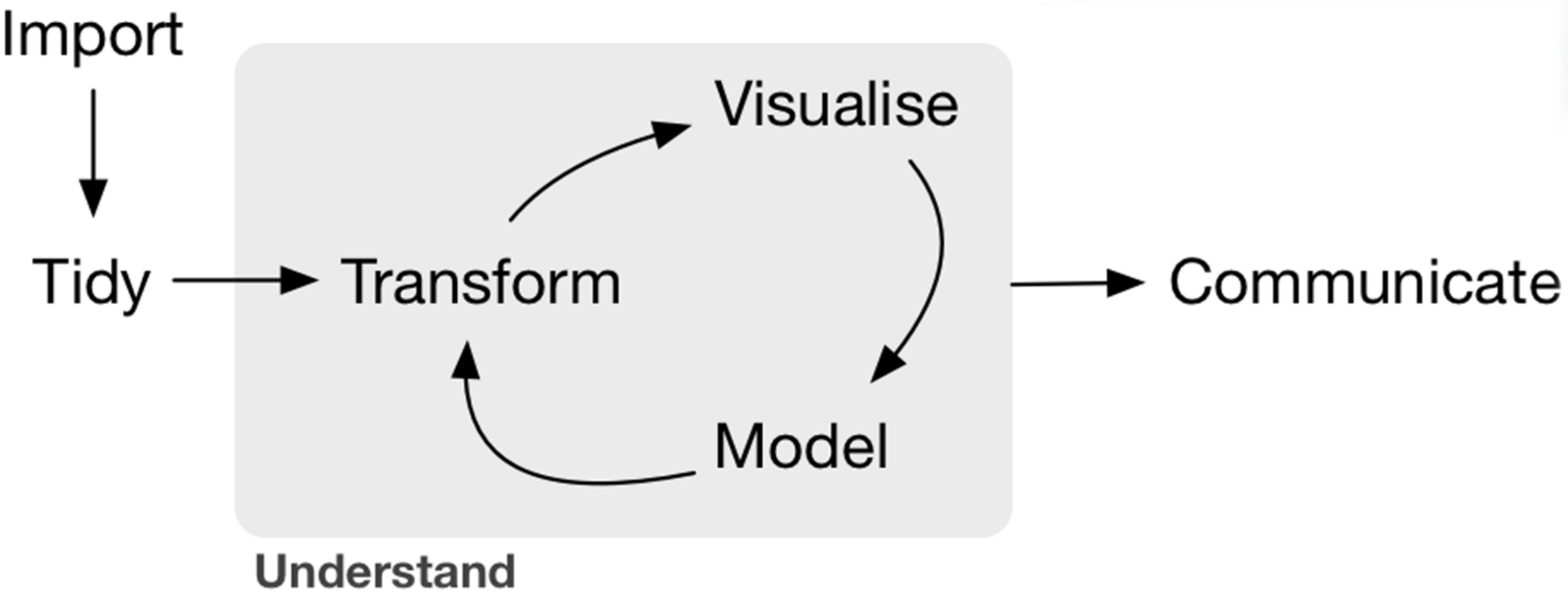
# What is “Scientific Computing”

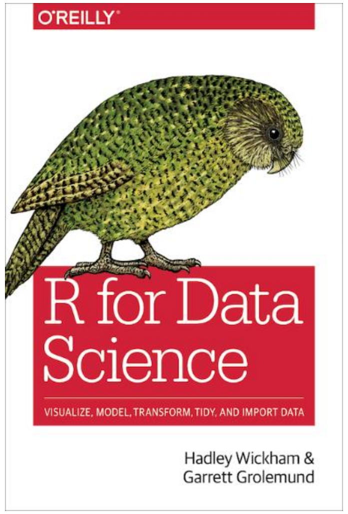


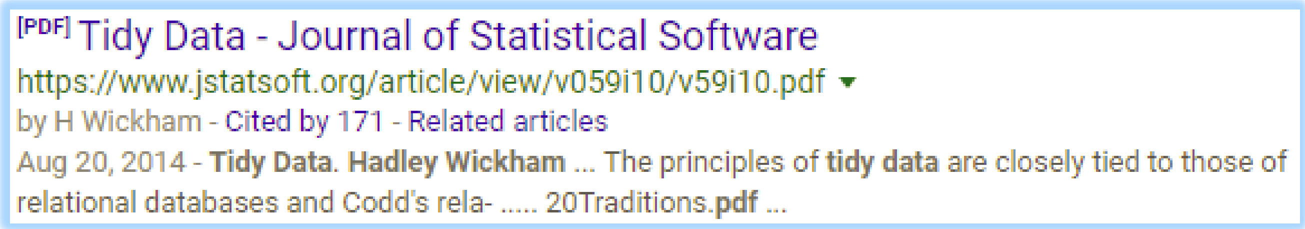
# What is "Data Science?"



# "Tidy Data"






[PDF] Tidy Data - Journal of Statistical Software

https://www.jstatsoft.org/article/view/v059i10/v59i10.pdf ▾

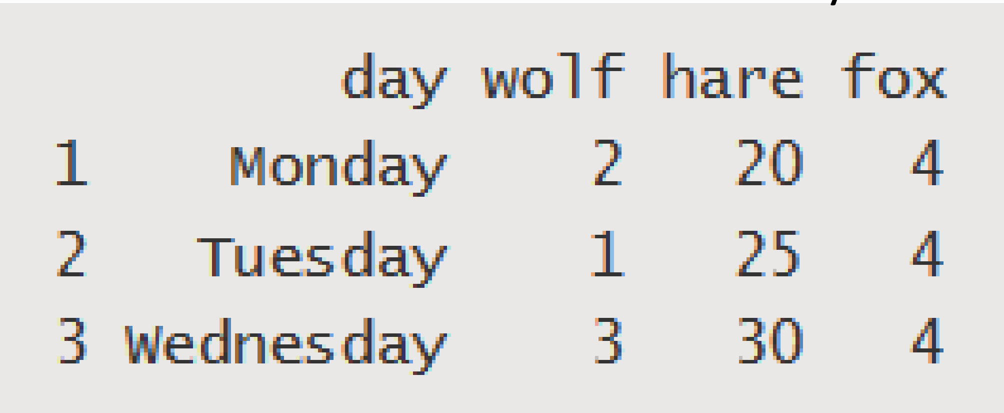
by H Wickham - Cited by 171 - Related articles

Aug 20, 2014 - **Tidy Data**. **Hadley Wickham** ... The principles of **tidy data** are closely tied to those of relational databases and Codd's rela- ..... 20Traditions.**pdf** ...

