Name: Punit Kawadkar

Roll no: 431

Batch: D2

- ## Practical no: 3

Prepare/Take datasets for any real-life application. Read a dataset into an array. Perform the following operations on it:

1. **Perform all matrix operations**
2. **Horizontal and vertical stacking of Numpy Arrays**
3. **Custom sequence generation**
4. **Arithmetic and Statistical Operations, Mathematical Operations, Bitwise Operators**
5. **Copying and viewing arrays**
6. **Data Stacking, Searching, Sorting, Counting, Broadcasting**

Dataset: Average monthly temperature of States in India.

1 to 10 of 10 entíiesFilteí

State Code	March	April	May	June	July
1	20.4	26.3	31	32	31
2	21.4	22.8	30	31.2	31.8
3	24.3	26.2	27.4	30	30.8
4	25.5	30.6	31	32	31
5	28	31	32.8	30	31.9
6	29	32	35	34.7	32.6
7	30	32.4	35.6	32.4	32
8	35	36	36.4	30	32
9	34.4	35.4	36.5	34	32
10	29	31	36.7	35.8	32.7

Show ▾ per page

1.Perform all matrix operations

Code:

```
import numpy as np
arr = np.loadtxt("/content/Practical 3.csv",delimiter=",",dtype=str)
arr

#Perform all matrix operations

# Define the given array
arr = np.array([
    ['State Code', 'March', 'April', 'May', 'June', 'July'],
    ['1', '20.4', '26.3', '31', '32', '31'],
    ['2', '21.4', '22.8', '30', '31.2', '31.8'],
    ['3', '24.3', '26.2', '27.4', '30', '30.8'],
    ['4', '25.5', '30.6', '31', '32', '31'],
    ['5', '28', '31', '32.8', '30', '31.9'],
    ['6', '29', '32', '35', '34.7', '32.6'],
    ['7', '30', '32.4', '35.6', '32.4', '32'],
    ['8', '35', '36', '36.4', '30', '32'],
    ['9', '34.4', '35.4', '36.5', '34', '32'],
    ['10', '29', '31', '36.7', '35.8', '32.7']
])

# Extract the numerical values as a float array
num_arr = arr[1:].astype(float)

# Transpose the array
transposed_arr = np.transpose(num_arr)

# Calculate the mean along axis 1
mean_arr = np.mean(num_arr, axis=1)

# Calculate the sum along axis 0
sum_arr = np.sum(num_arr, axis=0)

# Multiply the array by a scalar value
scalar = 2
scalar_mult_arr = num_arr * scalar

# Add two arrays element-wise
added_arr = num_arr + scalar_mult_arr

# Perform matrix multiplication
matrix_mult_arr = np.matmul(num_arr, transposed_arr)

print("Original Array:")
print(arr)
print()

print("Numerical Array:")
print(num_arr)
print()

print("Transposed Array:")
print(transposed_arr)
print()

print("Mean along Axis 1:")
print(mean_arr)
print()

print("Sum along Axis 0:")
print(sum_arr)
print()

print("Scalar Multiplication:")
print(scalar_mult_arr)
print()

print("Element-wise Addition:")
print(added_arr)
print()

print("Matrix Multiplication:")
print(matrix_mult_arr)
```

