In [ ]:
```python
# Write a program to distinguish between Array Indexing and Fancy Indexing.
import numpy as np

# NumPy array
arr = np.array([0, 1, 2, 3, 4, 5])

# Array Indexing
index_1 = arr[2]  # Access the element at index 2
index_2 = arr[1:4]  # Access elements from index 1(inclusive) to 3 (exclusive)
print("Array Indexing:")
print("Element at index 2:", index_1)
print("Elements from index 1 to 3 (exclusive):", index_2)

# Fancy Indexing
index_array = np.array([0, 4, 5])  # Array of integers
boolean_array = np.array([True, False, True, False, False, True])  # Boolean array

fancy_1 = arr[index_array]  # Access elements at specified indices using an integer
fancy_2 = arr[boolean_array]  # Access elements based on a boolean condition
print("\nFancy Indexing:")
print("Elements at specified indices:", fancy_1)
print("Elements based on a boolean condition:", fancy_2)
```

```
Array Indexing:
Element at index 2: 2
Elements from index 1 to 3 (exclusive): [1 2 3]

Fancy Indexing:
Elements at specified indices: [0 4 5]
Elements based on a boolean condition: [0 2 5]
```

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

import numpy as np

# Create 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 (exclus
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]
```

In [ ]:
```python
# Create the 5-Dimensional arrays using 'ndmin'.
import numpy as np

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

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

# Print 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]]]]]
```

In [ ]:
```python
# Reshape the array from 1-D to 2-D array.
import numpy as np

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

# Reshape the 1-D array to a 2-D array
arr_2d = arr_1d.reshape((2, 3))  # Specify 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]]
```

In [ ]:
```python
# Perform the Stack functions in Numpy arrays - Stack(), hstack(), vstack(), and ds
import numpy as np

# Create 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)

# Create 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]]]
```

In [ ]:
```python
# 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]
```

In [ ]:
```python
# Create Numpy Structured array using your domain features.
import numpy as np

# Define 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 ch
    ('url', 'U100'),              # Website URL as a Unicode string of up to 100 ch
    ('monthly_visitors', 'i8'),   # Monthly visitors as a 64-bit integer
    ('number_of_blogs', 'i4'),    # Number of blogs on the website as a 32-bit inte
    ('user_registration', bool)   # Whether user registration is allowed (boolean)
])

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

```
# Add 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)

# Access and manipulate 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])
```

In [ ]:

```
# Create Data frame using List and Dictionary.
import pandas as pd

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

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

# Display the DataFrame
print(df_list)


# 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
```

In [ ]:

```
# Create Data frame on your Domain area and perform the following operations to fin
# missing data from the dataset.
# • isnull()
# • notnull()
# • dropna()
# • fillna()
# • replace()
# • interpolate()
import pandas as pd
import numpy as np

# Create 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', '202
}

df = pd.DataFrame(data)

# Display the DataFrame
print("Original DataFrame:")
print(df)
# isnull()
# Identify missing values
missing_data = df.isnull()
print("\nMissing Data (isnull()):")
print(missing_data)


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

# fillna
# Replace 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

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
```

In [ ]:
```python
# Perform the Hierarchical Indexing in the above created dataset.
import pandas as pd
import numpy as np

# Create 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', '202
}

df = pd.DataFrame(data)

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

# Display 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
```