Numpy

Introduction to Numpy

- Numpy it stands for Numerical Python.
- Fundamental python package for scientific computation/programming
- It has tools and technique to solve mathematical problems.
- It is written in C and Python
- Operations which can be done on
    1. Arrays
    2. Linear Algebra
    3. Random Number Generation
    4. Broadcasting

List vs Arrays

➢Similarities

1. Storing Data
2. Can be indexed
3. Mutable
4. Slicing Operation

➢Differences

1. Different data types can be stored in list but in list same data type is used.
2. Operations can be performed directly on array but its not possible on list.
3. For arrays Numpy needs to be imported or installed whereas lists are inbuilt.

➢Advantages of Arrays over list

1. Faster to execute.
2. Needs less memory.
3. Convenient for performing Mathematical operations.

Array Representation

Arrays can be generated by using

1. Array module
2. Numpy library

But it is better to create a Numpy library rather than array module

Numpy array in python is also known as ndarray which means n-dimensional array

Numpy arrays

Numpy arrays can be created using various operations or functions

1. Array()
2. Arange()
3. Zeros()
4. Ones()
5. Linspace()
6. Eye()
7. Random()

Various ways to create array

- Array()

Use to create an array from lists or tuples.

- Arange()

Use to create array of evenly spaced values.

- Zeros()

Creates array filled with zeros.

- Ones

Creates array filled with ones.

- Empty

Use to create array

- Linspace

Creates array filled evenly spaced values

Various ways to create array

Eye()

Returns array filled with zeros except in the k-th diagonal , whose values are equal to 1.

Computation on NumPy Arrays

- NumPy Arrays
  - A grid that contains values of the same type.
  - NumPy Arrays come in two forms: **Vectors** and **Matrices**
  - Vectors are strictly one-dimensional arrays, while Matrices are multidimensional.
- Creating Numpy Arrays from Python lists
  - Create 1-D Array
  - Create 2-D Array
- Creating Numpy Arrays using arange() built-in function

Computation on NumPy Arrays

- Creating Numpy Array of “0’s” using zeros()
- Creating Numpy Array of “1’s” using ones()
- Creating Numpy Array using linspace()
- Creating an identity matrix in NumPy using the eye()
- Creating an array of random numbers using rand(), randn() , randint()

Computation on NumPy Arrays

- Converting a 1-D array to a 2-D array using reshape()
- Locating maximum and minimum values in an array using max() and min()
- Locating the index of the maximum and minimum values in an array using argmax() and argmin()

Computation on NumPy Arrays

- Indexing / Selecting elements or groups of elements from a NumPy array
- Conditional and Logical selection using <,>,==,& , |
- Compare two arrays using allclose()
- Broadcasting
- Nearest value using flat()
- Most frequent value using bincount()

Computation on NumPy Arrays

- Performing arithmetic operations using +, -, * , /
- Performing universal functions
  - np.sqrt(arr) #Returns the square root of each element
  - np.exp(arr) #Returns the exponentials of each element
  - np.sin(arr) #Returns the sin of each element
  - np.cos(arr) #Returns the cosine of each element
  - np.log(arr) #Returns the logarithm of each element
  - np.sum(arr) #Returns the sum total of elements in the array
  - np.std(arr) #Returns the standard deviation of in the array
  - arr.sum() #Returns the sum of all the values in mat
  - arr.sum(axis=0) #Returns the sum of all the columns in mat
  - arr.sum(axis=1) #Returns the sum of all the rows in mat

Aggregations

Function Name	NaN-Safe Version	Description
np.sum	np.nansum	Compute sum of elements
np.prod	np.nanprod	Compute product of elements
np.mean	np.nanmean	Compute mean of elements
np.std	np.nanstd	Compute standard deviation
np.var	np.nanvar	Compute variance
np.min	np.nanmin	Find minimum value
np.max	np.nanmax	Find maximum value
np.argmin	np.nanargmin	Find index of minimum value
np.argmax	np.nanargmax	Find index of maximum value
np.median	np.nanmedian	Compute median of elements
np.percentile	np.nanpercentile	Compute rank-based statistics of elements
np.any	N/A	Evaluate whether any elements are true
np.all	N/A	Evaluate whether all elements are true

Fancy Indexing

- Fancy indexing is conceptually simple: it means passing an array of indices to access multiple array elements at once
- Combined Indexing=> Fancy & Other indexing
- Modifying values with Fancy indexing

Sorting

- Using np.sort and np.argsort
- Sorting along rows or columns
- Partial sorts using partition()

NumPy's Structured Arrays

- A structured array in NumPy is an array of records.
- Each record can contain one or more items which can be of different types.
- A structured array can be created by using the dtype object to define the fields of a record
  - Passed as a list of tuples containing name and format of field
  - Passed as a dictionary of lists containing the key value pair for names and formats

TASK

1. Find the rows for which the first value (value in position 0) in the row is greater than 2.
   example_list=[[1, 2, 4, 3],[3, 5, 4 ,2],[4, 6, 1, 7],[5, 3, 3, 1],[0, 4, 3, 2], [1 ,9 ,8, 3]]
2. Find the rows where the element in position 2 in the row equals 4
3. Create a 5x5 array with random values and find the minimum and maximum values.
4. Check whether the two arrays are equal: ar_list1=[3,4,5] ; ar_list2=[3,5,4]
5. Create a random 10x4 array and extract the first five rows of the array and store them into a variable.
6. Create a random vector of size 10 and sort it in ascending and descending.
7. Find the nearest value from a given value in an array:
   example_list = [[12,3,4],[6,7,4],[10,8,11]] & value is 7.8
8. Create a random vector of size 15 and replace the maximum value by -1