Output:

```
Original Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]

Numerical Array:
[[ 1.   20.4 26.3 31.  32.  31. ]
 [ 2.   21.4 22.8 30.  31.2 31.8]
 [ 3.  24.3 26.2 27.4 30.  30.8]
 [ 4.  25.5 30.6 31.  32.  31. ]
 [ 5.  28.  31.  32.8 30.  31.9]
 [ 6.  29.  32.  35.  34.7 32.6]
 [ 7.  30.  32.4 35.6 32.4 32. ]
 [ 8.  35.  36.  36.4 30.  32. ]
 [ 9.  34.4 35.4 36.5 34.  32. ]
 [10.  29.  31.  36.7 35.8 32.7]]

Transposed Array:
[[ 1.   2.   3.   4.   5.   6.   7.   8.   9.  10. ]
 [20.4 21.4 24.3 25.5 28.  29.  30.  35.  34.4 29. ]
 [26.3 22.8 26.2 30.6 31.  32.  32.4 36.  35.4 31. ]
 [31.  30.  27.4 31.  32.8 35.  35.6 36.4 36.5 36.7]
 [32.  31.2 30.  32.  30.  34.7 32.4 30.  34.  35.8]
 [31.  31.8 30.8 31.  31.9 32.6 32.  32.  32.  32.7]]

Mean along Axis 1:
[23.61666667 23.2        23.61666667 25.68333333 26.45
28.21666667
 28.23333333 29.56666667 30.21666667 29.2       ]

Sum along Axis 0:
[ 55.  277.  303.7 332.4 322.1 317.8]

Scalar Multiplication:
[[ 2.  40.8 52.6 62.  64.  62. ]
 [ 4.  42.8 45.6 60.  62.4 63.6]
 [ 6.  48.6 52.4 54.8 60.  61.6]
 [ 8.  51.  61.2 62.  64.  62. ]
 [10.  56.  62.  65.6 60.  63.8]
 [12.  58.  64.  70.  69.4 65.2]
 [14.  60.  64.8 71.2 64.8 64. ]
 [16.  70.  72.  72.8 60.  64. ]
 [18.  68.8 70.8 73.  68.  64. ]
 [20.  58.  62.  73.4 71.6 65.4]]

Element-wise Addition:
[[  3.   61.2  78.9  93.   96.   93. ]
 [  6.   64.2  68.4  90.   93.6  95.4]
 [  9.   72.9  78.6  82.2  90.   92.4]
 [ 12.   76.5  91.8  93.   96.   93. ]
 [ 15.   84.   93.   98.4  90.   95.7]
 [ 18.   87.   96.  105.  104.1  97.8]
 [ 21.   90.   97.2 106.8  97.2  96. ]
 [ 24.  105.  108.  109.2  90.   96. ]
 [ 27.  103.2 106.2 109.5 102.   96. ]
 [ 30.   87.   93.  110.1 107.4  98.1]]

Matrix Multiplication:
[[4054.85 3952.4  3951.98 4274.98 4357.2  4645.2  4603.52 4749.2
4853.28
  4713.9 ]
 [3952.4  3866.48 3860.82 4165.58 4250.42 4531.52 4491.2  4631.4
4734.68
  4605.22]
 [3951.98 3860.82 3885.33 4197.57 4288.84 4565.18 4531.92 4700.66
4796.1
```

```
  4633.64]
 [4274.98 4165.58 4197.57 4548.61 4648.3  4948.7  4916.84 5106.5
5207.94
  5025.1 ]
 [4357.2  4250.42 4288.84 4648.3  4763.45 5062.94 5039.88 5250.72
5343.6
  5143.89]
 [4645.2  4531.52 4565.18 4948.7  5062.94 5392.85 5362.28 5573.2
5684.9
  5485.78]
 [4603.52 4491.2  4531.92 4916.84 5039.88 5362.28 5339.88 5564.24
5666.96
  5457.24]
 [4749.2  4631.4  4700.66 5106.5  5250.72 5573.2  5564.24 5833.96 5923.
  5667.28]
 [4853.28 4734.68 4796.1  5207.94 5343.6  5684.9  5666.96 5923.
6029.77
  5788.15]
 [4713.9  4605.22 4633.64 5025.1  5143.89 5485.78 5457.24 5667.28
5788.15
  5599.82]]
```

2.Horizontal Stacking of Numpy arrays

Code:

```
import numpy as np

# Define the given array
arr = np.array([
    ['State Code', 'March', 'April', 'May', 'June', 'July'],
    ['1', '20.4', '26.3', '31', '32', '31'],
    ['2', '21.4', '22.8', '30', '31.2', '31.8'],
    ['3', '24.3', '26.2', '27.4', '30', '30.8'],
    ['4', '25.5', '30.6', '31', '32', '31'],
    ['5', '28', '31', '32.8', '30', '31.9'],
    ['6', '29', '32', '35', '34.7', '32.6'],
    ['7', '30', '32.4', '35.6', '32.4', '32'],
    ['8', '35', '36', '36.4', '30', '32'],
    ['9', '34.4', '35.4', '36.5', '34', '32'],
    ['10', '29', '31', '36.7', '35.8', '32.7']
])

# Perform horizontal stacking
stacked_arr = np.hstack((arr, arr[:, 1:])) # Stack the array
horizontally with additional columns

print("Original Array:")
print(arr)
print()
print("Horizontally Stacked Array:")
print(stacked_arr)
```

