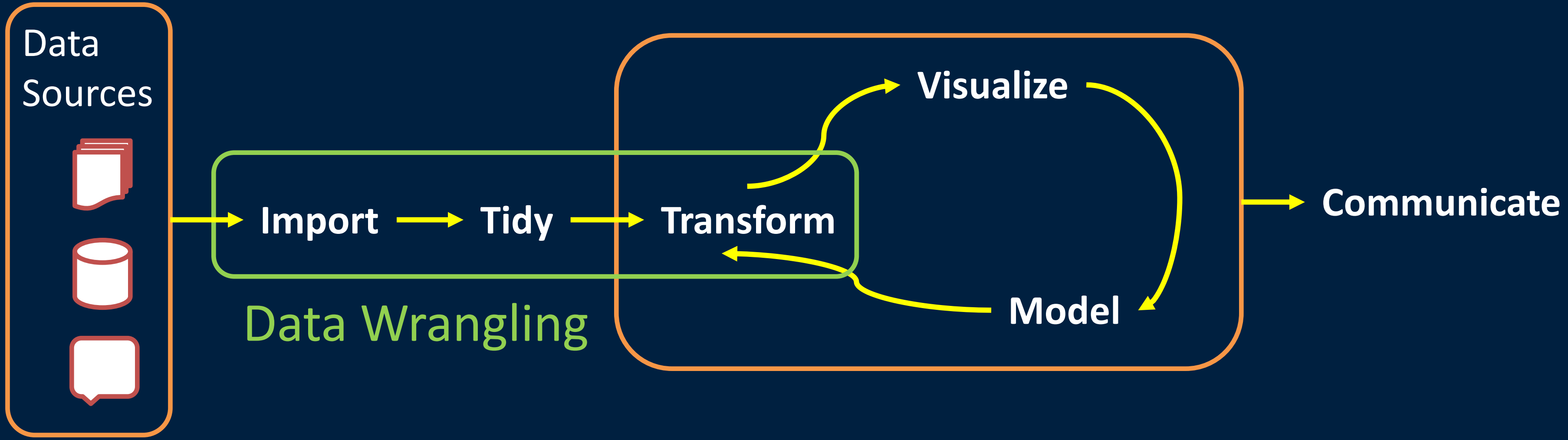
Data Wrangling

UBCO Master of Data Science – DATA 542

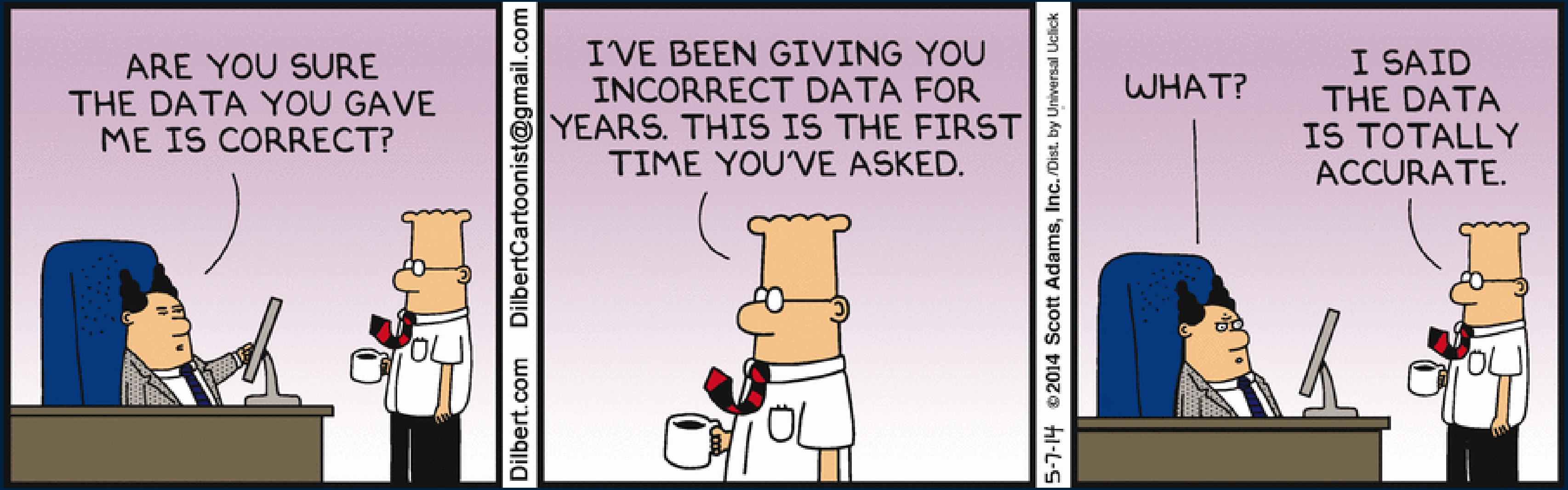
Fatemeh Fard

Data science workflow



Picture from: http://r4ds.had.co.nz/wrangle-intro.html

Garbage In Garbage Out


ARE YOU SURE THE DATA YOU GAVE ME IS CORRECT?
Dilbert.com DilbertCartoonist@gmail.com
I'VE BEEN GIVING YOU INCORRECT DATA FOR YEARS. THIS IS THE FIRST TIME YOU'VE ASKED.
5-7-14 © 2014 Scott Adams, Inc./Dist. by Universal Uclick
WHAT?
I SAID THE DATA IS TOTALLY ACCURATE.

DOB: 11/10/2003

DOB: 10/11/2003

Image from: http://alison.dbsdataprojects.com/2016/04/24/data-quality-garbage-in-garbage-out/

Dirty data in real world

incomplete: missing attribute values, lack of certain attributes of interest, or containing only aggregate data

- e.g., occupation=“”

noisy: containing errors or outliers

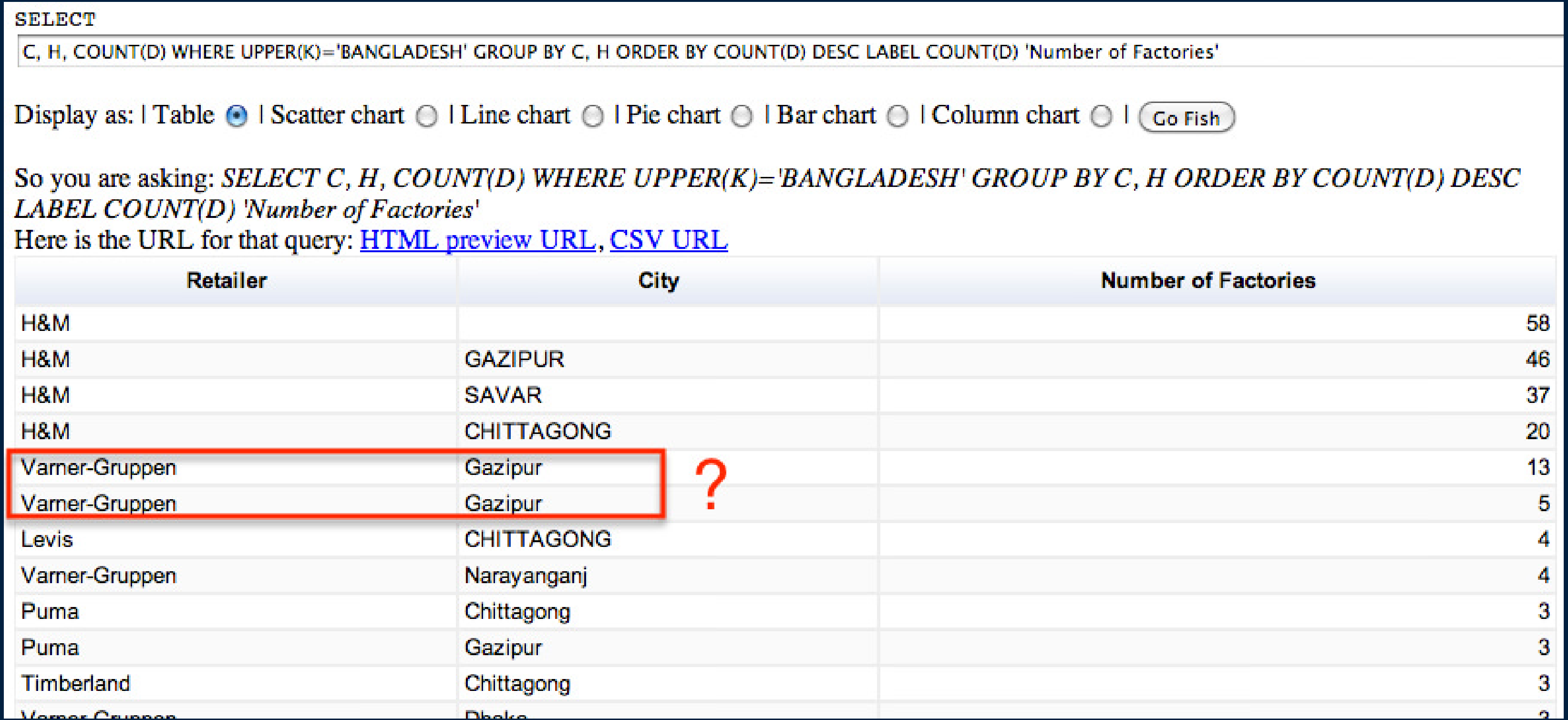
- e.g., Salary=“-10”

inconsistent: containing discrepancies in codes or names

- e.g., Age=“42” Birthday=“03/07/1997”
- e.g., Was rating “1,2,3”, now rating “A, B, C”
- e.g., discrepancy between duplicate records

Slide Source: https://www.cse.wustl.edu/~zhang/teaching/cs514/Spring11/Data-prep.pdf

Data wrangling importance



SELECT

```
C, H, COUNT(D) WHERE UPPER(K)='BANGLADESH' GROUP BY C, H ORDER BY COUNT(D) DESC LABEL COUNT(D) 'Number of Factories'
```

Display as: | Table ● | Scatter chart ○ | Line chart ○ | Pie chart ○ | Bar chart ○ | Column chart ○ | Go Fish

So you are asking: *SELECT C, H, COUNT(D) WHERE UPPER(K)='BANGLADESH' GROUP BY C, H ORDER BY COUNT(D) DESC LABEL COUNT(D) 'Number of Factories'*
Here is the URL for that query: HTML preview URL, CSV URL

