Key Points on Strings in Python

What are Strings?

- A **string** in Python is a sequence of characters enclosed in either single (`'`) or double (`"`) quotes.
- Strings are encoded using **Unicode**, allowing computers to interpret characters as binary code.

Declaring Strings

- **Single-line strings**: Enclose characters in single or double quotes.

```
single_quote_str = 'Hello'
double_quote_str = "World"
```

- **Multi-line strings**: Use triple quotes (`'''` or `"""`) or backslashes (`\`) to continue a string on the next line.

```
multi_line_str = '''This is a
multi-line string.'''

backslash_str = "This is a multi-\
line string example."
```

String Operations

Reassigning Strings

- Strings can be reassigned to new values.

```
name = "John"
name = "Paul"
print(name)  # Output: Paul
```

Concatenation

- Use the `+` operator to join strings.

```
a = "Hello, "
b = "World!"
print(a + b)  # Output: Hello, World!
```

Strings as Sequences

Indexing

- Strings are **zero-indexed**, meaning the first character has an index of 0.

```
name = "John"
print(name[0])  # Output: J
print(name[3])  # Output: n
```

Length

- Use the `len()` function to determine the length of a string.

```
print(len(name))  # Output: 4
```

Examples

Single vs Multi-line Strings

```
# Single-line string
greeting = "Hello, World!"

# Multi-line string
description = '''This is a multi-line
string in Python.'''
```

Accessing Characters

```
word = "Python"
print(word[0])  # Output: P
print(word[5])  # Output: n
```

Concatenating Strings

```
first_part = "Learn "
second_part = "Python!"
full_sentence = first_part + second_part
print(full_sentence)  # Output: Learn Python!
```

Using `len()`

```
sentence = "Hello, Python!"
print(len(sentence))  # Output: 14
```

Practice Tasks

1. Declare a string and print it.
2. Access specific characters using indexing.
3. Concatenate two strings and print the result.
4. Determine the length of a string using `len()`.