Numpy Exercises

Shanesh Shukla

In [1]:
```python
import numpy as np
```

In [2]:
```python
#creating 1D array with values 0 to 9
n = np.arange(10)
print(n)
```
```
[0 1 2 3 4 5 6 7 8 9]
```

In [9]:
```python
#Convert a 1D array to a 2D array with 2 rows
n = np.arange(10).reshape(2,-1)
print(n)
```
```
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

In [16]:
```python
#Multiply a 5x3 matrix by a 3x2 matrix.
n = np.arange(15).reshape(5,3)
n1 = np.arange(6).reshape(3,2)
print(np.dot(n,n1))
```
```
[[ 10  13]
 [ 28  40]
 [ 46  67]
 [ 64  94]
 [ 82 121]]
```

In [20]:
```python
#Extract all odd numbers from an array of 1-10.
n = np.arange(11)
n2 =(n[n%2 != 0])
print(n2)
```
```
[1 3 5 7 9]
```

In [22]:
```python
#Replace all odd numbers in an array of 1-10 with -1.
n = np.arange(1,11)
n[n % 2 != 0] = -1
print(n)
```
```
[-1  2 -1  4 -1  6 -1  8 -1 10]
```

In [26]:
```python
#Convert a 1D array to a boolean array where all positive values become True.
n = np.array([1,2,-1,2,-2,3,4,5])
n= [n>0]
print(n)
```
```
[array([ True,  True, False,  True, False,  True,  True,  True])]
```

In [33]:
```python
#Replace all even numbers in a 1D array with their negative.
n = np.arange(1,11)
n[n % 2 == 0] *= -1
print(n)
```
```
[  1  -2   3  -4   5  -6   7  -8   9 -10]
```

In [36]:
```python
#Create a random 3x3 matrix and normalize it.
n = np.arange(9).reshape(3,3)
nor_max = (n-np.mean(n))/np.std(n) #need to remember this formula
print(nor_max)
```
```
[[-1.54919334 -1.161895   -0.77459667]
 [-0.38729833  0.          0.38729833]
 [ 0.77459667  1.161895    1.54919334]]
```

In [39]:
```python
#Calculate the sum of the diagonal elements of a 3x3 matrix.
n = np.arange(9).reshape(3,3)
print(n)

print(np.trace(n))
```
```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
12
```

In [43]:
```python
#Find the indices of non-zero elements from [1,2,0,0,4,0].
n = np.array([1,2,0,0,4,0])
print(np.nonzero(n))
#np.nonzero() function will return indices of non zero elements in array
```
```
(array([0, 1, 4]),)
```

In [46]:
```python
#Reverse a 1D array (first element becomes the last).
n = np.arange(1,11)
print(np.flip(n))
#np.flip() function used to reverse the element in array
print(n[::-1]) # this is anpther way to do it
```
```
[10  9  8  7  6  5  4  3  2  1]
[10  9  8  7  6  5  4  3  2  1]
```

In [50]:
```python
#Create a 3x3 identity matrix.
n = np.eye(3,3)
print(n)
#np.eye function will create an identity matrix for given shape
```
```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [51]:
```python
#Reshape a 1D array to a 2D array with 5 rows and 2 columns.
n = np.arange(10)
n = n.reshape(5,2)
print(n)
```
```
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
```

In [54]:
```python
#Stack two arrays vertically.
Get the common items between two arrays..reshape(5,2)
n1 = np.arange(11,21).reshape(5,2)
print(np.vstack((n,n1)))
#np.vstack() function combine two arrays vertically
#always give parameters of two array in tuple
```
```
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [11 12]
 [13 14]
 [15 16]
 [17 18]
 [19 20]]
```

In [55]:
```python
#Get the common items between two arrays.
n = np.arange(10)
n2 = np.arange(5,9)
print(np.intersect1d(n,n2))
#np.intersect() function return common elements from two array
```
```
[5 6 7 8]
```

In [57]:
```python
n = np.random.randint(0,4,25)
print(n.reshape(5,5))
```
```
[[0 1 3 1 1]
 [3 2 1 3 0]
 [3 2 0 1 3]
 [2 0 0 3 1]
 [3 0 1 1 3]]
```

In [61]:
```python
#Create a 5x5 matrix with row values ranging from 0 to 4.
n = np.zeros((5,5))
n = n+ np.arange(5)
print(n)
```
```
[[0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]]
```

In [64]:
```python
#Find the index of the maximum value in a 1D array.
n = np.array([1,5,7,8,4,50,5,7,9])
print(np.argmax(n))
#np.argmax() return the index of maximum value in array
```
```
5
```

