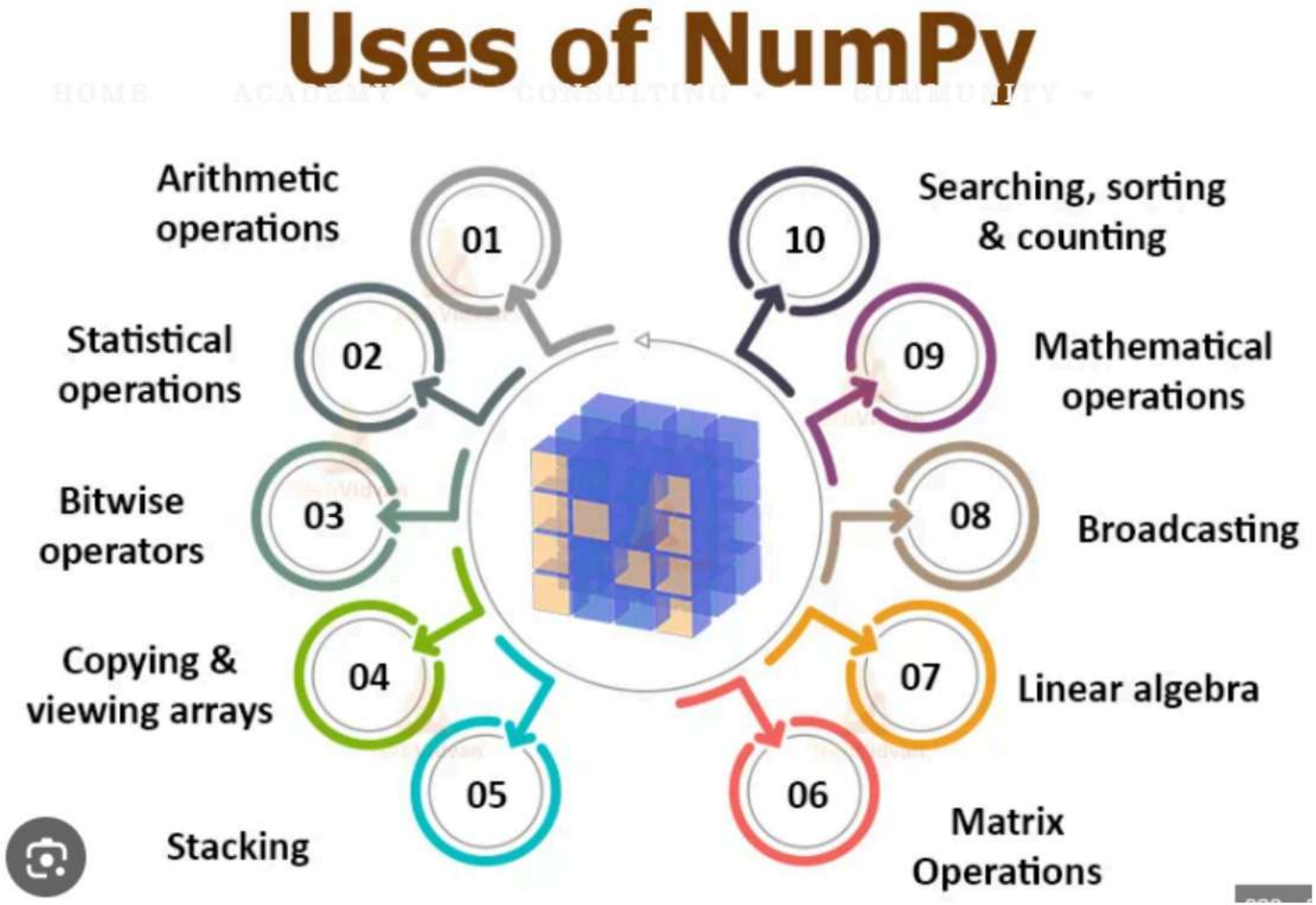


numpy and sys(jupytor)

In [162…
```
import numpy as np
```

In [164…
```
np.__version__
```

Out[164… '1.26.4'

In [15]:
```
import sys
sys.version
```

Out[15]: '3.12.7 | packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]'

creating array

In [21]:
```
mylist = [1,2,3,4]
mylist
```

Out[21]: [1, 2, 3, 4]

