Tips on Numpy

NumPy is a library for the Python programming language whose main objects are large and multi-dimensional arrays, which includes a large collection of high-level mathematical functions to operate on these arrays.

In [2]:

```
import numpy as np
```

DEFINING ARRAYS

In [54]:

```
A=np.array([[1,2,3],[4,5,6]],dtype=np.float64) #a list becomes an array #we will see som
e exemples for which the choice of the dtype may affect the result
B=np.array((1,2,3,4)) #a tuple becomes an array
C= np.arange(9)
print(A,'\n')
print(B,'\n')
print(C,'\n')
```

```
[[1. 2. 3.]
 [4. 5. 6.]]

[1 2 3 4]

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

In [56]:

```
print(A.size)
print(A.shape)
print(B.shape)
print(A.nbytes)
print(A.ndim)
print(A.dtype)

print(type(A.shape))
```

```
6
(2, 3)
(4,)
48
2
float64
<class 'tuple'>
```

In [12]:

```
M=np.array([A,A,A,A])
print(M.shape)
print(M ,'\n')

N=np.zeros((2,3,3))
print(N,'\n')
N[:,:,0]=A

print(N.shape)
print(N)
```

```
(4, 2, 3)
[[[1. 2. 3.]
  [4. 5. 6.]]

 [[1. 2. 3.]
```

```
  [4. 5. 6.]]

 [[1. 2. 3.]
  [4. 5. 6.]]

 [[1. 2. 3.]
  [4. 5. 6.]]]

[[[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]

 [[0. 0. 0.]
  [0. 0. 0.]
  [0. 0. 0.]]]

(2, 3, 3)
[[[1. 0. 0.]
  [2. 0. 0.]
  [3. 0. 0.]]

 [[4. 0. 0.]
  [5. 0. 0.]
  [6. 0. 0.]]]
```

In [13]:

```
print(np.diag(B))
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
```

In [14]:

```
print(np.zeros((3,3)))
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [15]:

```
print(np.ones((3,3)))
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

In [20]:

```
print(np.eye(3))
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [26]:

```
#Equidistant
D=np.linspace(0,10,num=10)
print(D)
```

```
[ 0.          1.11111111  2.22222222  3.33333333  4.44444444  5.55555556
  6.66666667  7.77777778  8.88888889 10.        ]
```

In [29]:

```
#From a callable function
f = lambda m,n: n + 10*m
```

```
E = np.fromfunction(f, (3,3), dtype=np.float64)
print(E)
```

```
[[ 0.  1.  2.]
 [10. 11. 12.]
 [20. 21. 22.]]
```

ARITMETIC OPERATION

Let's work with numpy arrays.

In [31]:

```
x = np.linspace(-4, 4, 3)
print(x,'\n')
print(np.outer(x, x),'\n')
```

```
[-4.  0.  4.]

[[ 16.  -0. -16.]
 [ -0.   0.   0.]
 [-16.   0.  16.]]
```

In [ ]:

```
help(np.outer)
```

In [32]:

```
x = np.ones((5))
y = np.linspace(1,5,5)
print(x.shape)
print(y.shape)
print(x)
print(y)
```

```
(5,)
(5,)
[1. 1. 1. 1. 1.]
[1. 2. 3. 4. 5.]
```

In [33]:

```
z= x+y
print(z)

z= x-y
print(z)

z= x*y
print(z)

z= x/y
print(z)

z= np.exp(x)
print(z)

z= np.sqrt(y)
print(z)

z= np.dot(x,y)
print(z)

print(type(z))
```

```
[2. 3. 4. 5. 6.]
[ 0. -1. -2. -3. -4.]
[1. 2. 3. 4. 5.]
[1.         0.5        0.33333333 0.25       0.2        ]
```

```
[2.71828183 2.71828183 2.71828183 2.71828183 2.71828183]
[1.         1.41421356 1.73205081 2.         2.23606798]
15.0
<class 'numpy.float64'>
```

In [34]:

```
x = np.ones((5,1))
y = np.linspace(1,5,5)
y = y.reshape((1,5))
print(x.shape)
print(y.shape)
print(x)
print(y)
```

```
(5, 1)
(1, 5)
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
[[1. 2. 3. 4. 5.]]
```

In [35]:

```
z= x+y
print(z)

z= x-y
print(z)

z= x*y
print(z)

z= x/y
print(z)

z= np.exp(x)
print(z)

z= np.sqrt(y)
print(z)

print(z.shape)
```

```
[[2. 3. 4. 5. 6.]
 [2. 3. 4. 5. 6.]
 [2. 3. 4. 5. 6.]
 [2. 3. 4. 5. 6.]
 [2. 3. 4. 5. 6.]]
[[ 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.]]
[[1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]]
[[1.         0.5        0.33333333 0.25       0.2       ]
 [1.         0.5        0.33333333 0.25       0.2       ]
 [1.         0.5        0.33333333 0.25       0.2       ]
 [1.         0.5        0.33333333 0.25       0.2       ]
 [1.         0.5        0.33333333 0.25       0.2       ]]
[[2.71828183]
 [2.71828183]
 [2.71828183]
 [2.71828183]
 [2.71828183]]
[[1.         1.41421356 1.73205081 2.         2.23606798]
```

```
(1, 5)
```

In [37]:

```
z= np.dot(y,x)
print(z)
```

```
[[15.]]
```

In [ ]:

```
help(np.dot)
```

MATRIX-VECTOR MOLTIPLICATION

In [38]:

```
A=[[1,2,3],[4,5,6],[7,8,9]]
A=np.array(A)
print(A)
print(A.shape)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
(3, 3)
```

In [39]:

```
v=np.ones((A.shape[1],1))
print(v)
print(v.shape)
```

```
[[1.]
 [1.]
 [1.]]
(3, 1)
```

In [43]:

```
result=np.matmul(A,v)
print(result)
print(result.shape)
```

```
[[ 6.]
 [15.]
 [24.]]
(3, 1)
```

In [44]:

```
result= A*v #watch out! This is not matrix-vector moltiplication
print(result)
```

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

In [45]:

```
x = np.ones((5,1))
y = np.linspace(1,5,5)
y = y.reshape((1,5))

print(np.dot(y,x))
print(np.matmul(y,x))
```

```
[[15.]]
[[15.]]
```

ARRAY MANIPULATION

In [46]:

```
A=np.random.rand(2,2)
print(A)
```

```
[[0.5717753  0.83953391]
 [0.16568948 0.13196589]]
```

In [48]:

```
B=np.transpose(A)
print(B,'\n')
C=A.T
print(B)
print(A==B)
print(C==B)
```

```
[[0.5717753  0.16568948]
 [0.83953391 0.13196589]]

[[0.5717753  0.16568948]
 [0.83953391 0.13196589]]
[[ True False]
 [False  True]]
[[ True  True]
 [ True  True]]
```

In [52]:

```
print(np.linalg.matrix_rank(A))
C=np.linalg.inv(A)
print(np.matmul(C,A),'\n')
print(C*A,'\n')
print(C@A)
```

```
2
[[ 1.00000000e+00 -1.02473257e-17]
 [ 2.39374759e-17  1.00000000e+00]]

[[-1.18551887 11.07383072]
 [ 0.43133157 -1.18551887]]

[[ 1.00000000e+00  6.30482243e-17]
 [-6.98540687e-17  1.00000000e+00]]
```

AGGREGATION OF ARRAYS AND CONDITIONAL EXPRESSION

In [57]:

```
t=np.sum(A)
print(t)
```

```
21.0
```

In [58]:

```
t = np.sum(A, axis=0)
print(t)
```

```
[5. 7. 9.]
```

In [59]:

```
t = np.sum(A, axis=1)
print(t)
```

```
[ 6. 15.]
```

In [60]:

```
t=np.mean(A)
print(t)
```

```
3.5
```

In [61]:

```
t=np.std(A)
print(t)
```

```
1.707825127659933
```

In [62]:

```
t=np.prod(A)
print(t)
```

```
720.0
```

In [ ]:

```
x=np.linspace(-10,10,21)
print(x,'\n')
print(x<0)

s=np.array(x[np.nonzero(x<0)])
print(s)

help(np.nonzero)
```

FOR inside an array

In [66]:

```
a=np.zeros((1,10))
print(a)

for i in range(np.size(a)):
  a[0,i]=1

print(a)
```

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

In [71]:

```
a=np.arange(30)
print(a,'\n')
a=np.reshape(a,(3,10))

print(a,'\n')

t = a[:,2]
print(t,'\n')

t = a[1,:]
print(t,'\n')

t = a[1:3,1:6]
print(t,'\n')
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29]

[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]]
```

```
[ 2 12 22]

[10 11 12 13 14 15 16 17 18 19]

[[11 12 13 14 15]
 [21 22 23 24 25]]
```

Matplotlib

In [73]:

```
import matplotlib.pyplot as plt
```

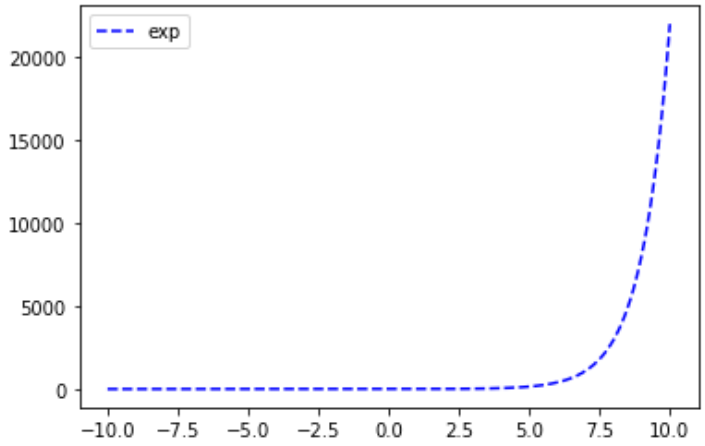
In [75]:

```
x = np.linspace(-10,10, num=200)
y = np.exp(x)

#plt.plot(x,y,color='red', linestyle='--')
plt.plot(x,y,color='blue', linestyle='--')

plt.legend(['exp'])

plt.show()
```



In [ ]:

```

```