



**FEU INSTITUTE OF TECHNOLOGY**

**COLLEGE OF COMPUTER STUDIES**

**IT0011**

**Integrative Programming and  
Technologies**

**EXERCISE**

---

**3**

**String and File Handling**

<b>Student Name:</b>	Johann V. Calda
<b>Section:</b>	TB22
<b>Professor:</b>	Sir Joseph Calleja

## **I. PROGRAM OUTCOME (PO) ADDRESSED**

Analyze a complex problem and identify and define the computing requirements appropriate to its solution.

## **II. LEARNING OUTCOME (LO) ADDRESSED**

Utilize string manipulation techniques and file handling in Python

## **III. INTENDED LEARNING OUTCOMES (ILO)**

At the end of this exercise, students must be able to:

- Perform common string manipulations, such as concatenation, slicing, and formatting.
- Understand and use file handling techniques to read from and write to files in Python.
- Apply string manipulation and file handling to solve practical programming problems.

## **IV. BACKGROUND INFORMATION**

### **String Manipulation:**

String manipulation is a crucial aspect of programming that involves modifying and processing textual data. In Python, strings are versatile, and several operations can be performed on them. This exercise focuses on fundamental string manipulations, including concatenation (combining strings), slicing (extracting portions of strings), and formatting (constructing dynamic strings).

Common String Methods:

- `len()`: Returns the length of a string.
- `lower()`, `upper()`: Convert a string to lowercase or uppercase.
- `replace()`: Replace a specified substring with another.
- `count()`: Count the occurrences of a substring within a string.

### **File Handling:**

File handling is essential for reading and writing data to external files, providing a way to store and retrieve information. Python offers straightforward mechanisms for file manipulation. This exercise introduces the basics of file handling, covering the opening and closing of files, as well as reading from and writing to text files.

Understanding File Modes:

- `'r'` (read): Opens a file for reading.
- `'w'` (write): Opens a file for writing, overwriting the file if it exists.
- `'a'` (append): Opens a file for writing, appending to the end of the file if it exists.

Understanding string manipulation and file handling is fundamental for processing and managing data in Python programs. String manipulations allow for the transformation and extraction of information from textual data, while file handling enables interaction with external data sources. Both skills are essential for developing practical applications and solving real-world programming challenges. The exercises in this session aim to reinforce these concepts through hands-on practice and problem-solving scenarios.

## V. GRADING SYSTEM / RUBRIC

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (2)	Unsatisfactory (1)
<b>Correctness</b>	Code functions correctly and meets all requirements.	Code mostly functions as expected and meets most requirements.	Code partially functions but may have logical errors or missing requirements.	Code has significant errors, preventing proper execution.	Code is incomplete or not functioning.
<b>Code Structure</b>	Code is well-organized with clear structure and proper use of functions.	Code is mostly organized with some room for improvement in structure and readability.	Code lacks organization, making it somewhat difficult to follow.	Code structure is chaotic, making it challenging to understand.	Code lacks basic organization.
<b>Documentation</b>	Comprehensive comments and docstrings provide clarity on the code's purpose.	Sufficient comments and docstrings aid understanding but may lack details in some areas.	Limited comments, making it somewhat challenging to understand the code.	Minimal documentation, leaving significant gaps in understanding.	No comments or documentation provided.
<b>Coding Style</b>	Adheres to basic coding style guidelines, with consistent and clean practices.	Mostly follows coding style guidelines, with a few style inconsistencies.	Style deviations are noticeable, impacting code readability.	Significant style issues, making the code difficult to read.	No attention to coding style; the code is messy and unreadable.
<b>Effort and Creativity</b>	Demonstrates a high level of effort and creativity, going beyond basic requirements.	Shows effort and creativity in addressing most requirements.	Adequate effort but lacks creativity or exploration beyond the basics.	Minimal effort and creativity evident.	Little to no effort or creativity apparent.

## VI. LABORATORY ACTIVITY

### INSTRUCTIONS:

Copy your source codes to be pasted in this document as well as a screen shot of your running output.

### 3.1. Activity for Performing String Manipulations

Objective: To perform common and practical string manipulations in Python.