In [67]:
```python
#Calculate the dot product of two arrays.
n = np.arange(10)
n2 = np.arange(11,21)
print(np.dot(n,n2))
```
```
780
```

In [68]:
```python
#Count the number of elements in an array within a specific range
arr = np.array([2, 5, 8, 10, 12, 15])
print(np.sum((arr>=5) & (arr<=12)))
```
```
4
```

In [70]:
```python
#Find the mean of each row in a 2D array.
n= np.arange(16).reshape(4,4)
n2 = np.mean(n,axis=1)
print(n2)
#np.mean() function retrun the mean of given values here values are rows
```
```
[ 1.5  5.5  9.5 13.5]
```

In [74]:
```python
#Create a random 4x4 matrix and extract the diagonal elements.
n = np.random.random((4,4))
diagonal_mat = np.diag(n)
print(diagonal_mat)
#np.diag() function return the diagonal elements of array
```
```
[0.30094283 0.88473439 0.86078038 0.32345374]
```

In [4]:
```python
#Count the number of occurrences of a specific value in an array.
n = np.array([1,2,3,1,3,4,1,4,5,6,7,6])
print(np.count_nonzero(n==5))
#np.count_nonzero function
```
```
1
```

In [6]:
```python
#Replace all values in a 1D array with the mean of the array.
n = np.arange(10)
n[:] = np.mean(n)
print(n)
```
```
[4 4 4 4 4 4 4 4 4 4]
```

In [8]:
```python
#Find the indices of the maximum and minimum values in a 1D array.
n = np.random.randint(10,50,20)
print(n)
print(np.argmax(n))
print(np.argmin(n))
```
```
[32 29 33 29 20 21 18 15 11 37 14 26 26 34 22 19 48 45 29 13]
16
8
```

In [14]:
```python
#Create a 2D array with 1 on the border and 0 inside.
n = np.ones((5,5))
print(n)
n[1:-1,1:-1] = 0
print(n)
```
```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

In [21]:
```python
#Find the unique values and their counts in a 1D array.
n = np.array([1,2,3,4,5,6,7,3,4,2,9,22,4])
n2 = np.unique(n,return_counts=True)
print(n2)
```
```
(array([ 1,  2,  3,  4,  5,  6,  7,  9, 22]), array([1, 2, 2, 3, 1, 1, 1, 1, 1]))
```

In [27]:
```python
#create a 3*3 matrix with values ranging from 0 to 8
n = np.arange(9).reshape(3,3)
print(n)
```
```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

In [36]:
```python
#Calculate the exponential of all elements in a 1D array.
n = np.arange(10)
print(np.exp(np.round(n)))
```
```
[1.00000000e+00 2.71828183e+00 7.38905610e+00 2.00855369e+01
 5.45981500e+01 1.48413159e+02 4.03428793e+02 1.09663316e+03
 2.98095799e+03 8.10308393e+03]
```

In [39]:
```python
#Swap two rows in a 2D array.
n = np.arange(12).reshape(4,3)
print(n)
n[[0,1]] =n[[1,0]]
print(n)
```
```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
[[ 3  4  5]
 [ 0  1  2]
 [ 6  7  8]
 [ 9 10 11]]
```

In [44]:
```python
#Create a random 3x3 matrix and replace all values greater than 0.5 with 1 and all others with 0.
n = np.random.random((3,3))
print(n)
n[n>0.5]=1
n[n<=0.5]=0
print(n)
```
```
[[0.03576149 0.28347812 0.72591172]
 [0.87865771 0.29048102 0.78539731]
 [0.86054702 0.68756731 0.00970668]]
[[0. 0. 1.]
 [1. 0. 1.]
 [1. 1. 0.]]
```

In [46]:
```python
#Calculate the mean of each column in a 2D array.
n =np.random.random((2,2))
print(np.mean(n,axis=0))
```
```
[0.74094834 0.9153858 ]
```

In [51]:
```python
#Concatenate two 1D arrays.
n = np.arange(10)
n1= np.arange(10)
print(np.concatenate((n,n1)))
```
```
[0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9]
```

In [54]:
```python
#Create a 2D array with random values and sort each row.
n = np.array([[1,3,5,7,3,4,6,9,24,56,24,3]]).reshape(4,3)
print(n)
print(np.sort(n,axis=1))
```
```
[[ 1  3  5]
 [ 7  3  4]
 [ 6  9 24]
 [56 24  3]]
[[ 1  3  5]
 [ 3  4  7]
 [ 6  9 24]
 [ 3 24 56]]
```