Retailer	City	Number of Factories
H&M		58
H&M	GAZIPUR	46
H&M	SAVAR	37
H&M	CHITTAGONG	20
Varner-Gruppen	Gazipur	13
Varner-Gruppen	Gazipur	5
Levis	CHITTAGONG	4
Varner-Gruppen	Narayanganj	4
Puma	Chittagong	3
Puma	Gazipur	3
Timberland	Chittagong	3
Varner-Gruppen	Dhaka	3

https://www.flickr.com/photos/psychemedia/8814981784

Naming …

U_id

User_id

User_Id

User_ID

user_id

User-Id

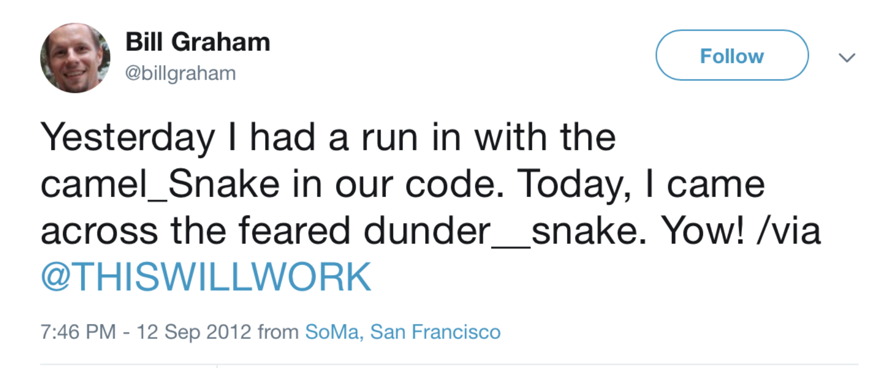
uid

user_Id

user_ID

userId

userID



Bill Graham
@billgraham

Yesterday I had a run in with the camel_Snake in our code. Today, I came across the feared dunder__snake. Yow! /via @THISWILLWORK

7:46 PM - 12 Sep 2012 from SoMa, San Francisco

+ TYPOS

https://twitter.com/billgraham/status/246077392408412160

Try It!

Open your files and folders and check the consistency of your own way of naming files/folders.

Data cleaning in practice

“It is impossible to overstress this: 80% of the work in any data project is in cleaning the data. “

- DJ Patil, Former US Chief Data Scientist

“If you can come up with strategies for data entry that are inherently clean (such as populating city and state fields from a zip code), you’re much better off. *Work done up front in getting clean data will be amply repaid over the course of the project*.”

Discuss some strategies to get cleaner data

http://radar.oreilly.com/2012/07/data-jujitsu.html

Group discussions: Strategies to collect cleaner data

Drop down menus to limit the choices (rather than collecting text)

Using one agent to collect data (consistency)

Limit the keywords

Outliers: consult wit domain experts

Reducing the ambiguity of data entry: giving options

Removal of duplicate data

Checking the data type

Implement automated solutions to clean

Data wrangling software: Trifacta

“Data wrangling is the process of cleaning, structuring and enriching raw data into a desired format for better decision making in less time. ”

https://www.trifacta.com/data-wrangling/

TRIFACTA is used by:
Target
Nissan
Bayer
Boston Consulting Group

TRIFACTA
Wrangler

Data wrangling software: OpenRefine

“OpenRefine (formerly Google Refine) is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data. ”

http://openrefine.org

Why Python in DATA 542?

