# HASHANALYTICS DATA ANALYSIS INTERNSHIP-Charity Wanjiru

## ASSIGNMENT 1

### NUMPY FUNDAMENTALS AND PANDAS

Installations needed

* numpy
* pandas

The first step after installing the needed packages, create new project and inside the project create new file where all the code goes. Then import the packages you will need. For exampleimport numpy as np

#### ARRAYS IN NUMPY

An array in numpy is similar to a python list. e.g this is a list of integers x=[1,2,3,4]. There are different types of arrays such as **one dimension arrays** my\_list=[1,2,3,4] my\_array=np.array(my\_list), **two dimension arrays** my\_list2=[5,6,7,8] my2d\_array=np.array([my\_list,my\_list2]).

In instances where you want to check how many rows and columns your array contains. use the.shape function like so print(my2d\_array.shape). To find out the data type that your array containsprint(my\_array.dtype).

#### Using Other Numpy Functions to create arrays

using **zero**,**ones,eye**, **empty**,**arange,exponential,sqrt,add,random**,**Maximum**

zero\_array=np.zeros(5) #creates a new array with(1-5)all elements are zero

ones\_array=np.ones([5,5]) #get all elements as 1(5rows/5columns)

empty\_array=np.empty(5)#creates randoms values from memory

eye\_array=np.eye(3,3)#n\*n matrix with ones and zeros

arange\_array=np.arange(5,50,2)#create array with specified sequence

expo=np.exp(arange\_array) #using exponetial function

sqrt\_func=np.sqrt(arange\_array) #using the sqrt function

added=np.add(array1,array1) #using add funct to add two arrays

maxed=np.maximum(added,array1)#using maximum function

#### scalar operations on array

scalar operations allow you to treat items of an array the same while dealing with it as arrays

array1=np.array([[1,2,3,4],[5,6,7,8]]) #two dimension array

**scalar multiplication**

array2=array1\*array1 #each number multiplies with each number in the array

**exponential multiplication**

array3=array1\*\*3#Raise each number to the 3rd power

**Scalar subtraction**

array4=array1-array1 #each number gets subtracted by itself

**Reciprocal**

print(1/array1) #each number has a reciprocal of 1

#### access members of array using array index

arr=np.arange(0,12)# give the range between 0-11

arr[0:4]=20 #assigns 20 to values 0-4

arr2d[:2,1:]=15 # assigns 15 to values

##### index in two dimension array

arr2d=np.array([[1,2,3],[4,5,6],[7,8,9]])

print(arr2d[0])#prints out 1st row

print(arr2d[1])#prints out 2nd row

print(arr2d[2])#prints out 3rd row

##### accessing each column

print(arr2d[0][2])

##### array slicing

slice1=arr2d[0:2,0:2]

##### using loops to index

for i in range(arr\_len):

arr2d[i]=i

When you would like to reuse your array without having the recent changes reflect in it. Use

arr2=arr.copy() #command to allocate extra memory while making changes

#### saving Numpy arrays

The first step have an array

single\_array=np.arange(50)

Then use the .save function then add the desired array name'saved\_single\_array' then include the array you want to save single\_array

np.save('saved\_single\_array',single\_array)#saving a single array

To read the array saved. use .load and add .npy to desired array name 'saved\_single\_array.npy'

load\_array=np.load('saved\_single\_array.npy')#reading a single array

#### saving multiple arrays in computer memory

Then use the .savez function then add the desired array name'saved\_multiple\_array' then assignx and y to the arrays you want to save respectively

np.savez('saved\_multiple\_array',x=single\_array,y=array1)

To read the array saved. use .load and add .npz to desired array name'saved\_multiple\_array.npz' load

Multiplearray=np.load(''savedmultiplearray.npz')

To print out each array separately, use [x] or [y]

print(load\_multiple\_array['y'])

### STATISTICAL AND MATHEMATICAL PROCESSING OF ARRAYS

axes\_values=np.arange(-100,100,10)

**using meshgrid to plot graphs**

dx,dy=np.meshgrid(axes\_values,axes\_values)

function=2\*(dx)+3\*(dy)

**plotting with matplotlib**

plt.imshow(function)# show graph for fuction

plt.title('function of plot 2\*(dx)+3\*(dy)') # title of graph function

plt.colorbar()#color key

plt.savefig('myfig.png')# save your image

### USING WHERE

To check whether a condition is true, one can use .where

y=np.array([10,15,20,25])

z3=np.where(y>0,0,1)

# PANDAS

To get started you need to import pandas as pd then from pandas import series,dataframe

#### series

object=Series([5,30,25,20])

series in pandas is one-dimensional array that can hold different data types.Pandas Series is equivalent to a column.

**To convert numpy array to a series**

data\_array=np.array(['a','b','c'])

s=Series(data\_array)

**Reading data from different file types**

revenue\_df=pd.read\_excel("revenue.xlsx") #reading excel files

revenue\_df=pd.read\_csv("revenue.csv") #reading csv files

revenue\_df=pd.read\_clipboard() #reading clipboard copied content

**Reading column names**

print(revenue\_df.columns) #reading column names

**Accessing specific column using column name**

print(revenue\_df['Rank'])

**Accessing specific column using column index**

print(revenue\_df[[0:2]) #returns 3 columns

**Re-indexing in pandas**

cars=Series(['Audi','Vitz','Corolla'], index=[0,4,8])

ranger=range(13)

cars=cars.reindex(ranger, method='ffill') #using the ffill function

**Using randnto create a dataframe**

df=DataFrame(np.random.randn(16).reshape(2,2),index=['a','b'],columns=['c1','c2'])

**re-indexing rows**

df\_1=df.reindex(['a','b','c'])

**re-indexing columns**

df\_2=df\_1.reindex(columns=['c1','c2','c3'])

**using loc(loc for label based indexing) to reindex**

df\_3=df\_2.loc[['a','b','c'],['c1','c2','c3']]

**Dropping entries**

`cars=Series(['Audi','merc','xtrail'], index=['a','b','c'])

cars=cars.drop('a')

**dropping dataframes rows/columns**

car\_df=DataFrame(np.arange(9).reshape(3,3),index=['Audi','merc'],columns=['revenue','profit'])

car\_df=car\_df.drop('merc',axis=0)#dropping row

car\_df=car\_df.drop('profit',axis=1)#dropping columns