# Tidy Data Concept…

- Each **variable** forms a *column*;
- Each **observation** forms a *row*; and
- The collection of **observational units** forms a *table*.

Count of individuals observed each day

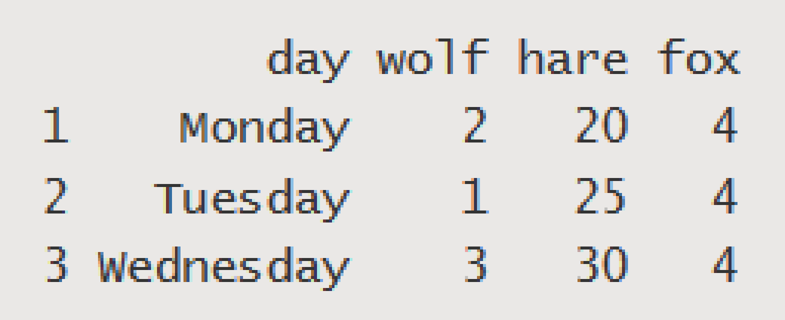


```
        day wolf hare fox
1    Monday    2   20   4
2   Tuesday    1   25   4
3 Wednesday    3   30   4
```

*Is this tidy?*

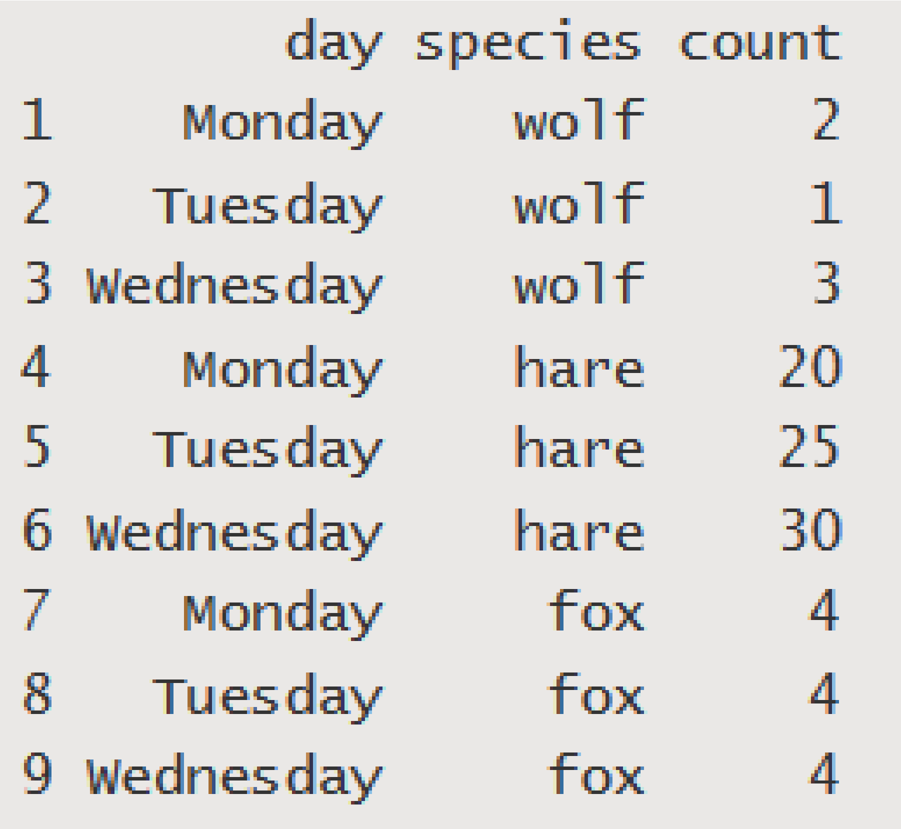
# Defining Tidy Data

*Messy…*



```
        day wolf hare fox
1    Monday    2   20   4
2   Tuesday    1   25   4
3 Wednesday    3   30   4
```

*Tidy!*



```
        day species count
1    Monday    wolf     2
2   Tuesday    wolf     1
3 Wednesday    wolf     3
4    Monday    hare    20
5   Tuesday    hare    25
6 Wednesday    hare    30
7    Monday     fox     4
8   Tuesday     fox     4
9 Wednesday     fox     4
```

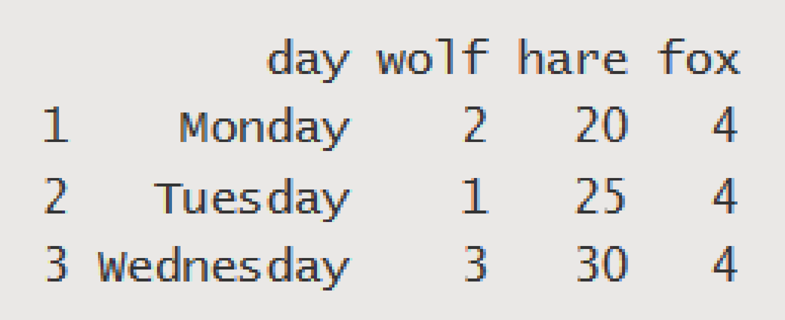
# Why tidy??

