



FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011

**Integrative Programming and
Technologies**

EXERCISE

3

String and File Handling

Student Name:	John Yeoj D. Bongon
Section:	TC21
Professor:	Mr. 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)
Correctness	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.
Code Structure	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.
Documentation	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.
Coding Style	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.
Effort and Creativity	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 screenshot 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.
```

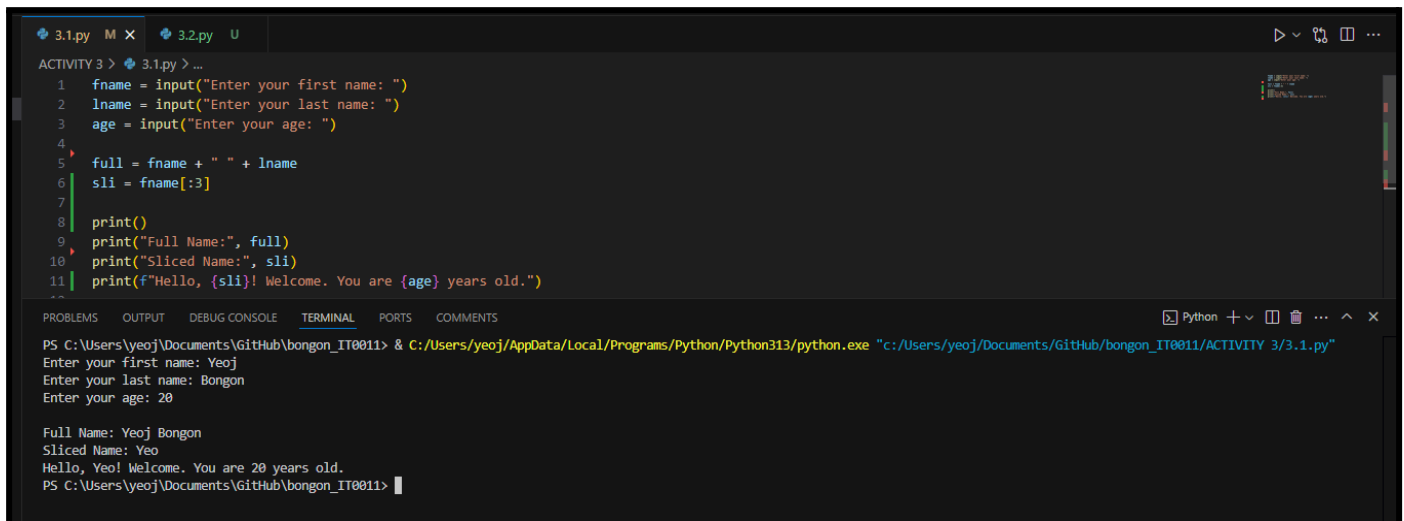
Source code:

```
fname = input("Enter your first name: ")
lname = input("Enter your last name: ")
age = input("Enter your age: ")

full = fname + " " + lname
sli = fname[:3]

print()
print("Full Name:", full)
print("Sliced Name:", sli)
print(f"Hello, {sli}! Welcome. You are {age} years old.")
```

Output:



```
3.1.py M X 3.2.py U
ACTIVITY 3 > 3.1.py > ...
1 fname = input("Enter your first name: ")
2 lname = input("Enter your last name: ")
3 age = input("Enter your age: ")
4
5 full = fname + " " + lname
6 sli = fname[:3]
7
8 print()
9 print("Full Name:", full)
10 print("Sliced Name:", sli)
11 print(f"Hello, {sli}! Welcome. You are {age} years old.")
12
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Users\yeoj\Documents\GitHub\bongon_IT0011> & C:/Users/yeoj/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/yeoj/Documents/GitHub/bongon_IT0011/ACTIVITY 3/3.1.py"
Enter your first name: Yeoj
Enter your last name: Bongon
Enter your age: 20

Full Name: Yeoj Bongon
Sliced Name: Yeo
Hello, Yeo! Welcome. You are 20 years old.
PS C:\Users\yeoj\Documents\GitHub\bongon_IT0011>
```

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

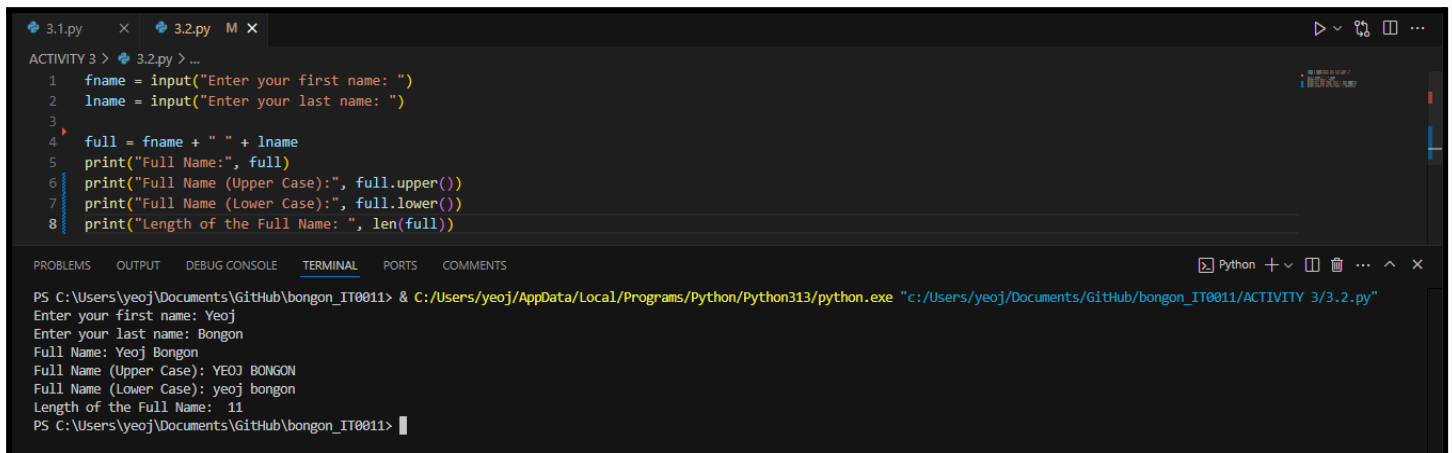
```
Enter your first name: Cloud
Enter your last name: Strife
Full Name: Cloud Strife
Full Name (Upper Case): CLOUD STRIFE
Full Name (Lower Case): cloud strife
Length of Full Name: 12
```

Source Code:

```
fname = input("Enter your first name: ")
lname = input("Enter your last name: ")

full = fname + " " + lname
print("Full Name:", full)
print("Full Name (Upper Case):", full.upper())
print("Full Name (Lower Case):", full.lower())
print("Length of the Full Name: ", len(full))
```

Output:



```
3.1.py 3.2.py M X
ACTIVITY 3 > 3.2.py > ...
1 fname = input("Enter your first name: ")
2 lname = input("Enter your last name: ")
3
4 full = fname + " " + lname
5 print("Full Name:", full)
6 print("Full Name (Upper Case):", full.upper())
7 print("Full Name (Lower Case):", full.lower())
8 print("Length of the Full Name: ", len(full))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Users\yeoj\Documents\GitHub\bongon_IT0011> & C:/Users/yeoj/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/yeoj/Documents/GitHub/bongon_IT0011/ACTIVITY 3/3.2.py"
Enter your first name: YeoJ
Enter your last name: Bongon
Full Name: YeoJ Bongon
Full Name (Upper Case): YEOJ BONGON
Full Name (Lower Case): yeoj bongon
Length of the Full Name: 11
PS C:\Users\yeoj\Documents\GitHub\bongon_IT0011>
```

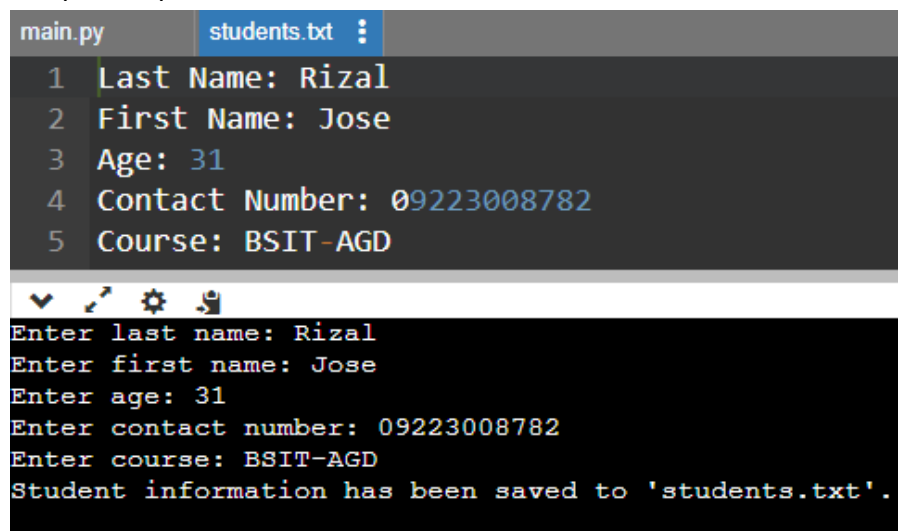
3.3. Practical Problem Solving with String Manipulation and File Handling

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



```
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'.
```

Source Code:

```
import os
os.makedirs("ACTIVITY 3", exist_ok=True)

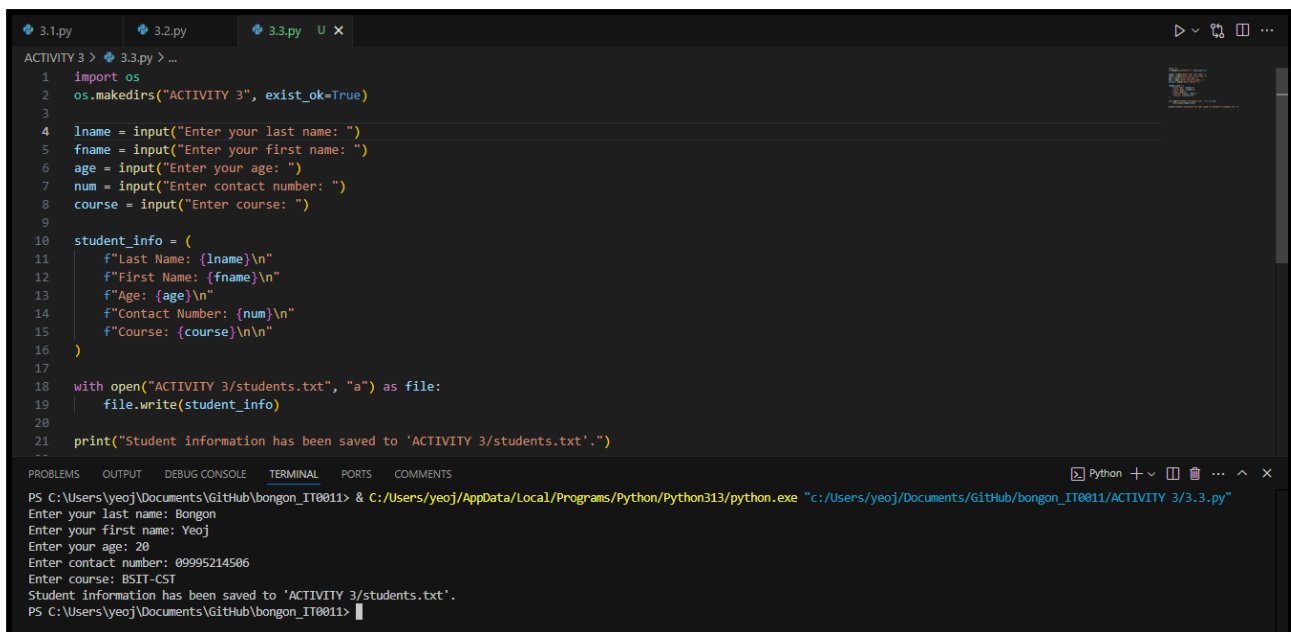
lname = input("Enter your last name: ")
fname = input("Enter your first name: ")
age = input("Enter your age: ")
num = input("Enter contact number: ")
course = input("Enter course: ")

student_info = (
    f"Last Name: {lname}\n"
    f"First Name: {fname}\n"
    f"Age: {age}\n"
    f"Contact Number: {num}\n"
    f"Course: {course}\n\n"
)

with open("ACTIVITY 3/students.txt", "a") as file:
    file.write(student_info)

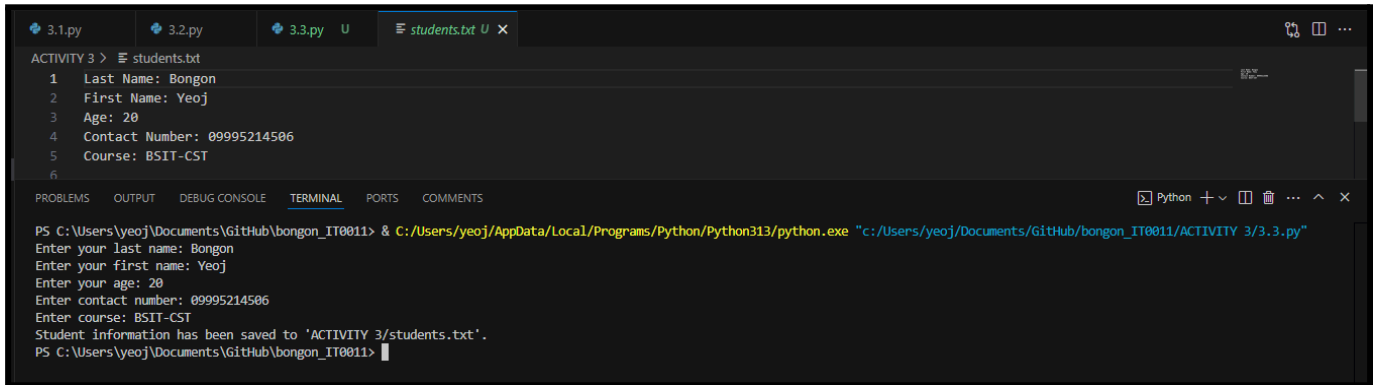
print("Student information has been saved to 'ACTIVITY 3/students.txt'.")
```

Output:



The screenshot displays a code editor with a dark theme. The top pane shows the Python script from the 'Source Code' section. The bottom pane is a terminal window with the following output:

```
PS C:\Users\yeoj\Documents\GitHub\bongon_IT0011> & C:/Users/yeoj/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/yeoj/Documents/GitHub/bongon_IT0011/ACTIVITY 3/3.3.py"
Enter your last name: Bongon
Enter your first name: Yeoj
Enter your age: 20
Enter contact number: 09995214506
Enter course: BSIT-CST
Student information has been saved to 'ACTIVITY 3/students.txt'.
PS C:\Users\yeoj\Documents\GitHub\bongon_IT0011>
```



The screenshot shows a VS Code editor with a file named `students.txt` open. The file contains the following text:

```
1 Last Name: Bongon
2 First Name: Yeoj
3 Age: 20
4 Contact Number: 09995214506
5 Course: BSIT-CST
6
```

Below the editor, the TERMINAL pane shows the command prompt output of running a Python script:

```
PS C:\Users\yeoj\Documents\Github\bongon_IT0011> & C:/Users/yeoj/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/yeoj/Documents/Github/bongon_IT0011/ACTIVITY 3/3.3.py"
Enter your last name: Bongon
Enter your first name: Yeoj
Enter your age: 20
Enter contact number: 09995214506
Enter course: BSIT-CST
Student information has been saved to 'ACTIVITY 3/students.txt'.
PS C:\Users\yeoj\Documents\Github\bongon_IT0011>
```

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

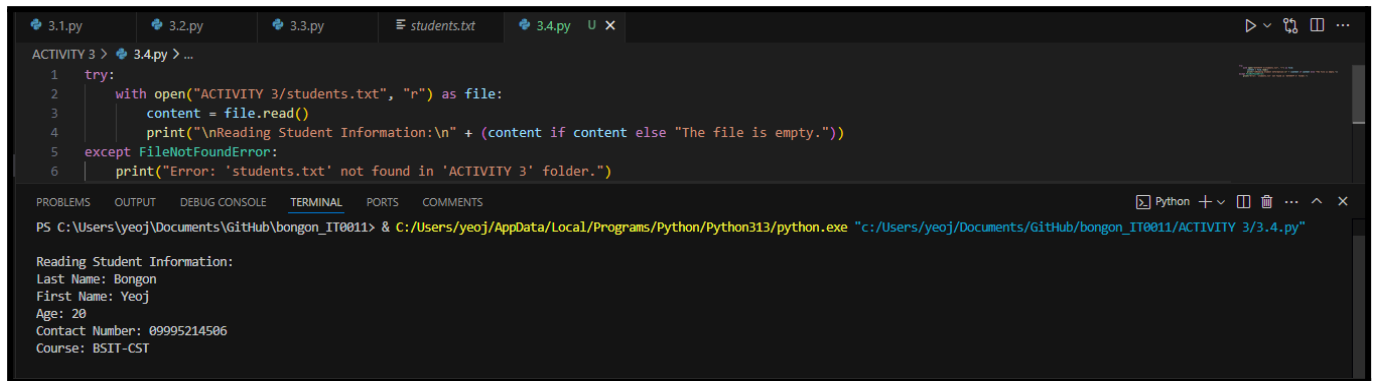
Sample Output

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

Source Code:

```
try:
    with open("ACTIVITY 3/students.txt", "r") as file:
        content = file.read()
        print("\nReading Student Information:\n" + (content if content else "The file is empty."))
except FileNotFoundError:
    print("Error: 'students.txt' not found in 'ACTIVITY 3' folder.")
```


Output:



The screenshot shows a Python IDE with a file named `3.4.py` open. The code in the editor is as follows:

```
1 try:
2     with open("ACTIVITY 3/students.txt", "r") as file:
3         content = file.read()
4         print("\nReading Student Information:\n" + (content if content else "The file is empty."))
5 except FileNotFoundError:
6     print("Error: 'students.txt' not found in 'ACTIVITY 3' folder.")
```

The terminal output at the bottom of the IDE shows the execution result:

```
PS C:\Users\yeoj\Documents\GitHub\bongon_IT0011> & C:\Users\yeoj\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/yeoj/Documents/GitHub/bongon_IT0011/ACTIVITY 3/3.4.py"

Reading Student Information:
Last Name: Bongon
First Name: Yeoj
Age: 20
Contact Number: 09995214506
Course: BSIT-CST
```

QUESTION AND ANSWER:

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

Answer. In Python, the `format()` function is a convenient way to dynamically create formatted strings based on data stored in variables; it functions essentially as a string concatenation tool with more formatting flexibility by allowing you to combine variables with text within a string by using placeholder curly braces (`{}`) where you want the variable values to be inserted.

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

Answer. When you open a file in Python in read mode ('r'), you can read its contents without changing it. There will be an error if the file is not found. When retrieving and displaying stored data—like reading student records from a file—this mode is helpful. However, to create or overwrite a file, use write mode ('w'). It will be produced if the provided file doesn't already exist. On the other hand, if the file already has data in it, opening it in write mode will remove everything before adding new data. This mode comes in handy when you wish to store new data or reset a file. For instance, the stored student data will be shown if you open `students.txt` in read mode. However, all prior records will be erased if you access the same file in write mode and add a new student record. Therefore, whether you wish to replace current data with fresh content or maintain it will determine which option is best.

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

Answer. In Python, the syntax `string[start:end]`, where start is the beginning index and end is the stopping index (not included), is used to slice a string into segments. `Text[0:5]` will extract "Hello" from `text = "Hello World"` since it takes characters from index 0 to 4.

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.

Answer. In Python, the 'w' mode overwrites the file and eliminates any prior data, whereas the 'a' mode adds new information to a file without erasing its existing content. Append mode is helpful for keeping numerous records because, for instance, if you open a file in this mode and add new student information, it will be added to the file without impacting the earlier entries.

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?

Answer.

try:

```
with open("data.txt", "r") as file:
    content = file.read()
    print(content if content else "The file is empty.")
```

except FileNotFoundError:

```
    print("Error: 'data.txt' not found.")
```