Output:

```
Original Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]

Horizontally Stacked Array:
[['State Code' 'March' 'April' 'May' 'June' 'July' 'March' 'April'
'May'
  'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8' '21.4' '22.8' '30' '31.2'
'31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8' '24.3' '26.2' '27.4' '30'
'30.8']
 ['4' '25.5' '30.6' '31' '32' '31' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7' '29' '31' '36.7' '35.8' '32.7']]
```

Vertical and Numerical stacking of Numpy arrays

Code:

```
import numpy as np

# Define the given array
arr = np.array([
    ['State Code', 'March', 'April', 'May', 'June', 'July'],
    ['1', '20.4', '26.3', '31', '32', '31'],
    ['2', '21.4', '22.8', '30', '31.2', '31.8'],
    ['3', '24.3', '26.2', '27.4', '30', '30.8'],
    ['4', '25.5', '30.6', '31', '32', '31'],
    ['5', '28', '31', '32.8', '30', '31.9'],
    ['6', '29', '32', '35', '34.7', '32.6'],
    ['7', '30', '32.4', '35.6', '32.4', '32'],
    ['8', '35', '36', '36.4', '30', '32'],
    ['9', '34.4', '35.4', '36.5', '34', '32'],
    ['10', '29', '31', '36.7', '35.8', '32.7']
])

# Extract the numerical values as a float array
num_arr = arr[1:].astype(float)

# Create additional arrays for horizontal and vertical stacking
horizontal_arr = np.array([
    ['11', '30.7', '32.4', '35', '34.7', '32.6'],
    ['12', '31', '33', '36', '35.3', '33'],
    ['13', '32', '34', '37', '35.5', '33.5']
])

vertical_arr = np.array([
    ['11', '30', '32', '35', '34.7', '32.6'],
    ['12', '31', '33', '36', '35', '33'],
    ['13', '32', '34', '37', '35.5', '33.5']
])


# Perform vertical stacking
vertical_stacked_arr = np.vstack((num_arr, vertical_arr))


print("Original Array:")
print(arr)
print()

print("Numerical Array:")
print(num_arr)
print()



print("Vertical Stacking:")
print(vertical_stacked_arr)
```

Output:

```
Original Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]

Numerical Array:
[[ 1.   20.4 26.3 31.   32.   31. ]
 [ 2.   21.4 22.8 30.   31.2 31.8]
 [ 3.   24.3 26.2 27.4 30.   30.8]
 [ 4.   25.5 30.6 31.   32.   31. ]
 [ 5.   28.   31.   32.8 30.   31.9]
 [ 6.   29.   32.   35.   34.7 32.6]
[ 7. 30.  32.4 35.6 32.4 32. ]
 [ 8.   35.   36.   36.4 30.   32. ]
 [ 9.   34.4 35.4 36.5 34.   32. ]
 [10.   29.   31.   36.7 35.8 32.7]]

Vertical Stacking:
[['1.0' '20.4' '26.3' '31.0' '32.0' '31.0']
 ['2.0' '21.4' '22.8' '30.0' '31.2' '31.8']
 ['3.0' '24.3' '26.2' '27.4' '30.0' '30.8']
 ['4.0' '25.5' '30.6' '31.0' '32.0' '31.0']
 ['5.0' '28.0' '31.0' '32.8' '30.0' '31.9']
 ['6.0' '29.0' '32.0' '35.0' '34.7' '32.6']
 ['7.0' '30.0' '32.4' '35.6' '32.4' '32.0']
 ['8.0' '35.0' '36.0' '36.4' '30.0' '32.0']
 ['9.0' '34.4' '35.4' '36.5' '34.0' '32.0']
 ['10.0' '29.0' '31.0' '36.7' '35.8' '32.7']
 ['11' '30' '32' '35' '34.7' '32.6']
 ['12' '31' '33' '36' '35' '33']
 ['13' '32' '34' '37' '35.5' '33.5']]
```

4.Arithmetic and Statistical Operations, Mathematical Operations, Bitwise Operators.

Code:

```python
import numpy as np

# Define the given array
arr = np.array([
    ['State Code', 'March', 'April', 'May', 'June', 'July'],
    ['1', '20.4', '26.3', '31', '32', '31'],
    ['2', '21.4', '22.8', '30', '31.2', '31.8'],
    ['3', '24.3', '26.2', '27.4', '30', '30.8'],
    ['4', '25.5', '30.6', '31', '32', '31'],
    ['5', '28', '31', '32.8', '30', '31.9'],
    ['6', '29', '32', '35', '34.7', '32.6'],
    ['7', '30', '32.4', '35.6', '32.4', '32'],
    ['8', '35', '36', '36.4', '30', '32'],
    ['9', '34.4', '35.4', '36.5', '34', '32'],
    ['10', '29', '31', '36.7', '35.8', '32.7']
])
```

```
# Extract the numerical values as a float array
num_arr = arr[1:].astype(float)

# Arithmetic and statistical operations
sum_row = np.sum(num_arr, axis=0) # Sum along the rows
mean_col = np.mean(num_arr, axis=1) # Mean along the columns
max_val = np.max(num_arr)  # Maximum value in the array

# Mathematical operations
sqrt_arr = np.sqrt(num_arr) # Square root of each element
exp_arr = np.exp(num_arr)  # Exponential of each element
log_arr = np.log(num_arr)  # Natural logarithm of each element

# Bitwise operations
bitwise_and = np.bitwise_and(num_arr.astype(int), 5) # Bitwise AND with 5
bitwise_or = np.bitwise_or(num_arr.astype(int), 3) # Bitwise OR with 3
bitwise_xor = np.bitwise_xor(num_arr.astype(int), 2)  # Bitwise XOR with 2

print("Original Array:")
print(arr)
print()

print("Arithmetic and Statistical Operations:")
print("Sum of each column:", sum_row)
print("Mean of each row:", mean_col)
print("Maximum value:", max_val)
print()

print("Mathematical Operations:")
print("Square root of each element:")
print(sqrt_arr)
print()
print("Exponential of each element:")
print(exp_arr)
print()
print("Natural logarithm of each element:")
print(log_arr)
print()

print("Bitwise Operations:")
print("Bitwise AND with 5:")
print(bitwise_and)
print()
print("Bitwise OR with 3:")
print(bitwise_or)
print()
print("Bitwise XOR with 2:")
print(bitwise_xor)
```