In [55]:
```python
#compute mean squared error between two arrays
a1 = np.array([1,2,3,4])
a2= np.array([2,3,4,5])
mse = np.mean((a1-a2)**2)
print(mse)
```
```
1.0
```

In [56]:
```python
#Replace all negative values in an array with 0.
n = np.array([1,2,3,-1,-4,5,-3,5,6,7,-8])
n[n<0]=0
print(n)
```
```
[1 2 3 0 0 5 0 5 6 7 0]
```

In [59]:
```python
#Find the 5th and 95th percentiles of an array.
n = np.random.randint(10,50,20)
print(n)
print(np.percentile(n,95))
print(np.percentile(n,5))
```
```
[26 29 14 15 13 28 12 21 10 20 31 45 39 23 27 22 28 28 13 39]
39.300000000000004
11.9
```

In [60]:
```python
#Create a random 2x2 matrix and compute its determinant
n = np.random.random((2,2))
print(np.linalg.det(n))
#np.linalg.det() calculate the determinant of array
```
```
0.17152095219407112
```

In [65]:
```python
#Count the number of elements in an array that are greater than the mean.
n = np.arange(20)
mean = np.mean(n)
print(mean)
print(np.count_nonzero(n[n>mean]))
```
```
9.5
10
```

In [68]:
```python
#Calculate the square root of each element in a 1D array.
n = np.arange(10)
print(np.round(np.sqrt(n)))
```
```
[0. 1. 1. 2. 2. 2. 2. 3. 3. 3.]
```

In [70]:
```python
#Create a 3x3 matrix and compute the matrix square root.
n = np.arange(9).reshape(3,3)
print(n)
print(np.linalg.matrix_power(n,2))
```
```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
[[ 15  18  21]
 [ 42  54  66]
 [ 69  90 111]]
```

In [80]:
```python
#Convert the data type of an array to float.
n = np.arange(20)
print(n)
print(n.dtype)
n2 =n.astype("f")
print(n2.dtype)
#np.astype() used to convert data type of an array
```
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
int64
float32
```

In [82]:
```python
#Calculate the element-wise absolute values of an array.
n = np.array([1,2,3-4,-2,5,-6,-9])
print(np.abs(n))
```
```
[1 2 1 2 5 6 9]
```

In [91]:
```python
#Find the indices where elements of two arrays match.
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([3, 2, 8, 4, 5])
print(np.where(arr1==arr2))
```
```
(array([1, 3, 4]),)
```

In [94]:
```python
#Calculate the cumulative sum of elements in a 1D array.
n = np.arange(10)
print(np.cumsum(n))
arr = np.array([1, 2, 3, 4, 5])
print(np.cumsum(arr))
```
```
[ 0  1  3  6 10 15 21 28 36 45]
[ 1  3  6 10 15]
```

In [98]:
```python
#Compute the inverse of a 2x2 matrix.
n = np.random.random((2,2))
print(n)
print(np.linalg.inv(n))
```
```
[[0.14433761 0.92028664]
 [0.94746499 0.15993139]]
[[-0.18840831  1.08415025]
 [ 1.1161679  -0.17003795]]
```

In [102…
```python
#Count the number of non-zero elements in a 2D array.
n = np.array([[1,2,0],[2,3,5],[0,0,5],[2,0,4]])
print(np.count_nonzero(n))
```
```
8
```

In [104…
```python
n = np.array([[1, np.nan, 3], [4, 5, np.nan], [7, 8, 9]])
n[np.isnan(n)] =0
print(n)
```
```
[[1. 0. 3.]
 [4. 5. 0.]
 [7. 8. 9.]]
```

In [106…
```python
#Find the correlation coefficient between two arrays
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([3, 4, 5, 6, 7])
cof = np.corrcoef(arr1,arr2)[0,1]
print(cof)
```
```
0.9999999999999999
```

In [107…
```python
#Create a 1D array and remove all duplicate values.
n = np.array([1,2,3,1,3,4,5,6,6,7,8,7])
print(np.unique(n))
```
```
[1 2 3 4 5 6 7 8]
```

In [109…
```python
#Compute the element-wise product of two arrays
n = np.arange(5)
n1 = np.arange(5)
print(np.multiply(n,n1))
print(np.dot(n,n1))
```
```
[ 0  1  4  9 16]
30
```

In [112…
```python
#Calculate the standard deviation of each column in a 2D array.
n = np.arange(12).reshape(4,3)
print(np.std(n,axis=0))
```
```
[3.35410197 3.35410197 3.35410197]
```

In [113…
```python
#Create a 2D array and set all values above a certain threshold to that threshold.
n = np.arange(12).reshape(4,3)
t = 4
n[n>t]=t
print(n)
```
```
[[0 1 2]
 [3 4 4]
 [4 4 4]
 [4 4 4]]
```

In [125…
```python
#Convert a 1D array of Fahrenheit temperatures to Celsius.
f_t = np.array([32, 68, 100, 212])
c_t = (f_t - 32)*5/9
print(c_t`)
```
```
[  0.          20.          37.77777778 100.        ]
```

In [126…
```python
#Compute the outer product of two arrays
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
outer_product = np.outer(arr1, arr2)
print(outer_product)
```
```
[[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
```

In [127…
```python
#Create a 1D array with 10 equidistant values between 0 and 1.
n =np.linspace(0,1,10)
print(n)
```
```
[0.         0.11111111 0.22222222 0.33333333 0.44444444 0.55555556
 0.66666667 0.77777778 0.88888889 1.        ]
```

In [135…
```python
#Compute the cross product of two 3D arrays.
n=np.array([[[1,2,3]]])
n2=np.array([[[4,5,6]]])
print(n.ndim)

print(np.cross(n,n2))
```
```
3
[[[-3  6 -3]]]
```

In [138…
```python
#Calculate the percentile along a specific axis of a 2D array.
n = np.arange(8).reshape(4,2)
print(n)
print(np.percentile(n,50,axis=0))
```
```
[[0 1]
 [2 3]
 [4 5]
 [6 7]]
[3. 4.]
```

In [139…
```python
#Create a 1D array and add a border of 0s around it
rr = np.array([1, 2, 3, 4])
arr_with_border = np.pad(arr, (1, 1), mode='constant', constant_values=0)
print(arr_with_border)
```
```
[0 1 2 3 4 5 0]
```

In [140…
```python
n = np.array([1,1,2,2,2,3,3,3,3])
print(np.histogram(n,bins=[1,2,3,4]))
```
```
(array([2, 3, 4]), array([1, 2, 3, 4]))
```

In [141…
```python
#Create a 2D array with random values and normalize each row.
n =np.random.random((4,3))
normalized = n/ np.linalg.norm(n,axis=1,keepdims=True)
print(normalized)
```
```
[[0.76564681 0.46998858 0.43919892]
 [0.13889458 0.77482311 0.61673125]
 [0.46248159 0.74466078 0.48123914]
 [0.67947615 0.21403641 0.70178385]]
```

In [146…
```python
#Create a random 2D array and sort it by the second column.
n =np.random.random((4,3))
print(n)
print(np.sort(n[:,1:]))
print(n)
```
```
[[0.11180702 0.9621821  0.60222725]
 [0.36556425 0.13141761 0.44605103]
 [0.45012001 0.16216043 0.02760459]
 [0.9037185  0.66942757 0.92086979]]
[[0.60222725 0.9621821 ]
 [0.13141761 0.44605103]
 [0.02760459 0.16216043]
 [0.66942757 0.92086979]]
[[0.11180702 0.9621821  0.60222725]
 [0.36556425 0.13141761 0.44605103]
 [0.45012001 0.16216043 0.02760459]
 [0.9037185  0.66942757 0.92086979]]
```

In [147…
```python
#Calculate the element-wise exponentiation of a 1D array.
n = np.arange(10)
print(np.exp(n))
```
```
[1.00000000e+00 2.71828183e+00 7.38905610e+00 2.00855369e+01
 5.45981500e+01 1.48413159e+02 4.03428793e+02 1.09663316e+03
 2.98095799e+03 8.10308393e+03]
```

In [152…
```python
#Compute the matrix multiplication of two 2D arrays.
n = np.arange(9).reshape(3,3)
n1 = np.arange(9,18).reshape(3,3)
print(n,n1)
print()
print(np.multiply(n,n1))
print()
n = np.arange(9).reshape(3,3)
n1 = np.arange(9,18).reshape(3,3)
print(n,n1)
print()
print(np.dot(n,n1))
```
```
[[0 1 2]
 [3 4 5]
 [6 7 8]] [[ 9 10 11]
 [12 13 14]
 [15 16 17]]

[[  0  10  22]
 [ 36  52  70]
 [ 90 112 136]]

[[0 1 2]
 [3 4 5]
 [6 7 8]] [[ 9 10 11]
 [12 13 14]
 [15 16 17]]

[[ 42  45  48]
 [150 162 174]
 [258 279 300]]
```

In [155…
```python
#Create a 1D array and set the values between 10 and 20 to 0.
n = np.arange(0,25)
n[(n>=10) & (n<=20)] = 0
print(n)
```
```
[ 0  1  2  3  4  5  6  7  8  9  0  0  0  0  0  0  0  0  0  0  0 21 22 23
 24]
```

In [156…
```python
#Compute the inverse hyperbolic sine of each element in a 1D array.
n = np.arange(10)
print(np.arcsinh(n))
```
```
[0.         0.88137359 1.44363548 1.81844646 2.09471255 2.31243834
 2.49177985 2.64412076 2.77647228 2.89344399]
```

In [158…
```python
#Compute the Kronecker product of two arrays.
n = np.array([1,2])
n1=np.array([3,4])
print(np.kron(n,n1))
```
```
[3 4 6 8]
```

In [160…
```python
#Count the number of occurrences of each unique value in a 1D array.
n = np.array([1,2,3,2,3,1,4,5,5,6,7,7])
uni,count = np.unique(n, return_counts=True)
print(uni,count)
```
```
[1 2 3 4 5 6 7] [2 2 2 1 2 1 2]
```

In [162…
```python
#Compute the cumulative product of elements along a given axis in a 2D array.
n = np.arange(12).reshape(4,3)
print(n)
print()
print(np.cumprod(n,axis=0))
```
```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]

[[  0   1   2]
 [  0   4  10]
 [  0  28  80]
 [  0 280 880]]
```

In [163…
```python
#Round elements of a 1D array to the nearest integer
arr = np.array([1.2, 2.7, 3.5, 4.9])
print(np.round(arr))
```
```
[1. 3. 4. 5.]
```

In [164…
```python
#Create a 1D array and append a new element to the end.
n = np.arange(10)
print(n)
print()
print(np.append(n,10))
```
```
[0 1 2 3 4 5 6 7 8 9]

[ 0  1  2  3  4  5  6  7  8  9 10]
```

In [165…
```python
#Calculate the element-wise absolute difference between two arrays.
arr1 = np.array([3, 7, 1, 10, 4])
arr2 = np.array([2, 5, 8, 1, 7])
print(np.abs(arr1-arr2))
```
```
[1 2 7 9 3]
```

In [167…
```python
#Create a 2D array with random values and replace the maximum value in each row with -1.
n = np.random.random((3,3))
```
Out[167… 2

In [176…
```python
#Create a 2D array with random values and replace the maximum value in each row with -1.
n = np.arange(12).reshape(4,3)
print(n)
max_idx = np.argmax(n,axis=1)
n[np.arange(n.shape[0]),max_idx]=-1
print(n)
# not able to understand what has happend here######
```
```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
[[ 0  1 -1]
 [ 3  4 -1]
 [ 6  7 -1]
 [ 9 10 -1]]
```

In [185…
```python
#Find the indices of the top N minimum values in a 1D array.
n =np.array([10,5,8,1,7])
print(np.argsort(n)[:2])
```
```
[3 1]
```

In [186…
```python
#Convert the elements of a 1D array to strings.
n = np.arange(5)
n2= n.astype(str)
print(n2.dtype)
```
```
<U21
```

In [187…
```python
#Compute the percentile rank of each element in a 1D array.
n = np.array([3,4,7,5,8,9,20,40,90,45,25,16,89])
print(np.percentile(n,100))
```
```
90.0
```

In [193…
```python
#Create a 1D array and shuffle its elements randomly.
n = np.array([2,3,4,5,6])
np.random.shuffle(n)
print(n)
```
```
[2 3 5 6 4]
```

In [197…
```python
#Check if all elements in a 1D array are non-zero.
n = np.array([2,3,5,5,6])
print(np.isnan(n))
#another easy way to do this
print(np.all(n != 0))
```
```
[False False False False False]
True
```

In [198…
```python
#Find the indices of the maximum value in each row of a 2D array.

n = np.arange(12).reshape(4,3)
print(np.argmax(n,axis=1))
```
```
[2 2 2 2]
```

In [202…
```python
#Create a 2D array and replace all nan values with the mean of the array.
n = np.array([[1, np.nan, 3], [4, 5, np.nan], [7, 8, 9]])
mean = np.nanmean(n)
n[np.isnan(n)]=mean
print(n)
```
```
[[1.         5.28571429 3.        ]
 [4.         5.         5.28571429]
 [7.         8.         9.        ]]
```

In [203…
```python
#Compute the sum of diagonal elements in a 2D array
n = np.random.random((3,3))
print(np.trace(n))
```
```
1.0768554280184155
```

In [205…
```python
#Check if any element in a 1D array is non-zero.
n = np.array([0,0,0,1,0])
print(np.any(n != 0))
```