**ASSIGNMENT 1: {Olasumbo Banjoko}**

**Getting started with Numpy**

|  |
| --- |
| Usage: linear, list =[,], 2dlist1 = [,] , 2dlist2 =[,], np.arange(start,end, dist)  Numpy works with list can be declared using np.array() or np.arange(), a list is created externally or internally but is embedded in arrays |
| Populating uses np.zero(),.ones([rows,columns]),.eye(),.empty(),**print()-python3** |

Note: np.arrange is used to populate and put values in arrays, likewise, np.zero() populates linearly with 0, np.ones(,) populates with 1 in 2dimension, np.eye populate diagonally with1s, np.empty() is similar to np.zero().

**Scalar operation on Numpy**

The library for numpy that uses a multiplication, exponential, subtraction, and reciprocal to perform a scalar operation on arrays.

|  |
| --- |
| Usage: create 2 arrays then multiply array1 \* array2 = assign, similarly,arr1\*\*arr2 |
| For exponential, array1-array2 (subtract), reciprocal = 1/array1; |

**Premium Arrays**

|  |
| --- |
| Create 2 by 2 dimensional array using np.arange, from function created use these:  Reference:docs.scipy.org on how to use arrays for math operations. Here operations can be done in 2d. |
| B=np.sqrt(),print(“B is”),print(B), np.exp(A), np.add(), np.maximum() |

**Saving and Loading Arrays**

|  |
| --- |
| Usage: np.save(‘titlefile’) = a single file called ‘titlefile’,np.savez(‘savmult.npz’, array1,array2) ,for txtfile saving np.save(‘txtfile.txt’,array1,delimiter=”,”)  We create a numpy array, print it. This operation is used to store very large data (save) to load when we want and delete when you do not need it. np.save() is used for 1 dimension as np.savez() used for 2dim. |
| np.load(‘titlefile.npy’)-1file,np.load(‘savmult.npz’),np.load(‘txtfile.txt’,delim=”,”) |

Note: to print multidimensional saved file use print(nameofarray[‘array1’] )

**Statistical Processing and Array Sketching**

Using matplotlib.pyplot as plt, we used the np.meshgrid() for grouping points seperated by dx and dy, like (-100,100,10) gives 20 elements lists.

|  |
| --- |
| defined as f1= np.cos(dx)+np.cos(dy), print(f1), can be plotted using plt.inshow() |
| To save figure plt.savefig(‘name.png’),plt.colorbar(),plt.title(f1ie function1) |

**Boolean's Operations on Numpy**

We create 2 equal length array, then create a condition array with element (boolean’s) ie True, False, using loops and np.where (we check the conditions of both arrays created using conditioned array), array.any()=or,array.all()=and, np.in1d(), array.sort(), np.unique(),x.mean(),x.sum(),x.std()

|  |
| --- |
| Usage: A, B=np.array(‘’), cond.=np.array(True,False,..),print A if true, else B = z[A if cond. B for a,cond,B in zip(x,y,condition)], np.where(cond.x,y) ie where condition is true accept and where false accept . np.where(x>0,0,1) = where x(array) > 0 print 0 otherwise print 1. x.mean(), x.std(),x.sum()=adds row wise |
| x.sum(0) adds column wise, npld(‘solid’,’gas’,’plasma’,A), array.sort(A) |

**Introduction to Series and Data-Frame using Pandas**

In the panda’s library we explore the use of panda and numpy. The list is embedded in array and array in series. Hence,list<array<Series. We can add two series, likes series+index allows us to rename the ordinal index. Array created used with array.to\_dict() matches string name in one column to values in another column. Checking for null, delete, access rows, getting head and tail of df, renaming array. Note: Dataframe is a 2d matrix of rows and columns heads.

|  |
| --- |
| Usage: series created= Series([5,…]), using numpy array to Series, s=Series(series created,index=[100/’index1’,200/’index2,..], print(s).array.to\_dict().pd.isnull(Series). To check index print(s[‘index1’] and so o |
| Using <https://en.wikipedia.org/wiki/List_of_largest_companies_by_revenue>, use pd.read\_clipboard() to copy first 6 table of the above link,deldf[‘nameofcolumn’], print(df) accessing rows with series.ix/iloc[0], df.head[0]= 1st element, df.tail[2], array.name(“ “) |

