Task 11

Exercise 1: Mean

Create a numpy array of 10 random integers between 1 and 100. Calculate the mean of the array using numpy's `mean()` function.

In [ ]:
```
import numpy as np

#random array of 10 integers between 1 and 100
arr = np.random.randint(1,100,10)
print(arr)
np.median(arr)
```

```
[15 53 25 65 51 27 79 56 94 23]
```

Out[ ]:
```
52.0
```

Exercise 2: Median

Create a numpy array of 10 random integers between 1 and 100. Calculate the median of the array using numpy's `median()` function.

In [ ]:
```
arr=np.random.randint(1,100,10)
print(arr)
np.median(arr)
```

```
[11 24 86 45 24 62 50  3 42 25]
```

Out[ ]:
```
33.5
```

Exercise 3: Mode

Create a numpy array of 10 random integers between 1 and 100. Calculate the mode of the array using numpy.

In [ ]:
```
arr=np.random.randint(1,100,10)
vals, counts=np.unique(arr,return_counts=True)
print(arr)
print(vals[np.argmax(counts)])
```

```
[34 53 96 61 61 67 19 80 91 24]
61
```

Exercise 4: Percentiles

Create a numpy array of 20 random integers between 1 and 100. Calculate the 10th, 50th, and 90th percentile of the array using numpy's `percentile()` function.

In [ ]:
```
arr=np.random.randint(1,100,20)
print(arr)
print("25th percentile:", np.percentile(arr,25))
print("50th percentile:", np.percentile(arr,50))
print("90th percentile:", np.percentile(arr,90))
```

```
[ 5 80 69 65  8  8 74 79 97 90 82 73  5 17 59 80 54 42 59 27]
25th percentile: 24.5
50th percentile: 62.0
90th percentile: 82.80000000000001
```

Exercise 5: Rank

Create a numpy array of 20 random integers between 1 and 100. Calculate the rank of each number in the array using numpy's `rank()` function.

In [ ]:
```
arr=np.random.randint(1,100,20)
print(arr)
rank=arr.argsort()
print(rank)
```

```
[77 21 81  1 52 11 49 64 68 71 81 63 34 87  1 86 15 75 25 96]
[ 3 14  5 16  1 18 12  6  4 11  7  8  9 17  0 10  2 15 13 19]
```

Exercise 6: Probability Distribution

Generate 1000 random numbers from a Poisson distribution with a lambda value of `5` using numpy's random module. Calculate the mean and standard deviation of the generated numbers using numpy's `mean()` and `std()` functions.

In [ ]:
```
values=np.random.poisson(lam=5, size=1000)
mean=np.mean(values)
stdDev=np.std(values)
print("Mean:", mean)
print("Standard Deviation:", stdDev)
```

```
Mean: 5.031
Standard Deviation: 2.308254535357832
```

Exercise 7: Linear Algebra

Exercise 7a: `dot()` function

Create two 2-dimensional NumPy arrays, `a` and `b`, with shapes `(2, 3)` and `(3, 2)` respectively, and fill them with random numbers. Then, calculate the matrix product of `a` and `b` using NumPy's `dot()` function.

In [ ]:
```
a=np.random.randint(0,10,(2,3))
b=np.random.randint(0,10,(3,2))
print("Matrix a:\n", a, "\n")
```

```
print("Matrix b:\n", b, "\n")
print("Dot product:\n", np.dot(a,b))
```

```
Matrix a:
 [[0 1 2]
 [0 2 1]]

Matrix b:
 [[6 3]
 [7 7]
 [4 5]]

Dot product:
 [[15 17]
 [18 19]]
```

Exercise 7b: `sum()` function

Create a 3-dimensional NumPy array, `a`, with shape `(2, 3, 4)`, and fill it with random numbers. Then, use NumPy's `sum()` function to calculate the sum of the elements in the second axis of `a`.

In [ ]:
```
a=np.random.randint(0,10,(2,3,4))
print("Matrix a:\n", a, "\n")
print("Sum of second axis of a:\n", np.sum(a,axis=1))
```

```
Matrix a:
 [[[9 3 7 5]
  [6 4 1 2]
  [8 1 0 2]]

 [[5 8 9 5]
  [5 1 3 2]
  [8 0 4 6]]]

Sum of second axis of a:
 [[23  8  8  9]
 [18  9 16 13]]
```

Exercise 7c: Determinant

Create a 2-dimensional NumPy array, `a`, with shape `(3, 3)`, and fill it with random numbers. Then, use NumPy's `linalg.det()` function to calculate the determinant of `a`.

In [ ]:
```
a=np.random.randint(0,10,(3,3))
print("Matrix a:\n", a, "\n")
print("Determinant of a:\n", np.linalg.det(a))
```

```
Matrix a:
 [[7 0 4]
 [1 0 1]
 [1 7 9]]

Determinant of a:
 -21.0
```

Exercise 7d: Inverse

Create a 2-dimensional NumPy array, `a`, with shape `(3, 3)`, and fill it with random numbers. Then, use NumPy's `linalg.inv()` function to calculate the inverse of `a`.

In [ ]:
```
a=np.random.randint(0,10,(3,3))
print("Matrix a:\n", a, "\n")
print("Inverse of a:\n", np.linalg.inv(a))
```

```
Matrix a:
 [[2 5 7]
 [0 2 2]
 [0 1 6]]

Inverse of a:
 [[ 0.5  -1.15 -0.2 ]
 [ 0.    0.6  -0.2 ]
 [ 0.   -0.1   0.2 ]]
```

Exercise 7e: Trace

Create a 2-dimensional NumPy array, `a`, with shape `(3, 3)`, and fill it with random numbers. Then, use NumPy's `trace()` function to calculate the trace of `a`.

In [ ]:
```
a=np.random.randint(0,10,(3,3))
print("Matrix a:\n", a, "\n")
print("Trace of a:\n", np.trace(a))
```

```
Matrix a:
 [[5 0 6]
 [2 5 0]
 [7 2 2]]

Trace of a:
 12
```

Exercise 7f: Eigenvalues

Create a 2-dimensional NumPy array, `a`, with shape `(3, 3)`, and fill it with random numbers. Then, use NumPy's `linalg.eig()` function to calculate the eigenvalues of `a`.

In [ ]:
```
a=np.random.randint(0,10,(3,3))
print("Matrix a:\n", a, "\n")
print("Eigenvalues of a:\n", np.linalg.eig(a))
```

```
Matrix a:
 [[7 3 9]
 [2 6 9]
 [0 3 2]]

Eigenvalues of a:
 (array([11.4695323,  4.6993204, -1.1688527]), array([[-0.78038294,  0.96674568, -
0.44055962],
       [-0.59610288, -0.17105703, -0.65191736],
       [-0.18884868, -0.19011122,  0.61717986]]))
```