Session - 7

Numpy

Numpy: Scientific Library for Data Science!

Numpy - Numerical Python

- ndarry: multi-dimensional array proving vectorized arithmetic operations
- Mathematical operations on entire array without having to use loops
- Tools to write, read data from disk, tools to work with memory-mapped files
- Linear Algebra, Random Number Generation, and Fourier Transform capabilities.
- Tools to integrate code with lower level languages like C, C++

Numpy (cont...)

Focus is on :

- Fast Vectorized Array Operations for data munging & Cleaning, subsetting, and filtering, transformation and any kind of computation.
- Common array operations - sorting, unique, & set operations
- Statistics operations - aggregation, summarizing data
- Merging, joining data sets together
- Play with conditional logic (if-else)
- Grouping & manipulating

Data Used

	2012 - 100 meter Olympics Result			
	Time	**Height (Inches)**	**Weight (Pounds)**	**Country**
Usain Bolt	9.63	76.77	209.439	JAM
Yohan Blake	9.75	70.86	167.551	JAM
Justin Gatlin	9.79	72.83	182.984	USA
Tyson Gay	9.8	70.07	165.347	USA
Ryan Bailey	9.88	75.98	216.053	USA
Churandy Martina	9.94	70.07	163.142	NED
Richard Thompson	9.98	70.01	176.37	TTO
Asafa Powell	11.99	70.8	191.802	JAM

Revisit Lists

- Variables with more than one variables, right? Collection of values
- Can Hold different data-types (strings, integers, float or objects - anything can go in single List, right)
- Append, Change, Delete, Iterate - etc operations
- Powered with lots of tricks - comprehensions

Revisit Lists - Count BMI List Way

```
#!/usr/bin/python3
# height in inches, weight in pounds
height = [76.77,  70.86,  72.83,  70.07,  75.98,  70.07,  70.01, 70.8]
weight = [209.439, 167.551, 182.984, 165.347, 216.053, 163.142, 176.37, 191.802]

m_height = [x * 0.0254 for x in height]
k_weight = [x * 0.453592 for x in weight]

#print (m_height)
#print (k_weight)

# kg/m^2

msqr_height = [x * x for x in m_height]
#print (msqr_height)

bmi  = [k_weight[i]/msqr_height[i] for i in range(len(msqr_height))]
print (bmi)
```

Numpy - Better to Handle Data Calculations

- Need is Mathematical or Statistical Calculations over Collections
- Speed
- Simplicity - should look natural Mathematics

Numpy - Look at bmi example

```
#!/usr/bin/python3

import numpy as np

# height in inches, weight in pounds
height = [76.77,  70.86,  72.83,  70.07,  75.98,  70.07,  70.01, 70.8]
weight = [209.439, 167.551, 182.984, 165.347, 216.053, 163.142, 176.37, 191.802]

np_height = np.array(height)
np_weight = np.array(weight)


np_height_meters  = np_height  * 0.0254
np_weight_kgs = np_weight *  0.453592

#print (m_height)
#print (k_weight)

# kg/m^2

np_height_meters_sqr  = np_height_meters ** 2
#print (msqr_height)

bmi = np_weight_kgs / np_height_meters_sqr

print (bmi)
```

Importing Numpy Library

Creating Numpy Array - from List

Multiply each element of Numpy Array

Square each element

Divide each element

Numpy - A few Observations

```
>>> import numpy as np
>>> my_numpy = np.array([11.00, 12, 'numpy', True, False])
>>> type(my_numpy)
<class 'numpy.ndarray'>
>>> my_numpy
array(['11.0', '12', 'numpy', 'True', 'False'],
      dtype='<U32')
>>> my_list = [11.00, 12, 'numpy', True, False]
>>> type(my_list)
<class 'list'>
>>> my_list + my_list
[11.0, 12, 'numpy', True, False, 11.0, 12, 'numpy', True, False]
>>> my_numpy + my_numpy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: ufunc 'add' did not contain a loop with signature matching types dtype('<U32') dtype('<U32') dtype('<U32')
>>> my_numpy_n = np.array([1, 2, 3])
>>> type(my_numpy_n)
<class 'numpy.ndarray'>
>>> my_numpy_n
array([1, 2, 3])
>>> my_numpy_n + my_numpy_n
array([2, 4, 6])
>>>
```

Numpy - Subsetting

```
>>> import numpy as np
>>> bmi = np.array([ 24.98460153,  23.46079266,  24.25439523,  23.67718338,  26.31235232, 23.36143414,  25.29895093,  26.9
005])
>>> type(bmi)
<class 'numpy.ndarray'>
>>> bmi[1]
23.460792659999999
>>> bmi[0]
24.984601529999999
>>> bmi[11]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: index 11 is out of bounds for axis 0 with size 8
>>> bmi > 24
array([ True, False,  True, False,  True, False,  True,  True], dtype=bool)
>>> bmi > 23
array([ True,  True,  True,  True,  True,  True,  True,  True], dtype=bool)
>>> bmi > 22
array([ True,  True,  True,  True,  True,  True,  True,  True], dtype=bool)
>>> bmi[bmi > 22]
array([ 24.98460153,  23.46079266,  24.25439523,  23.67718338,
        26.31235232,  23.36143414,  25.29895093,  26.90200005])
>>> bmi[bmi > 24]
array([ 24.98460153,  24.25439523,  26.31235232,  25.29895093,  26.90200005])
>>> bmi[bmi > 25]
array([ 26.31235232,  25.29895093,  26.90200005])
>>> bmi[bmi > 26]
array([ 26.31235232,  26.90200005])
>>> bmi[bmi > 26.9]
array([ 26.90200005])
>>>
```

Practice - One (Create Numpy Array)

- Import the **numpy** package as **np**
- Use **np.array()** to create a Numpy array from **m_runners**. Name this array **np_m_runners** Print out the **type** of **np_m_runners** to check that you got it right.

```
#          one.py

# This is 2012 Olympics Game 100 meter speed result
m_runners = [9.63, 9.75, 9.79, 9.80, 9.88, 9.94, 9.98, 11.99]

# Import the numpy package as np

# Create a Numpy array from m_runners: np_m_runners

# Print out type of np_m_runners

# Print np_m_runners
```

Practice Two - Count Hight

- Create a Numpy array from height. Name this new array np_height.
- Print np_height.
- Multiply np_height with 0.0254 to convert all height measurements from inches to meters. Store the new values in a new array, np_height_m.
- Print out np_height_m

```
#     #two.py
# Import numpy
import numpy as np

height = [76.77,  70.86,  72.83,  70.07,  75.98,  70.07,  70.01, 70.8]
# Create a Numpy array from height: np_height

# Print out np_height

# Convert np_height to m: np_height_m
# You need to multiply by  0.0254

# Print np_height_m
```

Practice Three - Count bmi

- Create a Numpy array from the weight list, height list with the correct units. Call them - np_weight_kg & np_height_m
- Use np_height_m and np_weight_kg to calculate the BMI of each player. Use the following equation: BMI=weight(kg)/height(m)^2
- Save the resulting numpy array as bmi.
- Print out bmi.

```
# height in inches & weight in pounds
height = [76.77,  70.86,  72.83,  70.07,  75.98,  70.07,  70.01, 70.8]
weight = [209.439, 167.551, 182.984, 165.347, 216.053, 163.142, 176.37, 191.802]
# Import numpy
# Create array from height with correct units: np_height_m  - multiplication by  0.0254
# Create array from weight with correct units: np_weight_kg - multiplication by 0.453592
# Calculate the BMI: bmi  - use kg/m^2
# Print out bmi
```

Practice Four - Lightweight

- Create a boolean Numpy array: the element of the array should be True if the corresponding runners BMI is below 23. You can use the < operator for this. Name the array light.
- Print the array light.
- Print out a Numpy array with the BMIs of all runners whose BMI is below 23. Use light inside square brackets to do a selection on the bmi array.

```
# height and weight are available as a regular lists
height = [76.77,  70.86,  72.83,  70.07,  75.98,  70.07,  70.01, 70.8]
weight = [209.439, 167.551, 182.984, 165.347, 216.053, 163.142, 176.37, 191.802]
# Import numpy

# Calculate the BMI: bmi
np_height_m = np.array(height) * 0.0254
np_weight_kg = np.array(weight) * 0.453592
bmi = np_weight_kg / np_height_m ** 2

# Create the light array
# Print out light
# Print out BMIs of all runners whose BMI is below 23
```

Practice Five - Match the Following

1

```
>>> demo = np.array([1, 2, 3, 4, 5, 6, 7, 8])
>>> demo>2
```

2

```
>>> demo = np.array([1, 2, 3, 4, 5, 6, 7, 8])
>>> demo[demo >5]
```

3

```
>>> a = np.array([1, 2, 3, True])
>>> b = np.array([False, 2, 3, 4])
>>> a + b
```

4

```
>>> demo = np.array([1, 2, 3, 4, 5, 6, 7, 8])
>>> demo[demo < 1]
```

A

```
array([6, 7, 8])
```

B

```
array([1, 4, 6, 5])
```

C

```
array([], dtype=int64)
```

D

```
array([False, False,  True,  True,  True,  True,  True,  True], dtype=bool)
```

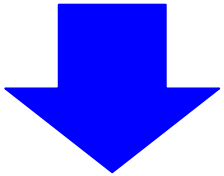
Practice Six - Subsetting

- Subset np_weight: print out the element at index 5.
- Print out a sub-array of np_height: It contains the elements at index 3 up to and including index 6

height = [76.77, 70.86, 72.83, 70.07, 75.98, 70.07, 70.01, 70.8]
weight = [209.439, 167.551, 182.984, 165.347, 216.053, 163.142, 176.37, 191.802]

Caching or Memoization

```
def some_function (arg1, arg2, arg3,..., argN):
        result = some computation involving arg1, arg2, arg3, ... argN
        return result
```



```
catche = {}
def some_function_with_catching(arg1, arg2, arg3, ..., argN):
        key = str(arg1) + str(arg2) + str(arg3) + .... + str(argN)
        if key in cache:
              return catche[key]
        else:
             result = same computation involving arg1, arg2, arg3, ..., argN
             catche[key]=result
             return result
```

Caching or Memoization - Fibonacci Numbers

The Fibonacci Sequence is the series of numbers:

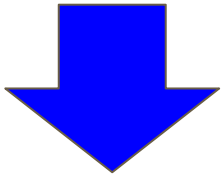
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The next number is found by adding up the two numbers before it.

- The 2 is found by adding the two numbers before it (1+1)
- The 3 is found by adding the two numbers before it (1+2),
- And the 5 is (2+3),
- and so on!

Caching or Memoization - Fibonacci Numbers

```
def fib(n):
    return n if n < 2 else fib(n-2) + fib(n-1)
```



```
__fib_cache = {}
def fib(n):
    if n in __fib_cache:
        return __fib_cache[n]
    else:
        __fib_cache[n] = n if n < 2 else fib(n-2) + fib(n-1)
        return __fib_cache[n]
```