



**FEU INSTITUTE OF TECHNOLOGY**

**COLLEGE OF COMPUTER STUDIES**

**IT0011**

**Integrative Programming and  
Technologies**

**EXERCISE**

---

**3**

**String and File Handling**

<b>Student Name:</b>	Camacho, Ann Valerie
<b>Section:</b>	TB22
<b>Professor:</b>	Prof. 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.
```

Source Code:

```
StringManip.py X
C: > Users > User > Downloads > StringManip.py > ...
1  #This asks the user to input their first name, last name, and age.
2  first_name = input("Enter your first name: ")
3  last_name = input("Enter your last name: ")
4  age = input("Enter your age: ")
5
6  #This concatenates the first name and last name into a full name
7  full_name = first_name + " " + last_name
8
9  #This slices the first name
10 sliced_name = first_name[:3] # Extracting the first three characters
11
12 #This creates a greeting message
13 greeting_message = f"Hello, {sliced_name}! Welcome. You are {age} years old."
14
15 #This prints all the results
16 print("\nFull Name:", full_name)
17 print("Sliced Name:", sliced_name)
18 print("Greeting Message:", greeting_message)
19
```

## Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\User> & C:/msys64/ucrt64/bin/python.exe c:/Users/User/Downloads/StringManip.py
Enter your first name: Valerie
Enter your last name: Camacho
Enter your age: 21

Full Name: Valerie Camacho
Sliced Name: Val
Greeting Message: Hello, Val! Welcome. You are 21 years old.
PS C:\Users\User> 
```

### 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:

```
StringManip.py x StringManip2.py •
C: > Users > User > Downloads > StringManip2.py > ...
1  #This asks the user to input their first and last name
2  first_name = input("Enter your first name: ")
3  last_name = input("Enter your last name: ")
4
5  #This concatenates the first name and last name into a full name
6  full_name = first_name + " " + last_name
7
8  # Convert full name to upper and lower case
9  upper_case_name = full_name.upper()
10 lower_case_name = full_name.lower()
11
12 #This calculates the length of the full name including whitespaces
13 name_length = len(full_name)
14
15 #This prints all the results
16 print("\nFull Name:", full_name)
17 print("Full Name (Upper Case):", upper_case_name)
18 print("Full Name (Lower Case):", lower_case_name)
19 print("Length of Full Name:", name_length)
20
```

## Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\User> & C:/msys64/ucrt64/bin/python.exe c:/Users/User/Downloads/StringManip2.py
Enter your first name: Valerie
Enter your last name: Camacho

Full Name: Valerie Camacho
Full Name (Upper Case): VALERIE CAMACHO
Full Name (Lower Case): valerie camacho
Length of Full Name: 15
PS C:\Users\User> █
```

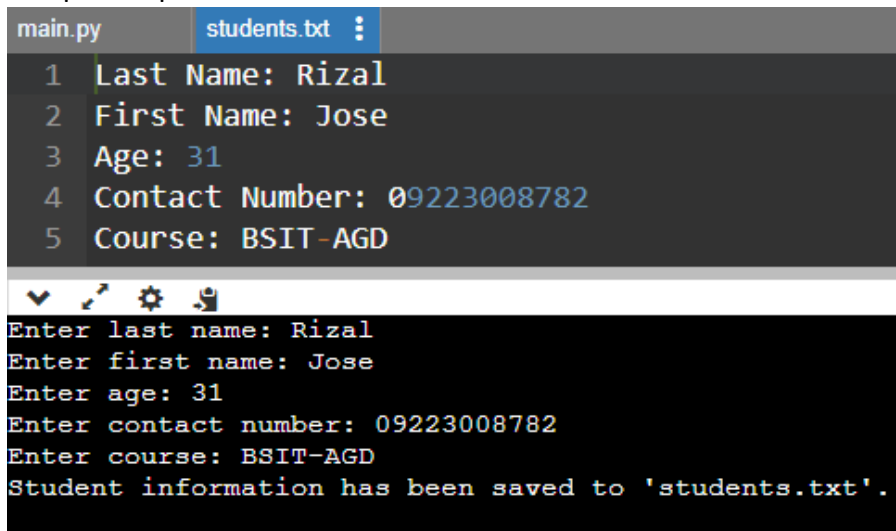
### 3.3. Practical Problem Solving with String Manipulation and File Handling

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 code editor with two tabs: 'main.py' and 'students.txt'. The 'main.py' tab is active, displaying a Python script with five lines of code. Below the code editor is a terminal window showing the program's execution. The user enters the following information: Last Name: Rizal, First Name: Jose, Age: 31, Contact Number: 09223008782, and Course: BSIT-AGD. The program then displays 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'.
```



## Source Code:

```
Welcome | StringManip3.py X | StringManip2.py
Val > StringManip3.py > ...
1  # This asks the user to input their first name, Last name, Age, Contact No#, and Course
2
3  last_name = input("Enter last name: ")
4  first_name = input("Enter first name: ")
5  age = input("Enter age: ")
6  contact_number = input("Enter contact number: ")
7  course = input("Enter course: ")
8
9  # This formats the user input
10 student_info = (
11     f"Last Name: {last_name}\n"
12     f"First Name: {first_name}\n"
13     f"Age: {age}\n"
14     f"Contact Number: {contact_number}\n"
15     f"Course: {course}\n"
16     "-----\n"
17 )
18
19 # This opens the file for appending and writes the information to the txt file
20 with open("students.txt", "a") as file:
21     file.write(student_info)
22
23 # This displays the confirmation message
24 print("\nStudent information has been saved to 'students.txt'.")
25
```

## Output:

```
PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | COMMENTS
Python + v [Icons] [Close]

PS C:\Users\User\Documents\ADVANCE WEB DES> & C:/msys64/ucrt64/bin/python.exe "c:/Users/User/Documents/ADVANCE WEB DES/Val/StringManip3.py"
Enter last name: Camacho
Enter first name: Valerie
Enter age: 21
Enter contact number: 0999-999-9999
Enter course: BSIT-BA

Student information has been saved to 'students.txt'.
PS C:\Users\User\Documents\ADVANCE WEB DES>

StringManip3.py X | students.txt X | StringManip2.py

students.txt
1  Last Name: Camacho
2  First Name: Valerie
3  Age: 21
4  Contact Number: 0999-999-9999
5  Course: BSIT-BA
6  -----
7
```

### 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:

```
StringManip4.py > ...
1  # This opens the file in read mode
2
3  with open("students.txt", "r") as file:
4      # This reads the contents of the file
5      student_data = file.read()
6
7      # This displays the student information
8  print("\nReading Student Information:\n")
9  print(student_data)
10
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
PS C:\Users\User\Documents\ADVANCE WEB DES> & C:/msys64/ucrt64/bin/python.exe "c:/Users/User/Documents/ADVANCE WEB DES/StringManip4.py"

Reading Student Information:

Last Name: Camacho
First Name: Valerie
Age: 21
Contact Number: 0999-999-9999
Course: BSIT-BA
-----
```

## 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 allows you to insert variables into a string in a structured way. It replaces placeholders {} with specified values, making string formatting more readable.

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

'r' is used to read an existing file. If the file doesn't exist, an error occurs. 'w' is used to create or overwrite a file. If the file exists, its contents are erased. 'r' is used whenever you need to retrieve and process stored data and 'w' is used for creating new files or replacing existing content.

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

String slicing allows you to extract a portion of a string using the syntax string[start:end].

Example:

```
example.py > ...
1 text = "Hello, World!"
2 substring = text[0:5] # Extracts characters from index 0 to 4
3 print(substring)
4
```

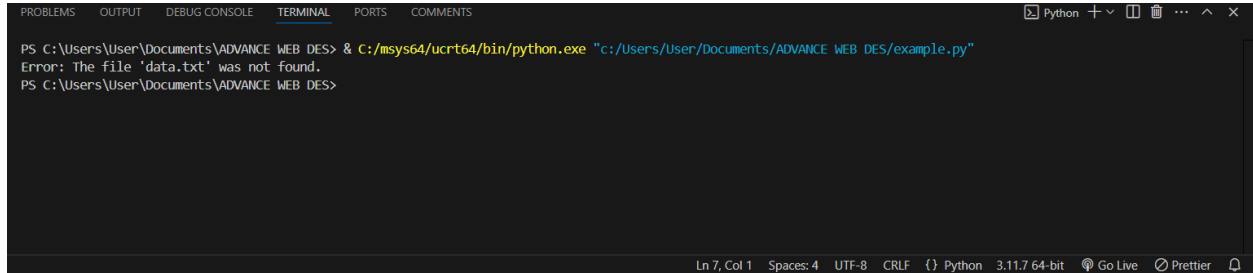
4. When saving information to a file in Python, what is the purpose of using the 'a' mode instead of the 'w' mode? Provide a straightforward example.

'a' adds new data to a file without deleting existing content. 'w' however, overwrites the file.

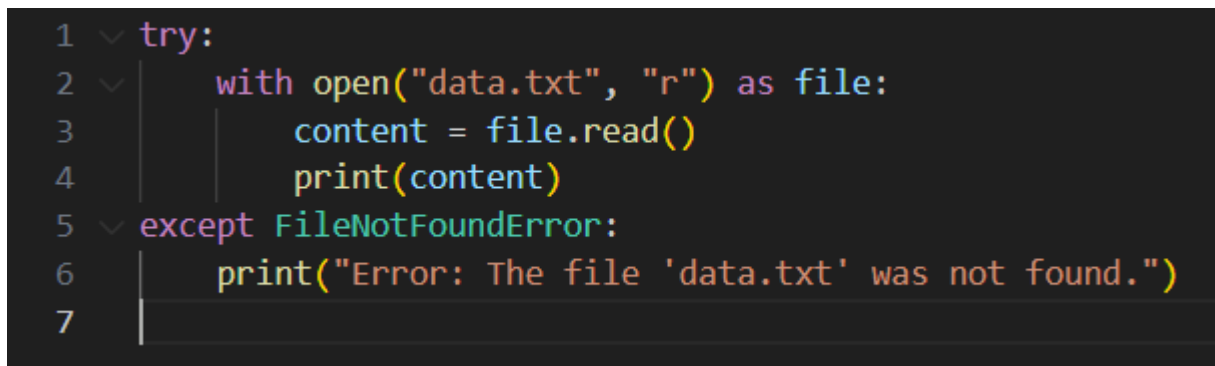
```
example.py > ...
1 with open("sample.txt", "a") as file:
2     file.write("New log entry\n")
3
```

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?

The 'try:' statement is needed to handle potential errors that may occur while attempting to open and read the file, and the 'except FileNotFoundError' is used to catch if a file is missing and an error message will be displayed instead.

A screenshot of a terminal window with a dark background. The terminal shows a command prompt where a Python script was executed. The output shows an error message: "Error: The file 'data.txt' was not found." The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and COMMENTS. The status bar at the bottom indicates the file is at line 7, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and it's a Python 3.11.7 64-bit file. There are also icons for Go Live and Prettier.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
Python + - [ ] ... ^ x
PS C:\Users\User\Documents\ADVANCE WEB DES> & C:/msys64/ucrt64/bin/python.exe "c:/Users/User/Documents/ADVANCE WEB DES/example.py"
Error: The file 'data.txt' was not found.
PS C:\Users\User\Documents\ADVANCE WEB DES>
Ln 7, Col 1  Spaces: 4  UTF-8  CRLF  {} Python  3.11.7 64-bit  Go Live  Prettier
```

A screenshot of a code editor showing a Python script. The code uses a try-except block to handle a FileNotFoundError. The try block attempts to open 'data.txt' in read mode, read its content, and print it. The except block catches the FileNotFoundError and prints a custom error message: "Error: The file 'data.txt' was not found." The code is syntax-highlighted with a dark background.

```
1 try:
2     with open("data.txt", "r") as file:
3         content = file.read()
4         print(content)
5 except FileNotFoundError:
6     print("Error: The file 'data.txt' was not found.")
7
```