Lecture 4: Text Data Type (`str`) in Python

1. Introduction to Strings (`str`)

A **string** in Python is a sequence of characters enclosed in quotes (either single `''`, double `""`, or triple quotes `'''`/`"""` for multi-line strings). Strings are one of the most commonly used data types in Python, and they allow us to work with text.

Example:

```
name = "Alice"  # A string enclosed in double quotes

greeting = 'Hello, World!'  # A string enclosed in single quotes
```

Strings are **immutable**, meaning that once a string is created, it cannot be changed.

2. Basic String Operations

Python provides a variety of operations that can be performed on strings, such as indexing, slicing, and concatenation. Let's explore them in detail.

---

3. String Indexing

String **indexing** allows us to access individual characters in a string using their position (or index). In Python, indexing starts at 0, meaning the first character of the string has an index of 0, the second character has an index of 1, and so on.

- **Positive Indexing**: Starts from the beginning (0, 1, 2, ...).
- **Negative Indexing**: Starts from the end (-1, -2, -3, ...).

Example:

```
text = "Python"

# Positive Indexing
print(text[0])  # Output: P (first character)
print(text[3])  # Output: h (fourth character)

# Negative Indexing
print(text[-1])  # Output: n (last character)
print(text[-3])  # Output: t (third character from the end)
```

In this example:

- `text[0]` refers to the first character 'P'.
- `text[-1]` refers to the last character 'n'.

---

4. String Slicing

Slicing allows us to extract a part (or substring) of a string by specifying a range of indices. The syntax for slicing is:

```
string[start:end]
```

Where `start` is the index where slicing begins, and `end` is the index where slicing stops (the character at the `end` index is **not included**).

Example:

```
text = "Python"

# Slicing from index 0 to 3 (not including 3)
print(text[0:3])  # Output: Pyt

# Slicing from index 1 to 4
print(text[1:4])  # Output: yth

# Slicing the whole string
print(text[:])  # Output: Python (entire string)

# Slicing with negative indices
print(text[-4:-1])  # Output: tho
```

In this example:

- `text[0:3]` extracts characters from index 0 to 2, resulting in "Pyt".
- `text[:3]` means the same as `text[0:3]`.

---

5. String Concatenation

Concatenation is the process of combining two or more strings into one. In Python, this is done using the `+` operator.

Example:

```
first_name = "John"
last_name = "Doe"

# Concatenating strings
full_name = first_name + " " + last_name
print(full_name)  # Output: John Doe
```

In this example, the two strings `first_name` and `last_name` are combined, with a space in between.

---

6. Other Common String Operations

Here are some other useful string operations in Python:

1. **String Length (`len()`)**: You can find the length (number of characters) of a string using the `len()` function.
   Example:

```
text = "Python"
print(len(text))  # Output: 6
```

2. **Changing Case: You can change the case of strings using the following methods:**

- **`.upper()`: Converts the string to uppercase.**
- **`.lower()`: Converts the string to lowercase.**
- **`.capitalize()`: Capitalizes the first letter of the string.**

Example:

```
text = "hello world"

print(text.upper())  # Output: HELLO WORLD
```