FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011
Integrative Programming and Technologies

EXERCISE

3

String and File Handling

Student Name:	Ian Rey Angelo Tamayo
Section:	
Professor:	

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

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

Output

```
#3.1 Activity for performing String Manipulations
fname = input("Enter your first name: ")
lname = input("Enter your last name: ")
age = input("Enter your age: ")


full_name = fname + " " + lname


nickname = fname[:3]

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


print("\nFull Name:", full_name)
print("Nickname:", nickname)
print("Greeting Message:", greeting)
```

```
Enter your first name: Joey
Enter your last name: Kangaroo
Enter your age: 45

Full Name: Joey Kangaroo
Nickname: Joe
Hello, Joe! Welcome. You are 45 years old.
```

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

Output

```
#3.2 Activity for Performing String Manipulations
fname = input("Enter your first name: ")
lname = input("Enter your last name: ")

full_name = fname + " " + lname

full_nameUP = full_name.upper()
full_namelow = full_name.lower()

full_nameLen = len(full_name)


print("\nFull Name:", full_name)
print("Full Name (Upper Case):", full_nameUP)
print("Full Name (Lower Case):", full_namelow)
print("Length of Full Name:", full_nameLen)
```

```
Enter your first name: Ian
Enter your last name: Tamayo

Full Name: Ian Tamayo
Full Name (Upper Case): IAN TAMAYO
Full Name (Lower Case): ian tamayo
Length of Full Name: 10
```

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.

Output

```python
#3.3. Practical Problem Solving with String Manipulation and File Handling
lname = input("Enter your last name: ")
fname = input("Enter your first name: ")
age = input("Enter your age: ")
contactno = input("Enter contact number: ")
course = input("Enter course: ")

student_info = f"Last Name: {lname}\nFirst Name: {fname}\nAge: {age}\nContact Number: {contactno}\nCourse: {course}\n"

with open("Activity_03\students.txt", "w") as file:
    file.write(student_info)

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

```
Enter your last name: Tamayo
Enter your first name: Ian
Enter your age: 21
Enter contact number: 1234567890
Enter course: BSITCST

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

Activity_03 > students.txt

```
1  Last Name: Tamayo
2  First Name: Ian
3  Age: 21
4  Contact Number: 1234567890
5  Course: BSITCST
6
```

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

Output

```
#3.4 Activity for Reading File Contents and Display
file_path = "Activity_03/students.txt"

try:
    with open(file_path, "r") as file:
        print("\nReading Student Information:")
        print(file.read())
except FileNotFoundError:
    print("Error: 'students.txt' not found. Please ensure the file exists.")
```

```
Reading Student Information:
Last Name: Tamayo
First Name: Ian
Age: 21
Contact Number: 1234567890
Course: BSITCST
```

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 inserts variables into a string.

```
name = "Alice"
age = 25
print("My name is {} and I am {} years old.".format(name, age))
```

Output: My name is Alice and I am 25 years old.

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

Opening a file in 'read' mode ('r') allows you to access and read its contents without modifying it. If the file doesn't exist, Python raises an error. This mode is used when you only need to retrieve data, such as reading a configuration file or displaying stored information.

While 'write' mode ('w') creates a new file or overwrites an existing one. Any previous data in the file is lost. This mode is used when saving new data, such as logging information or writing user input to a file.

Use 'r' when reading data and 'w' when writing new content.

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

String slicing extracts a portion of a string using index positions. It follows the syntax string[start:end], where start is inclusive, and end is exclusive.

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

Output: 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' (append) mode adds new content to a file without deleting existing data, while 'w' (write) mode overwrites the entire file.

```
with open("students.txt", "a") as file:
   file.write("\nNew student: John Doe")
```

This appends "New student: John Doe" to the file 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?

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