**Assignment 9 String Manipulation**

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

Strings in Python are sequences of characters and are one of the most commonly used data types..

**1. Creating a String**

You can create a string by enclosing characters in single quotes (') or double quotes (").

**Example:**

# Creating strings

string1 = 'Hello, World!'

string2 = "Python is fun!"

**2. Accessing Characters in a String**

You can access individual characters in a string using indexing. Python uses zero-based indexing, which means the first character is at index 0.

**Example:**

# Accessing characters

first\_char = string1[0] # 'H'

fifth\_char = string1[4] # 'o'

last\_char = string1[-1] # '!'

**3. Slicing Strings**

You can extract a substring (a part of the string) using slicing. The syntax is string[start:end], where start is the index of the first character and end is the index of the character just after the last character you want.

**Example:**

# Slicing strings

substring = string1[0:5] # 'Hello'

substring2 = string1[7:] # 'World!'

substring3 = string1[:5] # 'Hello'

**4 String Length:**

You can find the length of a string using the len() function.

Example:

# Finding the length of a string

length = len(string1) # 13

**5 String Concatenation:**

You can combine (concatenate) strings using the + operator.

**Example:**

# Concatenating strings

greeting = string1 + " " + string2 # 'Hello, World! Python is fun!'

**6. String Methods**

Python provides many built-in methods to manipulate strings. Here are a few common ones:

**lower():** Converts all characters to lowercase.

**upper():** Converts all characters to uppercase.

**strip():** Removes whitespace from the beginning and end of the string.

**replace(old, new):** Replaces occurrences of a substring with another substring.

**split(separator):** Splits the string into a list based on a separator.

**7 Formatting Strings**

You can format strings using f-strings (available in Python 3.6 and later) or the format() method.

**Example:**

name = "Alice"

age = 30

formatted\_string = f"My name is {name} and I am {age} years old." # 'My name is Alice and I am 30 years old.'

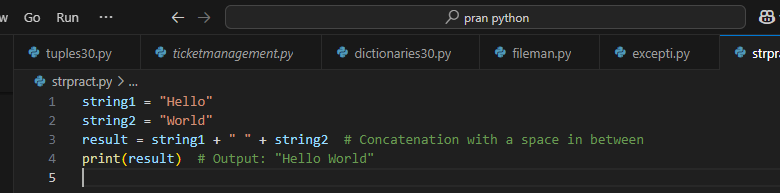
formatted\_string2 = "My name is {} and I am {} years old.".format(name, age) # 'My name is Alice and I am 30 years old.'

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

In Python, strings support various basic operations and methods that allow you to manipulate and work with text. Here's an overview of the operations you mentioned:

**1. Concatenation**

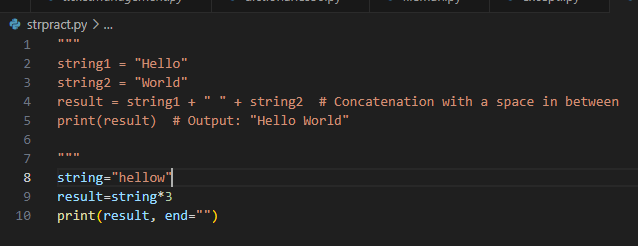
Concatenation refers to joining two or more strings together using the + operator.



### 2. ****Repetition****

Repetition allows you to repeat a string multiple times using the \* operator.

Example:



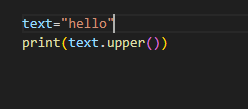
### 3. ****String Methods****

Python provides several useful methods for string manipulation. Some of the most common ones are:

#### upper()

Converts all characters in the string to uppercase.

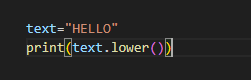
Example:



Lower()

Converts all characters in the string to lowercase

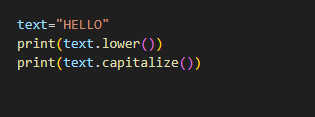
Example:



#### capitalize()

Capitalizes the first letter of the string and makes all other characters lowercase.

Example:



#### title()

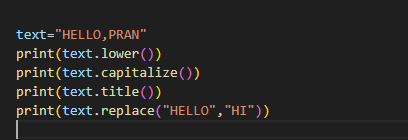
Capitalizes the first letter of each word in the string.

#### 

#### replace()

Replaces all occurrences of a substring with another substring.

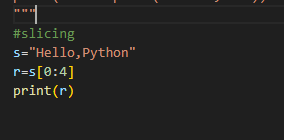
Example:



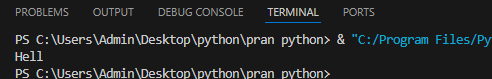
String slicing in python

**String slicing** in Python is a way to get specific parts of a string by using start, end, and step values**.** It’s especially useful for text manipulation and data parsing.

Example:



Output:



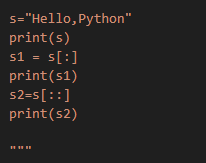
## Syntax of String Slicing in Python

*substring = s[start : end : step]*

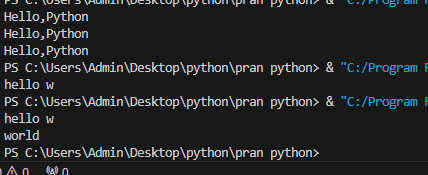
### Retrieve All Characters

To retrieve the entire string, use slicing without specifying any parameters.

Example:

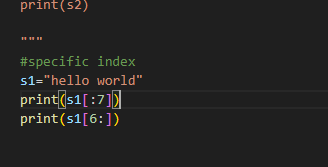


Output:



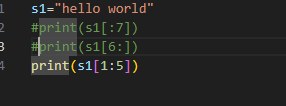
### Get All Characters Before or After a Specific Position

Example:



### Extract Characters Between Two Positions

Example:



### Get Characters at Specific Intervals

### Example:

