**9. String Manipulation**

1. **Understanding how to access and manipulate strings.**

String Access Methods

# 1. Indexing

* Strings are sequences of characters that can be accessed via zero-based indexing
* Positive indexes start from the beginning (0 = first character)
* Negative indexes start from the end (-1 = last character)
* Attempting to access an out-of-range index raises an IndexError

# 2. Slicing

* Syntax: string[start:stop:step]
* Extracts a substring from start to stop-1
* Default start is 0, default stop is length of string, default step is 1
* Omitting values: [:] creates a copy of the string
* Negative values count from the end
* Step value allows skipping characters ( [::2] gets every second character)

1. **Basic operations: concatenation, repetition, string methods (upper(), lower(), etc.).**

# 1. String Concatenation

Definition: Combining two or more strings into a single string Methods:

Using the + operator

* Using the join() method (more efficient for multiple strings)
* Using f-strings (Python 3.6+) or format() for embedded expressions

# 2. String Repetition

Definition: Creating new strings by repeating an existing string Method:

 Using the \* operator

**String Methods :**

1. Case Conversion Methods
   * upper() → Converts all characters to uppercase
   * lower() → Converts all characters to lowercase
   * title() → Converts first letter of each word to uppercase
   * capitalize() → Converts first character to uppercase, rest lowercase
   * swapcase() → Swaps uppercase to lowercase and vice versa
   * casefold() → Aggressive lowercase conversion (for case-insensitive comparisons)
2. Search & Validation Methods
   * find(sub) → Returns lowest index where substring is found (else -1)
   * index(sub) → Like find() but raises ValueError if not found
   * count(sub) → Counts occurrences of substring
   * startswith(prefix) → Checks if string starts with prefix
   * endswith(suffix) → Checks if string ends with suffix
   * isalnum() → Checks if all characters are alphanumeric
   * isalpha() → Checks if all characters are alphabetic

isdigit() → Checks if all characters are digits

* + isnumeric() → Checks if all characters are numeric (including Unicode)
  + isspace() → Checks if all characters are whitespace
  + islower() → Checks if all characters are lowercase
  + isupper() → Checks if all characters are uppercase
  + istitle() → Checks if string follows title case rules

1. Formatting & Cleaning Methods
   * strip([chars]) → Removes leading/trailing characters (default: whitespace)
   * lstrip([chars]) → Removes leading characters  rstrip([chars]) → Removes trailing characters
   * center(width[, fillchar]) → Centers string in given width
   * expandtabs(tabsize) → Replaces tabs with spaces
2. Transformation Methods
   * + replace(old, new[, count]) → Replaces occurrences of substring
     + split([sep[, maxsplit]]) → Splits string into list using separator
     + encode(encoding, errors) → Returns encoded version of string
     + translate(table) → Performs character-level translations

1. **String slicing.**

String slicing allows you to extract portions of strings by specifying start, stop, and step values. Here's a complete explanation:

**Basic Syntax**

string[start:stop:step]

# Key Components

1. Start Index (inclusive) - Where slicing begins (default: 0)
2. Stop Index (exclusive) - Where slicing ends (default: end of string)
3. Step - Interval between characters (default: 1)

# Common Use Cases

1. Extracting substrings
2. Reversing strings
3. Skipping characters
4. Processing fixed-width data formats
5. Removing prefixes/suffixes String slicing is:

* Zero-based indexing
* Does not modify original string (strings are immutable)
* Returns a new string object

This powerful feature works similarly for other sequence types like lists and tuples.