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

In [ ]:

```
import numpy as np

# Create a sample NumPy array
arr = np.array([11, 21, 32, 47, 59 , 65, 75, 89 , 90 ,97 ])

# Array Indexing: Accessing individual elements using integer indices
print("Array Indexing:")
print("arr[0]:", arr[0])  # Access the first element
print("arr[3]:", arr[3])  # Access the fourth element
print()

# Fancy Indexing: Accessing elements using arrays of indices
print("Fancy Indexing:")
indices = np.array([0, 2, 4])  # Create an array of indices
print("arr[indices]:", arr[indices])  # Access elements at the specified
print()
```

```
Array Indexing:
arr[0]: 11
arr[3]: 47

Fancy Indexing:
arr[indices]: [11 32 59]
```

Q2. Execute the 2D array Slicing.

In [ ]:

```
import numpy as np

# Create a sample 2D NumPy array
data = np.array([[1, 2, 3, 4],
                 [5, 6, 7, 8],
                 [9, 10, 11, 12]])

# Display the original array
print("Original Array:")
print(data)

# Slicing the array
# Syntax: array[row_start:row_end, col_start:col_end]

# Selecting a single row (row 1)
row_1 = data[1, :]
print("\nRow 1:")
print(row_1)

# Selecting a single column (column 2)
col_2 = data[:, 2]
print("\nColumn 2:")
print(col_2)

# Slicing a subarray (rows 0 and 1, columns 1 and 2)
subarray1 = data[0:2, 1:3]
```

```
print("\nSubarray1 (rows 0:2, columns 1:3):")
print(subarray1)

# Slicing with step (every other row, every other column)
step_slice = data[::2, ::2]
print("\nStep Slice (every other row, every other column):")
print(step_slice)

subarray2= data[1:3, 2:4]
print("\nSubarray2 (rows 1:3, columns 2:4):")
print(subarray2)
```

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

Row 1:
[5 6 7 8]

Column 2:
[ 3  7 11]

Subarray1 (rows 0:2, columns 1:3):
[[2 3]
 [6 7]]

Step Slice (every other row, every other column):
[[ 1  3]
 [ 9 11]]

Subarray2 (rows 1:3, columns 2:4):
[[ 7  8]
 [11 12]]
```

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

In [ ]:

```
import numpy as np

# Create a 5-D array with ndmin
array_5d = np.array([11, 22, 23, 44], ndmin=5)

# Check the shape of the array
print("Shape of the 5-D array:", array_5d.shape)

# Display the 5-D array
print("5-D Array:")
print(array_5d)
```

```
Shape of the 5-D array: (1, 1, 1, 1, 4)
5-D Array:
[[[[[11 22 23 44]]]]]
```

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

In [ ]:

```
import numpy as np

# Create a 1-D array
array_1d = np.array([11, 22, 33, 44, 55, 66 ,77,88,99])
```

```
#  Using numpy.reshape()
array_2d_1 = np.reshape(array_1d, (3, 3))



# Display the original and reshaped arrays
print("Original 1-D array:")
print(array_1d)

print("\nReshaped 2-D array (using reshape):")
print(array_2d_1)
```

```
Original 1-D array:
[11 22 33 44 55 66 77 88 99]

Reshaped 2-D array (using reshape):
[[11 22 33]
 [44 55 66]
 [77 88 99]]
```

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

In [ ]:

```
import numpy as np

# Create two sample arrays
array1 = np.array([11, 22, 33])
array2 = np.array([34, 55, 86])

# Perform stacking operations

# 1. stack() - Stacking along a new axis (axis=0 by default)
stacked_array = np.stack((array1, array2))
print("stacked_array (stack along a new axis):")
print(stacked_array)

# 2. hstack() - Horizontal stacking (column-wise)
hstacked_array = np.hstack((array1, array2))
print("\nhstacked_array (horizontal stacking):")
print(hstacked_array)

# 3. vstack() - Vertical stacking (row-wise)
vstacked_array = np.vstack((array1, array2))
print("\nvstacked_array (vertical stacking):")
print(vstacked_array)

# Create two 2-D sample arrays
array3 = np.array([[1, 2, 3], [4, 5, 6]])
array4 = np.array([[7, 8, 9], [10, 11, 12]])

# 4. dstack() - Stacking along a new 3rd axis (depth-wise)
dstacked_array = np.dstack((array3, array4))
print("\ndstacked_array (depth-wise stacking):")
print(dstacked_array)
```

```
stacked_array (stack along a new axis):
[[11 22 33]
 [34 55 86]]

hstacked_array (horizontal stacking):
[11 22 33 34 55 86]

vstacked_array (vertical stacking):
[[11 22 33]
 [34 55 86]]

dstacked_array (depth-wise stacking):
[[[ 1  7]
  [ 2  8]
  [ 3  9]]

 [[ 4 10]
  [ 5 11]
  [ 6 12]]]
```

Q6. Perform the searchsort method in Numpy array.

In [ ]:

```python
import numpy as np

#Create a sample NumPy array
array = np.array([4, 2, 8, 6, 10, 1])

#Sort the array in ascending order
sorted_array = np.sort(array)
print("Sorted Array (ascending order):")
print(sorted_array)

# Sort the array in descending order
reverse_sorted_array = np.sort(array)[::-1]
print("\nSorted Array (descending order):")
print(reverse_sorted_array)


# Perform a binary search to find the position to insert a value while ma
value_to_insert = 5
position_to_insert = np.searchsorted(sorted_array, value_to_insert)
print(f"\nPosition to Insert {value_to_insert} to Maintain Sorted Order:"

# Perform a binary search to find the indices where a value should be ins
values_to_insert = [3, 7, 9]
positions_to_insert = np.searchsorted(sorted_array, values_to_insert)
print("\nPositions to Insert Multiple Values to Maintain Sorted Order:")
print(positions_to_insert)
```

```
Sorted Array (ascending order):
[ 1  2  4  6  8 10]

Sorted Array (descending order):
[10  8  6  4  2  1]

Position to Insert 5 to Maintain Sorted Order: 3

Positions to Insert Multiple Values to Maintain Sorted Order:
[2 4 5]
```

Q7. Create Numpy Structured array using your domain features.

In [ ]:

```
import numpy as np

# Define the structured data type for the election system
election_dtype = np.dtype([
    ('voter_id', 'int32'),
    ('voter_name', 'U50'),  # 'U50' specifies Unicode string of up to 50
    ('age', 'int32'),
    ('address', 'U100'),   # 'U100' specifies Unicode string of up to 100
    ('vote_cast', 'bool')
])

# Create a structured array with sample data
election_data = np.array([
    (1, 'Alok Misra', 30, '123 Main St', True),
    (2, 'John Smith', 28, '456 Elm St', False),
    (3, 'Anand Patel', 35, '789 Oak St', True)
], dtype=election_dtype)

# Print the structured array
print("Election Data Structured Array:")
print(election_data)
```

```
Election Data Structured Array:
[(1, 'Alok Misra', 30, '123 Main St',  True)
 (2, 'John Smith', 28, '456 Elm St', False)
 (3, 'Anand Patel', 35, '789 Oak St',  True)]
```

Q8. Create Data frame using List and Dictionary.

In [ ]:

```
import pandas as pd

# Creating a DataFrame using Lists
data_list = [['RAM', 28, 'Engineer'],
             ['MOHAN', 24, 'Data Scientist'],
             ['KISHAN', 22, 'Student']]

columns_list = ['Name', 'Age', 'Occupation']

df_list = pd.DataFrame(data_list, columns=columns_list)

# Creating a DataFrame using Dictionaries
data_dict = {
    'Name': ['RAHIM', 'RISHI', 'FEROZ'],
    'Age': [32, 30, 26],
    'Occupation': ['Doctor', 'Teacher', 'Artist']
}

df_dict = pd.DataFrame(data_dict)

# Displaying the DataFrames
print("DataFrame created using Lists:")
print(df_list)
print("\nDataFrame created using Dictionaries:")
print(df_dict)
```

```
DataFrame created using Lists:
     Name  Age      Occupation
0     RAM   28        Engineer
1   MOHAN   24  Data Scientist
2  KISHAN   22         Student

DataFrame created using Dictionaries:
    Name  Age Occupation
0  RAHIM   32     Doctor
1  RISHI   30    Teacher
2  FEROZ   26     Artist
```

Q9. Create Data frame on your Domain area and perform the following operations to find and eliminate themissing data from the dataset.

• isnull() • notnull() • dropna() • fillna() • replace() • interpolate()

In [ ]:

```
import pandas as pd
import numpy as np

# Create a sample DataFrame for the online election system
data = {
    'voter_id': [1, 2, 3, 4, 5],
    'voter_name': ['Jane', 'Jaffri', 'Aman', 'Munda', 'Aalok'],
    'age': [30, None, 25, 40, None],
    'address': ['123 Main St', None, '456 Elm St', '789 Oak St', None],
    'vote_cast': [True, False, True, True, False]
}

df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Check for missing data

# isnull() - Check if a value is missing (returns a DataFrame of boolean
print("\nCheck for Missing Data (isnull()):")
print(df.isnull())

# notnull() - Check if a value is not missing (returns a DataFrame of boo
print("\nCheck for Non-Missing Data (notnull()):")
print(df.notnull())

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

# fillna() - Fill missing values with a specified value or strategy
df_filled = df.fillna({'age': df['age'].mean(), 'address': 'Unknown'})
print("\nDataFrame after filling missing values (fillna()):")
print(df_filled)
```

```
# replace() - Replace specific values with another value
df_replaced = df.replace({'vote_cast': {True: 'Yes', False: 'No'}})
print("\nDataFrame after replacing values (replace()):")
print(df_replaced)

# interpolate() - Interpolate missing values (works well with numeric dat
df_interpolated = df.interpolate()
print("\nDataFrame after interpolating missing values (interpolate()):")
print(df_interpolated)
```

```
Original DataFrame:
   voter_id voter_name   age      address  vote_cast
0         1       Jane  30.0  123 Main St       True
1         2     Jaffri   NaN         None      False
2         3       Aman  25.0   456 Elm St       True
3         4      Munda  40.0   789 Oak St       True
4         5      Aalok   NaN         None      False

Check for Missing Data (isnull()):
   voter_id  voter_name    age  address  vote_cast
0     False       False  False    False      False
1     False       False   True     True      False
2     False       False  False    False      False
3     False       False  False    False      False
4     False       False   True     True      False

Check for Non-Missing Data (notnull()):
   voter_id  voter_name    age  address  vote_cast
0      True        True   True     True       True
1      True        True  False    False       True
2      True        True   True     True       True
3      True        True   True     True       True
4      True        True  False    False       True

DataFrame after dropping rows with missing values (dropna()):
   voter_id voter_name   age      address  vote_cast
0         1       Jane  30.0  123 Main St       True
2         3       Aman  25.0   456 Elm St       True
3         4      Munda  40.0   789 Oak St       True

DataFrame after filling missing values (fillna()):
   voter_id voter_name        age      address  vote_cast
0         1       Jane  30.000000  123 Main St       True
1         2     Jaffri  31.666667      Unknown      False
2         3       Aman  25.000000   456 Elm St       True
3         4      Munda  40.000000   789 Oak St       True
4         5      Aalok  31.666667      Unknown      False

DataFrame after replacing values (replace()):
   voter_id voter_name   age      address vote_cast
0         1       Jane  30.0  123 Main St       Yes
1         2     Jaffri   NaN         None        No
2         3       Aman  25.0   456 Elm St       Yes
3         4      Munda  40.0   789 Oak St       Yes
4         5      Aalok   NaN         None        No

DataFrame after interpolating missing values (interpolate()):
   voter_id voter_name   age      address  vote_cast
0         1       Jane  30.0  123 Main St       True
1         2     Jaffri  27.5         None      False
2         3       Aman  25.0   456 Elm St       True
3         4      Munda  40.0   789 Oak St       True
4         5      Aalok  40.0         None      False
```

```
/var/folders/fz/sw830djj40x8hdx5bbn5fd3h0000gn/T/ipykernel_6494/115054643
8.py:45: FutureWarning: DataFrame.interpolate with object dtype is depreca
ted and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  df_interpolated = df.interpolate()
```

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

In [ ]:

```
import pandas as pd

# Create a sample DataFrame for the online election system
data = {
    'voter_id': [1, 2, 3, 4, 5],
    'voter_name': ['Jane', 'Jaffri', 'Aman', 'Birla', 'Aalok'],
    'age': [30, None, 25, 40, None],
    'address': ['123 Main St', None, '456 Elm St', '789 Oak St', None],
    'vote_cast': [True, False, True, True, False]
}

df = pd.DataFrame(data)

# Create hierarchical indexing with 'voter_id' and 'voter_name'
df.set_index(['voter_id', 'voter_name'], inplace=True)

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

```
DataFrame with Hierarchical Indexing:
                      age      address  vote_cast
voter_id voter_name
1        Jane        30.0  123 Main St       True
2        Jaffri       NaN         None      False
3        Aman        25.0   456 Elm St       True
4        Birla       40.0   789 Oak St       True
5        Aalok        NaN         None      False
```