Easy manipulation of the data…

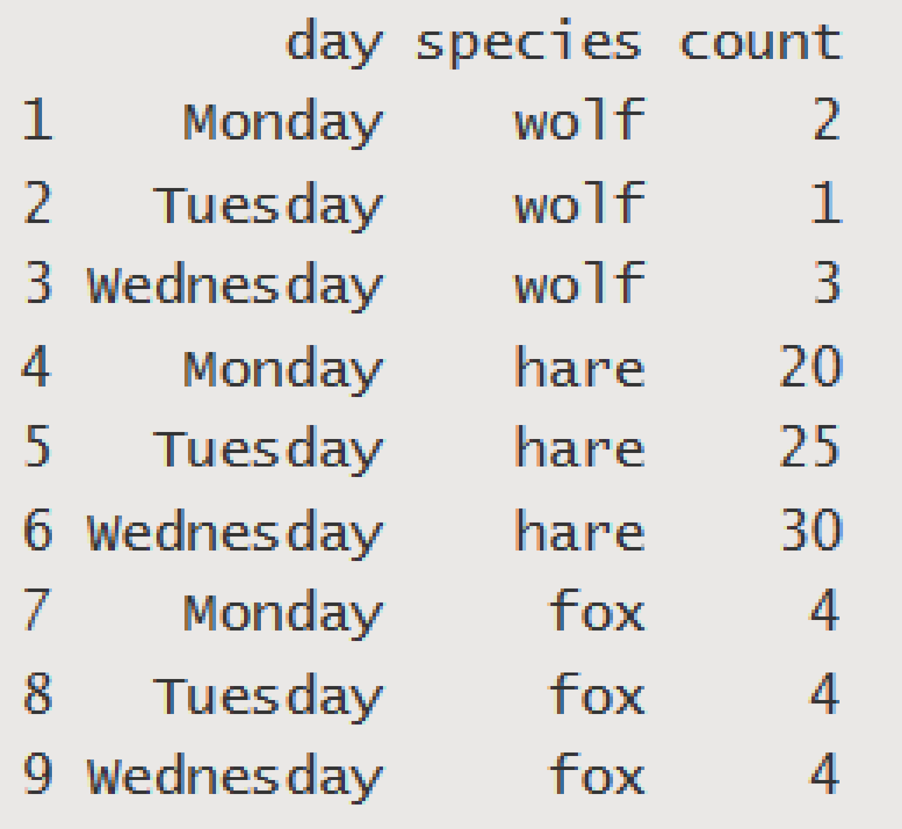
- Filtering rows (observations)
- Transforming data (derived columns)
- Aggregating
- Sorting

Plotting…

Modeling…



```
        day wolf hare fox
1    Monday    2   20   4
2   Tuesday    1   25   4
3 Wednesday    3   30   4
```

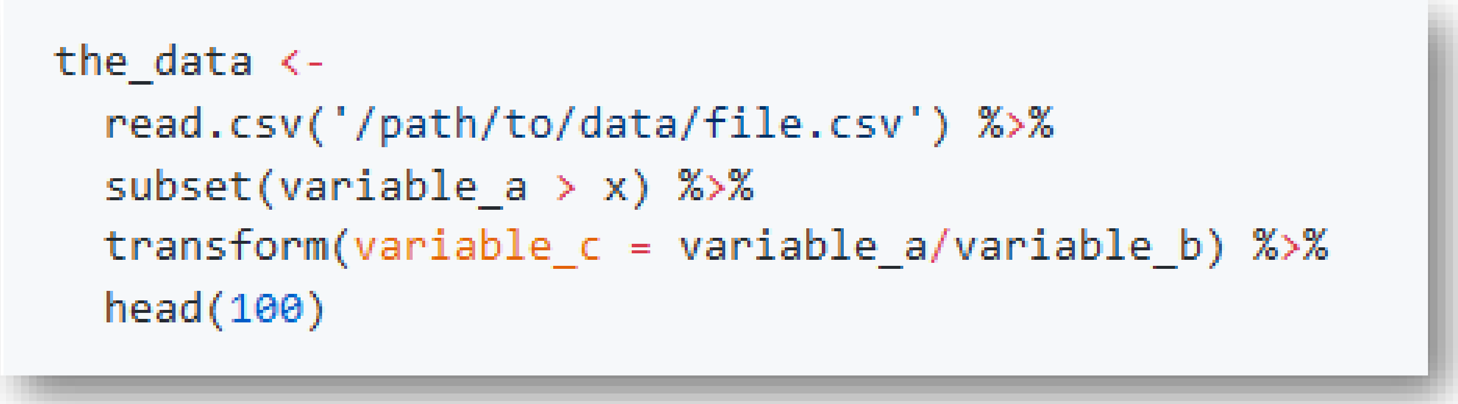


```
        day species count
1    Monday    wolf     2
2   Tuesday    wolf     1
3 Wednesday    wolf     3
4    Monday    hare    20
5   Tuesday    hare    25
6 Wednesday    hare    30
7    Monday     fox     4
8   Tuesday     fox     4
9 Wednesday     fox     4
```