**Indexing and Reindexing in Pandas**

Indexing allows us to slice and pull out an index from negative indexing. However, indexing does not allow us to change the value of an index. Reindexing allows us to add elements to an index created from series or arrays. We use with fill\_value with the reindex dataframe to give new element values instead. Method =’ffill’ is used with range argument from an index created to replicate ‘strings’ from (0,4,8)

|  |
| --- |
| Usage: index1=index[‘a’,’b’],in1=index1.index, slicing print(index1[2:] = prints(first 0,1 rows of index) similarly negative index applies this as index[-2:]. |
| series2 = series2.reindex(['e', 'f', 'g', 'h', 'i', 'j', 'k'], fill\_value=10) # adds k = 10 new values added, using ffill then cars = Series(['Audi', 'Merc', 'BMW'], index=[0, 4, 8]),ranger = range(13),cars.reindex(ranger, method="ffill") # values from 0 to 4 get audi, values from 4 to 8 merci, 8 to 12 gets bmw. To reindex columns create df, then df.reindex(columns =[‘c1’..],rows=df.iloc[[‘a’][‘c’..]] |

**Dropping Entries from Data-types and Handling Null Data in Pandas**

To drop values in Series and df(dataframes) we use .dropna(‘col’/’row’), for df we include .dropna(‘bmwie column name’ axis=1, 0). inspecifying df and series index=rows, columns = column. In dropna(axis=1,thresh=3), delete nan in columns, threshold drop row if threshold exceed 3. fillna(0) populates na with 0 when fillna is used with {dictionaries} it replaces row 0, 1,2 with values.

|  |
| --- |
| Series.drop(‘a’) then print(df.dropna())=rows get deleted. Given df1 = DataFrame([[1, 2, 3, 4], [3, np.nan, np.nan, 6], [np.nan, np.nan, np.nan, np.nan]]),  print(df1.dropna(how='all')) = drop the rows when ALL elements are null.  print(df1.dropna(axis=1))= to delete nan in columns. print(df2.dropna(thresh=3)),  x.sum(0) sums columns when (1) sums rows. |
| print(df2.fillna(0)) = fillna replaces not available values with values of 0. print(df2.fillna({0: 0, 1: 50, 2: 100, 3: 200})) = for column fill na value to be 50, column 0 na value is filled with 0. {} means pass dictionary to values. cars = Series(['BMW', 'Merc', 'Audi'], index=['a', 'b', 'c']),cars = cars.drop('a') |

**Selecting , Modifying Entries in Pandas and Co-ordinate Data Frame.**

|  |
| --- |
| series1 = Series([100, 200, 300], index=['A', 'B', 'C']) then print(series1[['A', 'B']]) to access two element. print(series1[series1 > 200]) gives condition T/F output. print(series1[0:2]).  To Access series we apply slice. Also, double braces [[]] is used to access the rows. Furthermore, we use function .ix to access rows of df.  To Align to series that are not equalwe sum. similarly we do so with dataframes. Unavailable columns are not added hence it is NaN. |
| To align series, we add ser\_1+ser\_2. For dataframes, df1 =df(np.arange(4). reshape(2, 2), columns=['a', 'b'], index=['cars', 'bike']) to create df2 do the same. Then add print(df1+df2), we can add using df1.add(df2,fill\_values = 0) adds values by replacing nan with 0. we can subract using df1(dataframe-serc (series)). |

**Sorting, Ranking from Series, Statistics and Graph**

Using numpy and pandas, we sort according to index, and actual values and also ranking series for sorting.

To apply statistics, we sum dataframe and series along each column, rows. Using min. and max. function, using .idmax() and .cumsum(). We plt using plot, legend,savefig, and show. From series, we can also get unique values. Describe gives us summary statistics like mean, median of data sets.

|  |
| --- |
| ser1 = Series([500, 1000, 1500], index=['a', 'c', 'b']). print(ser1.sort\_index()) = sorts according to index. print(ser1.sort\_values()) = this sorts according to actual values. print(ser1.rank()) |
| array1 = np.array([[10, np.nan, 20], [30, 40, np.nan]]), df1 = DataFrame(array1, index=[1, 2], columns=list('ABC')). print(df1.sum()) = sum() operation along each columns, print(df1.sum(axis=1)) = sum along indexes. print(df1.describe()) = gives summary statistics. plt.plot(df1),plt.legend(df1.columns, loc=”upper right”), plt.savefig(‘pic.png’), plt.show() = plots dataframes. print(ser1.unique()) = to get unique values. |