Work with different data sources

Gain programming skills for data wrangling

Free for various data sources

Main language in big data analytics technologies

Good for almost everything! From web scraping to data analytics …

Gain insights about data and the background process

Numerical arrays

Data from various format

- Documents
- Images
- Sounds
- Tables
- Numerical values

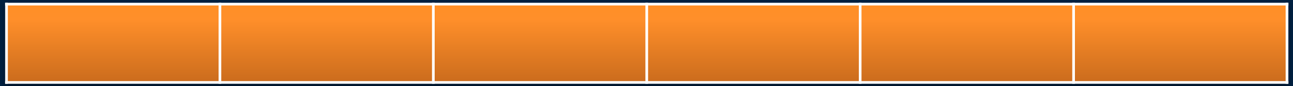
Arrays

Numpy package

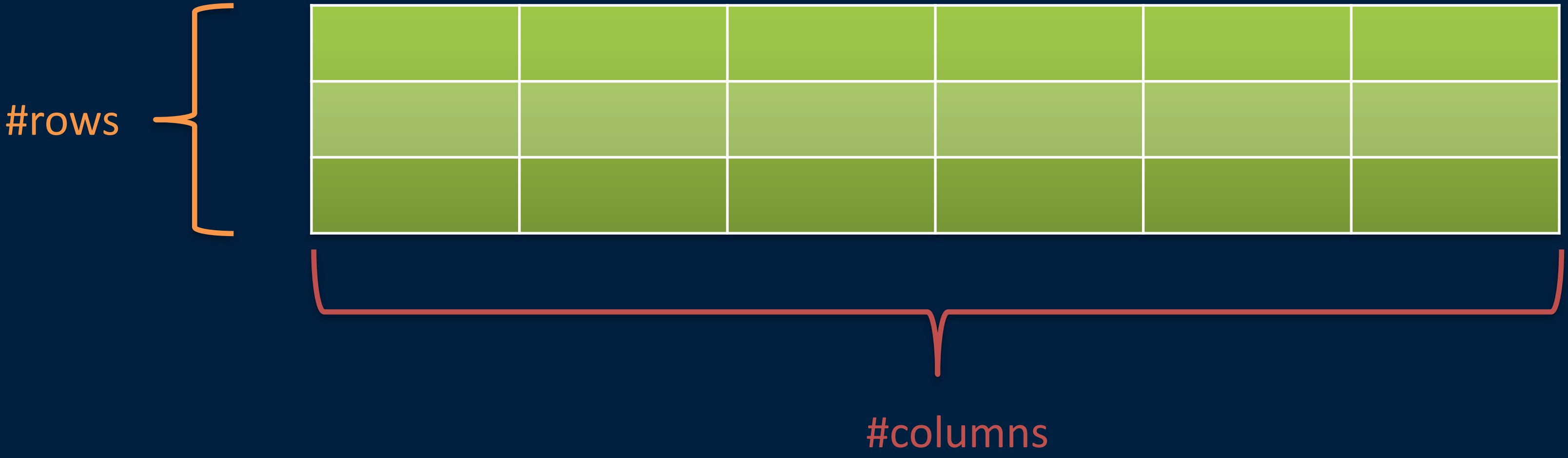
Pandas package

Arrays

1-D arrays: 1 X 6 or array of size 6

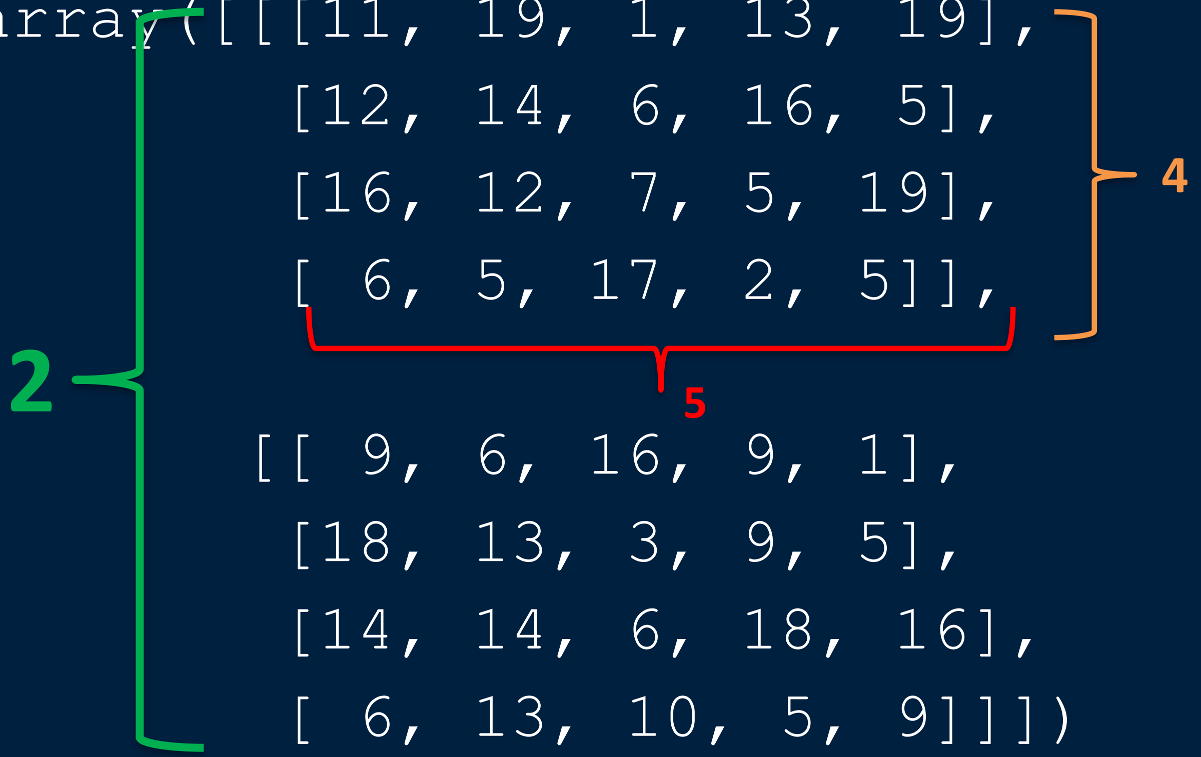


2-D arrays: rows X columns



Arrays > 2D

```
np.random.randint(20, size=(2, 4, 5))
```



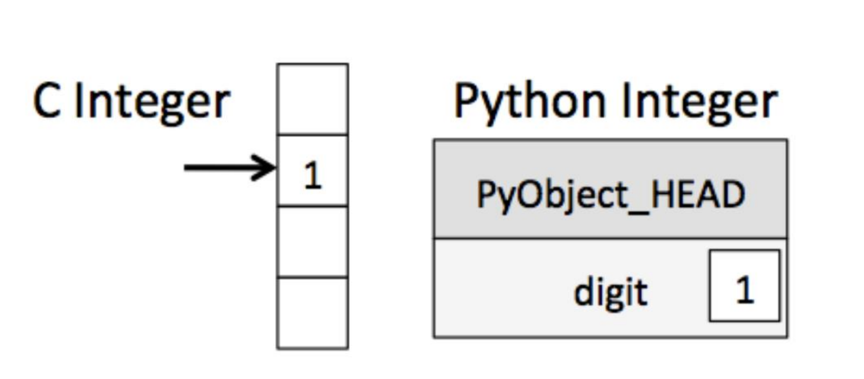
```
array([[[11, 19, 1, 13, 19],
        [12, 14, 6, 16, 5],
        [16, 12, 7, 5, 19],
        [ 6, 5, 17, 2, 5]],

       [[ 9, 6, 16, 9, 1],
        [18, 13, 3, 9, 5],
        [14, 14, 6, 18, 16],
        [ 6, 13, 10, 5, 9]]])
```

Python data types

Dynamic type allocation

```
x = 4
x = “four”
```



Static type in C:

```
int x = 4;
x = “four” //FAILS
```

- ob_refcnt, a reference count for Python to handle memory allocation and deallocation
- ob_type, which encodes the type of the variable
- ob_size, which specifies the size of the following data members
- ob_digit, which contains the actual integer value that we expect the Python variable to represent.