# Data science – in R

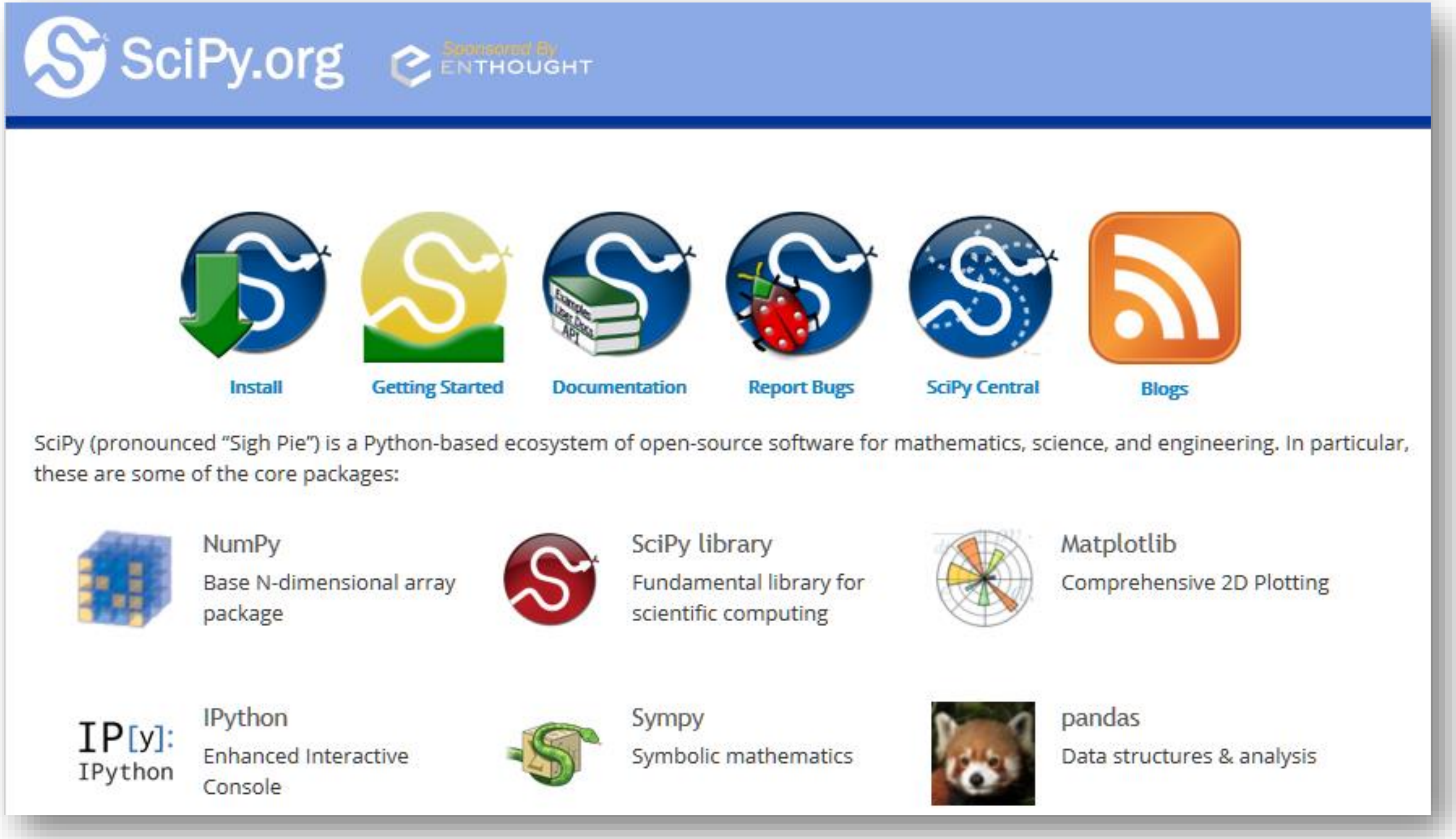
- TidyVerse
  Set of R Tools for tidying data and working with tidy data
  - https://www.tidyverse.org/packages/

- Tools are designed to string – or “pipe” – commands together
  - Output of one tool becomes the input of another…



```
the_data <-
  read.csv('/path/to/data/file.csv') %>%
  subset(variable_a > x) %>%
  transform(variable_c = variable_a/variable_b) %>%
  head(100)
```

# Data science – in Python



SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:

NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

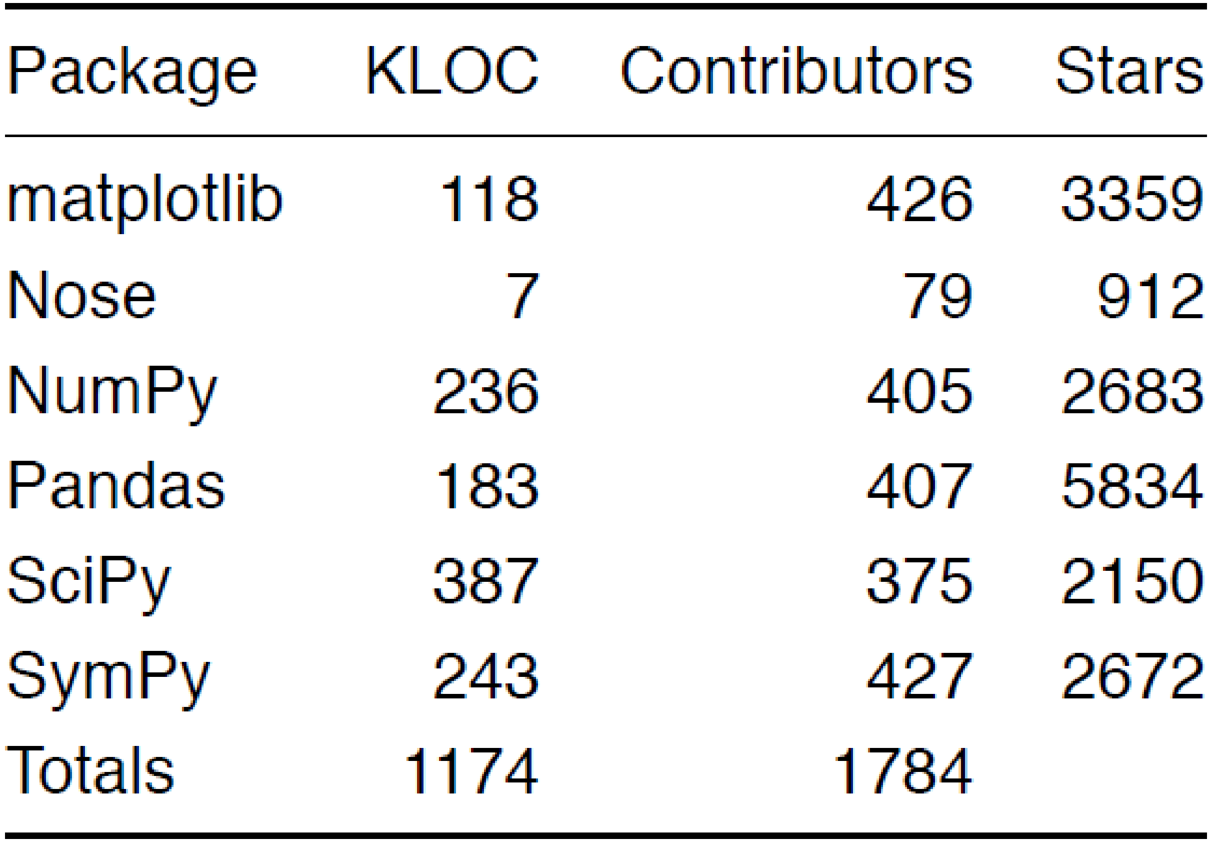
Matplotlib
Comprehensive 2D Plotting

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

# The SciPy ‘stack’



| Package | KLOC | Contributors | Stars |
|---|---|---|---|
| matplotlib | 118 | 426 | 3359 |
| Nose | 7 | 79 | 912 |
| NumPy | 236 | 405 | 2683 |
| Pandas | 183 | 407 | 5834 |
| SciPy | 387 | 375 | 2150 |
| SymPy | 243 | 427 | 2672 |
| Totals | 1174 | 1784 | |

Unit testing

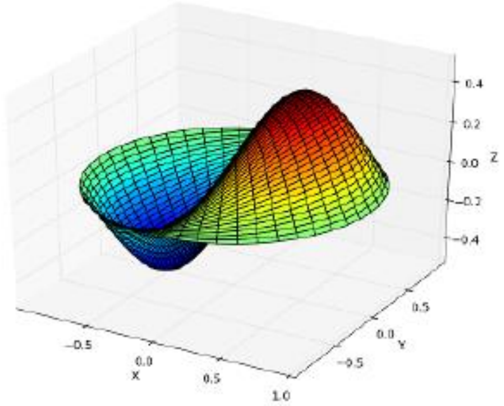
Algebraic computation

https://github.com/scw/scipy-devsummit-2016-talk/blob/master/slides/devsummit-2016-scipy-arcgis-presentation-full.pdf

KLOC = Thousands of lines of [actual] code
Stars = # of people following projects on GitHub

# SciPy modules

- **matplotlib** – object oriented plotting



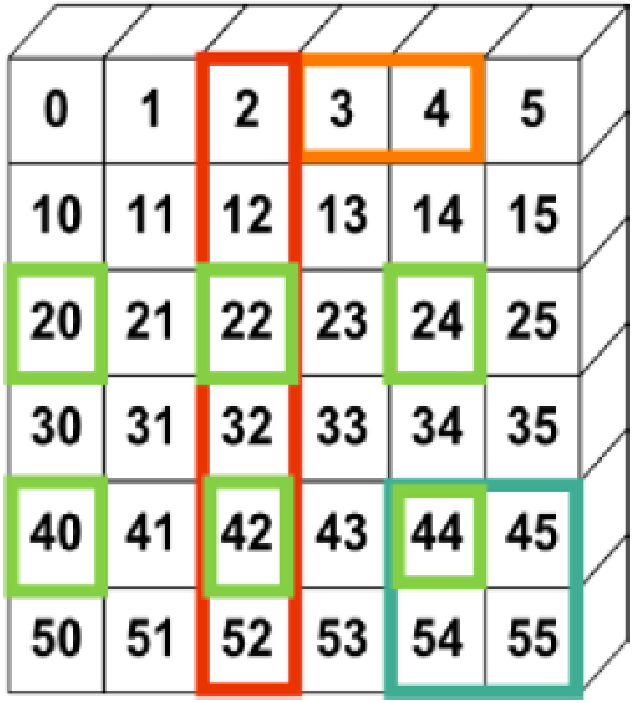
- **SciPy** – Computational methods for:
  - Integration (scipy.integrate)
  - Optimization (scipy.optimize)
  - Interpolation (scipy.interpolate)
  - Fourier Transforms (scipy.fftpack)
  - Signal Processing (scipy.signal)
  - Linear Algebra (scipy.linalg)
  - Spatial (scipy.spatial)
  - Statistics (scipy.stats)
  - Multidimensional image processing (scipy.ndimage)

SciKits

# NumPy

- Provides an n-dimensional data structure: **Array**
  - Absence has been holding Python back as a rigorous scientific coding platform.
  - Allows for *array programming*

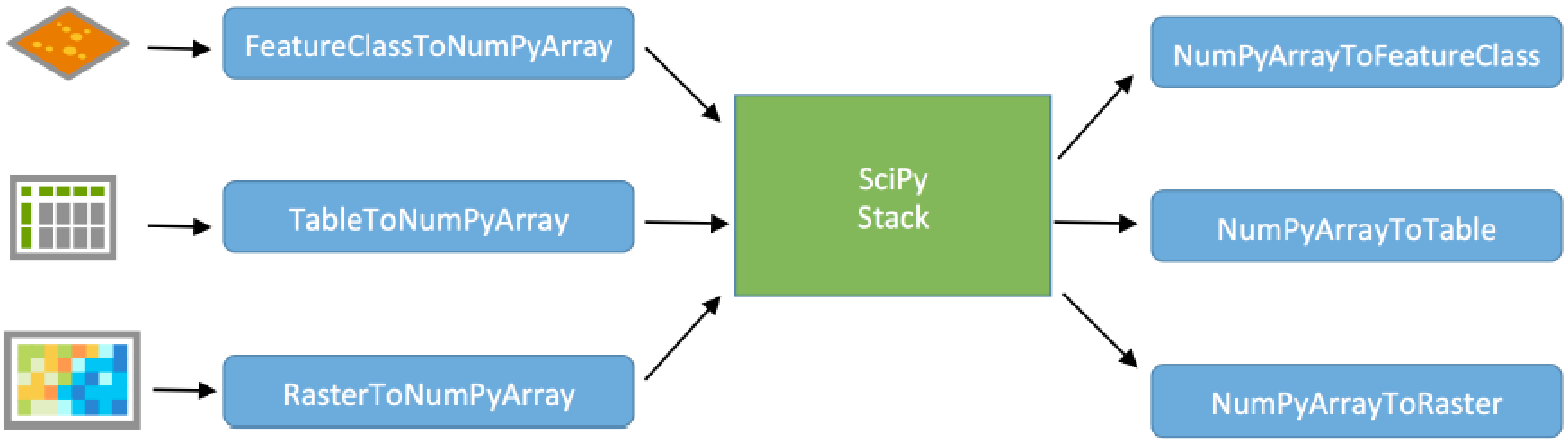
- *Why important??*
  - Easily extract specific data
  - Fast and efficient w/ large data sets



