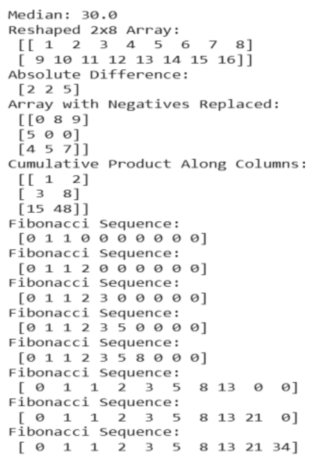
Experiment-4

Code :

```python
import numpy as np

import pandas as pd
# 4.a - Calculate the median
arr = np.array([10, 20, 30, 40, 50])
print("Median:", np.median(arr))

# 4.b - 4x4 array reshaped to 2x8
arr = np.arange(1, 17)
print("Reshaped 2x8 Array:\n", arr.reshape(2, 8))

  # 4.c - Absolute difference between arrays
arr1 = np.array([5, 10, 15])
arr2 = np.array([3, 8, 20])
print("Absolute Difference:\n", np.abs(arr1 - arr2))

  # 4.d - Replace negative values with 0 in 3x3 array
arr = np.random.randint(-10, 10, (3, 3))
arr[arr < 0] = 0
print("Array with Negatives Replaced:\n", arr)

  # 4.e - Cumulative product along columns
arr = np.array([[1, 2], [3, 4], [5, 6]])
print("Cumulative Product Along Columns:\n", np.cumprod(arr, axis=0))

  # 4.f - First 10 Fibonacci numbers using NumPy
fib = np.zeros(10, dtype=int)
fib[1] = 1
for i in range(2, 10):
  fib[i] = fib[i - 1] + fib[i - 2]
  print("Fibonacci Sequence:\n", fib)
```

Output :



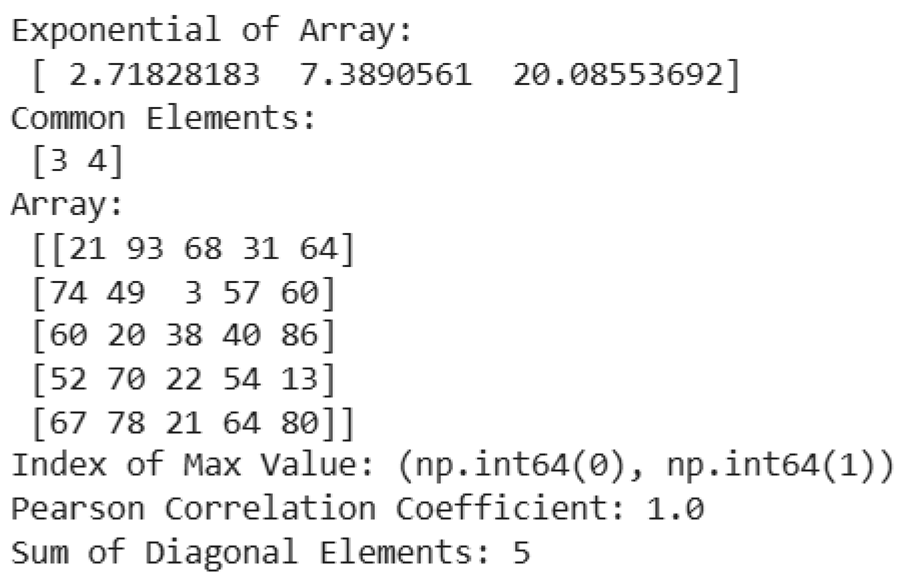
```
Median: 30.0
Reshaped 2x8 Array:
 [[ 1  2  3  4  5  6  7  8]
 [ 9 10 11 12 13 14 15 16]]
Absolute Difference:
 [2 2 5]
Array with Negatives Replaced:
 [[0 8 9]
 [5 0 0]
 [4 5 7]]
Cumulative Product Along Columns:
 [[ 1  2]
 [ 3  8]
 [15 48]]
Fibonacci Sequence:
 [0 1 1 0 0 0 0 0 0 0]
Fibonacci Sequence:
 [0 1 1 2 0 0 0 0 0 0]
Fibonacci Sequence:
 [0 1 1 2 3 0 0 0 0 0]
Fibonacci Sequence:
 [0 1 1 2 3 5 0 0 0 0]
Fibonacci Sequence:
 [0 1 1 2 3 5 8 0 0 0]
Fibonacci Sequence:
 [ 0  1  1  2  3  5  8 13  0  0]
Fibonacci Sequence:
 [ 0  1  1  2  3  5  8 13 21  0]
Fibonacci Sequence:
 [ 0  1  1  2  3  5  8 13 21 34]
```

Experiment-5

Code :

```
# 5.a - Exponential of each element
arr = np.array([1, 2, 3])
print("Exponential of Array:\n", np.exp(arr))

# 5.b - Common elements between two arrays
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([3, 4, 5, 6])
print("Common Elements:\n", np.intersect1d(arr1, arr2))

# 5.c - 5x5 array and index of max value
arr = np.random.randint(0, 100, (5, 5))
index = np.unravel_index(np.argmax(arr), arr.shape)
print("Array:\n", arr)
print("Index of Max Value:", index)

# 5.d - Pearson correlation coefficient
arr1 = np.array([10, 20, 30])
arr2 = np.array([15, 25, 35])
print("Pearson Correlation Coefficient:", np.corrcoef(arr1, arr2)[0, 1])

# 5.e - Sum of diagonal elements
matrix = np.array([[1, 2], [3, 4]])
print("Sum of Diagonal Elements:", np.trace(matrix))
```

Output :



```
Exponential of Array:
 [ 2.71828183  7.3890561  20.08553692]
Common Elements:
 [3 4]
Array:
 [[21 93 68 31 64]
 [74 49  3 57 60]
 [60 20 38 40 86]
 [52 70 22 54 13]
 [67 78 21 64 80]]
Index of Max Value: (np.int64(0), np.int64(1))
Pearson Correlation Coefficient: 1.0
Sum of Diagonal Elements: 5
```

Experiment-6

Code :

```
# 6.a - Read CSV into DataFrame
import pandas as pd
from google.colab import files

uploaded = files.upload()

df = pd.read_csv("titanic.csv")

print("Titanic CSV Contents:")

# 6.b - Display first 5 rows
df = pd.DataFrame({'A': range(10), 'B': range(10, 20)})
print("First 5 Rows:\n", df.head())

# 6.c - Create DataFrame from dictionary
data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [23, 25, 22]}
df = pd.DataFrame(data)
print("DataFrame from Dictionary:\n", df)

# 6.d - Average of a specific column
df = pd.DataFrame({'Score': [80, 90, 85, 95]})
print("Average of 'Score':", df['Score'].mean())

# 6.e - Max value in a column
df = pd.DataFrame({'Marks': [70, 85, 60, 90]})
print("Max of 'Marks':", df['Marks'].max())

# 6.f - Random DataFrame with 5 rows & 3 columns
data = np.random.rand(5, 3)
df = pd.DataFrame(data, columns=['A', 'B', 'C'])
print("Random DataFrame:\n", df)
```

Output :

Choose Files titanic.csv

- **titanic.csv**(text/csv) - 67808 bytes, last modified: 12/4/2025 - 100% done

```
Saving titanic.csv to titanic.csv
Titanic CSV Contents:
First 5 Rows:
   A   B
0  0  10
1  1  11
2  2  12
3  3  13
4  4  14
DataFrame from Dictionary:
      Name  Age
0    Alice   23
1      Bob   25
2  Charlie   22
Average of 'Score': 87.5
Max of 'Marks': 90
Random DataFrame:
          A         B         C
0  0.686460  0.492368  0.804302
1  0.933059  0.976990  0.155729
2  0.382309  0.370291  0.488393
3  0.663099  0.869508  0.558561
4  0.455677  0.955339  0.765634
```