Output:

```
Original Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]

Arithmetic and Statistical Operations:
Sum of each column: [ 55.  277.  303.7 332.4 322.1 317.8]
Mean of each row: [23.61666667 23.2        23.61666667 25.68333333
26.45       28.21666667
 28.23333333 29.56666667 30.21666667 29.2        ]
Maximum value: 36.7

Mathematical Operations:
Square root of each element:
[[1.         4.51663592 5.12835256 5.56776436 5.65685425 5.56776436]
 [1.41421356 4.6260134  4.77493455 5.47722558 5.58569602 5.63914887]
 [1.73205081 4.92950302 5.11859356 5.23450093 5.47722558 5.54977477]
 [2.         5.04975247 5.53172667 5.56776436 5.65685425 5.56776436]
 [2.23606798 5.29150262 5.56776436 5.72712843 5.47722558 5.6480085 ]
 [2.44948974 5.38516481 5.65685425 5.91607978 5.89067059 5.70964097]
 [2.64575131 5.47722558 5.69209979 5.96657356 5.69209979 5.65685425]
 [2.82842712 5.91607978 6.         6.03324125 5.47722558 5.65685425]
 [3.         5.86515132 5.94978991 6.04152299 5.83095189 5.65685425]
 [3.16227766 5.38516481 5.56776436 6.05805249 5.98331012 5.71839138]]

Exponential of each element:
[[2.71828183e+00 7.23781421e+08 2.64207337e+11 2.90488497e+13
  7.89629602e+13 2.90488497e+13]
 [7.38905610e+00 1.96744188e+09 7.97837026e+09 1.06864746e+13
  3.54803451e+13 6.46494039e+13]
 [2.00855369e+01 3.57565748e+10 2.39064685e+11 7.93722706e+11
  1.06864746e+13 2.37831866e+13]
 [5.45981500e+01 1.18716009e+11 1.94720262e+13 2.90488497e+13
  7.89629602e+13 2.90488497e+13]
 [1.48413159e+02 1.44625706e+12 2.90488497e+13 1.75735300e+14
  1.06864746e+13 7.14486410e+13]
 [4.03428793e+02 3.93133430e+12 7.89629602e+13 1.58601345e+15
  1.17494766e+15 1.43879894e+14]
 [1.09663316e+03 1.06864746e+13 1.17798894e+14 2.88990493e+15
  1.17798894e+14 7.89629602e+13]
 [2.98095799e+03 1.58601345e+15 4.31123155e+15 6.43160170e+15
  1.06864746e+13 7.89629602e+13]
 [8.10308393e+03 8.70422638e+14 2.36605404e+15 7.10801915e+15
```

```
  5.83461743e+14 7.89629602e+13]
 [2.20264658e+04 3.93133430e+12 2.90488497e+13 8.68175420e+15
  3.52973785e+15 1.59011875e+14]]

Natural logarithm of each element:
[[0.         3.0155349  3.26956894 3.4339872  3.4657359  3.4339872 ]
 [0.69314718 3.06339092 3.12676054 3.40119738 3.44041809 3.45946629]
 [1.09861229 3.19047635 3.26575941 3.31054301 3.40119738 3.42751469]
 [1.38629436 3.23867845 3.42100001 3.4339872  3.4657359  3.4339872 ]
 [1.60943791 3.33220451 3.4339872  3.49042852 3.40119738 3.46260601]
 [1.79175947 3.36729583 3.4657359  3.55534806 3.54673969 3.48431229]
 [1.94591015 3.40119738 3.47815842 3.57234564 3.47815842 3.4657359 ]
 [2.07944154 3.55534806 3.58351894 3.59456877 3.40119738 3.4657359 ]
 [2.19722458 3.53805656 3.56671182 3.59731226 3.52636052 3.4657359 ]
 [2.30258509 3.36729583 3.4339872  3.60277676 3.57794789 3.48737508]]

Bitwise Operations:
Bitwise AND with 5:
[[1 4 0 5 0 5]
 [0 5 4 4 5 5]
 [1 0 0 1 4 4]
 [4 1 4 5 0 5]
 [5 4 5 0 4 5]
 [4 5 0 1 0 0]
 [5 4 0 1 0 0]
 [0 1 4 4 4 0]
 [1 0 1 4 0 0]
 [0 5 5 4 1 0]]

Bitwise OR with 3:
[[ 3 23 27 31 35 31]
 [ 3 23 23 31 31 31]
 [ 3 27 27 27 31 31]
 [ 7 27 31 31 35 31]
 [ 7 31 31 35 31 31]
 [ 7 31 35 35 35 35]
 [ 7 31 35 35 35 35]
 [11 35 39 39 31 35]
 [11 35 35 39 35 35]
 [11 31 31 39 35 35]]

Bitwise XOR with 2:
[[ 3 22 24 29 34 29]
 [ 0 23 20 28 29 29]
 [ 1 26 24 25 28 28]
 [ 6 27 28 29 34 29]
 [ 7 30 29 34 28 29]
 [ 4 31 34 33 32 34]
 [ 5 28 34 33 34 34]
 [10 33 38 38 28 34]
 [11 32 33 38 32 34]
 [ 8 31 29 38 33 34]]
```