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 10 | 11 | 12 | 13 | 14 | 15 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 30 | 31 | 32 | 33 | 34 | 35 |
| 40 | 41 | 42 | 43 | 44 | 45 |
| 50 | 51 | 52 | 53 | 54 | 55 |

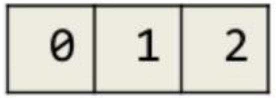
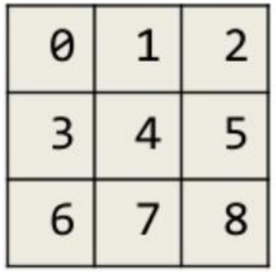
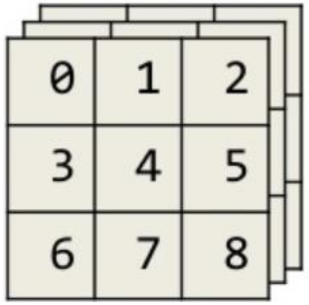
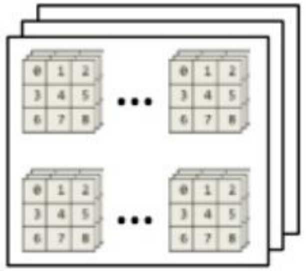
# ArcGIS and NumPy

- NumPy ships with ArcGIS (since 9.x)
- Easy to switch between ArcGIS data types and NumPy arrays that work with SciPy Stack



http://desktop.arcgis.com/en/arcmap/latest/analyze/python/working-with-numpy-in-arcgis.htm

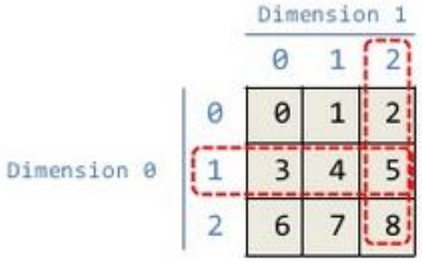
# NumPy's *n-dimensional array*

| Dimensions | Example | Terminology |
|---|---|---|
| **1** |  | Vector |
| **2** |  | Matrix |
| **3** |  | 3D Array<br>(3rd order Tensor) |
| **N** |  | ND Array |

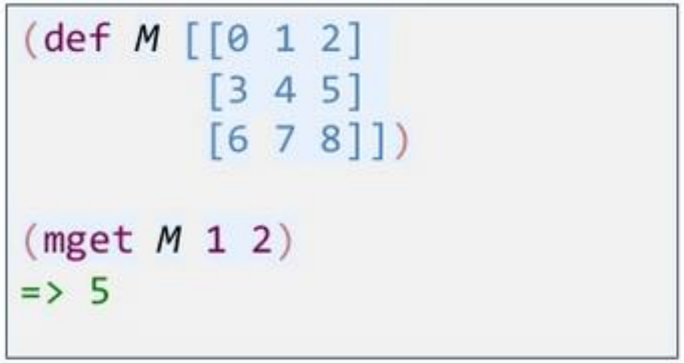
*Elements within are all the same data type…*

# NumPy’s n-dimensional array

- Allow quick access to: rows, columns, cells



|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 3 | 4 | 5 |
| 2 | 6 | 7 | 8 |



```
(def M [[0 1 2]
        [3 4 5]
        [6 7 8]])

(mget M 1 2)
=> 5
```

