sagar-250-lab9

September 16, 2023

[1]:
```
#Q1. Program to distinguish between Array Indexing and Fancy Indexing.

import numpy as np

#Index array
array = np.array([1, 2, 3, 4, 5])
second_element = array[1]
second_to_last = array[1:]

print("Index array")
print("The second element is: ",second_element)
print("Second element to last element: ", second_to_last)
print()

#Fancy array
array = np.array([10, 20, 30, 40, 50])
second_and_fourth_elements = array[[1, 3]]
boolean_array = np.array([True, False, True, False, False, True])
print("Fancy array")
print("Second and fourth element: ", second_and_fourth_elements)
print("Boolean array: ", boolean_array)
```

```
Index array
The second element is:  2
Second element to last element:  [2 3 4 5]

Fancy array
Second and fourth element:  [20 40]
Boolean array:  [ True False  True False False  True]
```

[2]:
```
# Q2. Execute the 2D array Slicing.

import numpy as np

# Creating a sample 2D NumPy array
arr_2d = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
```

```
# Slicing the 2D array
subarray_1 = arr_2d[0:2, 1:3]  # Rows 0 to 1 (exclusive) and Columns 1 to 2 (exclusive)
subarray_2 = arr_2d[:, 1]  # All rows and Column 1
subarray_3 = arr_2d[1, :]  # Row 1 and All columns

print("Original 2D Array:")
print(arr_2d)
print("\nSliced Subarrays:")
print("Subarray 1:")
print(subarray_1)
print("Subarray 2:")
print(subarray_2)
print("Subarray 3:")
print(subarray_3)
```

```
Original 2D Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Sliced Subarrays:
Subarray 1:
[[2 3]
 [5 6]]
Subarray 2:
[2 5 8]
Subarray 3:
[4 5 6]
```

[3]:
```
# Q3. Create the 5-Dimensional arrays using 'ndmin'.

import numpy as np

# Creating a 5-dimensional array with ndmin
arr_5d = np.array([1, 2, 3], ndmin=5)

# Checking the shape of the 5-dimensional array
print("Shape of the 5-dimensional array:", arr_5d.shape)

# Printing the 5-dimensional array
print("5-dimensional array:")
print(arr_5d)
```

```
Shape of the 5-dimensional array: (1, 1, 1, 1, 3)
5-dimensional array:
```

```
[[[[[1 2 3]]]]]
```

[4]:
```
#Q4. Reshape the array from 1-D to 2-D array.

import numpy as np

# Creating a 1-D array
arr_1d = np.array([1, 2, 3, 4, 5, 6])

# Reshaping the 1-D array to a 2-D array
arr_2d = arr_1d.reshape((2, 3))  # Specifying the desired shape (2 rows, 3 columns)

# Alternatively, you can use np.reshape() function:
arr_2d = np.reshape(arr_1d, (2, 3))

# Print the original and reshaped arrays
print("Original 1-D array:")
print(arr_1d)

print("\nReshaped 2-D array:")
print(arr_2d)
```

```
Original 1-D array:
[1 2 3 4 5 6]

Reshaped 2-D array:
[[1 2 3]
 [4 5 6]]
```

[5]:
```
#Q5. Perform the Stack functions in Numpy arrays - Stack(), hstack(), vstack(), and dstack().

import numpy as np

# Creating sample arrays
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# np.stack(): Stacking along a new axis
stacked_axis0 = np.stack((arr1, arr2), axis=0)
stacked_axis1 = np.stack((arr1, arr2), axis=1)

print("np.stack() along axis 0:")
print(stacked_axis0)
print("\nnp.stack() along axis 1:")
print(stacked_axis1)
```

```
# np.hstack(): Stacking horizontally
hstacked = np.hstack((arr1, arr2))
print("\nnp.hstack():")
print(hstacked)

# np.vstack(): Stacking vertically
vstacked = np.vstack((arr1, arr2))
print("\nnp.vstack():")
print(vstacked)

# Creating 2D arrays
arr3 = np.array([[7], [8], [9]])
arr4 = np.array([[10], [11], [12]])

# np.dstack(): Stacking along the third axis (depth-wise)
dstacked = np.dstack((arr3, arr4))
print("\nnp.dstack():")
print(dstacked)
```

```
np.stack() along axis 0:
[[1 2 3]
 [4 5 6]]

np.stack() along axis 1:
[[1 4]
 [2 5]
 [3 6]]

np.hstack():
[1 2 3 4 5 6]

np.vstack():
[[1 2 3]
 [4 5 6]]

np.dstack():
[[[ 7 10]]

 [[ 8 11]]

 [[ 9 12]]]
```

[6]:
```
#Q6. Perform the searchsort method in Numpy array.

import numpy as np
```

```
# Create a NumPy array
arr = np.array([3, 1, 2, 5, 4])

# Sort the array
sorted_arr = np.sort(arr)

print("Original array:")
print(arr)
print("Sorted array:")
print(sorted_arr)
```

```
Original array:
[3 1 2 5 4]
Sorted array:
[1 2 3 4 5]
```

[7]:
```
# Q7. Create Numpy Structured array using your domain features.

import numpy as np

# Defining the data types for blog application website features
feature_dtype = np.dtype([
    ('website_name', 'U50'),      # Website name as a Unicode string of up to␣
 ↪50 characters
    ('url', 'U100'),              # Website URL as a Unicode string of up to␣
 ↪100 characters
    ('monthly_visitors', 'i8'),   # Monthly visitors as a 64-bit integer
    ('number_of_blogs', 'i4'),    # Number of blogs on the website as a 32-bit␣
 ↪integer
    ('user_registration', bool)   # Whether user registration is allowed␣
 ↪(boolean)
])

# Creating an empty structured array with the defined data type
website_features = np.array([], dtype=feature_dtype)

# Adding website features to the structured array
website1 = ('BlogSite1', 'https://www.blogsite1.com', 1000000, 5000, True)
website2 = ('BlogSite2', 'https://www.blogsite2.com', 500000, 3000, True)
website3 = ('BlogSite3', 'https://www.blogsite3.com', 2000000, 10000, False)

website_features = np.array([website1, website2, website3], dtype=feature_dtype)

# Accessing and manipulating the structured array
print("Structured Array:")
print(website_features)
```

```
# Accessing individual features
print("\nFirst Website Feature:")
print("Website Name:", website_features['website_name'][0])
print("Website URL:", website_features['url'][0])
print("Monthly Visitors:", website_features['monthly_visitors'][0])
print("Number of Blogs:", website_features['number_of_blogs'][0])
print("User Registration Allowed:", website_features['user_registration'][0])
```

```
Structured Array:
[('BlogSite1', 'https://www.blogsite1.com', 1000000,  5000,  True)
 ('BlogSite2', 'https://www.blogsite2.com',  500000,  3000,  True)
 ('BlogSite3', 'https://www.blogsite3.com', 2000000, 10000, False)]

First Website Feature:
Website Name: BlogSite1
Website URL: https://www.blogsite1.com
Monthly Visitors: 1000000
Number of Blogs: 5000
User Registration Allowed: True
```

[9]:
```
# Q8. Create Data frame using List and Dictionary.

import pandas as pd

# Creating a list of data
data_list = [
    ['Alice', 25],
    ['Bob', 30],
    ['Charlie', 35],
    ['David', 40]
]

# Creating a DataFrame from the LIST
df_list = pd.DataFrame(data_list, columns=['Name', 'Age'])

# Displaying the DataFrame
print(df_list)
print()

# Create a dictionary of data
data_dict = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 30, 35, 40]
}

# Create a DataFrame from the DICTIONARY
df_dict = pd.DataFrame(data_dict)
```

```
# Display the DataFrame
print(df_dict)
```

```
      Name  Age
0    Alice   25
1      Bob   30
2  Charlie   35
3    David   40

      Name  Age
0    Alice   25
1      Bob   30
2  Charlie   35
3    David   40
```

[11]:
```
# Q9. Create Data frame on your Domain area and perform the following␣
 ↪operations to find and eliminate the
# missing data from the dataset.
# • isnull()
# • notnull()
# • dropna()
# • fillna()
# • replace()
# • interpolate()

import pandas as pd
import numpy as np

# Creating a sample DataFrame
data = {
    'Post_ID': [1, 2, 3, 4, 5],
    'Title': ['Post 1', 'Post 2', np.nan, 'Post 4', 'Post 5'],
    'Author': ['Author A', 'Author B', 'Author C', np.nan, 'Author E'],
    'Content': ['Content 1', np.nan, 'Content 3', 'Content 4', 'Content 5'],
    'Date_Published': ['2023-01-01', '2023-02-15', '2023-03-10', '2023-04-20',␣
 ↪'2023-05-30']
}

df = pd.DataFrame(data)

# Displaying the DataFrame
print("Original DataFrame:")
print(df)

# isnull()
# Identifying missing values
```

```
missing_data = df.isnull()
print("\nMissing Data isnull():")
print(missing_data)

# notnull()
# Identifying not null values
notnull_data = df.notnull()
print("\nNot null values notnull();")
print(notnull_data)

# dropna()
# Removing rows with missing values
df_clean = df.dropna()
print("\nDataFrame after dropna():")
print(df_clean)

# fillna()
# Replacing missing values with a default value
df_filled = df.fillna('No Data')
print("\nDataFrame after fillna():")
print(df_filled)

# replace()
# Replace specific values
df_replaced = df.replace('Author C', 'Author D')
print("\nDataFrame after replace():")
print(df_replaced)

# interpolate
# Interpolate missing values
df_interpolated = df.interpolate()
print("\nDataFrame after interpolate():")
print(df_interpolated)
```

```
Original DataFrame:
   Post_ID   Title    Author    Content Date_Published
0        1  Post 1  Author A  Content 1     2023-01-01
1        2  Post 2  Author B        NaN     2023-02-15
2        3     NaN  Author C  Content 3     2023-03-10
3        4  Post 4       NaN  Content 4     2023-04-20
4        5  Post 5  Author E  Content 5     2023-05-30

Missing Data isnull():
   Post_ID  Title  Author  Content  Date_Published
0    False  False   False    False           False
1    False  False   False     True           False
2    False   True   False    False           False
```

```
3    False  False    True    False           False
4    False  False   False    False           False

Not null values notnull();
   Post_ID  Title  Author  Content  Date_Published
0     True   True    True     True            True
1     True   True    True    False            True
2     True  False    True     True            True
3     True   True   False     True            True
4     True   True    True     True            True

DataFrame after dropna():
   Post_ID   Title    Author    Content Date_Published
0        1  Post 1  Author A  Content 1     2023-01-01
4        5  Post 5  Author E  Content 5     2023-05-30

DataFrame after fillna():
   Post_ID    Title    Author    Content Date_Published
0        1   Post 1  Author A  Content 1     2023-01-01
1        2   Post 2  Author B    No Data     2023-02-15
2        3  No Data  Author C  Content 3     2023-03-10
3        4   Post 4   No Data  Content 4     2023-04-20
4        5   Post 5  Author E  Content 5     2023-05-30

DataFrame after replace():
   Post_ID   Title    Author    Content Date_Published
0        1  Post 1  Author A  Content 1     2023-01-01
1        2  Post 2  Author B        NaN     2023-02-15
2        3     NaN  Author D  Content 3     2023-03-10
3        4  Post 4       NaN  Content 4     2023-04-20
4        5  Post 5  Author E  Content 5     2023-05-30

DataFrame after interpolate():
   Post_ID   Title    Author    Content Date_Published
0        1  Post 1  Author A  Content 1     2023-01-01
1        2  Post 2  Author B        NaN     2023-02-15
2        3     NaN  Author C  Content 3     2023-03-10
3        4  Post 4       NaN  Content 4     2023-04-20
4        5  Post 5  Author E  Content 5     2023-05-30

C:\Users\DELL\AppData\Local\Temp\ipykernel_3572\822406727.py:60: FutureWarning:
DataFrame.interpolate with object dtype is deprecated and will raise in a future
version. Call obj.infer_objects(copy=False) before interpolating instead.
  df_interpolated = df.interpolate()
```

[12]:
```
# Q10. Perform the Hierarchical Indexing in the above created dataset.

import pandas as pd
```

```
import numpy as np

# Creating a sample DataFrame
data = {
    'Post_ID': [1, 2, 3, 4, 5],
    'Title': ['Post 1', 'Post 2', np.nan, 'Post 4', 'Post 5'],
    'Author': ['Author A', 'Author B', 'Author C', np.nan, 'Author E'],
    'Content': ['Content 1', np.nan, 'Content 3', 'Content 4', 'Content 5'],
    'Date_Published': ['2023-01-01', '2023-02-15', '2023-03-10', '2023-04-20',␣
 ↪'2023-05-30']
}

df = pd.DataFrame(data)

# Creating hierarchical indexing
df.set_index(['Author', 'Post_ID'], inplace=True)

# Displaying the DataFrame with hierarchical indexing
print("DataFrame with Hierarchical Indexing:")
print(df)
```

```
DataFrame with Hierarchical Indexing:
                   Title    Content Date_Published
Author   Post_ID
Author A 1        Post 1  Content 1     2023-01-01
Author B 2        Post 2        NaN     2023-02-15
Author C 3           NaN  Content 3     2023-03-10
NaN      4        Post 4  Content 4     2023-04-20
Author E 5        Post 5  Content 5     2023-05-30
```