```
import numpy as np
from io import StringIO

# ============================================================================
# Topic 1: Masks and Boolean Arrays
# Description: Boolean masks are True/False arrays used to filter elements.
# ============================================================================
print("\n" + "="*50 + "\nTopic 1: Masks and Boolean Arrays\n" + "="*50)

# Problem 1 (Simple): Create mask for elements >5
arr1 = np.array([1, 6, 3, 8, 2])
mask1 = arr1 > 5
print("\nProblem 1 - Mask for elements >5:", mask1)

# Problem 2 (Simple): Extract even numbers
arr2 = np.array([2, 3, 4, 5, 6])
evens = arr2[arr2 % 2 == 0]
print("\nProblem 2 - Even numbers:", evens)

# Problem 3 (Intermediate): Replace elements >10 with 0
arr3 = np.array([5, 12, 8, 15, 3])
arr3[arr3 > 10] = 0
print("\nProblem 3 - Modified array:", arr3)

# Problem 4 (Intermediate): 2D mask for elements 3-8
arr4 = np.array([[1, 5], [9, 4]])
print("\nProblem 4 - Original array:\n", arr4)
print("Mask:\n", (arr4 >= 3) & (arr4 <= 8))
```

```
# Problem 5 (Complex): Rows with any negative element
arr5 = np.array([[1, -2], [3, 4], [-5, 6]])
rows_with_neg = arr5[np.any(arr5 < 0, axis=1)]
print("\nProblem 5 - Rows with negatives:\n", rows_with_neg)

# Problem 6 (Complex): Combine masks (>5 and <10)
arr6 = np.array([2, 6, 12, 7, 3])
combined = arr6[(arr6 > 5) & (arr6 < 10)]
print("\nProblem 6 - Combined mask result:", combined)

#
==========================================================================
=
# Topic 2: Fancy Indexing
# Description: Indexing using arrays/lists of indices
#
==========================================================================
=
print("\n" + "="*50 + "\nTopic 2: Fancy Indexing\n" + "="*50)

# Problem 1 (Simple): Select indices 0,2,4
arr7 = np.array([10, 20, 30, 40, 50])
print("\nProblem 7 - Selected elements:", arr7[[0, 2, 4]])

# Problem 2 (Simple): Select rows from 2D array
arr8 = np.array([[1, 2], [3, 4], [5, 6]])
print("\nProblem 8 - Selected rows:\n", arr8[[2, 0]])

# Problem 3 (Intermediate): Select specific columns
arr9 = np.array([[1, 2, 3], [4, 5, 6]])
print("\nProblem 9 - Selected columns:\n", arr9[:, [1, 0]])
```

```
# Problem 4 (Intermediate): Boolean array indexing
arr10 = np.array([5, 10, 15, 20])
print("\nProblem 10 - Selected elements:",
    arr10[np.array([True, False, True, False])])

# Problem 5 (Complex): Array permutation
arr11 = np.array([10, 20, 30, 40, 50])
perm = np.random.permutation(5)
print("\nProblem 11 - Permuted array:", arr11[perm])

# Problem 6 (Complex): 3D array indexing
arr12 = np.arange(27).reshape(3,3,3)
print("\nProblem 12 - Selected 3D elements:\n", arr12[[2, 1, 0]])

#
==========================================================================
=
# Topic 3: Sorting Arrays
# Description: Array sorting and related operations
#
==========================================================================
=
print("\n" + "="*50 + "\nTopic 3: Sorting Arrays\n" + "="*50)

# Problem 1 (Simple): Sort 1D array
arr13 = np.array([3, 1, 4, 2, 5])
print("\nProblem 13 - Sorted array:", np.sort(arr13))

# Problem 2 (Simple): Get sorted indices
arr14 = np.array([5, 3, 7, 2])
print("\nProblem 14 - Sorted indices:", np.argsort(arr14))
```

```
# Problem 3 (Intermediate): Sort 2D array rows
arr15 = np.array([[3, 2], [1, 4]])
print("\nProblem 15 - Sorted rows:\n", np.sort(arr15, axis=1))

# Problem 4 (Intermediate): Sort columns descending
arr16 = np.array([[5, 3], [2, 6]])
print("\nProblem 16 - Sorted cols descending:\n",
    np.sort(arr16, axis=0)[::-1])

# Problem 5 (Complex): Sort using another array
keys = np.array([3, 1, 2])
values = np.array(['a', 'b', 'c'])
print("\nProblem 17 - Sorted values:", values[np.argsort(keys)])

# Problem 6 (Complex): Partial sort
arr17 = np.array([5, 3, 8, 1, 6])
print("\nProblem 18 - Partially sorted:", np.partition(arr17, 3))

#
==========================================================================
=
# Topic 4: Structured Arrays
# Description: Arrays with complex data types
#
==========================================================================
=
print("\n" + "="*50 + "\nTopic 4: Structured Arrays\n" + "="*50)

# Problem 1 (Simple): Create structured array
dtype = [('name', 'U10'), ('age', 'i4'), ('height', 'f4')]
data = np.array([('Alice', 25, 165.5), ('Bob', 30, 175.0)], dtype=dtype)
```

```
print("\nProblem 19 - Structured array:", data)

# Problem 2 (Simple): Access 'age' field
print("\nProblem 20 - Ages:", data['age'])

# Problem 3 (Intermediate): Sort by age
print("\nProblem 21 - Sorted by age:", np.sort(data, order='age'))

# Problem 4 (Intermediate): Filter height > 170
print("\nProblem 22 - Tall people:", data[data['height'] > 170])

# Problem 5 (Complex): Load from CSV
csv_data = "Name,Age,Height\nAlice,25,165.5\nBob,30,175.0"
data_file = StringIO(csv_data)
structured = np.genfromtxt(data_file, delimiter=',', skip_header=1,
                           dtype=[('Name', 'U10'), ('Age', 'i4'), ('Height', 'f4')])
print("\nProblem 23 - From CSV:", structured)

# Problem 6 (Complex): Modify and add field
new_data = np.copy(data)
new_data['age'] += 1
new_dtype = np.dtype(data.dtype.descr + [('weight', 'f4')])
expanded = np.zeros(new_data.shape, dtype=new_dtype)
for field in data.dtype.names:
    expanded[field] = new_data[field]
expanded['weight'] = [60.5, 70.2]
print("\nProblem 24 - Modified array:", expanded)
```