List: mutable multi-element container in Python

Image from: https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html

Try it: Python list

Python's dynamic typing => we can even create heterogeneous lists

Question 1: What is the output of the following code in Python?

```
L = [True, "2", 3.0, 4]
[type(item) for item in L]
```

Try it: Python list

Python's dynamic typing => we can even create heterogeneous lists

Question 1: What is the output of the following code in Python?

```
L = [True, "2", 3.0, 4]
[type(item) for item in L]
```

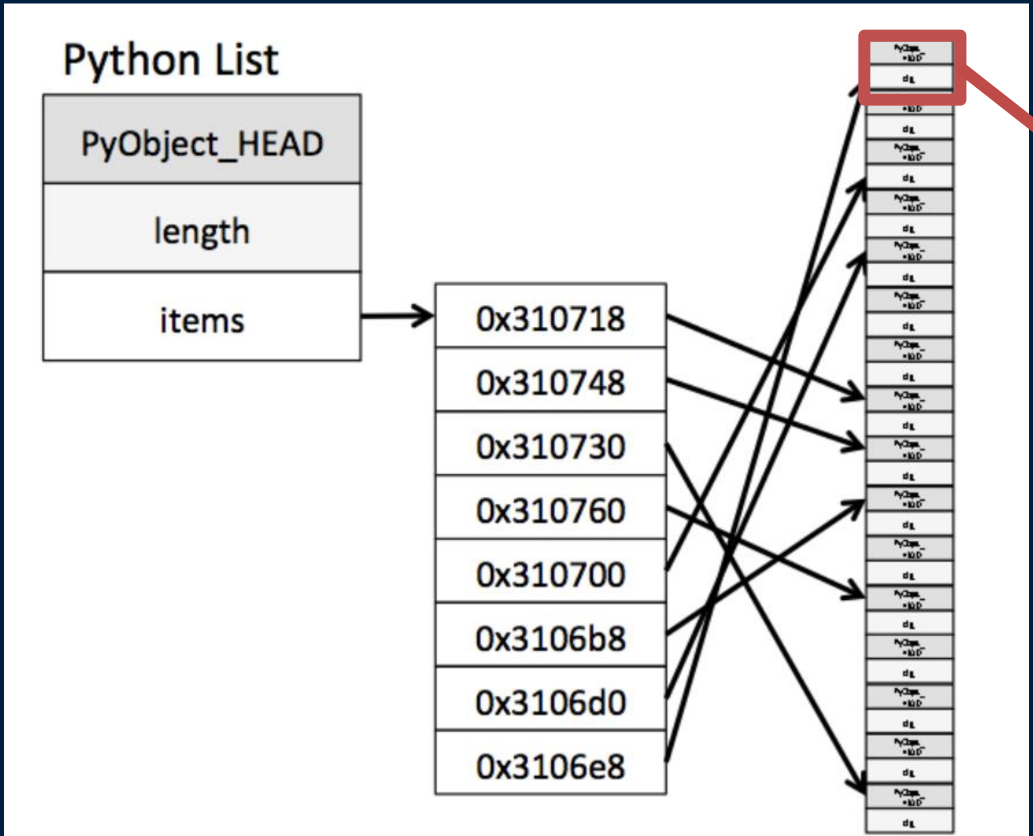
Discuss the results

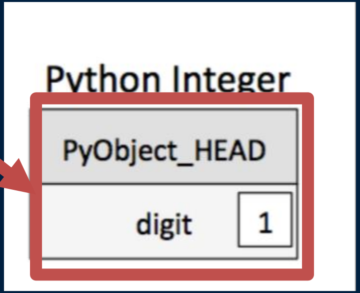
```
Output: [bool, str, float, int]
```

Python data types

Why does this matter?

List

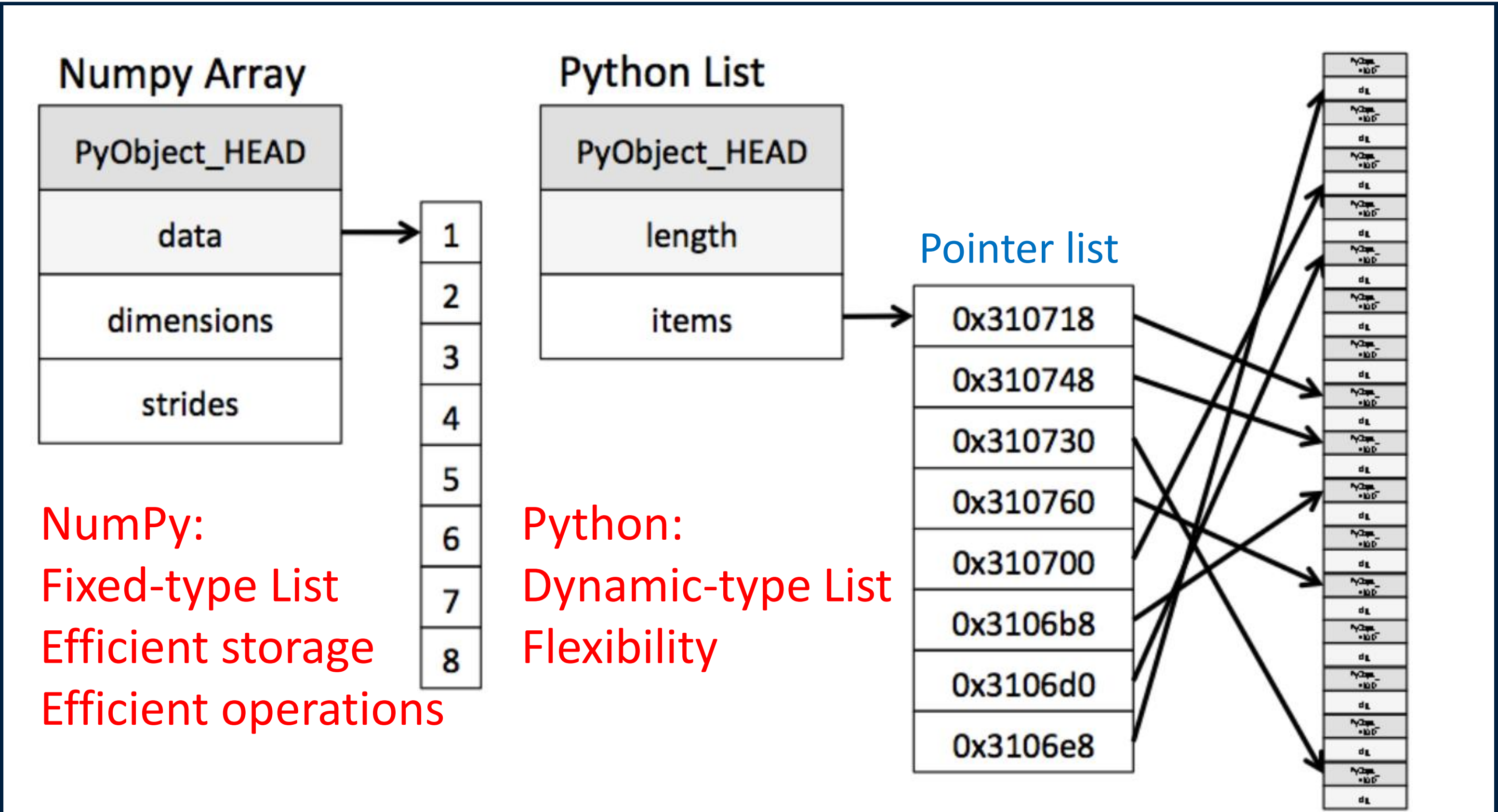




Each item is a Python Object:
reference count
object type
object size
object digit
(Python 3.4 integer)

https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html

Python data types vs NumPy data types



NumPy:
Fixed-type List
Efficient storage
Efficient operations

Python:
Dynamic-type List
Flexibility

https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html

Question

Question 1: Which one represents a dynamic type allocation in Python?

A) `for i in range(100): result += i`

B) `my_list =[“2”, 5, ‘3’, 6.0]`

C)
```
my_val = 5
my_val = True
```

D) All of the above

Question 2: NumPy list is a static-type not a dynamic type?

1) `True`

2) `False`

In Practice

NumPy introduction

Numerical Python: NumPy

Like List in Python

More efficient storage and data operations as the arrays size increase

Core of data science in Python

Install NumPy:

http://www.numpy.org/

```
 git clone https://github.com/numpy/numpy.git
numpy
```

```
 pip install numpy
```

import NumPy

```
import numpy as np
```

See np documentation/available modules

```
np.<TAB>
```

```
np?
```

Fixed-Type arrays

Since Python 3.3 we have built-in fixed-types arrays in Python:

```
import array
L = list(range(10))
A = array.array('i', L)
Output:
array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

↓

'i' is a type code

NumPy adds more efficient operations to this efficient arrays

Creating NumPy arrays from Python lists

1- From Python lists:

```
np.array([1, 4, 2, 5, 3])
Output: array([1, 4, 2, 5, 3])
```

2- Up-casting type:

```
np.array([3.14, 4, 2, 3])
Output: array([3.14, 4. , 2. , 3. ])
```

3- Explicitly set the data type:

```
np.array([1, 2, 3, 4], dtype='float32’)
Output: array([1., 2., 3., 4.], dtype=float32)
```

Creating NumPy arrays from Python lists cont.

4- From Python nested lists:

```
np.array([range(i, i + 3) for i in [2, 4, 6]])


Output:
array([[2, 3, 4],
       [4, 5, 6],
       [6, 7, 8]])
```

Question

Question 1: What is the result of the following code?

```
my_list = [2, 'm', "me", 5.6]
np.array(my_list)
```

A) `array(['2', 'm', 'me', '5.6'], dtype='<U21')`

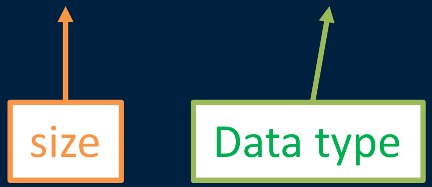
B) `Type Error`

Creating NumPy arrays from scratch

More efficient

1- One dimensional array of zeros

```
np.zeros(10, dtype=int)
```



size

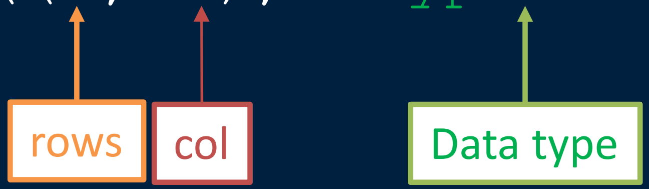
Data type

```
Output:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Creating NumPy arrays from scratch cont.

2- Two dimensional array of ones

```
np.ones((2, 5), dtype=float)
```



rows

col

Data type

```
Output:
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

Creating NumPy arrays from scratch cont.

3- Filing array with a number

```
np.full((3, 5), 3.14)
```

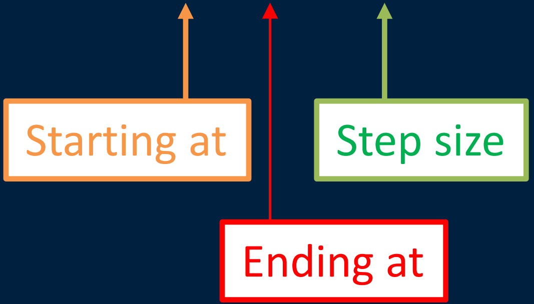
Fixed Number

```
Output:
array([[ 3.14, 3.14, 3.14, 3.14, 3.14],
       [ 3.14, 3.14, 3.14, 3.14, 3.14],
       [ 3.14, 3.14, 3.14, 3.14, 3.14]])
```

Creating NumPy arrays from scratch cont.

4- Filing array with a linear sequence

```
np.arange(0,10,3)
```



Starting at

Step size

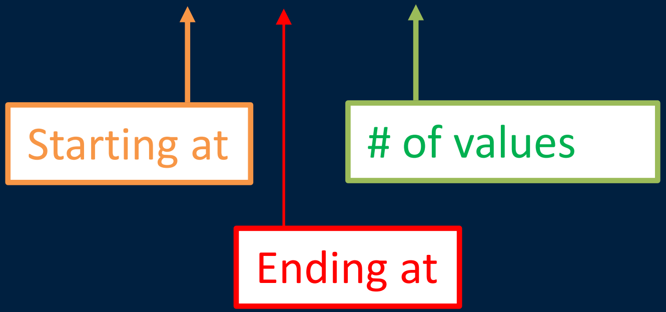
Ending at

```
Output:
array([0, 3, 6, 9])
```

Creating NumPy arrays from scratch cont.

5- Filing array with a number of values evenly spaced values in a range

```
np.linspace(0, 2, 5)
```



Starting at

# of values

Ending at

```
Output:
array([0. , 0.5, 1. , 1.5, 2. ])
```

Creating NumPy arrays from scratch cont.

6- Filing array with random values

- Uniformly distributed random values between 0 and 1 (3 X 3 array)

```
np.random.random((3, 3))
```

- Normally distributed random values with mean 0 and standard deviation 1 (3 X 3 array)

```
np.random.normal(0, 1, (3, 3))
```

- Random integers in the interval [0, 10) (3 X 3 array)

```
np.random.randint(0, 10, (3, 3))
```

Creating NumPy arrays from scratch cont.

7- identity matrix

```
np.eye(3)    #You can pass (i,j)
Output: array([[1, 0, 0],
               [0, 1, 0],
               [0, 0, 1]])
```

8- Uninitialized array

```
 np.empty(3)
 Output: array([0. , 0.5, 1. , 1.5, 2. ])
```

The values will be whatever happens to already exist at that memory location

Question

Question 1: Create an identity matrix of size 5 with integer values

A) `np.eye(5, dtype = int)`

B) `np.eye((5,5), dtype = int)`

C) `np.eye(5 )`

D) `np.eye(5,5)`

Discuss

Question 2: Create an array of values in linear sequence in range (10, 45) with step size 4, where the data type is float.

Use `np.arange()`

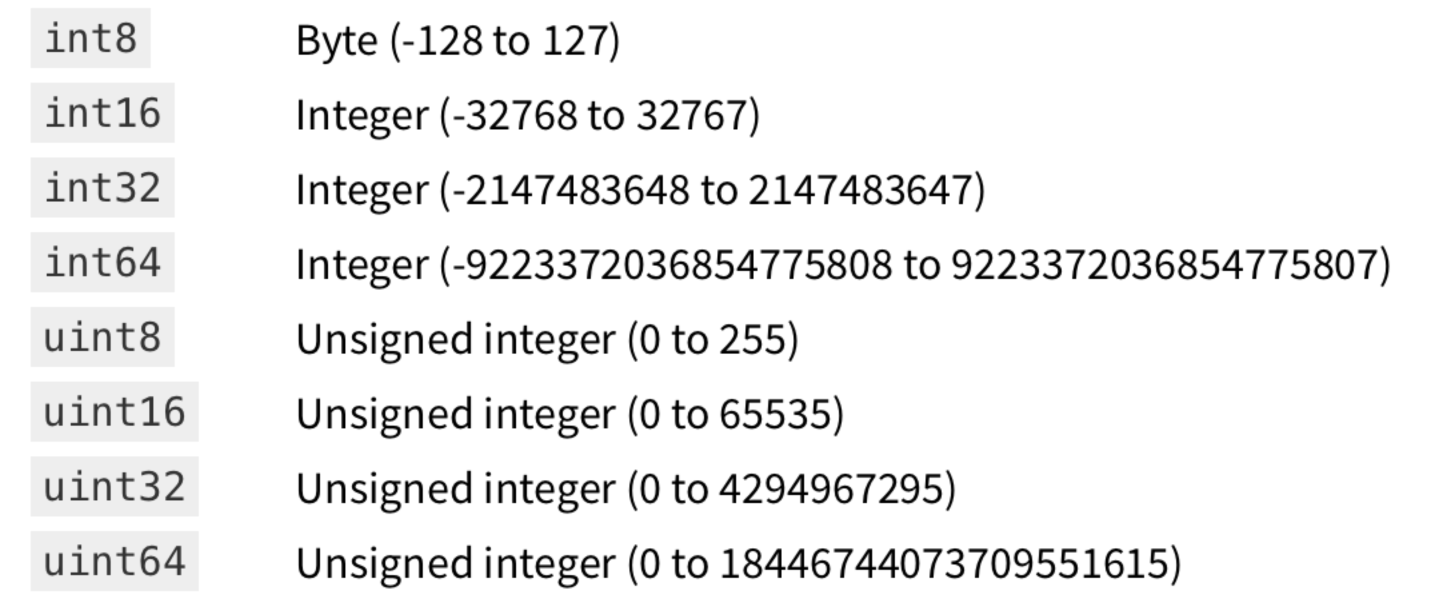
```
result = np.arange(10,45,4,dtype=float)
```

Question 3: How many numbers are in the array you created in Question 2?

```
array([10., 14., 18., 22., 26., 30., 34., 38., 42.])
result.size
```

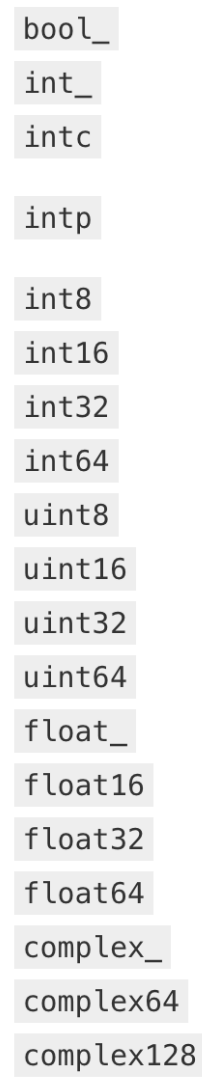
Numpy standard data types

```
dtype='int16’
dtype=np.int16
```



`int8`	Byte (-128 to 127)
`int16`	Integer (-32768 to 32767)
`int32`	Integer (-2147483648 to 2147483647)
`int64`	Integer (-9223372036854775808 to 9223372036854775807)
`uint8`	Unsigned integer (0 to 255)
`uint16`	Unsigned integer (0 to 65535)
`uint32`	Unsigned integer (0 to 4294967295)
`uint64`	Unsigned integer (0 to 18446744073709551615)

https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html



`bool_`

`int_`

`intc`

`intp`

`int8`

`int16`

`int32`

`int64`

`uint8`

`uint16`

`uint32`

`uint64`

`float_`

`float16`

`float32`

`float64`

`complex_`

`complex64`

`complex128`

NumPy array attributes

Each array has attributes

ndim: the number of dimensions

shape: the size of each dimension

size: the total size of the array

dtype: the data type of the array

itemsize: lists the size (in bytes) of each array element

nbytes: lists the total size (in bytes) of the array (itemsize X size)

NumPy array attributes - example

```
arr = np.random.randint(20, size=(3, 4, 5))


arr.ndim                output: 3
arr.shape               output: (3, 4, 5)
arr.size                output: 60 (3 X 4 X 5)
arr.dtype               output: dtype('int64')
arr.itemsize            output: 8
arr.nbytes              output: 480  (size X itemsize)
```

Array Indexing and Slicing

Array indexing

Indexing starts at 0

Negative numbers count backwards

Shown in []

Separated with , in the [] such as:

[row , col]

Array indexing - example

```
arr1 = array([9, 1, 1, 0, 7, 8])
arr2 = array([[2, 6, 8, 7, 6],
              [6, 1, 3, 4, 5],
              [7, 2, 7, 1, 5]])


arr1[-2]           output: 7
arr2[0,0]          output: 2
arr2[1,2]          output: 3
```

Question

Question 1: What is the value for `arr2[-2,-3]`

```
arr2 = array([[2, 6, 8, 7, 6],
              [6, 1, 3, 4, 5],
              [7, 2, 7, 1, 5]])
```

A) 1 **B)** 3 **C)** 7 **D)** 8 **E)** 6

Array slicing

```
x[start:stop:step]
```

Default: `start=0`, `stop=size of dimension`, `step=1`

```
x[:5] # first five elements
x[5:] # elements after index 5
x[::2] # every other element
x[::-1] # all elements, reversed
x[5::-2] # reversed every other from index 5
```

Multi dimensional arrays: separate with commas

x2: (3,4) => `x2[:3, ::2] # all rows, every other column`

Array slicing and sub-arrays

NumPy sub-array slicing:

Returns ***views***

Not *copies* of the array data

Copy them using: `x2[:2, :2].copy()`

Other useful operations

Reshape arrays: `reshape()`

Concatenate arrays : `np.concatenate([list_of_arrays])`

Split arrays: `np.split(array,[list_of_splitting_indices])`

```
x = np.array([1, 2, 3]);  y = np.array([3, 2, 1])
np.concatenate([x, y])
Output: array([1, 2, 3, 3, 2, 1])


x = [1, 2, 3, 99, 99, 3, 2, 1]
x1, x2, x3 = np.split(x, [3, 5])
Output: [1 2 3] [99 99] [3 2 1]
```

Try it: Example Questions

Question 1: create a numpy array of shape (3,4) with random numbers.

Create a numpy array of ones with the same shape.

Concatenate the two arrays

Question 2: create two one-D arrays of size 9.

Add two arrays together.

Calculate the sum, mean, min and max value of each array

Lecture 1 learning outcomes

You should be able to:

- Specify the process of data wrangling
- Explain the data pre-processing in real world
- List some of the available tools for data wrangling
- Understand NumPy arrays
- Perform functions on NumPy arrays in your Python program

THE UNIVERSITY OF BRITISH COLUMBIA