new

April 15, 2023

[39]:
```
#importing Numpy
import numpy as np
import pandas as pd
```

0.0.1 1) You have an array of shape (5, 5). Using NumPy, create a new array that contains the diagonal elements of the original array.

[4]:
```
# create a (5, 5) array
arr = np.array([['Mahesh Babu', 'Jr. NTR', 'Allu Arjun', 'Ram Charan',␣
 ↪'Prabhas',],
                ['Nani', 'Vijay Deverakonda', 'Nagarjuna', 'Ravi Teja',␣
 ↪'Venkatesh'],
                ['Pawan Kalyan', 'Chiranjeevi', 'Balakrishna', 'Naga␣
 ↪Chaitanya', 'Varun Tej',],
                ['Sudheer Babu', 'Nikhil Siddharth', 'Bellamkonda Sreenivas',␣
 ↪'Sai Dharam Tej', 'Sumanth'],
                ['Sundeep Kishan', 'Sharwanand', 'Karthikeya', 'Kalyan Ram',␣
 ↪'Nithiin']])

#diagonal elements
diagonal_arr = arr.diagonal()

print(diagonal_arr)
```

```
['Mahesh Babu' 'Vijay Deverakonda' 'Balakrishna' 'Sai Dharam Tej'
 'Nithiin']
```

[5]:
```
arr = np.array([[1,0,0,0,0],
                [0,1,0,0,0],
                [0,0,1,0,0],
                [0,0,0,1,0],
                [0,0,0,0,1]])
diagonal_arr = np.diag(arr)

print('Diagonal Array is',diagonal_arr)
```

```
Diagonal Array is [1 1 1 1 1]
```

0.0.2 2) You have two arrays of shape (3, 3) and (3, 1). Using NumPy, perform matrix multiplication of these arrays.

[6]:
```
#3x3 array
arr1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
#3x1 array
arr2 = np.array([[2], [3], [4]])
#np.matmul() for multiplying two arrays is used here
product = np.matmul(arr1, arr2)
product
```

[6]:
```
array([[20],
       [47],
       [74]])
```

[7]:
```
#3x3 array
arr1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
#3x1 array
arr2 = np.array([[2], [3], [4]])
#np.dot() for multiplying two arrays is used here
product = np.dot(arr1, arr2)
print('Product :')
print(product)
```

```
Product :
[[20]
 [47]
 [74]]
```

0.0.3 4) You have two arrays of shape (3, 3) and (3, 4). Using NumPy, concatenate these arrays along the first axis

[9]:
```
# create two 3x3 and 3x4 arrays
arr1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
arr2 = np.array([[10, 11, 12, 13], [14, 15, 16, 17], [18, 19, 20, 21]])

# concatenate the arrays along the first axis
result = np.concatenate((arr1, arr2), axis=1)

# print the result
print('After concatenate:')
print(result)
```

```
After concatenate:
[[ 1  2  3 10 11 12 13]
 [ 4  5  6 14 15 16 17]
 [ 7  8  9 18 19 20 21]]
```

0.0.4 5) You have an array of shape (2, 3, 4). Using NumPy, reshape it into an array of shape (2, 4, 3).

[10]:
```
# shape: (2, 3, 4)
chocolate = np.array([
    [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
    [[13, 14, 15, 16], [17, 18, 19, 20], [21, 22, 23, 24]]
])

# print the original array
print("Original array:")
print(chocolate)

# reshape the array
# new shape: (2, 4, 3)
new_chocolate = np.transpose(chocolate, (0, 2, 1))

# print the reshaped array
print("Reshaped array:")
print(new_chocolate)
```

```
Original array:
[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
Reshaped array:
[[[ 1  5  9]
  [ 2  6 10]
  [ 3  7 11]
  [ 4  8 12]]

 [[13 17 21]
  [14 18 22]
  [15 19 23]
  [16 20 24]]]
```

0.0.5 6) You have an array of shape (4, 4). Using NumPy, split it into two equal parts horizontally

