# assignment

March 6, 2024

## 1 LAB 6 Assignment

### 1.0.1 1. Reverse the columns of 2d-array.

[ ]:
```
import numpy as np
# Create a 2D array
arr = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

# Reverse the columns
reversed_arr = np.flip(arr, axis=1)

print(reversed_arr)
```

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

### 1.0.2 2. Create a 2d-array containing random floats between 5 and 10

[ ]:
```
import numpy as np

# Create a 2D array of shape (3, 3) with random floats between 5 and 10
random_arr = np.random.uniform(5, 10, size=(3, 3))

print(random_arr)
```

```
[[7.78685857 5.21227678 8.06400359]
 [7.64024059 7.43540211 5.70082165]
 [5.9197782  7.24431083 9.92569624]]
```

### 1.0.3 3. Find the percentile scores of a numpy array

[ ]:
```
percentiles = np.percentile(arr, [3, 5, 7])
print(percentiles)
```

```
[1.24 1.4  1.56]
```

### 1.0.4 4. Find the position of a missing value in a numpy array

[ ]:
```
# Importing the NumPy library with an alias 'np'
import numpy as np

# Creating a NumPy array 'nums' with provided values, including NaN (Not a␣
 ↪Number)
nums = np.array([[3, 2, np.nan, 1],
                 [10, 12, 10, 9],
                 [5, np.nan, 1, np.nan]])
# Printing a message indicating the original array 'nums'
print("Original array:")
print(nums)
# Printing a message indicating finding the missing data (NaN) in the array␣
 ↪using np.isnan()
# This function returns a boolean array of the same shape as 'nums', where True␣
 ↪represents NaN values
print("\nFind the missing data of the said array:")
print(np.isnan(nums))
```

```
Original array:
[[ 3.  2. nan  1.]
 [10. 12. 10.  9.]
 [ 5. nan  1. nan]]

Find the missing data of the said array:
[[False False  True False]
 [False False False False]
 [False  True False  True]]
```

### 1.0.5 5. Sort a 2d-array by column.

[ ]:
```
import numpy as np

# Create a 2D array
arr = np.array([[9, 8, 7],
                [6, 5, 4],
                [3, 2, 1]])

# Sort the array by column
sorted_arr = arr[:, np.argsort(arr[0])]
print(sorted_arr)
```

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

### 1.0.6 6. Find the most frequent value in a numpy array

[ ]:
```
import numpy as np

# Create a numpy array of single dimension
single_dim_arr = np.array([1, 2, 3, 2, 1, 2, 3, 3, 3])

# Find the most frequent value in the numpy array
most_frequent_value = np.argmax(np.bincount(single_dim_arr))

print(most_frequent_value)
```

```
3
```

### 1.0.7 7. Convert an array of arrays into a flat 1-d array

[ ]:
```
import numpy as np

# Create an array of arrays
array_of_arrays = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Convert the array of arrays into a flat 1-d array
flat_array = array_of_arrays.flatten()

print(flat_array)
```

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

### 1.0.8 8. Consider the numerical data set to perform some statistical and mathematical operations like mean, median, mode, standard deviation, eculidaian distance, manhatton distance etc using numpy library

[ ]:
```
import numpy as np
import scipy.stats as sp
# Consider a numerical dataset as a numpy array
data = np.array([1,2,3,4,5,6,7,8,9,10,11,12,1,2,3,4,4,4,2,5])

# Calculate the mean of the dataset
mean = np.mean(data)

# Calculate the median of the dataset
median = np.median(data)

# Calculate the mode of the dataset
mode = sp.mode(data)

# Calculate the standard deviation of the dataset
std_dev = np.std(data)
```

```python

# Calculate the Euclidean distance between two points in the dataset
point1 = data[0]
point2 = data[1]
euclidean_distance = np.linalg.norm(point1 - point2)

# Calculate the Manhattan distance between two points in the dataset
manhattan_distance = np.sum(np.abs(point1 - point2))

# Print the results
print("Mean:", mean)
print("Median:", median)
print("Mode:", mode)
print("Standard Deviation:", std_dev)
print("Euclidean Distance:", euclidean_distance)
print("Manhattan Distance:", manhattan_distance)
```

```
Mean: 5.15
Median: 4.0
Mode: ModeResult(mode=4, count=4)
Standard Deviation: 3.2446109165815242
Euclidean Distance: 1.0
Manhattan Distance: 1
```