



**FEU INSTITUTE OF TECHNOLOGY**

**COLLEGE OF COMPUTER STUDIES**

**IT0011**

**Integrative Programming and  
Technologies**

**EXERCISE**

---

**3**

**String and File Handling**

<b>Student Name:</b>	Mike Deniel R. Acosta
<b>Section:</b>	TB22

<b>Professor:</b>	Mr. 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:

```
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 four characters of the first name
```

```
sliced_name = first_name[:4]
```

```
# Greeting message that includes the sliced first name
```

```
greeting = f"Hello, {sliced_name}! Welcome. You are {age} years old."
```

```
#Result
```

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

Source Code:

Sample Output:

```
Enter your first name: Mike Deniel
Enter your last name: Acosta
Enter your age: 21

Full Name: Mike Deniel Acosta
Sliced Name: Mike
Greeting: Hello, Mike! Welcome. You are 21 years old.
PS C:\Users\Cheetos\Documents\Practice Coding\Activity 03> █
```

### 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:

```
# Get user input for first and last name
first_name = input("Enter your first name: ").strip()
last_name = input("Enter your last name: ").strip()

# Concatenate the first and last name into a full name
full_name = f"{first_name} {last_name}"

# Display full name
print("\nFull Name:", full_name)

# Display full name in uppercase
print("Full Name (Uppercase):", full_name.upper())

# Display full name in lowercase
print("Full Name (Lowercase):", full_name.lower())

# Count and display the length of the full name (excluding spaces)
name_length = len(full_name.replace(" ", ""))
print("Full Name Length (without spaces):", name_length)
```

Sample Output:

```
Enter your first name: Mike Deniel
Enter your last name: Acosta

Full Name: Mike Deniel Acosta
Full Name (Uppercase): MIKE DENIEL ACOSTA
Full Name (Lowercase): mike deniel acosta
Full Name Length (without spaces): 16
PS C:\Users\Cheetos\Documents\Practice Coding\Activity 03> |
```

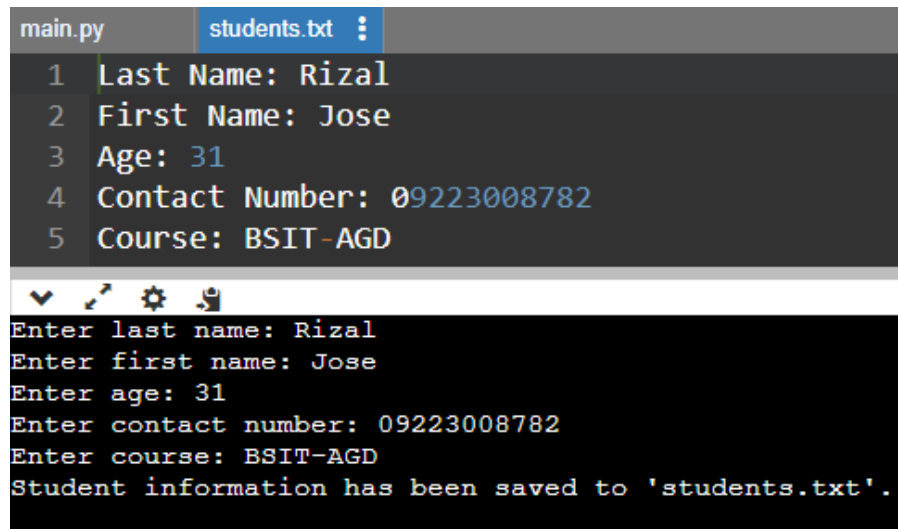
### 3.3. Practical Problem Solving with String Manipulation and File Handling

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 that prompt the user for their last name, first name, age, contact number, and course. Below the code editor, a terminal window shows the execution of the program, with the user's input for each prompt and a final 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:

```
# Accept input for student details

last_name = input("Enter Last Name: ").strip()
first_name = input("Enter First Name: ").strip()
age = input("Enter Age: ").strip()
contact_number = input("Enter Contact Number: ").strip()
course = input("Enter Course: ").strip()
```

```
# 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}
-----
"""
```

**Student info to the file using 'with open()' for better file handling**

```
file_path = "C:\\Users\\Cheetos\\Documents\\Practice Coding\\Activity 03\\Student.txt"
```

```
try:
```

```
    with open(file_path, "a") as file:
```

```
        file.write(student_info)
```

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

```
except Exception as e:
```

```
    print(f"\nError saving student information: {e}")
```



Sample Output:

```
Enter Last Name: Acosta
Enter First Name: Mike Deniel
Enter Age: 21
Enter Contact Number: 09618397465
Enter Course: BSIT-BA

Student information has been successfully saved to 'students.txt'.
PS C:\Users\Cheetos\Documents\Practice Coding\Activity 03> 
```

```
Student.txt
1
2 Last Name: Acosta
3 First Name: Mike Deniel
4 Age: 21
5 Contact Number: 09618397465
6 Course: BSIT-BA
7 -----
8
```

### 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:

```
file_path = "C:\\Users\\Cheetos\\Documents\\Practice Coding\\Activity 03\\Student.txt"
```

try:

```
# Open the file in read mode using 'with' to ensure proper handling
```

```

with open(file_path, "r") as file:

    student_info = file.read().strip() # Read and remove leading/trailing whitespace


if student_info:

    print("\nReading Student Information:")

    print(student_info)

else:

    print("No student information found in 'students.txt'.")


except FileNotFoundError:

    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}")

```

Sample Output:

```

Reading Student Information:
Last Name: Acosta
First Name: Mike Deniel
Age: 21
Contact Number: 09618397465
Course: BSIT-BA
-----
PS C:\Users\Cheetos\Documents\Practice Coding\Activity 03>

```

## QUESTION AND ANSWER:

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

The function format() in Python enables the integration of variables into strings. This is done when the placeholder or an expression within curly braces {} is replaced with the variable passed inside the

function. This greatly adds to the readability of the code and enables easy creation of dynamic strings. For example, a person's name and age are integrated into a sentence describing them.

```
first_name = "Mike"
age = 21

# Using format() to insert variables into a string
message = "Hello, my name is {} and I am {} years old.".format(
    first_name, age)

print(message)
```

Hello, my name is Mike and I am 21 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

While working with file handling in Python, the choice of 'read' ('r') and 'write' ('w') modes dictates how a file has been accessed. 'Read' mode is for reading an existing file. It positions the file pointer at the beginning of the file and raises an error if this file is absent. Whereas the 'write' mode is for writing: it opens a file for writing, discarding existing content, valid only if a file existed or a new file was created if it wasn't; thus, it is where one has to write some data to the file.

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

The program allows the user to cut substrings from a string by specifying ranges of indices within a string. Using the string[start:stop:step], the portion of the string can be isolated with a standardized starting index at position start and an ending index at position stop maintaining a specific step size. Such technique makes retrieval of certain segments infinitely more accurate and easy from a larger string.

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.

In the phase of file operations in Python, 'append' ('a') mode is different from 'write' mode as initially referred by the write method of an object in Python. It appends new data at the end of a pre-existing file, instead of overwriting whatever is currently there. An attempt to open a file in 'append' mode will create a new file if the file does not already exist. The append mode is ideal for logging or accumulating information over time-whatever information was already in the file won't be lost.

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?

To open a file, "data.txt," in Python, it has to be opened in 'read' mode and exceptions must be handled where a FileNotFoundError may occur. The block will help avoid termination of the program by providing

a message when the required file is not found. The with open() statement ensures that files are properly closed after use, subsequently making the code robust.