In [23]:
```
type(mylist)
```

Out[23]: list

converting list to array

In [34]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\jayes\anaconda3\lib\site-packages (1.26.4)
```

In [36]:
```
arr = np.array(mylist)
```

In [38]:
```
arr
```

Out[38]: array([1, 2, 3, 4])

In [40]:
```
type(arr)
```

Out[40]: numpy.ndarray

In [44]:
```
type(mylist)
```

Out[44]:
```
list
```

In [48]:
```
np.arange(15)
```

Out[48]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [50]:
```
np.arange(3.0)
```

Out[50]:
```
array([0., 1., 2.])
```

In [52]:
```
np.arange(15)
```

Out[52]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [54]:
```
np.arange(0,5)
```

Out[54]:
```
array([0, 1, 2, 3, 4])
```

In [56]:
```
np.arange(10,20)
```

Out[56]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [58]:
```
np.arange(20,10)
```

Out[58]:
```
array([], dtype=int32)
```

In [78]:
```
np.arange(-20,10)
```

Out[78]:
```
array([-20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10,  -9,  -8,
        -7,  -6,  -5,  -4,  -3,  -2,  -1,   0,   1,   2,   3,   4,   5,
         6,   7,   8,   9])
```

In [82]:
```
array_1d = np.array([1,2,3,4,5])
print("1D Array:")
print(array_1d)
print("\nelements of array :",array_1d.size)
print("\n size of array :",array_1d.shape)
```

```
1D Array:
[1 2 3 4 5]

elements of array : 5

 size of array : (5,)
```

In [86]:
```
array_with_nan = np.array([1, 2, np.nan, 4,5])
print("Array with np.nan:")
print(array_with_nan)
```

```
Array with np.nan:
[ 1.  2. nan  4.  5.]
```

In [64]:
```
array_2d = np.array([[1,2,3], [4,5,6], [7,8,9]])
print("\n2D Array:")
print(array_2d)
```

```
2D Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [66]:
```
array_3d = np.array([[[1,2], [3,4]], [[5,6], [7,8]], [[9,10], [11,12]]])
print("\n3D Array:")
print(array_3d)
```

```
3D Array:
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
```

In [90]:
```
a = np.arange(10,50,5)
a
```

Out[90]:
```
array([10, 15, 20, 25, 30, 35, 40, 45])
```

In [170…
```
# Create an array with linearly spaced values
c = np.linspace(0, 1, 5)  # 5 values evenly spaced between 0 and 1
print("Array c:", c)
```

```
Array c: [0.   0.25 0.5  0.75 1.  ]
```

In [168…
```
np.linspace(1,2,5)
```

Out[168… array([1. , 1.25, 1.5 , 1.75, 2.])

In [172…
```
# Create an identity matrix
f = np.eye(4)  # 4x4 identity matrix
print("Identity matrix f:\n", f)
```

```
Identity matrix f:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

In [92]:
```
a1 = np.zeros(3)                                #parameter tuning. by default system prints float
a1
```

Out[92]: array([0., 0., 0.])

In [100…
```
a2 = np.zeros(3, dtype=int)                      #hyper parameter tuning. user prints it as int
a2
```

Out[100… array([0, 0, 0])

In [108…
```
a3 = np.zeros((2,2),dtype=int)
print(a3)
print(type(a3))
```

```
[[0 0]
 [0 0]]
<class 'numpy.ndarray'>
```

In [112…
```
np.zeros((2,10))
```

Out[112…
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [116…
```
n = (3,4)
np.zeros(n, dtype=int)
```

Out[116…
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

In [120…
```
n = (4,5)
np.ones(n, dtype=int)
```

Out[120…
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

In [142…
```
n = (4,3,2)
np.ones(n,dtype=int)
```

Out[142…
```
array([[[1, 1],
        [1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1],
        [1, 1]]])
```

In [144…
```
rand(3,5)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[144], line 1
----> 1 rand(3,5)

NameError: name 'rand' is not defined
```

In [124…
```
random.rand(3,5)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[124], line 1
----> 1 random.rand(3,5)

NameError: name 'random' is not defined
```

In [257…
```
np.random.rand()
```

Out[257…
```
0.7277717534105705
```

In [128…
```
np.random.rand(3,5)                                    # by default generate float value between 0 and 1
```

Out[128…
```
array([[0.39602538, 0.64180677, 0.47336426, 0.34810518, 0.61187416],
       [0.70180038, 0.75752145, 0.31751809, 0.9110231 , 0.80899422],
       [0.39370051, 0.32051793, 0.33905493, 0.03275241, 0.17442674]])
```

In [132…
```
np.random.randint(3,8)
```

Out[132…
```
3
```

In [134…
```
np.random.randint(3,9,3)
```

Out[134…
```
array([6, 5, 3])
```

In [138…
```
np.random.randint(10,40,(2,3))
```

Out[138…
```
array([[35, 18, 16],
       [36, 20, 31]])
```

In [152…
```
b = np.random.randint(10,40,(3,4,2))
b
```

Out[152…
```
array([[[26, 38],
        [38, 25],
        [11, 27],
        [23, 33]],