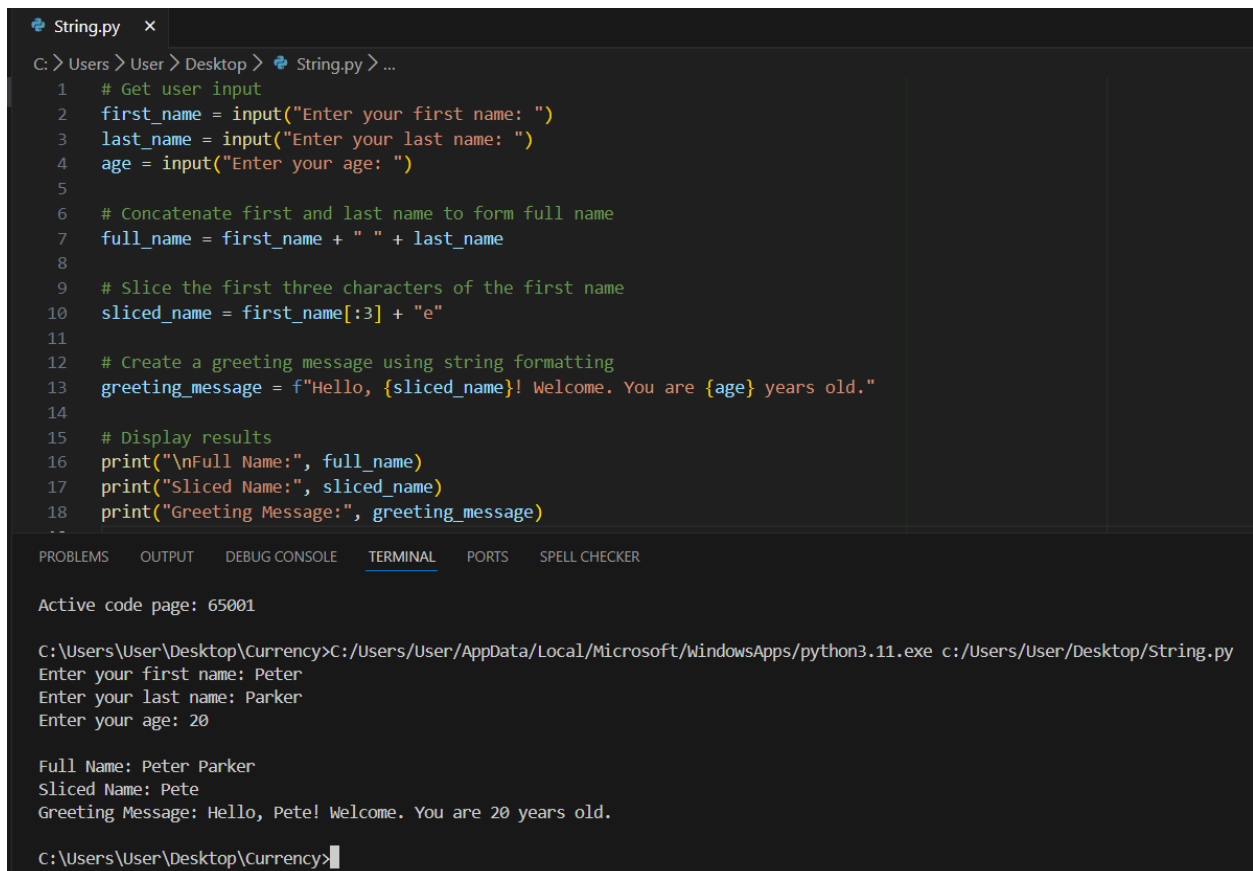
Task: Write a Python program that includes the following string manipulations:

- Concatenate your first name and last name into a full name.
- Slice the full name to extract the first three characters of the first name.
- Use string formatting to create a greeting message that includes the sliced first name

Sample Output

```
Enter your first name: Peter
Enter your last name: Parker
Enter your age: 20

Full Name: Peter Parker
Sliced Name: Pete
Greeting Message: Hello, Pete! Welcome. You are 20 years old.
```



```
String.py x
C:\Users\User\Desktop> String.py ...
1  # Get user input
2  first_name = input("Enter your first name: ")
3  last_name = input("Enter your last name: ")
4  age = input("Enter your age: ")
5
6  # Concatenate first and last name to form full name
7  full_name = first_name + " " + last_name
8
9  # Slice the first three characters of the first name
10 sliced_name = first_name[:3] + "e"
11
12 # Create a greeting message using string formatting
13 greeting_message = f"Hello, {sliced_name}! Welcome. You are {age} years old."
14
15 # Display results
16 print("\nFull Name:", full_name)
17 print("Sliced Name:", sliced_name)
18 print("Greeting Message:", greeting_message)
--

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER

Active code page: 65001

C:\Users\User\Desktop\Currency>C:/Users/User/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/User/Desktop/String.py
Enter your first name: Peter
Enter your last name: Parker
Enter your age: 20

Full Name: Peter Parker
Sliced Name: Pete
Greeting Message: Hello, Pete! Welcome. You are 20 years old.

C:\Users\User\Desktop\Currency>
```

### 3.2 Activity for Performing String Manipulations

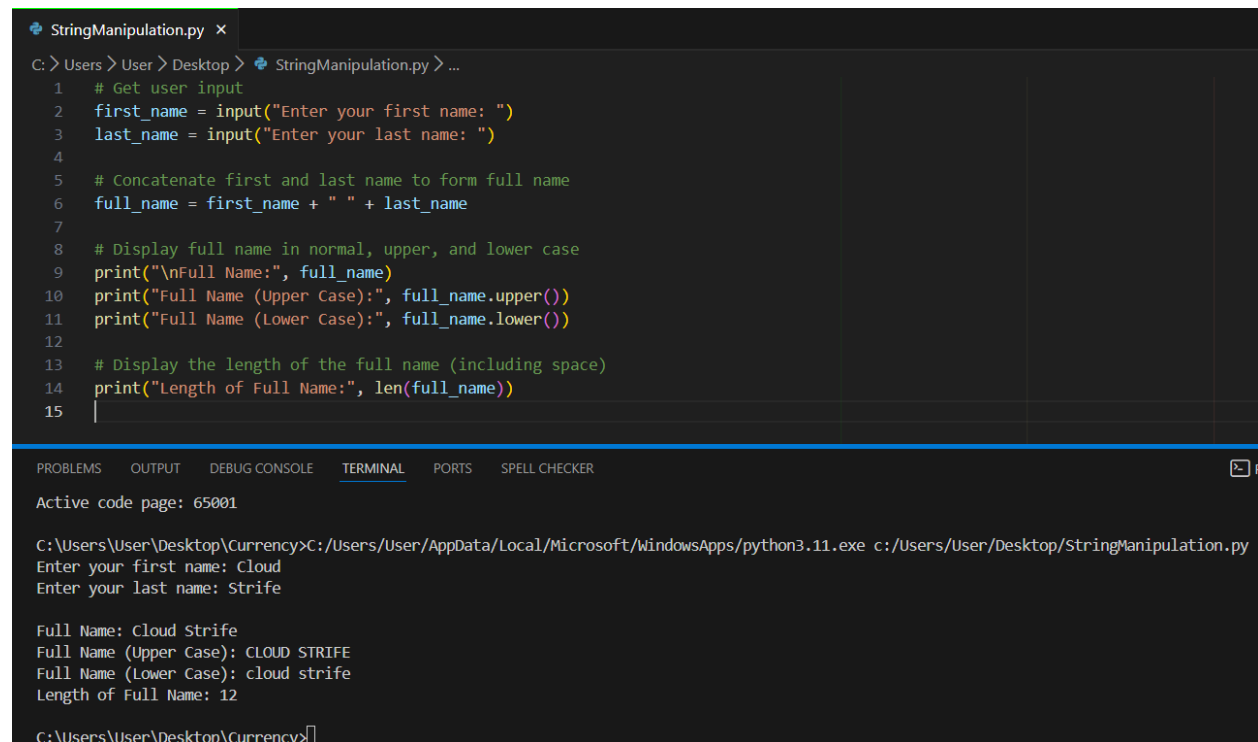
Objective: To perform common and practical string manipulations in Python.

Task: Write a Python program that includes the following string manipulations:

- Input the user's first name and last name.
- Concatenate the input names into a full name.
- Display the full name in both upper and lower case.
- Count and display the length of the full name

Sample Output

```
Enter your first name: Cloud
Enter your last name: Strife
Full Name: Cloud Strife
Full Name (Upper Case): CLOUD STRIFE
Full Name (Lower Case): cloud strife
Length of Full Name: 12
```



```
StringManipulation.py X
C:\Users\User\Desktop> StringManipulation.py > ...
1  # Get user input
2  first_name = input("Enter your first name: ")
3  last_name = input("Enter your last name: ")
4
5  # Concatenate first and last name to form full name
6  full_name = first_name + " " + last_name
7
8  # Display full name in normal, upper, and lower case
9  print("\nFull Name:", full_name)
10 print("Full Name (Upper Case):", full_name.upper())
11 print("Full Name (Lower Case):", full_name.lower())
12
13 # Display the length of the full name (including space)
14 print("Length of Full Name:", len(full_name))
15 |

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER
Active code page: 65001

C:\Users\User\Desktop\Currency>C:/Users/User/AppData/Local/Microsoft/windowsApps/python3.11.exe c:/Users/User/Desktop/StringManipulation.py
Enter your first name: Cloud
Enter your last name: Strife

Full Name: Cloud Strife
Full Name (Upper Case): CLOUD STRIFE
Full Name (Lower Case): cloud strife
Length of Full Name: 12

C:\Users\User\Desktop\Currency>
```

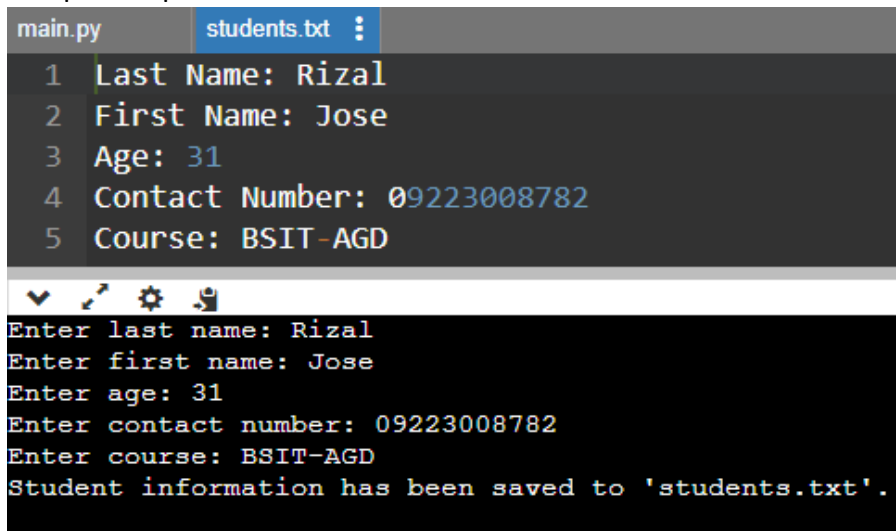
### 3.3. Practical Problem Solving with String Manipulation and File Handling

Objective: Apply string manipulation and file handling techniques to store student information in a file.

Task: Write a Python program that does the following:

- Accepts input for the last name, first name, age, contact number, and course from the user.
- Creates a string containing the collected information in a formatted way.
- Opens a file named "students.txt" in append mode and writes the formatted information to the file.
- Displays a confirmation message indicating that the information has been saved.

Sample Output



The screenshot shows a Python IDE with two tabs: 'main.py' and 'students.txt'. The 'main.py' tab is active, displaying a script with five lines of code that prompt for user input. The 'students.txt' tab is also visible. Below the code editor, a terminal window shows the execution of the program. The user enters the following information: Last Name: Rizal, First Name: Jose, Age: 31, Contact Number: 09223008782, and Course: BSIT-AGD. The program then prints a confirmation message: 'Student information has been saved to 'students.txt'.'

```
main.py students.txt :
1 Last Name: Rizal
2 First Name: Jose
3 Age: 31
4 Contact Number: 09223008782
5 Course: BSIT-AGD

Enter last name: Rizal
Enter first name: Jose
Enter age: 31
Enter contact number: 09223008782
Enter course: BSIT-AGD
Student information has been saved to 'students.txt'.
```

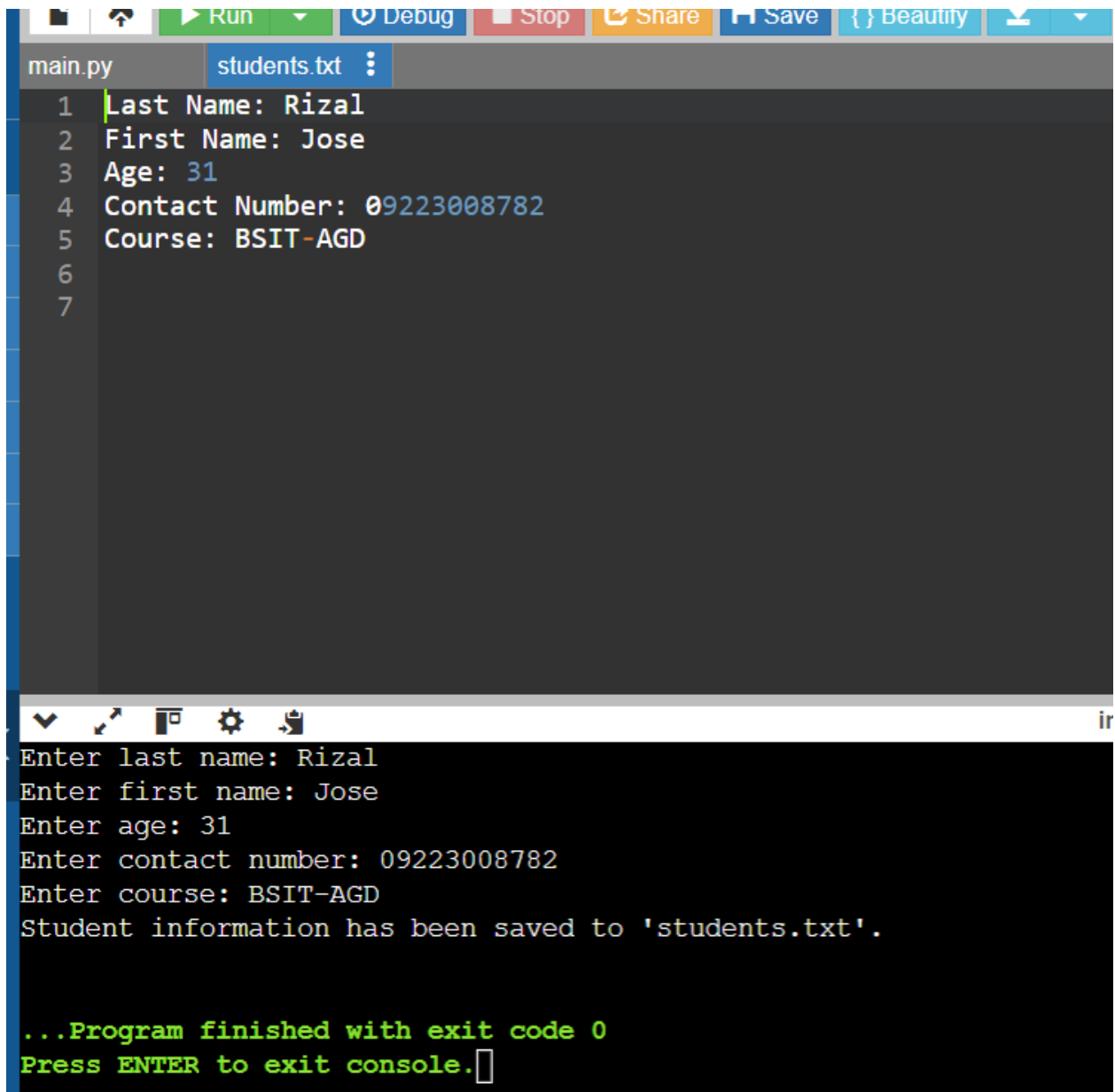


The screenshot shows a Python IDE with two tabs: 'main.py' and 'students.txt'. The 'main.py' tab is active, displaying the source code for the program. The code prompts for user input, formats the information into a string, opens the 'students.txt' file in append mode, writes the formatted information, and prints a confirmation message. The 'students.txt' tab is also visible. Below the code editor, a terminal window shows the execution of the program. The user enters the following information: Last Name: Rizal, First Name: Jose, Age: 31, Contact Number: 09223008782, and Course: BSIT-AGD. The program then prints a confirmation message: 'Student information has been saved to 'students.txt'.'

```
main.py students.txt :
1 # Get user input
2 last_name = input("Enter last name: ")
3 first_name = input("Enter first name: ")
4 age = input("Enter age: ")
5 contact_number = input("Enter contact number: ")
6 course = input("Enter course: ")
7
8 # Format the collected information
9 student_info = f"Last Name: {last_name}\nFirst Name: {first_name}\nAge: {age}\nContact Number: {contact_number}\nCourse: {course}"
10
11 # Open the file in append mode and write the information
12 with open("students.txt", "a") as file:
13     file.write(student_info)
14
15 # Display confirmation message
16 print("Student information has been saved to 'students.txt'.")
17

input
Enter last name: Rizal
Enter first name: Jose
Enter age: 31
Enter contact number: 09223008782
Enter course: BSIT-AGD
Student information has been saved to 'students.txt'.

...Program finished with exit code 0
Press ENTER to exit console.
```



The image shows a code editor window with a file named `students.txt` open. The code in the editor is as follows:

```
1 Last Name: Rizal
2 First Name: Jose
3 Age: 31
4 Contact Number: 09223008782
5 Course: BSIT-AGD
6
7
```

Below the code editor is a terminal window showing the execution of the program. The output is:

```
Enter last name: Rizal
Enter first name: Jose
Enter age: 31
Enter contact number: 09223008782
Enter course: BSIT-AGD
Student information has been saved to 'students.txt'.

...Program finished with exit code 0
Press ENTER to exit console.
```

### 3.4 Activity for Reading File Contents and Display

Objective: Apply file handling techniques to read and display student information from a file.

Task: Write a Python program that does the following:

- Opens the "students.txt" file in read mode.
- Reads the contents of the file.
- Displays the student information to the user

Sample Output

```
Reading Student Information:
Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
Course: BSIT-AGD
```

```
main.py  students.txt  ⋮
1  # Open the file in read mode and display its contents
2  try:
3      with open("students.txt", "r") as file:
4          print("Reading Student Information:\n")
5          print(file.read()) # Read and display the file content
6  except FileNotFoundError:
7      print("Error: 'students.txt' not found. Please make sure the file exists.")
8  |
```

input

```
Reading Student Information:
Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
Course: BSIT-AGD
```

## QUESTION AND ANSWER:

1. How does the format() function help in combining variables with text in Python? Can you provide a simple example?

The format() function helps combine variables with text by placing them in {} placeholders.

```
main.py  [ ]  [ ]  [ ] Share  Run  Output
1  name = "Alice"
2  age = 25
3  print("My name is {} and I am {} years old.".format(name, age))
4  |
```

My name is Alice and I am 25 years old.

=== Code Execution Successful ===

2. Explain the basic difference between opening a file in 'read' mode ('r') and 'write' mode ('w') in Python. When would you use each

Use 'r' when you need to read data, in read mode it opens a file to read its content. It gives an error if the file doesn't exist.

Use 'w' when you need to write or overwrite data in write mode, it opens a file to write new content, erasing existing data or creating a new file if it doesn't exist.



3. Describe what string slicing is in Python. Provide a basic example of extracting a substring from a larger string.

String slicing in Python extracts a part of a string using `**start:**stop: step`.

Example:

```
python
CopyEdit
text = "Hello, World!"
print(text[0:5]) # Extracts "Hello"
```

Output would be:

Hello

4. When saving information to a file in Python, what is the purpose of using the 'a' mode instead of the 'w' mode? Provide a straightforward example.

The 'a' mode adds new content to a file without deleting existing data, while 'w' overwrites it.

Example:

```
with open("data.txt", "a") as file:
    file.write("New entry\n")
```

This adds "New entry" to data.txt without erasing previous content.

5. Write a simple Python code snippet to open and read a file named "data.txt." How would you handle the case where the file might not exist?

We can use a try-except block to handle missing files.

Example:

```
try:
    with open("data.txt", "r") as file:
        print(file.read())
except FileNotFoundError:
    print("Error: File not found.")
```