



FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011

**Integrative Programming and
Technologies**

EXERCISE

3

String and File Handling

Student Name:	Rona Mae L. Serrano
Section:	TB22
Professor:	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)
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 screen shot of your running output.

3.1. Activity for Performing String Manipulations

Source Code:

```
first_name = input("Enter your first name:")
last_name = input("Enter your last name:")
age = input("Enter your age:")

# Concatenate your first name and last name
full_name = first_name + " " + last_name

# Slice the full name to extract the first three characters of the first name
sliced_name = first_name[:4]

# create a greeting message that includes the sliced first name
greeting = f"Hello, {sliced_name}! Welcome. You are {age} years old."

# print the result
print("\nFull Name:", full_name)
print("Sliced Name:", sliced_name)
print("Greeting:", greeting)
```

Output:

```
Enter your first name:Rona Mae
Enter your last name:Serrano
Enter your age:21

Full Name: Rona Mae  Serrano
Sliced Name: Rona
Greeting: Hello, Rona! Welcome. You are 21 years old.
```

3.2 Activity for Performing String Manipulations

Source Code:

```
first_name = input("Enter your first name:")
last_name = input("Enter your last name:")

# Concatenate your first name and last name into a full name
full_name = first_name + " " + last_name

print("Full Name:", full_name)

#display the full name in UPPERCASE
print("Full Name (Upper Case):", full_name.upper())
#display the full name in lowercase
print("Full Name (Lower Case):", full_name.lower())

#count and display the length of the full name
print("Full Name Length:", len(full_name))
```

Output:

```
Enter your first name:Rona Mae
Enter your last name:Serrano
Full Name: Rona Mae Serrano
Full Name (Upper Case): RONA MAE SERRANO
Full Name (Lower Case): rona mae serrano
Full Name Length: 16
```

3.3. Practical Problem Solving with String Manipulation and File Handling

Source Code:

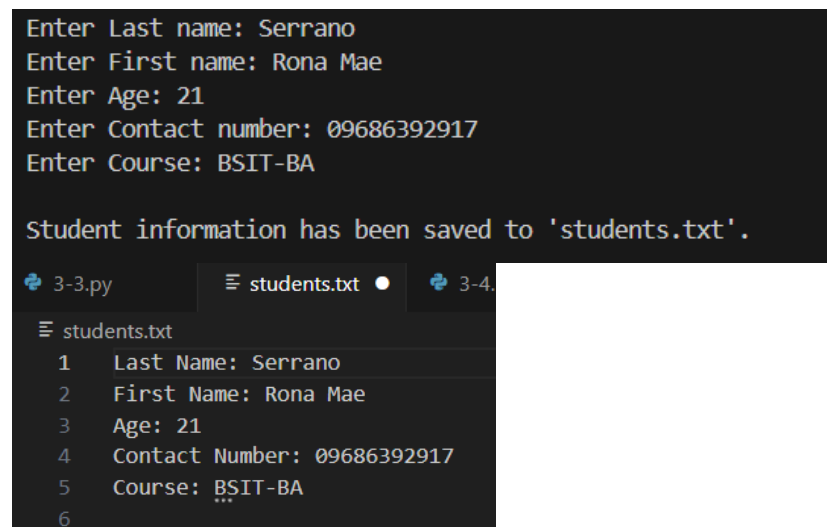
```
# Accept input for student details
last_name = input("Enter Last name: ")
first_name = input("Enter First name: ")
age = input("Enter Age: ")
contact_number = input("Enter Contact number: ")
course = input("Enter Course: ")

# Create a formatted string with student details
student_info = f"""
Last Name: {last_name}
First Name: {first_name}
Age: {age}
Contact Number: {contact_number}
Course: {course}
"""

# Open the file in append mode and write student info
f = open("Activity_03/students.txt", "a") # Using explicit file handling
f.write(student_info)
f.close() # Ensure the file is properly closed

# Confirmation message
print("\nStudent information has been saved to 'students.txt'.")
```

Output:



```
Enter Last name: Serrano
Enter First name: Rona Mae
Enter Age: 21
Enter Contact number: 09686392917
Enter Course: BSIT-BA

Student information has been saved to 'students.txt'.
```

```
3-3.py students.txt 3-4.
students.txt
1 Last Name: Serrano
2 First Name: Rona Mae
3 Age: 21
4 Contact Number: 09686392917
5 Course: BSIT-BA
6
```

3.4 Activity for Reading File Contents and Display

Source Code:

```
try:
    f = open("Activity_03/students.txt", "r") # Open the file in read mode
    student_info = f.read().strip() # Read the content and remove leading/trailing whitespace
    f.close() # Close the file explicitly

    if not student_info: # Check if the file is empty
        print("No student information found in 'students.txt'.")
    else:
        print("\nReading Student Information:")
        print(student_info)

except FileNotFoundError: # Handle case when the file does not exist
    print("Error: 'students.txt' not found.")
except PermissionError:
    print("Error: You don't have permission to access 'students.txt'.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

Output:

```
Reading student Information:
Last Name: Serrano
First Name: Rona Mae
Age: 21
Contact Number: 09686839619
Course: BSIT
```

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 enables the seamless insertion of variables into a string by replacing {} placeholders with actual values, improving readability and flexibility.

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' (read mode) allows access to a file's content but results in an error if the file is missing.
- 'w' (write mode) creates a new file or replaces an existing one, erasing any prior content.

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

String slicing refers to extracting a specific portion of a string using index values, allowing precise selection of characters within a given range.

Example:

```
text = "Hello, Python!"  
substring = text[7:13]  
print(substring)
```

Output:

Python

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' (append mode) preserves existing data while adding new content to a file, whereas 'w' completely overwrites any previous content.

Example:

```
with open("example.txt", "a") as file:  
    file.write("This is new content.\n")
```

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?

A try-except block can be used to safely open and read the file, preventing errors if the file is missing.

Example:

```
try:  
    with open("data.txt", "r") as file:  
        content = file.read() print(content)  
except FileNotFoundError:  
    print("Error: The file 'data.txt' does not exist.")
```