Python Course - Day 005

Special Characters and Escape Sequences in Python (Part 1)

In Python, escape sequences are special combinations of characters that represent actions or characters that are difficult to type directly. They always start with a backslash (\) followed by a character or a set of characters. Here's a detailed explanation of common escape sequences and special characters:

Quotation Marks

`\'` and `\"`: These allow you to include single and double quotes within strings.

```
print("He said, \"Hello!\"")  # Output: He said, "Hello!"
print('It\'s a beautiful day')  # Output: It's a beautiful day
```

Backslash

`\\`: Represents a literal backslash character.

```
print("This is a backslash: \\")  # Output: This is a backslash: \
```

New Line and Carriage Return

`\n`: Inserts a new line.

`\r`: Moves the cursor to the beginning of the current line (rarely used in Python).

```
print("Hello\nWorld")
# Output:
# Hello
# World

print("Hello\rWorld")  # Output: World (overwrites "Hello")
```

⌨ Special Characters and Escape Sequences in Python (Part 2)

Tab

`\t`: Inserts a horizontal tab.

```
print("Name:\tJohn")  # Output: Name:    John
```

Backspace

`\b`: Moves the cursor one space back (rarely used in strings).

```
print("Hello\bWorld")  # Output: HellWorld (erases 'o')
```

</> Unicode and Hex Characters

`\uXXXX`: Represents a Unicode character by its 4-digit hex value.

`\xhh`: Represents a character by its 2-digit hex value.

```
print("\u263A")  # Output: ☺ (smiley face)
print("\x41")    # Output: A (ASCII character for 'A')
```

➡ Line Continuation

`\ + newline`: Allows a statement to continue on the next line.

```
long_string = "This is a very long string that \
continues on the next line"
print(long_string)
# Output: This is a very long string that continues on the next line
```

Understanding these special characters and escape sequences allows you to handle various text formatting scenarios and include special characters in your strings effectively.

+ Concatenation in Python (Part 1)

Concatenation in Python is the process of combining two or more strings into a single string. It's a fundamental operation when working with text data. Here's how you can perform concatenation in Python:

Using the + Operator

The simplest way to concatenate strings is by using the + operator:

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

% Using the % Operator (String Formatting)

You can use the % operator for string formatting, which is a form of concatenation:

```
name = "Alice"
age = 30
message = "My name is %s and I am %d years old" % (name, age)
print(message)  # Output: My name is Alice and I am 30 years old
```

+ Concatenation in Python (Part 2)

Using the format() Method

The format() method provides a more flexible way to concatenate strings:

```
item = "apple"
price = 1.99
quantity = 3
order = "I want {} {}s at ${} each".format(quantity, item, price)
print(order)  # Output: I want 3 apples at $1.99 each
```

Using f-strings (Python 3.6+)

F-strings provide a concise and readable way to embed expressions inside string literals:

```
name = "Bob"
age = 25
intro = f"Hi, I'm {name} and I'm {age} years old"
print(intro)  # Output: Hi, I'm Bob and I'm 25 years old
```

These methods allow you to easily combine strings and other data types in Python, making your code more readable and efficient.