- Efficient computation (bulk operations)
- *Data driven* representation

# Pandas

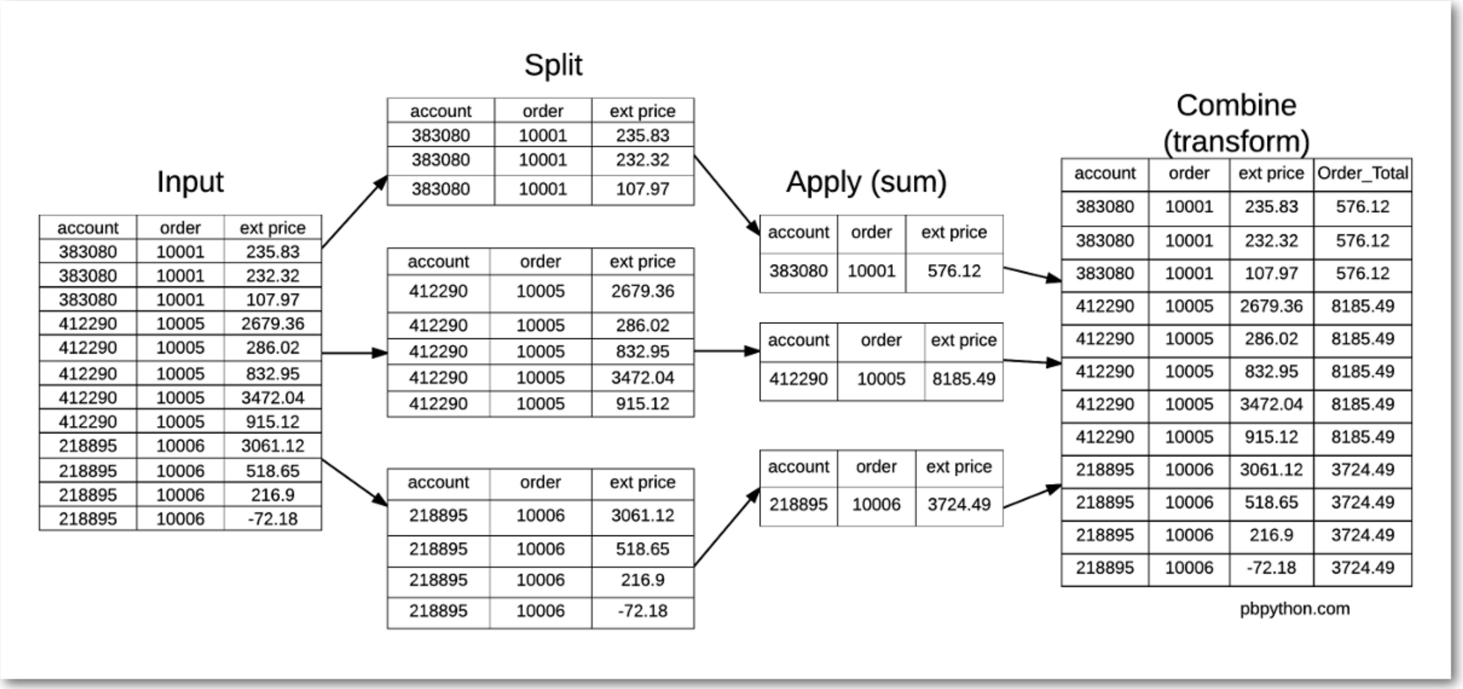
- “Swiss-army knife of data manipulation in Python
- Brings the “Data Frame” to Python
  - 2-dimensional (tabular) data structure (i.e. ‘tidy data’)

pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

# Pandas

- “Swiss-army knife” of data manipulation in Python
- Brings the “Data Frame” to Python
  - 2-dimensional (tabular) data structure (i.e. ‘tidy data’)

Wes McKinney



Input

| account | order | ext price |
|---|---|---|
| 383080 | 10001 | 235.83 |
| 383080 | 10001 | 232.32 |
| 383080 | 10001 | 107.97 |
| 412290 | 10005 | 2679.36 |
| 412290 | 10005 | 286.02 |
| 412290 | 10005 | 832.95 |
| 412290 | 10005 | 3472.04 |
| 412290 | 10005 | 915.12 |
| 218895 | 10006 | 3061.12 |
| 218895 | 10006 | 518.65 |
| 218895 | 10006 | 216.9 |
| 218895 | 10006 | -72.18 |

Split

| account | order | ext price |
|---|---|---|
| 383080 | 10001 | 235.83 |
| 383080 | 10001 | 232.32 |
| 383080 | 10001 | 107.97 |

| account | order | ext price |
|---|---|---|
| 412290 | 10005 | 2679.36 |
| 412290 | 10005 | 286.02 |
| 412290 | 10005 | 832.95 |
| 412290 | 10005 | 3472.04 |
| 412290 | 10005 | 915.12 |

| account | order | ext price |
|---|---|---|
| 218895 | 10006 | 3061.12 |
| 218895 | 10006 | 518.65 |
| 218895 | 10006 | 216.9 |
| 218895 | 10006 | -72.18 |

Apply (sum)

| account | order | ext price |
|---|---|---|
| 383080 | 10001 | 576.12 |

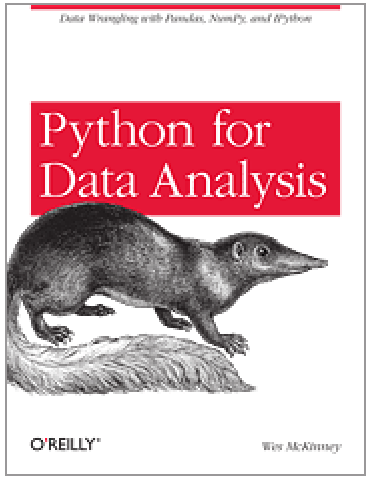
| account | order | ext price |
|---|---|---|
| 412290 | 10005 | 8185.49 |

| account | order | ext price |
|---|---|---|
| 218895 | 10006 | 3724.49 |

Combine (transform)

| account | order | ext price | Order_Total |
|---|---|---|---|
| 383080 | 10001 | 235.83 | 576.12 |
| 383080 | 10001 | 232.32 | 576.12 |
| 383080 | 10001 | 107.97 | 576.12 |
| 412290 | 10005 | 2679.36 | 8185.49 |
| 412290 | 10005 | 286.02 | 8185.49 |
| 412290 | 10005 | 832.95 | 8185.49 |
| 412290 | 10005 | 3472.04 | 8185.49 |
| 412290 | 10005 | 915.12 | 8185.49 |
| 218895 | 10006 | 3061.12 | 3724.49 |
| 218895 | 10006 | 518.65 | 3724.49 |
| 218895 | 10006 | 216.9 | 3724.49 |
| 218895 | 10006 | -72.18 | 3724.49 |

pbpython.com



https://www.slideshare.net/wesm/a-look-at-pandas-design-and-development

# Pandas

- “Swiss-army knife” of data manipulation in Python
- Brings the “Data Frame” to Python
  - 2-dimensional (tabular) data structure (i.e. ‘tidy data’)
  - Facilitates:
    - sorting/transforming/pivoting/melting of data
    - sub-setting/querying/selection of specific rows and/or columns
    - aggregation and summarizing of [selected] rows and columns
    - input & output; merging/appending/joining of multiple tables into one
    - plotting

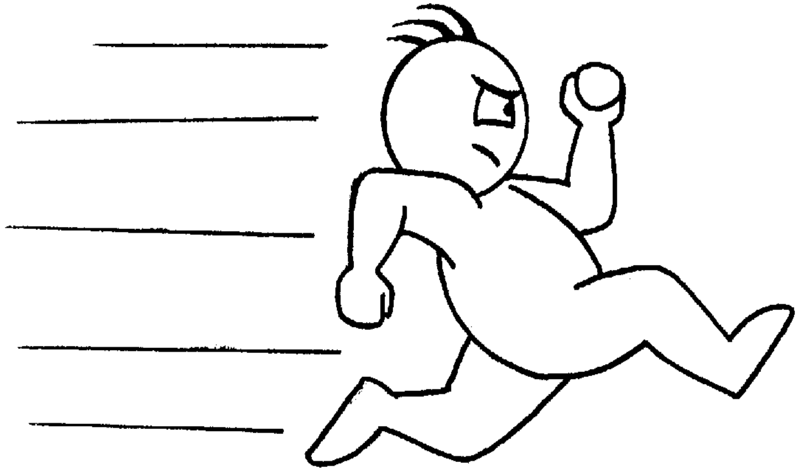
https://www.slideshare.net/wesm/a-look-at-pandas-design-and-development

# Pandas' DataFrame

- Same as a table in ArcGIS
  - Multiple data types (but same in each column)
  - All columns contain equal # of rows
  - Indexed: Rows are like Python dictionaries
- Allows for easy selection of: rows, columns, values
  - Slicing and query
- Can be sorted, subset, re-shaped easily
- Can be merged and joined with other data frames

# Pandas’ DataFrame

- Filter rows meeting a criteria
- Select specific columns
- Sort rows on values in one/many columns
- Merge/append/join other arrays or frames
- Group and summarize values
- Reshape tables
- Time series
- Plotting



# Diving In

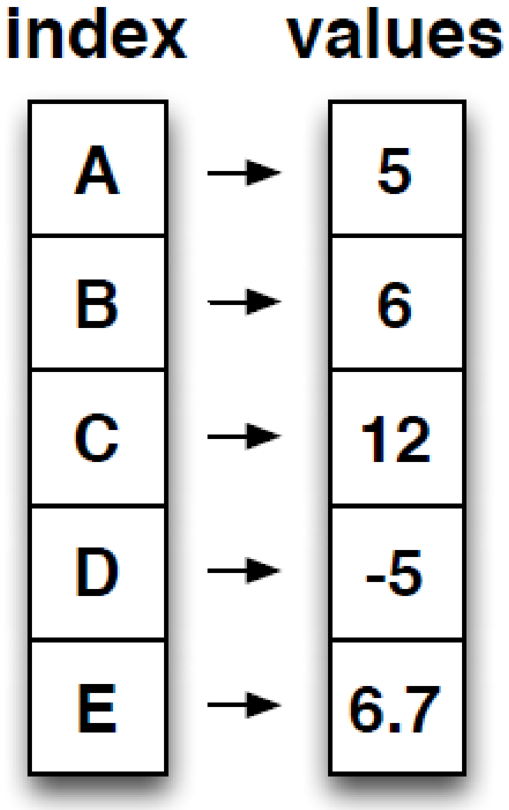
**NumPy**

- Intro to NumPy – Why NumPy’s array is useful
- Using NumPy with feature classes
- Using NumPy with Raster datasets

**Pandas**

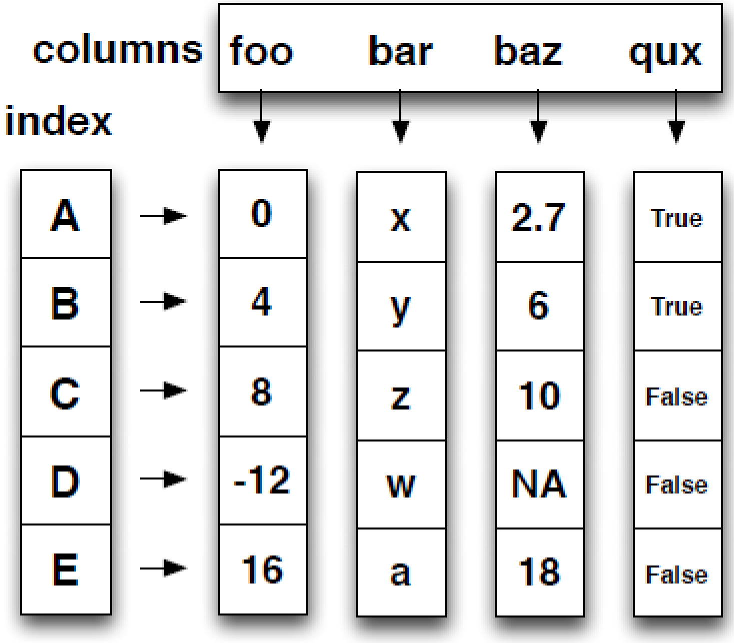
- Sara-the-Turtle *redux*: How indices work
- Getting to know Pandas
- *more research examples…*

# Recap: Pandas’ *Series* object



- 1-dimensional data collection
- Data can be of any type,
  but all members are of that type
- Indexed values
  - Need not be sequential numbers!
  - Can be anything?
  - Duplicates possible
    (but reduces functionality)

# Recap: Pandas' DataFrame object



| | foo | bar | baz | qux |
|---|---|---|---|---|
| A | 0 | x | 2.7 | True |
| B | 4 | y | 6 | True |
| C | 8 | z | 10 | False |
| D | -12 | w | NA | False |
| E | 16 | a | 18 | False |

- Each column is a *series*
  - *A column can be any data type, but contents must all be of the same data type*
- Rows and columns have *implicit* & *explicit* indices
  - Can reference values by row & column number…
  - Or by row index and column name…
- The size is mutable: can append rows, columns.
- Can join to other tables