Lecture Plan: 03 - Accessing Elements

1. Basic Indexing & Slicing
    - Accessing Individual Elements
    - Accessing Range of Elements
2. Boolean Indexing/Masking
3. Array Indexing/Fancy Indexing
4. Working with 2D Arrays
    - Accessing specific rows and columns.

1. Basic Indexing & Slicing

Explain this to students by referring back to python lists, as this type of indexing is same as python lists.

```
temperatures = np.array([25, 26, 28, 24, 23, 26, 27])

### Accessing Individual Elements
print(temperatures[0])   # Access the temperature on the first
day
print(temperatures[3])   # Access the temperature on the fourth
day

### Accessing Range of Elements
print(temperatures[1:4])   # Access temperatures from the second
to the fourth day
print(temperatures[2:])    # Access temperatures from the third
day to the last day
```

2. Boolean Indexing & Masking

Explain the concept of boolean indexing to the students.

Must do tips:

1. Do not directly use the mask within array. `passing_students = scores[scores >= 90]`. Instead, break it into two steps and show them first *what is a mask?* Print the mask so that they can understand it is an array of boolean values where we get `True` where the condition is met, otherwise `False`.
2. Explain them - when it is used? Whenever we need to filter the array based on a condition, boolean indexing is used. Again, this is one of the ways of Numpy to avoid looping.

```
scores = np.array([85, 90, 78, 92, 88, 95, 80])

### Step 1: Create the mask
passing_mask = scores >= 90
print(passing_mask)
```

```
### Step 2: Use the mask
passing_students = scores[passing_mask]
print(passing_students)
```

3. Array Indexing/Fancy Indexing

Explain the concept of integer array indexing in NumPy, that allows you to select elements from an array using an array of integer indices.

```
student_ids = np.array([101, 105, 110, 115, 120, 125])

indices = np.array([1, 3, 5])
selected_students = student_ids[indices]
print(selected_students)
```

4. Working with 2D array

Here we want to cover how to access rows and columns in 2D arrays. We want to drive this using the practical example. So, you must follow the example given below.

Imagine you have a 2D array representing the daily closing prices of different stocks over a certain period of time. Each row corresponds to a stock, and each column represents a trading day.

```
# Create a 2D array of daily closing prices (5 stocks x 30
trading days)
closing_prices = np.random.randint(low=50, high=200, size=(5,
30))
print("Closing Prices:")
print(closing_prices)
```

1. Accessing a Row

```
# Access the closing prices of the 3rd stock (row index 2)
stock_3_prices = closing_prices[2]
print("Closing Prices for Stock 3:")
print(stock_3_prices)
```

2. Accessing Multiple Rows

```
# Access the closing prices for stocks 2, 4, and 5 (row indices
1, 3, and 4)
selected_stocks_prices = closing_prices[[1, 3, 4]]
print("Closing Prices for Selected Stocks:")
print(selected_stocks_prices)
```

3. Accessing a Column

```
# Access the closing prices for the 5th trading day (column index
4)
day_5_prices = closing_prices[:, 4]
print("Closing Prices for Day 5:")
print(day_5_prices)
```

4. Accessing Multiple Columns

```
# Access the closing prices for the 3rd, 5th, and 7th trading
days (column indices 2, 4, and 6)
selected_days_prices = closing_prices[:, [2, 4, 6]]
print("Closing Prices for Selected Days:")
print(selected_days_prices)
```

5. Boolean Indexing Example

```
# Access the closing prices for stocks with prices above 150
high_price_stocks = closing_prices[closing_prices > 150]
print("Closing Prices for High Price Stocks:")
print(high_price_stocks)
```

6. Array Indexing Example

```
# Define the indices for the desired data points
row_indices = np.array([0, 2, 4])
column_indices = np.array([1, 5, 9])

# Access the closing prices for the specified data points
selected_prices = closing_prices[row_indices[:, np.newaxis],
column_indices]
print("Selected Closing Prices:")
print(selected_prices)
```

End with the note that, these situations demonstrate how to access rows, multiple rows, columns, multiple columns, and how to use array indexing for rows and columns in NumPy arrays. By applying appropriate indexing techniques, you can extract and analyze specific subsets of data for further analysis or processing.