       [[26, 23],
        [22, 36],
        [13, 23],
        [24, 27]],

       [[33, 36],
        [29, 19],
        [25, 30],
        [39, 24]]])
```

In [288…
```
b[::-1,:,:]                                # reverse by 3rd parameter of a 3D array
```

Out[288…
```
array([[[33, 36],
        [29, 19],
        [25, 30],
        [39, 24]],

       [[26, 23],
        [22, 36],
        [13, 23],
        [24, 27]],

       [[26, 38],
        [38, 25],
        [11, 27],
        [23, 33]]])
```

In [290…
```
type(b)
```

Out[290…
```
numpy.ndarray
```

In [291…
```
np.arange(1,13).reshape(3,4)
```

Out[291…
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [178…
```
# Reshape an array
a1 = np.array([[5, 8], [4,6]])
print(a1)
reshaped = np.reshape(a1, (1, 4))  # Reshape to 1x3
print("\nReshaped array:", reshaped)
```

```
[[5 8]
 [4 6]]

Reshaped array: [[5 8 4 6]]
```

In [180…
```
# Flatten an array
f1 = np.array([[1, 2], [3, 4]])
flattened = np.ravel(f1)  # Flatten to 1D array
print("Flattened array:", flattened)
```

```
Flattened array: [1 2 3 4]
```

In [182…
```
# Transpose an array
e1 = np.array([[1, 2], [3, 4]])
transposed = np.transpose(e1)  # Transpose the array
print("Transposed array:\n", transposed)
```

```
Transposed array:
 [[1 3]
 [2 4]]
```

In [184…
```
# Stack arrays vertically
a2 = np.array([1, 2])
b2 = np.array([3, 4])
stacked = np.vstack([a2, b2])  # Stack a and b vertically
print("Stacked arrays:\n", stacked)
```

```
Stacked arrays:
 [[1 2]
 [3 4]]
```

In [158…
```
c = np.random.randint(10,20,(3,4))
c
```

Out[158…
```
array([[13, 16, 19, 16],
       [12, 17, 16, 10],
       [17, 18, 10, 19]])
```

mathematical functions

In [187…
```
# Add two arrays
g = np.array([1, 2, 3, 4])
added = np.add(g, 2)  # Add 2 to each element
print("Added 2 to g:", added)
```

```
Added 2 to g: [3 4 5 6]
```

In [189…
```
# Square each element
squared = np.power(g, 2)  # Square each element
print("Squared g:", squared)
```

```
Squared g: [ 1  4  9 16]
```

In [191…
```
# Square root of each element
sqrt_val = np.sqrt(g)  # Square root of each element
print("Square root of g:", sqrt_val)
```

```
Square root of g: [1.         1.41421356 1.73205081 2.        ]
```

In [199…
```
a=np.array([1,2,3])
a1=np.array([1,2,3])
print(a)
print(a1)
```

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

In [201…
```
dot_product = np.dot(a1, a)  # Dot product of a1 and a
print("Dot product of a1 and a:", dot_product)
```

```
Dot product of a1 and a: 14
```

slicing

In [355…
```
slice = np.random.randint(10,40,(5,4))
slice
```

Out[355…
```
array([[27, 22, 24, 20],
       [19, 15, 36, 26],
       [10, 35, 21, 21],
       [36, 34, 34, 22],
       [27, 39, 29, 16]])
```

In [356…
```
slice
```

Out[356…
```
array([[27, 22, 24, 20],
       [19, 15, 36, 26],
       [10, 35, 21, 21],
       [36, 34, 34, 22],
       [27, 39, 29, 16]])
```

In [360…
```
print(type(slice))
```

```
<class 'numpy.ndarray'>
```

In [212…
```
slice[:]
```

Out[212…
```
array([[38, 20, 17, 23],
       [33, 36, 23, 32],
       [16, 30, 39, 27],
       [25, 16, 22, 39],
       [14, 29, 12, 14]])
```

In [362…
```
s1 = slice[1:3]
s1
```

Out[362…
```
array([[19, 15, 36, 26],
       [10, 35, 21, 21]])
```

In [254…
```
print(type(s1))
```