[11]:
```
#4x4 array
arr = np.array([[1, 2, 3, 4],
                [5, 6, 7, 8],
                [9, 10, 11, 12],
                [13, 14, 15, 16]])
```

```
#split the array horizontally into two equal parts
new_arr = np.split(arr, 2)

# print the original and new arrays
print("Original array:\n", arr)
print("\nNew array:\n", new_arr)
```

```
Original array:
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

New array:
 [array([[1, 2, 3, 4],
       [5, 6, 7, 8]]), array([[ 9, 10, 11, 12],
       [13, 14, 15, 16]])]
```

0.0.6 10) You have a data frame containing the names, ages and salaries of employees. Using Pandas, create a new data frame that contains only the names and salaries of employees who are older than 30 years

[20]:
```
#a sample data frame
df = pd.DataFrame({
    'name': ['Kedhar','Sathivk','Hari','Veda','Vinay','Adithya'],
    'age': [29, 30, 28, 31, 45, 50],
    'salary': [50000, 80000, 65000, 45000, 95000, 55000]
})

# filter the data frame to keep only employees older than 30
df_filtered = df.loc[df['age'] > 30, ['name', 'salary']]
df
df_filtered
```

[20]:
```
      name  salary
3     Veda   45000
4    Vinay   95000
5  Adithya   55000
```

0.0.7 11) You have two data frames containing the names, ages and genders of students from two different classes. Using Pandas, merge these data frames on the basis of the names of the students and add a new column that contains the average age of the students from both classes.

[55]:
```
# Load the data frames
class1 = pd.DataFrame({'name': ['Kedhar', 'Hari', 'Sita'], 'age': [21, 20, 19],␣
 ↪'gender': ['M', 'M', 'F']})
class2 = pd.DataFrame({'name': ['Hari', 'Kedhar', 'Sita'], 'age': [18, 22, 20],␣
 ↪'gender': ['F', 'M', 'F']})

# Merge the data frames on the 'name' column
merged_df = pd.merge(class1, class2, on='name')

# Compute the average age of the students from both classes
merged_df['avg_age'] = (merged_df['age_x'].fillna(0) + merged_df['age_y'].
 ↪fillna(0)) / 2

# Rename the columns to be more descriptive
merged_df = merged_df.rename(columns={'age_x': 'age_class1', 'age_y':␣
 ↪'age_class2'})

# Drop the redundant 'gender' column

# Print the merged data frame
print(merged_df)
```

```
     name  age_class1 gender_x  age_class2 gender_y  avg_age
0  Kedhar          21        M          22        M     21.5
1    Hari          20        M          18        F     19.0
2    Sita          19        F          20        F     19.5
```

0.0.8 12) You have a data frame containing the names and grades of students. Using Pandas, group the data frame by grades and calculate the mean, median and standard deviation of the grades for each group.

[49]:
```
import pandas as pd

# create a sample DataFrame
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', 'Dave', 'Eve', 'Frank'],
    'Grade': [90, 85, 75, 90, 80, 95]
})
print(pd.DataFrame(df))
# group the DataFrame by grades and calculate the mean, median, and standard␣
 ↪deviation
```

```
grouped_df = df.groupby('Grade').agg({'Name': list, 'Grade': ['mean', 'median',␣
 ↪'std']})

# rename the columns
grouped_df.columns = ['Students', 'Mean Grade', 'Median Grade', 'Std Deviation']

# print the results
print(grouped_df)
```

```
      Name  Grade
0    Alice     90
1      Bob     85
2  Charlie     75
3     Dave     90
4      Eve     80
5    Frank     95
             Students  Mean Grade  Median Grade  Std Deviation
Grade
75          [Charlie]        75.0          75.0            NaN
80              [Eve]        80.0          80.0            NaN
85              [Bob]        85.0          85.0            NaN
90     [Alice, Dave]        90.0          90.0            0.0
95            [Frank]        95.0          95.0            NaN
```

[ ]: