FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011
Integrative Programming and Technologies

EXERCISE

3

String and File Handling

Student Name:	Tara Victoria E. Gerones
Section:	TW23
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

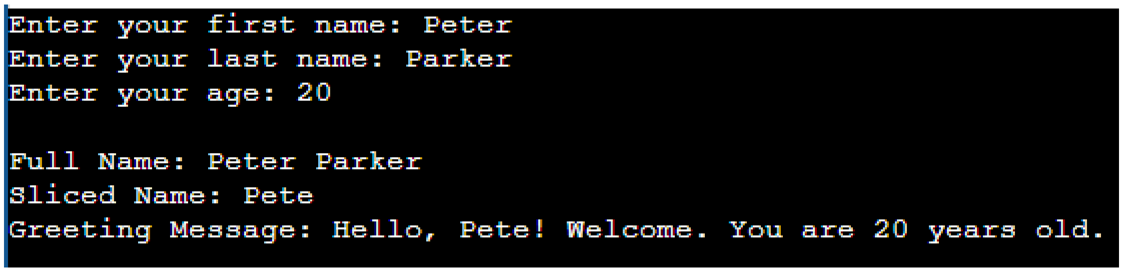
INSTRUCTIONS:
Copy your source codes to be pasted in this document as well as a screen shot of your running output.

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



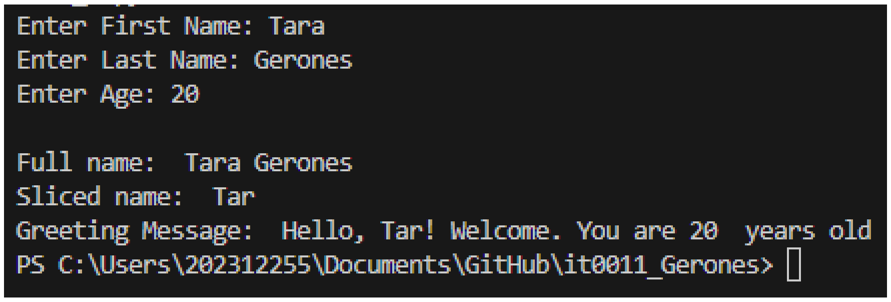
```
Enter your first name: Peter
Enter your last name: Parker
Enter your age: 20

Full Name: Peter Parker
Sliced Name: Pete
Greeting Message: Hello, Pete! Welcome. You are 20 years old.
```

SOURCE CODE:

```
firstName = input("Enter First Name: ")
lastName = input("Enter Last Name: ")
age = input("Enter Age: ")
fullName = firstName + ' ' + lastName
sliceName = firstName[:3]
print()
message = "Hello, {fName}! Welcome. You are {Age}  years
old".format(fName = sliceName, Age=age)
print("Full name: ", fullName)
print("Sliced name: ", sliceName)
print("Greeting Message: ", message)
```

OUTPUT:



```
Enter First Name: Tara
Enter Last Name: Gerones
Enter Age: 20

Full name:  Tara Gerones
Sliced name:  Tar
Greeting Message:  Hello, Tar! Welcome. You are 20  years old
PS C:\Users\202312255\Documents\GitHub\it0011_Gerones> 
```

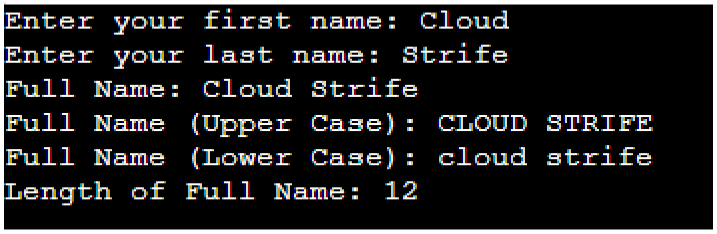
3.2 Activity for Performing String Manipulations

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



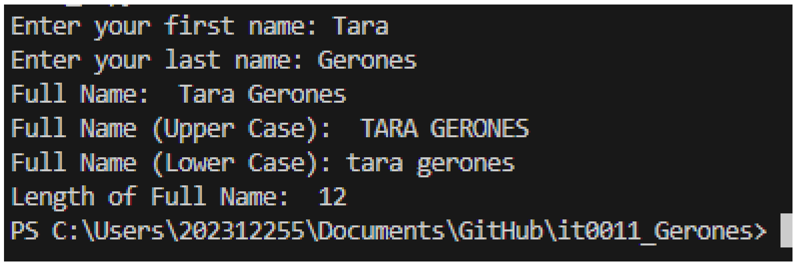
```
Enter your first name: Cloud
Enter your last name: Strife
Full Name: Cloud Strife
Full Name (Upper Case): CLOUD STRIFE
Full Name (Lower Case): cloud strife
Length of Full Name: 12
```

SOURCE CODE:

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

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

OUTPUT:



```
Enter your first name: Tara
Enter your last name: Gerones
Full Name:  Tara Gerones
Full Name (Upper Case):  TARA GERONES
Full Name (Lower Case): tara gerones
Length of Full Name:  12
PS C:\Users\202312255\Documents\GitHub\it0011_Gerones>
```

3.3. Practical Problem Solving with String Manipulation and File Handling

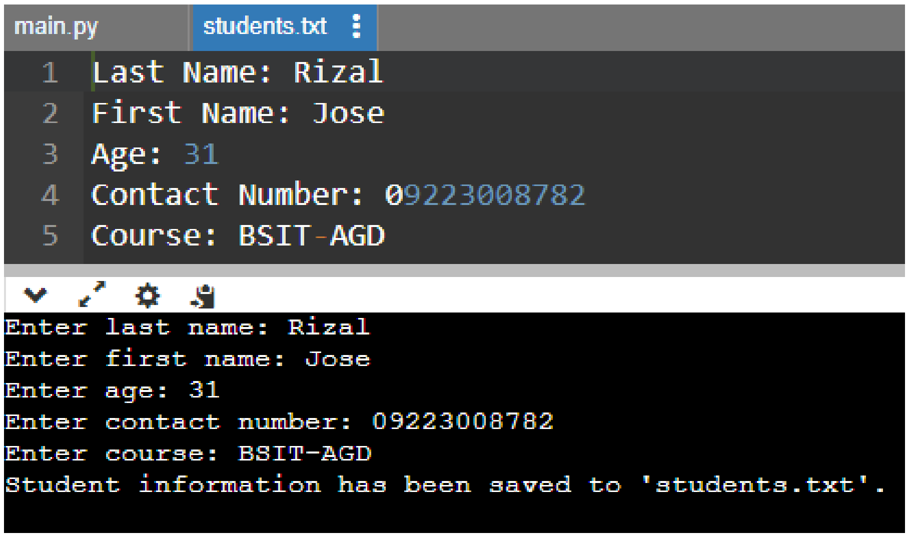
Objective: Apply string manipulation and file handling techniques to store student information in a file.

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



```
Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
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'.
```

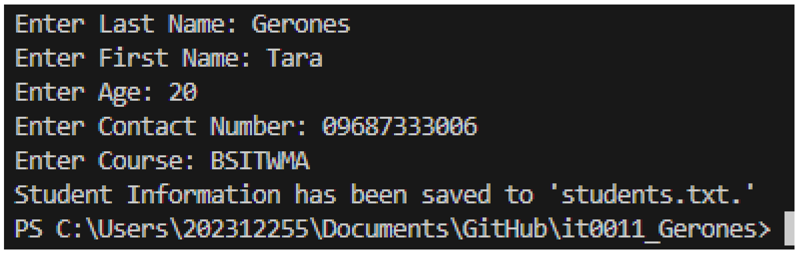
SOURCE CODE:

```
lname = input('Enter Last Name: ')
fname = input('Enter First Name: ')
age = input('Enter Age: ')
contact = input("Enter Contact Number: ")
course = input("Enter Course: ")

stud_info = f'Last Name: {lname} \nFirst Name: {fname} \nAge: {age}
\nContact Number: {contact} \nCourse: {course}'
file = open('students.txt', 'a')
file.write(stud_info)
print("Student Information has been saved to 'students.txt.'")
```

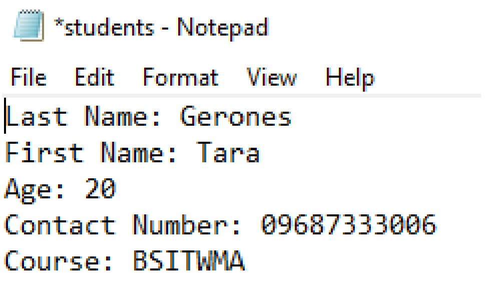
OUTPUT:

Console:



```
Enter Last Name: Gerones
Enter First Name: Tara
Enter Age: 20
Enter Contact Number: 09687333006
Enter Course: BSITWMA
Student Information has been saved to 'students.txt.'
PS C:\Users\202312255\Documents\GitHub\it0011_Gerones>
```

TXT File:



*students - Notepad

File Edit Format View Help

```
Last Name: Gerones
First Name: Tara
Age: 20
Contact Number: 09687333006
Course: BSITWMA
```

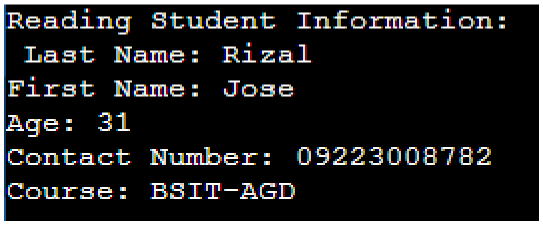
3.4 Activity for Reading File Contents and Display

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

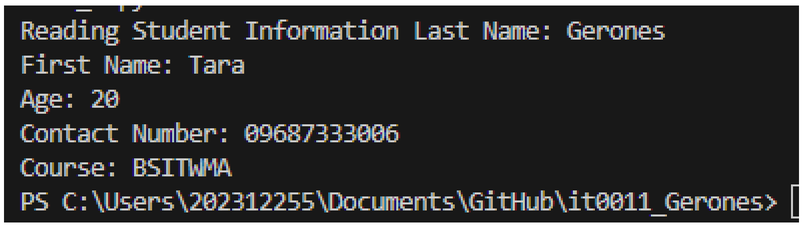


```
Reading Student Information:
 Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
Course: BSIT-AGD
```

SOURCE CODE:

```
file = open('students.txt', 'r')
print('Reading Student Information', file.read())
```

OUTPUT:



```
Reading Student Information Last Name: Gerones
First Name: Tara
Age: 20
Contact Number: 09687333006
Course: BSITWMA
PS C:\Users\202312255\Documents\GitHub\it0011_Gerones>
```

QUESTION AND ANSWER:

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

Another way to use `format()` is by specifying positional arguments inside the placeholders. For instance, `print("The temperature in {0} is {1}°C.".format("Manila", 30))` will produce: "The temperature in Manila is 30°C.

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

The `'r'` mode is useful when you need to retrieve data from a file without modifying it, whereas `'w'` mode is used when you want to write new content. A safer approach to reading a file is using `try-except` to handle missing files, such as `try: with open("example.txt", "r") as f: print(f.read()) except FileNotFoundError: print("File not found.")`.

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

A more detailed slicing example involves extracting part of a longer string. Given `sentence = "Python Programming"`, using `sentence[7:18]` retrieves `"Programming"`. String slicing is useful for extracting substrings, manipulating text, and processing data.

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.

Using `'a'` mode is beneficial when maintaining a history of updates, such as appending notes to a file. For example, `with open("notes.txt", "a") as f: f.write("Remember to review Python files handling.\n")` ensures that previous notes remain while adding new ones.

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?

Another method to handle missing files is checking their existence before attempting to read them. Using `import os`, the condition `if os.path.exists("data.txt"):` ensures the file exists before opening it. If the file is missing, a message such as `"File not found."` is displayed instead of causing an error.