```
<class 'numpy.ndarray'>
```

In [248…
```
s = slice[1,2]
s
```

Out[248…
```
23
```

In [250…
```
print(type(s))
```

```
<class 'numpy.int32'>
```

In [226…
```
slice[1,3]
```

Out[226…
```
32
```

In [228…
```
slice[1,-1]
```

Out[228…
```
32
```

In [230…
```
slice
```

Out[230…
```
array([[38, 20, 17, 23],
       [33, 36, 23, 32],
       [16, 30, 39, 27],
       [25, 16, 22, 39],
       [14, 29, 12, 14]])
```

In [232…
```
slice[2:3]                                    #forward slicing
```

Out[232…
```
array([[16, 30, 39, 27]])
```

In [262…
```
slice[-3:-2]                                    #backward slicing
```

Out[262…
```
array([[16, 30, 39, 27]])
```

In [267…
```
slice[0:-2]                                     #mixed slicing
```

Out[267…
```
array([[38, 20, 17, 23],
       [33, 36, 23, 32],
       [16, 30, 39, 27]])
```

In [271…
```
slice[-2:]
```

Out[271…
```
array([[25, 16, 22, 39],
       [14, 29, 12, 14]])
```

In [273…
```
slice[:]
```

Out[273…
```
array([[38, 20, 17, 23],
       [33, 36, 23, 32],
       [16, 30, 39, 27],
       [25, 16, 22, 39],
       [14, 29, 12, 14]])
```

In [281…
```
slice[::-1,:]                                    #reverse rows of matrix
```

Out[281…
```
array([[14, 29, 12, 14],
       [25, 16, 22, 39],
       [16, 30, 39, 27],
       [33, 36, 23, 32],
       [38, 20, 17, 23]])
```

In [283…
```
slice[:,::-1]                                     #reverse columns of matrix
```

Out[283…
```
array([[23, 17, 20, 38],
       [32, 23, 36, 33],
       [27, 39, 30, 16],
       [39, 22, 16, 25],
       [14, 12, 29, 14]])
```

In [275…
```
slice[-1]
```

Out[275…
```
array([14, 29, 12, 14])
```

In [279…
```
slice[:-1]
```

Out[279…
```
array([[38, 20, 17, 23],
       [33, 36, 23, 32],
       [16, 30, 39, 27],
       [25, 16, 22, 39]])
```

In [242…
```
b[1:3]
```

Out[242…
```
array([[[26, 23],
        [22, 36],
        [13, 23],
        [24, 27]],

       [[33, 36],
        [29, 19],
        [25, 30],
        [39, 24]]])
```

In [236…
```
b[1,2]
```

Out[236…
```
array([13, 23])
```

In [296…
```
slice
```

Out[296…
```
array([[38, 20, 17, 23],
       [33, 36, 23, 32],
       [16, 30, 39, 27],
       [25, 16, 22, 39],
       [14, 29, 12, 14]])
```

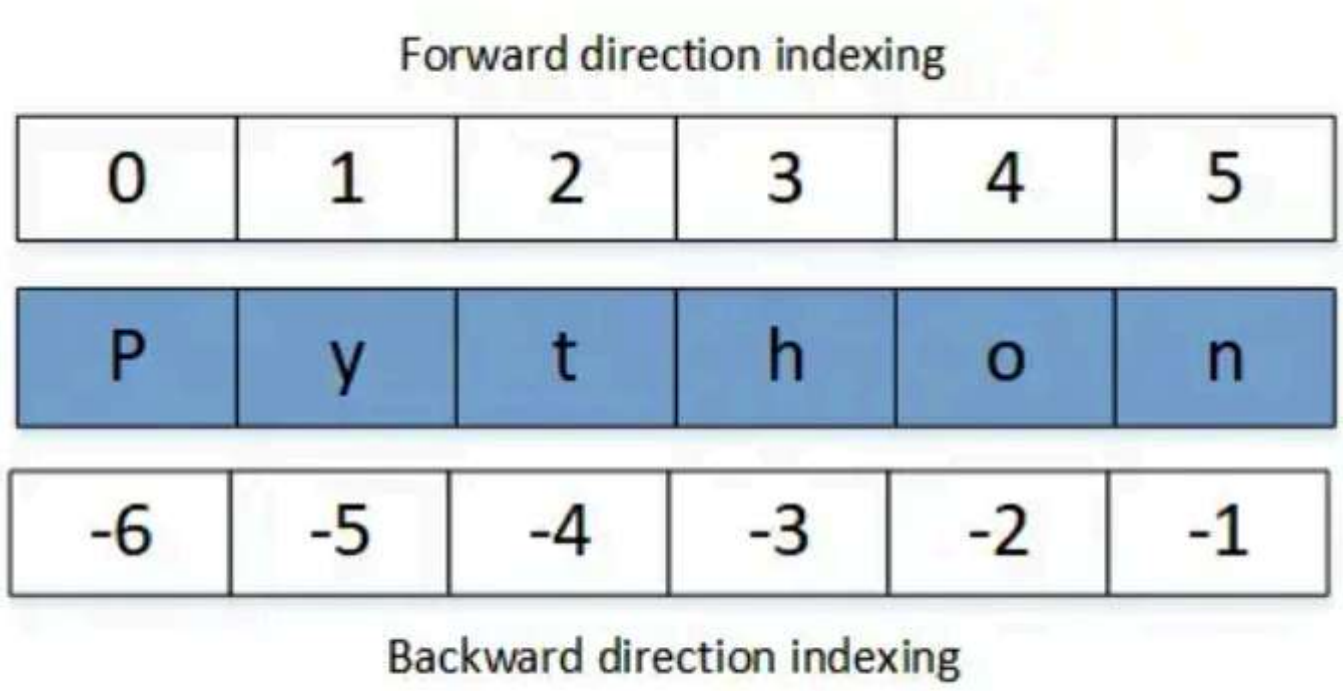
In [302…
```
slice[1:4]
```

Out[302…
```
array([[33, 36, 23, 32],
       [16, 30, 39, 27],
       [25, 16, 22, 39]])
```

In [300…
```
slice[-4,2]
```

Out[300…
```
23
```

- : slicing row by row in 2D matrix(slicing)
- , prints single element in 2D matrix(indexing)
- : slicing 2D matrix by 2D matrix in 3D matrix
- , prints a row in 3D matrix
- single element is of type numpy.int or numpy.float etc and other is of type numpy.ndarray
- use '-1' in step part to reverse string, 1D, 2D and 3D arrays



In [388…
```
from numpy import *
a = array([1,2,3,4,9])
median(a)
```

Out[388… 3.0

Without work on import* can you please find the median, mode)

In [390…
```
# median is the middle number for odd number list and average of middle two numbers for even number list

def find_median(lst):
    lst.sort()  # Sort the list
    n = len(lst)
    mid = n // 2  # Middle index
```

```
    if n % 2 == 0:
        return (lst[mid - 1] + lst[mid]) / 2  # Average of two middle values
    else:
        return lst[mid]  # Middle value

# Example
numbers = [7, 3, 9, 2, 6]
print("Median:", find_median(numbers))  # Output: 6
```

```
Median: 6
```

In [392…
```
# The mode is the number that appears most frequently in a list.

def find_mode(lst):
    frequency = {}  # Dictionary to store count of each number

    for num in lst:
        frequency[num] = frequency.get(num, 0) + 1  # Count occurrences

    max_count = max(frequency.values())  # Find highest count

    modes = [key for key, val in frequency.items() if val == max_count]  # Find all modes

    return modes if len(modes) > 1 else modes[0]  # Return single value if only one mode

# Example
numbers = [4, 1, 2, 2, 3, 4, 4]
print("Mode:", find_mode(numbers))  # Output: 4
```

```
Mode: 4
```

In [309…
```
slice
```

Out[309…
```
array([[38, 20, 17, 23],
       [33, 36, 23, 32],
       [16, 30, 39, 27],
       [25, 16, 22, 39],
       [14, 29, 12, 14]])
```

In [313…
```
slice[4,:]                                # prints 5th row
```

Out[313…
```
array([14, 29, 12, 14])
```

In [318…
```
slice[:,3]                                  # prints 4th column
```

Out[318…
```
array([23, 32, 27, 39, 14])
```

In [323…
```
slice[2:4,1:3]
```

Out[323…
```
array([[30, 39],
       [16, 22]])
```

masking or filters

In [328…
```
slice<20
```

Out[328…
```
array([[False, False,  True, False],
       [False, False, False, False],
       [ True, False, False, False],
       [False,  True, False, False],
       [ True, False,  True,  True]])
```

In [332…
```
slice[slice<20]
```

Out[332…
```
array([17, 16, 16, 14, 12, 14])
```

In [334…
```
import numpy as np
```

In [336…
```
import matplotlib.pyplot as plt
```

In [340…
```
from PIL import Image
```

In [352…
```
jayesh = Image.open(r'C:\Users\jayes\OneDrive\Desktop\jayesh.jpg')
jayesh
```

Out[352…

In [ ]: