Day 7: String Handling

1. String Operations and Methods

Strings are one of the most fundamental data types in Python, and mastering string operations allows us to manipulate and process text effectively. Below are some key concepts, descriptions, and examples.

Basic String Operations

- **Concatenation**: Combining two or more strings using the + operator.
- first_name = "John"
- last_name = "Doe"
- full_name = first_name + " " + last_name
- print(full_name) # Output: John Doe
- **Repetition**: Repeating a string multiple times using the * operator.
- greeting = "Hello! "
- repeated_greeting = greeting * 3
- print(repeated_greeting) # Output: Hello! Hello! Hello!
- **Indexing**: Accessing individual characters of a string.
- text = "Python"
- print(text[0]) # Output: P
- print(text[-1]) # Output: n (last character)
- **Slicing**: Extracting substrings using slice notation.
- text = "Programming"
- print(text[0:5]) # Output: Progr
- print(text[3:]) # Output: gramming
- print(text[:6]) # Output: Progra

Common String Methods

Python provides a wide range of built-in methods for strings. Some of the most commonly used methods include:

- **Changing Case**
- text = "python programming"
- print(text.upper()) # Output: PYTHON PROGRAMMING
- print(text.lower()) # Output: python programming
- print(text.capitalize()) # Output: Python programming
- print(text.title()) # Output: Python Programming
- **Checking Properties**
- text = "123abc"
- print(text.isalpha()) # Output: False (contains digits)
- print(text.isdigit()) # Output: False (contains alphabets)
- print("123".isdigit()) # Output: True
- print("hello".isalpha()) # Output: True
- **Searching and Replacing**
- text = "I love programming. Programming is fun!"
- print(text.find("programming")) # Output: 7 (first occurrence)
- print(text.replace("programming", "Python"))
- # Output: I love Python. Programming is fun!
- **Splitting and Joining**
- sentence = "Python is a versatile language."
- words = sentence.split() # Splits by space by default
- print(words) # Output: ['Python', 'is', 'a', 'versatile', 'language.']
- joined = "-".join(words)
- print(joined) # Output: Python-is-a-versatile-language.

2. String Formatting

String formatting allows you to insert variables or values into strings dynamically. Python provides multiple methods for string formatting:

1. f-strings (Formatted String Literals)

Introduced in Python 3.6, f-strings allow for easy and readable string formatting using expressions inside curly braces {}.

- **Example:**
- name = "Alice"
- age = 25
- print(f"My name is {name} and I am {age} years old.")
- # Output: My name is Alice and I am 25 years old.
- **Expressions in f-strings:**
- radius = 5
- print(f"The area of a circle with radius {radius} is {3.14 * radius**2:.2f}.")
- # Output: The area of a circle with radius 5 is 78.50.

2. .format() Method

The .format() method allows you to insert values into placeholders {}.

- **Example:**
- name = "Bob"
- city = "Paris"
- print("Hello, my name is {} and I live in {}.".format(name, city))
- # Output: Hello, my name is Bob and I live in Paris.
- **Positional and Keyword Arguments:**
- # Using positional arguments
- print("The coordinates are: ({}, {}).".format(3, 5))
- # Output: The coordinates are: (3, 5).

- # Using keyword arguments
- print("The capital of {country} is {capital}.".format(country="France", capital="Paris"))
- # Output: The capital of France is Paris.
- **Reordering placeholders:**
- print("{1} is older than {0}.".format("Alice", "Bob"))
- # Output: Bob is older than Alice.

3. % Formatting (Legacy)

The % operator is a legacy method for string formatting, similar to printf in C. While still supported, it's less preferred compared to f-strings or .format().

- **Example:**
- name = "Charlie"
- score = 95
- print("Hello %s, you scored %d points!" % (name, score))
- # Output: Hello Charlie, you scored 95 points!

Exercises for Practice

1. Write a program that takes a user's first and last name as input, then prints their full name in all uppercase letters.
2. Ask the user for a sentence, then:
   - Print the number of words in the sentence.
   - Replace every instance of the word "is" with "was."
3. Write a program to display a dynamic message like:
   "Hello John! You have 5 new messages."
   Use both f-strings and the .format() method to demonstrate this.
4. Write a program to take a string and display:
   - The reversed string.

- The string with vowels removed.
  (Hint: Use a loop or string methods like .replace().)

Answers for the given questions:

Program 1: Full Name in Uppercase

```
# Take user input

first_name = input("Enter your first name: ")

last_name = input("Enter your last name: ")


# Combine and convert to uppercase

full_name = f"{first_name} {last_name}".upper()


# Print the result

print(f"Your full name in uppercase is: {full_name}")
```

Program 2: Sentence Analysis

```
# Take user input

sentence = input("Enter a sentence: ")


# Count the number of words

word_count = len(sentence.split())


# Replace "is" with "was"

modified_sentence = sentence.replace("is", "was")


# Display the results

print(f"The number of words in the sentence: {word_count}")

print(f"Modified sentence: {modified_sentence}")
```

Program 3: Dynamic Message

Using f-strings:

```
# Variables
name = "John"
new_messages = 5

# Dynamic message with f-strings
message = f"Hello {name}! You have {new_messages} new messages."
print(message)
```

Using .format() method:

```
# Dynamic message with .format()
message = "Hello {}! You have {} new messages.".format("John", 5)
print(message)
```

Program 4: String Reversal and Vowel Removal

```
# Take user input
text = input("Enter a string: ")

# Reverse the string
reversed_string = text[::-1]

# Remove vowels
vowels = "aeiouAEIOU"
vowel_removed = "".join([char for char in text if char not in vowels])
```

```
# Display the results

print(f"Reversed string: {reversed_string}")

print(f"String without vowels: {vowel_removed}")
```

These solutions provide concise and clear answers to the questions while reinforcing key string handling concepts.