🔢 Numpy

An amazing module in Python for playing with Matrix and 3D data representation.

On this page

Numpy is a Python Module used to create and modify matrices and 3D Arrays.
Below are some of the advanced operations we can perform using Numpy:-
Linear Algebraic Operations using Numpy-
A quick look at 3D arrays created using Numpy

Numpy is a Python Module used to create and modify matrices and 3D Arrays.

Basic Operations that can be done :-

- First of all we will import Numpy package into our coding environment. `import numpy as np`
- We can create an Array using `.array()` method, Like for example., `arr=np.array(6)` This code block will create an array which can contain 6 elements. Note:- We have inserted only one element in our Array. Hence, our newly created array has only one value, for now!

```
print("array is: ",arr)
print("a dim: ",arr.ndim)
print("a shape: ",arr.shape)
print("a size: ",arr.size)
print("a dtype: ",arr.dtype)
```

- Above are some of the code blocks, by using these we can print out various important informations like Size, Dimension, Shape, Data Type,etc. Note:- We have to use the name of the variable we have used to create our array.

```
vc=np.array([10,15,22])
```

- Above is an example of 1 Dimensional array created using Numpy, This arrya has only 1 dimension and 3 elements inside it. We can apply same operations to print information about the array similarly as we did above 👆.

```
m=np.array([[1,2],[3,4]])
```

- ☝This is a 2 Dimensional array, it has 2 Dimensions similar to common Graph, Similar operations can be done this as well for taking information.

```
te=np.array([[[1,2],[5,7]],[[10,11],[11,15]]])
```

- 👆This right here is a 3-D array, which has a lot of usage in Real-World Applications. Operations on this type of 3D array are much difficult to interprete as we have only learned two axed X and Y uptil now. All the information can be derived using above listed operations.

```
print("np.zeros(2,2) \n",np.zeros((2,2)))
print("np.ones(2,2) \n",np.ones((2,2)))
print("np.eye(2) \n",np.eye((2)))
print("np.random.random(2,2) \n",np.random.random((2,2)))
```

- Above 👆 are some of the built in functions in Numpy which can be used to quickly create some arrays. 1. `zeros()` :-
Creates an array with all elements being zero('0'), takes two or one arguments, being the number of Zeros to print and other argument being the dimesions.
2. `ones()` :- Creates a matrix containing '1' as element, arguments are given with the number of dimension and the number of '1' to be printed. 3. `eye()` :-
Creates an Identity Matrix, takes one argument that is the size or number of elements.
4. `random.random()` :- Creates an array with random elements, just takes two arguments the size of the array and number of elements.

```
ni=np.array([1,2,3])
print(ni[0])
ni[1]=10
```

- Also, we can print a specific element from an array using normal indexing method. And we can assign a specific value on a respective index of our own choice as shown above ☝.

Below are some of the advanced operations we can perform using Numpy:-

- Stacking Operations using `hstack()` and `vstack()` :- These two built in functions have the responsibility to stack / combine two arrays into one. `hstack()` is used for horizontal stacking while `vstack()` is used for vertical stacking.

```
a=np.array([[1,2],
    [4,6]])
b=np.array([[4,5],
    [7,9]])
print("\n Vertical Stacking: \n",np.vstack((a,b)))
print("\n Horizontal Stacking: \n",np.hstack((a,b)))
```

- Above 👆 is the example of how we can implement `hsatck()` and `vstack()` in our code. Note:- The dimensions of the two arrays should be the same while stacking.
- We can also use `row_stack()` and `column_stack()` below 👇is an example of how to do it

```
c=np.array([10,21])
print("\n Column Stacking: \n",np.column_stack((a,c)))
print("\n Row Stacking: \n",np.row_stack((b,c)))
```

- `concatenate()` is also an option for combining two arrays together, that can be done using

```
print("\n Concatenation : \n",np.concatenate((arr1,arr2),1))
```

- An array can be splitted vertically or horizontally using `hsplit()` or `vsplit()` , 👇 below is the example of implementation-

```
x=np.array([[1, 3, 5, 7, 9, 11],
            [2, 4, 6, 8, 10, 12]])
print("\n Splitting Horizontally into two part: \n", np.hsplit(x,2))
print("\n Splitting Vertically into two part: \n", np.vsplit(x,2))
```

This method will split the arrays into two parts equally.

- Note:- The array dimension should be of equal elements(even count) for perfect splitting.

Here are some of the datetime oprations available in Numpy:-

```
today=np.datetime64('2022-12-08')
print("\n Date is: ",today)
print("\n Year is: ",np.datetime64(today,'Y'))
print("\n Month is: ",np.datetime64(today,'M'))
print("\n Day is: ",np.datetime64(today,'D'))

# Creating array of dates in month
dates=np.arange('2022-01','2022-02',dtype='datetime64[D]')
print("\n Dates: ",dates)
print("\n Today is December: ",today in dates)
```

- These are used to print the date,month,year,time of the respective date called inside the variable, in this case named `'today'`. The syntax for calling Year, Month and Day are in single quotes and capital letters only!
- We can also set our own custom range of dates as `var_name=np.arange('2022-01','2022-02',dtype='datetime64[D]')`
- Some operations on datetime are possible like the "Number of weeks in a given time range" OR the "Number of days in a given time range".

Linear Algebraic Operations using Numpy-

```
A=np.array([[1,2,3],
        [4,5,6],
        [7,8,7]])
print("\n Matrix A: \n",A)
print("\n Rank of A: \n",np.linalg.matrix_rank(A))
print("\n Trace of A: \n",np.trace(A)) # Trace is the sum of Diagonal elements in the
print("\n Determinant of A: \n",np.linalg.det(A))
print("\n Inverse of A: \n",np.linalg.inv(A))
print("\n Matrix A raised to power of 2: \n",np.linalg.matrix_power(A,2))
```

- Above 👆are some of the Algebraic operations that can be done is Numpy.

```
# Solving Linear Equations using Numpy
# x + 2*y = 8
# 3*x + 4*y = 18, are two equations, we just need to write their coefficients in each
# separate array for a variable with values

m=np.array([[1,2],
        [3,4]])
n=np.array([8,18])
print("\n Solution of Linear Equations is: \n",np.linalg.solve(m,n))
```

- Numpy also gives us the power to solve Linear Equations without writing the entire equations with variables.

A quick look at 3D arrays created using Numpy

```
a=np.array([[[1,2],[3,4]],
            [[5,6],[7,8]]])
print(a)
print("a ndim: ", a.ndim)
print("a shape: ", a.shape)
print("a size: ", a.size)
print("a dtype: ", a.dtype)
```

- Above 👆 is an example of how to create 3D array / matrix.
- ```
  p=np.array([[1,2,3,4,5,6]])
  t=np.reshape(p,(2,3))
  print("\n",t)
  ```
- We can also reshape our matrix to look in a particular way without even changing its Data. `reshape()` takes 3 arguments., the array_name the newshape and the order respectively.

← Previous
Pandas

Last modified 23h ago
  ```