Q1. Write a program to distinguish between Array Indexing and Fancy Indexing.

In [ ]:
```
import numpy as np

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

# Array Indexing
indexing_result = arr[0, 1]
print("Array Indexing Result:", indexing_result)

# Fancy Indexing
fancy_indexing_result = arr[[0, 1], [1, 2]]
print("Fancy Indexing Result:", fancy_indexing_result)
```

```
Array Indexing Result: 2
Fancy Indexing Result: [2 6]
```

Q2. Execute the 2D array Slicing.

In [ ]:
```
import numpy as np

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

slice_result = arr[1:3, 0:2]
print("Slicing Result:\n", slice_result)
```

```
Slicing Result:
 [[4 5]
 [7 8]]
```

Q3. Create the 5-Dimensional arrays using ‘ndmin’.

In [ ]:
```
import numpy as np
# Creating a 5-D array with ndmin
arr = np.array([1, 2, 3], ndmin=5)
print("5-Dimensional Array:\n", arr)
```

```
5-Dimensional Array:
 [[[[[1 2 3]]]]]
```

Q4. Reshape the array from 1-D to 2-D array.

In [ ]:
```
import numpy as np
# Creating a 1-D array
arr = np.array([1,5,8,6,2,7])
print(arr)

#reshaping the 1-D into 2-D
arr1 = arr.reshape(2,3)
print("Reshaped Array: ","\n", arr1)
```

```
[1 5 8 6 2 7]
Reshaped Array:
 [[1 5 8]
 [6 2 7]]
```

Q5. Perform the Stack functions in Numpy arrays – Stack(), hstack(), vstack(), and dstack().

In [ ]:
```python
import numpy as np

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

# Stack arrays vertically
vstack_result = np.vstack((arr1, arr2))
print("vstack Result:\n", vstack_result)

# Stack arrays horizontally
hstack_result = np.hstack((arr1, arr2))
print("hstack Result:", hstack_result)

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

# Stack arrays along the third axis
dstack_result = np.dstack((arr3, arr4))
print("dstack Result:\n", dstack_result)
```

```
vstack Result:
 [[1 2 3]
 [4 5 6]]
hstack Result: [1 2 3 4 5 6]
dstack Result:
 [[[ 7 10]
  [ 8 11]
  [ 9 12]]]
```

Q6. Perform the searchsort method in Numpy array.

In [ ]:
```python
import numpy as np

# Creating a sorted 1-D array
arr = np.array([2, 4, 6, 8, 10])

# Search for the index where 6 should be inserted to maintain sorting
index = np.searchsorted(arr, 6)
print("Index for 6:", index)
```

```
Index for 6: 2
```

Q7. Create Numpy Structured array using your domain features.

In [ ]:
```python
import numpy as np

# Creating a structured array with two fields: 'name' and 'price'
data = [('Shirt', 250), ('Pant', 300), ('Tie', 280)]
dtype = [('name', 'U10'), ('price', int)]
```

```
structured_array = np.array(data, dtype=dtype)
print("Structured Array:\n", structured_array)
```

```
Structured Array:
 [('Shirt', 250) ('Pant', 300) ('Tie', 280)]
```

Q8. Create Data frame using List and Dictionary.

In [ ]:
```
import pandas as pd

# Creating a DataFrame from a list of dictionaries
data = [{'Name': 'Chris', 'Age': 25},
        {'Name': 'Manoj', 'Age': 30},
        {'Name': 'Clement', 'Age': 28}]

df_from_list = pd.DataFrame(data)
print("DataFrame from List of Dictionaries:\n", df_from_list)

# Creating a DataFrame from a dictionary
data = {'Name': ['Chris', 'Manoj', 'Clement'],
        'Age': [25, 30, 28]}

df_from_dict = pd.DataFrame(data)
print("DataFrame from Dictionary:\n", df_from_dict)
```

```
DataFrame from List of Dictionaries:
       Name  Age
0    Chris   25
1    Manoj   30
2  Clement   28
DataFrame from Dictionary:
       Name  Age
0    Chris   25
1    Manoj   30
2  Clement   28
```

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()

In [ ]:
```
import pandas as pd
import numpy as np

# Creating a sample DataFrame with missing data
data = {'Name': ['Chris', 'Manoj', 'Clement', 'Deepak'],
        'Age': [25, np.nan, 28, 32],
        'Salary': [50000, 60000, np.nan, 75000]}

df = pd.DataFrame(data)

# Check for missing data
print("isnull():\n", df.isnull())
print("notnull():\n", df.notnull())

# Drop rows with missing data
df_cleaned = df.dropna()
print("DataFrame after dropna():\n", df_cleaned)

# Fill missing values with a specific value
```

```
df_filled = df.fillna({'Age': 0, 'Salary': 0})
print("DataFrame after fillna():\n", df_filled)

# Replace specific values
df_replaced = df.replace(np.nan, -1)
print("DataFrame after replace():\n", df_replaced)

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

```
isnull():
     Name    Age  Salary
0  False  False   False
1  False   True   False
2  False  False    True
3  False  False   False
notnull():
    Name    Age  Salary
0  True   True    True
1  True  False    True
2  True   True   False
3  True   True    True
DataFrame after dropna():
      Name   Age   Salary
0   Chris  25.0  50000.0
3  Deepak  32.0  75000.0
DataFrame after fillna():
       Name   Age   Salary
0    Chris  25.0  50000.0
1    Manoj   0.0  60000.0
2  Clement  28.0      0.0
3   Deepak  32.0  75000.0
DataFrame after replace():
       Name   Age   Salary
0    Chris  25.0  50000.0
1    Manoj  -1.0  60000.0
2  Clement  28.0     -1.0
3   Deepak  32.0  75000.0
DataFrame after interpolate():
       Name   Age   Salary
0    Chris  25.0  50000.0
1    Manoj  26.5  60000.0
2  Clement  28.0  67500.0
3   Deepak  32.0  75000.0
```

Q10. Perform the Hierarchical Indexing in the above created dataset.

In [ ]:

```
import pandas as pd

# Creating a sample DataFrame with hierarchical indexing
data = {'Name': ['Chris', 'Manoj', 'Clement', 'Vishnu'],
        'Age': [25, 30, 28, 32],
        'Salary': [70000, 90000, 95000, 45000]}

df = pd.DataFrame(data)

# Creating a hierarchical index based on 'Name' and 'Age'
df.set_index(['Name', 'Age'], inplace=True)
```

```python
# Accessing data using hierarchical index
print("Data for Chris and Age 25:\n", df.loc[('Chris', 25)])

# Resetting the index
df.reset_index(inplace=True)
print("DataFrame after resetting the index:\n", df)
```

```
Data for Chris and Age 25:
 Salary    70000
Name: (Chris, 25), dtype: int64
DataFrame after resetting the index:
       Name  Age  Salary
0    Chris   25   70000
1    Manoj   30   90000
2  Clement   28   95000
3   Vishnu   32   45000
```