3.Custom Sequence generation

Code:

```
import numpy as np

# Define the given array
arr = np.array([
    ['State Code', 'March', 'April', 'May', 'June', 'July'],
    ['1', '20.4', '26.3', '31', '32', '31'],
    ['2', '21.4', '22.8', '30', '31.2', '31.8'],
    ['3', '24.3', '26.2', '27.4', '30', '30.8'],
    ['4', '25.5', '30.6', '31', '32', '31'],
    ['5', '28', '31', '32.8', '30', '31.9'],
    ['6', '29', '32', '35', '34.7', '32.6'],
    ['7', '30', '32.4', '35.6', '32.4', '32'],
    ['8', '35', '36', '36.4', '30', '32'],
    ['9', '34.4', '35.4', '36.5', '34', '32'],
    ['10', '29', '31', '36.7', '35.8', '32.7']
])

# Extract the numerical values as a float array
num_arr = arr[1:].astype(float)

# Generate a custom sequence based on the given array
custom_sequence = np.linspace(0, 1, num=num_arr.shape[0])

print("Original Array:")
print(arr)
print()

print("Numerical Array:")
print(num_arr)
print()

print("Custom Sequence:")
print(custom_sequence)
```

Output:

```
Original Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]
```

```
Numerical Array:
[[ 1.   20.4 26.3 31.  32.  31. ]
 [ 2.   21.4 22.8 30.  31.2 31.8]
 [ 3.   24.3 26.2 27.4 30.  30.8]
 [ 4.   25.5 30.6 31.  32.  31. ]
 [ 5.   28.  31.  32.8 30.  31.9]
 [ 6.   29.  32.  35.  34.7 32.6]
 [ 7.   30.  32.4 35.6 32.4 32. ]
 [ 8.   35.  36.  36.4 30.  32. ]
 [ 9.   34.4 35.4 36.5 34.  32. ]
 [10.   29.  31.  36.7 35.8 32.7]]

Custom Sequence:
[0.         0.11111111 0.22222222 0.33333333 0.44444444 0.55555556
 0.66666667 0.77777778 0.88888889 1.        ]
```

5.Copying and viewing arrays

Code:

```python
import numpy as np

# Define the given array
arr = np.array([
    ['State Code', 'March', 'April', 'May', 'June', 'July'],
    ['1', '20.4', '26.3', '31', '32', '31'],
    ['2', '21.4', '22.8', '30', '31.2', '31.8'],
    ['3', '24.3', '26.2', '27.4', '30', '30.8'],
    ['4', '25.5', '30.6', '31', '32', '31'],
    ['5', '28', '31', '32.8', '30', '31.9'],
    ['6', '29', '32', '35', '34.7', '32.6'],
    ['7', '30', '32.4', '35.6', '32.4', '32'],
    ['8', '35', '36', '36.4', '30', '32'],
    ['9', '34.4', '35.4', '36.5', '34', '32'],
    ['10', '29', '31', '36.7', '35.8', '32.7']
])

# Create a copy of the array
arr_copy = arr.copy()

# View a portion of the array using indexing
view_arr = arr[1:4, 2:5] # Rows 1 to 3 (excluding the header) and
columns 2 to 4

print("Original Array:")
print(arr)
print()

print("Copy of the Array:")
print(arr_copy)
print()
print("View of a Portion of the Array:")
print(view_arr)
```

Output:

```
Original Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]

Copy of the Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]

View of a Portion of the Array:
[['26.3' '31' '32']
 ['22.8' '30' '31.2']
 ['26.2' '27.4' '30']]
```

6.Data Stacking , Searching, Sorting, Counting, Broadcasting.

Code:

```
import numpy as np

# Define the given array
arr = np.array([
    ['State Code', 'March', 'April', 'May', 'June', 'July'],
    ['1', '20.4', '26.3', '31', '32', '31'],
    ['2', '21.4', '22.8', '30', '31.2', '31.8'],
    ['3', '24.3', '26.2', '27.4', '30', '30.8'],
    ['4', '25.5', '30.6', '31', '32', '31'],
    ['5', '28', '31', '32.8', '30', '31.9'],
    ['6', '29', '32', '35', '34.7', '32.6'],
    ['7', '30', '32.4', '35.6', '32.4', '32'],
    ['8', '35', '36', '36.4', '30', '32'],
    ['9', '34.4', '35.4', '36.5', '34', '32'],
    ['10', '29', '31', '36.7', '35.8', '32.7']
])

# Data stacking
stacked_arr = np.vstack((arr[1:4], arr[6:9]))  # Vertically stack rows
1 to 3 and rows 6 to 8
concatenated_arr = np.concatenate((arr[1:4], arr[6:9]), axis=0) #
Concatenate along the row axis

# Searching
index = np.where(arr == '32') # Find indices where '32' is present in
the array

# Sorting
sorted_arr = np.sort(arr[1:], axis=0) # Sort rows (excluding the
header) along the column axis

# Counting
unique_vals, counts = np.unique(arr[1:, 2], return_counts=True) #
Count unique values in column 'April'

# Broadcasting
broadcasted_arr = arr[1:].astype(float) + 5 # Add 5 to each element in
the numerical portion of the array

print("Original Array:")
print(arr)
print()

print("Data Stacking:")
print("Vertically Stacked Array:")
print(stacked_arr)
print()
print("Concatenated Array:")
print(concatenated_arr)
print()

print("Searching:")
print("Indices of '32':")
print(index)
print()

print("Sorting:")
print("Sorted Array:")
print(sorted_arr)
```

```python
print()

print("Counting:")
print("Unique Values in 'April':")
print(unique_vals)
print("Counts:")
print(counts)
print()

print("Broadcasting:")
print("Array with 5 added to each element:")
print(broadcasted_arr)
```

Output:

```
Original Array:
[['State Code' 'March' 'April' 'May' 'June' 'July']
 ['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['4' '25.5' '30.6' '31' '32' '31']
 ['5' '28' '31' '32.8' '30' '31.9']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']
 ['9' '34.4' '35.4' '36.5' '34' '32']
 ['10' '29' '31' '36.7' '35.8' '32.7']]

Data Stacking:
Vertically Stacked Array:
[['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']]

Concatenated Array:
[['1' '20.4' '26.3' '31' '32' '31']
 ['2' '21.4' '22.8' '30' '31.2' '31.8']
 ['3' '24.3' '26.2' '27.4' '30' '30.8']
 ['6' '29' '32' '35' '34.7' '32.6']
 ['7' '30' '32.4' '35.6' '32.4' '32']
 ['8' '35' '36' '36.4' '30' '32']]

Searching:
Indices of '32':
(array([1, 4, 6, 7, 8, 9]), array([4, 4, 2, 5, 5, 5]))

Sorting:
Sorted Array:
[['1' '20.4' '22.8' '27.4' '30' '30.8']
 ['10' '21.4' '26.2' '30' '30' '31']
 ['2' '24.3' '26.3' '31' '30' '31']
 ['3' '25.5' '30.6' '31' '31.2' '31.8']
 ['4' '28' '31' '32.8' '32' '31.9']
 ['5' '29' '31' '35' '32' '32']
 ['6' '29' '32' '35.6' '32.4' '32']
 ['7' '30' '32.4' '36.4' '34' '32']
 ['8' '34.4' '35.4' '36.5' '34.7' '32.6']
 ['9' '35' '36' '36.7' '35.8' '32.7']]

Counting:
Unique Values in 'April':
['22.8' '26.2' '26.3' '30.6' '31' '32' '32.4' '35.4' '36']
Counts:
[1 1 1 1 2 1 1 1 1]

Broadcasting:
Array with 5 added to each element:
[[ 6.  25.4 31.3 36.  37.  36. ]
 [ 7.  26.4 27.8 35.  36.2 36.8]
 [ 8.  29.3 31.2 32.4 35.  35.8]
 [ 9.  30.5 35.6 36.  37.  36. ]
 [10.  33.  36.  37.8 35.  36.9]
 [11.  34.  37.  40.  39.7 37.6]
 [12.  35.  37.4 40.6 37.4 37. ]
 [13.  40.  41.  41.4 35.  37. ]
 [14.  39.4 40.4 41.5 39.  37. ]
 [15.  34.  36.  41.7 40.8 37.7]]
```