# Lab7-FullNarrative

September 22, 2020

```
[1]: %%html
     <!--Script block to left align Markdown Tables-->
     <style>
       table {margin-left: 0 !important;}
     </style>
```

```
[1]: <IPython.core.display.HTML object>
```

```
[2]: # Preamble script block to identify host, user, and kernel
     import sys
     ! hostname
     ! whoami
     print(sys.executable)
     print(sys.version)
     print(sys.version_info)
```

```
project-89cf5069-98c0-48bd-be52-58fb94d82c15
user
/usr/bin/python3
3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0]
sys.version_info(major=3, minor=8, micro=2, releaselevel='final', serial=0)
```

## 0.1 Zachary Dansby

## 0.2 R#: 11722464

## 0.3 HEX: 0xb2dee0

## 0.4 Lab 7

## 0.5 9/22/2020

### 0.5.1 Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. The library's name is short for "Numeric Python" or "Numerical Python". If you are curious about NumPy, this cheat sheet is recommended: https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf

**Arrays** A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension. In other words, an array contains information about the raw data, how to locate an element and how to interpret an element.To make a numpy array, you can just use the np.array() function. All you need to do is pass a list to it. Don't forget that, in order to work with the np.array() function, you need to make sure that the numpy library is present in your environment. If you want to read more about the differences between a Python list and NumPy array, this link is recommended: https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference

### 0.5.2 Example- 1D Arrays

**Let's create a 1D array from the 2000s (2000-2009):**

[1]:
```
import numpy as np                  #First, we need to impoty "numpy"
mylist = [2000,2001,2002,2003,2004,2005,2006,2007,2008,2009]          #Create a␣
 ↪list of the years
print(mylist)                      #Check how it looks
np.array(mylist)                   #Define it as a numpy array
```

```
[2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009]
```

[1]:
```
array([2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009])
```

### 0.5.3 Example- n-Dimensional Arrays

**Let's create a 5x2 array from the 2000s (2000-2009):**

[8]:
```
myotherlist = [[2000,2001],[2002,2003],[2004,2005],[2006,2007],[2008,2009]]    ␣
 ↪#Since I want a 5x2 array, I should group the years two by two
print(myotherlist)                  #See how it looks as a list
np.array(myotherlist)               #See how it looks as a numpy array
```

```
[[2000, 2001], [2002, 2003], [2004, 2005], [2006, 2007], [2008, 2009]]
```

[8]:
```
array([[2000, 2001],
       [2002, 2003],
       [2004, 2005],
       [2006, 2007],
       [2008, 2009]])
```

### 0.5.4 Exercise 1

**Using the numbers below:** #### 9,18,333,22,25,120,7,43,50,3,7,94,953,57,11,154

**A) Create a 1D array B) Create a 4x4 array so that:** - all numbers in the first row are single digits - all numbers in the second row are two digit and even numbers - all numbers in the third row are two digit and odd numbers - all numbers in the fourth row are three digit numbers

```
[9]: import numpy as np
     #A
     funlist = [9,18,333,22,25,120,7,43,50,3,7,94,953,57,11,154]
     print(funlist)
     #B
     funar = [[9,7,3,7],[18,22,50,94],[25,43,57,11],[333,120,953,154]]
     print(funar)

     np.array(funar)
```

```
[9, 18, 333, 22, 25, 120, 7, 43, 50, 3, 7, 94, 953, 57, 11, 154]
[[9, 7, 3, 7], [18, 22, 50, 94], [25, 43, 57, 11], [333, 120, 953, 154]]
```

```
[9]: array([[  9,   7,   3,   7],
            [ 18,  22,  50,  94],
            [ 25,  43,  57,  11],
            [333, 120, 953, 154]])
```

```
[0]:
```

**Arrays Arithmetic** Once you have created the arrays, you can do basic Numpy operations. Numpy offers a variety of operations applicable on arrays. From basic operations such as summation, subtraction, multiplication and division to more advanced and essential operations such as matrix multiplication and other elementwise operations. In the examples below, we will go over some of these:

### 0.5.5 Example- 1D Array Arithmetic

- Define a 1D array with [0,12,24,36,48,60,72,84,96]
- Multiple all elements by 2
- Take all elements to the power of 2
- Find the maximum value of the array and its position
- Find the minimum value of the array and its position
- Define another 1D array with [-12,0,12,24,36,48,60,72,84]
- Find the summation and subtraction of these two arrays
- Find the multiplication of these two arrays

```
[29]: import numpy as np         #import numpy
      Array1 = np.array([0,12,24,36,48,60,72,84,96])     #Step1: Define Array1
      print(Array1)
      print(Array1*2)     #Step2: Multiple all elements by 2
      print(Array1**2)     #Step3: Take all elements to the power of 2
      print(np.power(Array1,2)) #Another way to do the same thing, by using a␣
       ↪function in numpy
      print(np.max(Array1))     #Step4: Find the maximum value of the array
      print(np.argmax(Array1))     ##Step4: Find the postition of the maximum value
      print(np.min(Array1))     #Step5: Find the minimum value of the array
      print(np.argmin(Array1))      ##Step5: Find the postition of the minimum value
      Array2 = np.array([-12,0,12,24,36,48,60,72,84])     #Step6: Define Array2
      print(Array2)
      print(Array1+Array2)         #Step7: Find the summation of these two arrays
      print(Array1-Array2)         #Step7: Find the subtraction of these two arrays
      print(Array1*Array2)         #Step8: Find the multiplication of these two arrays
```

```
[ 0 12 24 36 48 60 72 84 96]
[  0  24  48  72  96 120 144 168 192]
[   0  144  576 1296 2304 3600 5184 7056 9216]
[   0  144  576 1296 2304 3600 5184 7056 9216]
96
8
0
0
[-12   0  12  24  36  48  60  72  84]
[-12  12  36  60  84 108 132 156 180]
[12 12 12 12 12 12 12 12 12]
[   0    0  288  864 1728 2880 4320 6048 8064]
```

### 0.5.6 Example- n-Dimensional Array Arithmetic

- Define a 2x2 array with [5,10,15,20]
- Define another 2x2 array with [3,6,9,12]
- Find the summation and subtraction of these two arrays
- Find the minimum number in the multiplication of these two arrays
- Find the position of the maximum in the multiplication of these two arrays
- Find the mean of the multiplication of these two arrays
- Find the mean of the first row of the multiplication of these two arrays

[42]:
```
import numpy as np          #import numpy
Array1 = np.array([[5,10],[15,20]])     #Step1: Define Array1
print(Array1)
Array2 = np.array([[3,6],[9,12]])     #Step2: Define Array2
print(Array2)
print(Array1+Array2)     #Step3: Find the summation
print(Array1-Array2)     #Step3: Find the subtraction
MultArray = Array1@Array2         #Step4: To perform a typical matrix␣
 ↪multiplication (or matrix product)
MultArray1 = Array1.dot(Array2)         #Step4: Another way To perform a ␣
 ↪matrix multiplication
print(MultArray)
print(MultArray1)
print(np.min(MultArray))      #Step4: Find the minimum value of the␣
 ↪multiplication
print(np.argmax(MultArray))      ##Step5: Find the postition of the maximum␣
 ↪value
```

```
print(np.mean(MultArray))      ##Step6: Find the mean of the multiplication of␣
 ↪these two arrays
print(np.mean(MultArray[0,:]))      ##Step7: Find the mean of the first row of␣
 ↪the multiplication of these two arrays
```

```
[[ 5 10]
 [15 20]]
[[ 3  6]
 [ 9 12]]
[[ 8 16]
 [24 32]]
[[2 4]
 [6 8]]
[[105 150]
 [225 330]]
[[105 150]
 [225 330]]
105
3
202.5
127.5
```

### 0.5.7 Exercise 2

**Using the numbers below:** #### 88,-5,9,22,46,2,-1,0

**A) Create a 2x4 array and print it out B) Add 6 to all elements, store it as a new object, and print it out C) Multiply all elements by 5, store it as a new object, and print it out D) Divide all elements by 7, store it as a new object, and print it out**

[63]:
```
import numpy as np
x = 0
i = 0
#A
funr = [[88,-5],[9,22],[46,2],[-1,0]]
funr = np.array([[88,-5],[9,22],[46,2],[-1,0]])
print(funr)
#B
funsix =  funr + 6
print(funsix)
#C
funfive =  funr * 5
print(funfive)
#D
funsev = funr / 7
print(funsev)
```

```
[[88 -5]
```

```
 [ 9 22]
 [46  2]
 [-1  0]]
[[94  1]
 [15 28]
 [52  8]
 [ 5  6]]
[[440 -25]
 [ 45 110]
 [230  10]
 [ -5   0]]
[[12.57142857 -0.71428571]
 [ 1.28571429  3.14285714]
 [ 6.57142857  0.28571429]
 [-0.14285714  0.        ]]
```

**Arrays Comparison** Comparing two NumPy arrays determines whether they are equivalent by checking if every element at each corresponding index are the same.

### 0.5.8 Example- 1D Array Comparison

- Define a 1D array with [1.0,2.5,3.4,7,7]
- Define another 1D array with [5.0/5.0,5.0/2,6.8/2,21/3,14/2]
- Compare and see if the two arrays are equal
- Define another 1D array with [6,1.4,2.2,7.5,7]
- Compare and see if the first array is greater than or equal to the third array

[1]:
```python
import numpy as np          #import numpy
Array1 = np.array([1.0,2.5,3.4,7,7])     #Step1: Define Array1
print(Array1)
Array2 = np.array([5.0/5.0,5.0/2,6.8/2,21/3,14/2])     #Step2: Define Array1
print(Array2)
print(np.equal(Array1, Array2))          #Step3: Compare and see if the two␣
 ↪arrays are equal
Array3 = np.array([6,1.4,2.2,7.5,7])     #Step4: Define Array3
print(Array3)
print(np.greater_equal(Array1, Array3))          #Step3: Compare and see if␣
 ↪the two arrays are equal
```

```
[1.  2.5 3.4 7.  7. ]
[1.  2.5 3.4 7.  7. ]
[ True  True  True  True  True]
[6.  1.4 2.2 7.5 7. ]
[False  True  True False  True]
```

### 0.5.9 Exercise 3- n-Dimensional Array Comparison

**Using the numbers below:** #### 4,9,23,39,40,55,68,72

**A) Create a 2x2 array with all the even numbers and print it out B) Create a 2x2 array with all the odd numbers and print it out C) Compare the two arrays using the "less_equal" function from numpy package**

[67]:
```
import numpy as np
funev = np.array([[4,40],[68,72]])
print(funev)

funodd = np.array([[9,23],[39,55]])
print(funodd)

funev <= funodd
```

```
[[ 4 40]
 [68 72]]
[[ 9 23]
 [39 55]]
```

[67]:
```
array([[ True, False],
       [False, False]])
```

**Arrays Manipulation** numpy.copy() allows us to create a copy of an array. This is particularly useful when we need to manipulate an array while keeping an original copy in memory. The numpy.delete() function returns a new array with sub-arrays along an axis deleted. Let's have a look at the examples.

### 0.5.10 Example- Copying and Deleting Arrays and Elements

- Define a 1D array, named "x" with [1,2,3]
- Define "y" so that "y=x"
- Define "z" as a copy of "x"
- Discuss the difference between y and z
- Delete the second element of x

[7]:
```
import numpy as np          #import numpy
x = np.array([1,2,3])      #Step1: Define x
print(x)
y = x                      #Step2: Define y as y=x
print(y)
z = np.copy(x)             #Step3: Define z as a copy of x
print(z)
# For Step4: They look similar but check this out:
x[1] = 8                   # If we change x ...
print(x)
print(y)
print(z)
# By modifying x, y changes but z remains as a copy of the initial version of x.
x = np.delete(x, 1)       #Step5: Delete the second element of x
```

```
print(x)
```

```
[1 2 3]
[1 2 3]
[1 2 3]
[1 8 3]
[1 8 3]
[1 2 3]
[1 3]
```

### 0.5.11 Exercise 4

**Using the numbers below:** #### 1,2,3,4,5,6,7,8,9

**A) Create a 1D array and print it out B) Delete all the even numbers and print out the new version C) Make a copy of the array from step B and print it out**

[12]:
```
import numpy as np
#A
funone = np.array([1,2,3,4,5,6,7,8,9])
print(funone)
#B
funone = funone[(funone %2 != 0)]
print(funone)
#C
funeven = np.copy(funone)
print(funeven)
```

```
[1 2 3 4 5 6 7 8 9]
[1 3 5 7 9]
[1 3 5 7 9]
```

**Sorting Arrays** Sorting means putting elements in an ordered sequence. Ordered sequence is any sequence that has an order corresponding to elements, like numeric or alphabetical, ascending or descending. If you use the sort() method on a 2-D array, both arrays will be sorted.

### 0.5.12 Example- Sorting 1D Arrays

**Define a 1D array as [‘FIFA 2020’,‘Red Dead Redemption’,‘Fallout’,‘GTA’,‘NBA 2018’,‘Need For Speed’] and print it out. Then, sort the array alphabetically.**

[10]:
```
import numpy as np          #import numpy
games = np.array(['FIFA 2020','Red Dead Redemption','Fallout','GTA','NBA␣
 ↪2018','Need For Speed'])
print(games)
print(np.sort(games))
```

```
['FIFA 2020' 'Red Dead Redemption' 'Fallout' 'GTA' 'NBA 2018'
 'Need For Speed']
```

```
['FIFA 2020' 'Fallout' 'GTA' 'NBA 2018' 'Need For Speed'
 'Red Dead Redemption']
```

### 0.5.13 Example- Sorting n-Dimensional Arrays

**Define a 3x3 array with 17,-6,2,86,-12,0,0,23,12 and print it out. Then, sort the array.**

[18]:
```
import numpy as np          #import numpy
a = np.array([[17,-6,2],[86,-12,0],[0,23,12]])
print(a)
print ("Along columns : \n", np.sort(a,axis = 0) ) #This will be sorting in␣
↪each column
print ("Along rows : \n", np.sort(a,axis = 1) ) #This will be sorting in each␣
↪row
print ("Sorting by default : \n", np.sort(a) )          #Same as above
print ("Along None Axis : \n", np.sort(a,axis = None) ) #This will be sorted␣
↪like a 1D array
```

```
[[ 17  -6   2]
 [ 86 -12   0]
 [  0  23  12]]
Along columns :
 [[  0 -12   0]
 [ 17  -6   2]
 [ 86  23  12]]
Along rows :
 [[ -6   2  17]
 [-12   0  86]
 [  0  12  23]]
Sorting by default :
 [[ -6   2  17]
 [-12   0  86]
 [  0  12  23]]
Along None Axis :
 [-12  -6   0   0   2  12  17  23  86]
```

### 0.5.14 Exercise 5

**A) Google: "Python how to Sort Numpy", find a good link with appropriate explanation, and add the link to your notebook Then, use the 4x4 array from Exercise 1, part2 and ... B) Print it out C) Sort it along the rows and print it out D) Sort it along the columns and print it out**

[14]:
```
#A
#https://jakevdp.github.io/PythonDataScienceHandbook/02.08-sorting.html
#B
funar = np.array([[9,7,3,7],[18,22,50,94],[25,43,57,11],[333,120,953,154]])
print(funar)
```

```
#C
print(np.sort(funar,axis = 1))
#D
print(np.sort(funar,axis = 0))
```

```
[[  9   7   3   7]
 [ 18  22  50  94]
 [ 25  43  57  11]
 [333 120 953 154]]
[[  3   7   7   9]
 [ 18  22  50  94]
 [ 11  25  43  57]
 [120 154 333 953]]
[[  9   7   3   7]
 [ 18  22  50  11]
 [ 25  43  57  94]
 [333 120 953 154]]
```

**Partitioning (Slice) Arrays** Slicing in python means taking elements from one given index to another given index.

We can do slicing like this: [start:end].

We can also define the step, like this: [start:end:step].

If we don't pass start its considered 0

If we don't pass end its considered length of array in that dimension

If we don't pass step its considered 1

### 0.5.15 Example- Slicing 1D Arrays

**Define a 1D array as [1,3,5,7,9], slice out the [3,5,7] and print it out.**

[34]:
```
import numpy as np          #import numpy
a = np.array([1,3,5,7,9])     #Define the array
print(a)
aslice = a[1:4]              #slice the [3,5,7]
print(aslice)                #print it out
```

```
[1 3 5 7 9]
[3 5 7]
```

### 0.5.16 Example- Slicing n-Dimensional Arrays

**Define a 5x5 array with "Superman, Batman, Jim Hammond, Captain America, Green Arrow, Aquaman, Wonder Woman, Martian Manhunter, Barry Allen, Hal Jordan, Hawkman, Ray Palmer, Spider Man, Thor, Hank Pym, Solar, Iron Man, Dr. Strange, Daredevil, Ted Kord, Captian Marvel, Black Panther, Wolverine, Booster Gold, Spawn" and print it out. Then:** - Slice the first column and print it out - Slice the third row and print it out - Slice 'Wolverine' and print it out - Slice a 3x3 array with 'Wonder Woman, Ray Palmer, Iron Man, Martian Manhunter, Spider Man, Dr. Strange, Barry Allen, Thor, Daredevil'

[35]:
```
import numpy as np          #import numpy
Superheroes = np.array([['Superman', 'Batman', 'Jim Hammond', 'Captain␣
 ↪America', 'Green Arrow'],
                ['Aquaman', 'Wonder Woman', 'Martian Manhunter', 'Barry Allen',␣
 ↪'Hal Jordan'],
                ['Hawkman', 'Ray Palmer', 'Spider Man', 'Thor', 'Hank Pym'],
                ['Solar', 'Iron Man', 'Dr. Strange', 'Daredevil', 'Ted Kord'],
                ['Captian Marvel', 'Black Panther', 'Wolverine', 'Booster Gold',␣
 ↪'Spawn']])
print(Superheroes) #Step1
print(Superheroes[:,0])
print(Superheroes[2,:])
print(Superheroes[4,2])
print(Superheroes[1:4,1:4])
```

```
[['Superman' 'Batman' 'Jim Hammond' 'Captain America' 'Green Arrow']
 ['Aquaman' 'Wonder Woman' 'Martian Manhunter' 'Barry Allen' 'Hal Jordan']
 ['Hawkman' 'Ray Palmer' 'Spider Man' 'Thor' 'Hank Pym']
 ['Solar' 'Iron Man' 'Dr. Strange' 'Daredevil' 'Ted Kord']
 ['Captian Marvel' 'Black Panther' 'Wolverine' 'Booster Gold' 'Spawn']]
['Superman' 'Aquaman' 'Hawkman' 'Solar' 'Captian Marvel']
['Hawkman' 'Ray Palmer' 'Spider Man' 'Thor' 'Hank Pym']
Wolverine
[['Wonder Woman' 'Martian Manhunter' 'Barry Allen']
 ['Ray Palmer' 'Spider Man' 'Thor']
 ['Iron Man' 'Dr. Strange' 'Daredevil']]
```

### 0.5.17 Exercise 6

**Using the names below:**

"Cartman, Kenny, Kyle, Stan, Butters, Wendy, Chef, Mr. Mackey, Randy, Sharon, Sheila, Towelie"

**A) Create a 3x4 array and print it out B) Sort the array alphabetically by column and print it out C) From the sorted array, slice out a 2x2 array with ('Cartman, Randy, Sharon, Wendy'), and print it out**

[7]:
```
import numpy as np
#A
names = np.array([['Cartman', 'Kenny', 'Stan'],['Kyle', 'Butters',␣
 ↪'Wendy'],['Chef', 'Mr.Mackey', 'Randy'],['Sharon', 'Sheila', 'Towelie']])
print(names)
#B
sname = np.sort(names)
print(np.sort(names))
```

```python
#C
sort = np.array([[sname[0,0],sname[2,2]],[sname[3,0],sname[1,2]]])
print(sort)
```

```
[['Cartman' 'Kenny' 'Stan']
 ['Kyle' 'Butters' 'Wendy']
 ['Chef' 'Mr.Mackey' 'Randy']
 ['Sharon' 'Sheila' 'Towelie']]
[['Cartman' 'Kenny' 'Stan']
 ['Butters' 'Kyle' 'Wendy']
 ['Chef' 'Mr.Mackey' 'Randy']
 ['Sharon' 'Sheila' 'Towelie']]
[['Cartman' 'Randy']
 ['Sharon' 'Wendy']]
```

## 0.6 References

Johnson, J. (2020). Python Numpy Tutorial (with Jupyter and Colab). Retrieved September 15, 2020, from https://cs231n.github.io/python-numpy-tutorial/

Willems, K. (2019). (Tutorial) Python NUMPY Array TUTORIAL. Retrieved September 15, 2020, from https://www.datacamp.com/community/tutorials/python-numpy-tutorial?utm_source=adwords_ppc

Willems, K. (2017). NumPy Cheat Sheet: Data Analysis in Python. Retrieved September 15, 2020, from https://www.datacamp.com/community/blog/python-numpy-cheat-sheet

W3resource. (2020). NumPy: Compare two given arrays. Retrieved September 15, 2020, from https://www.w3resource.com/python-exercises/numpy/python-numpy-exercise-28.php