FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011 Integrative Programming and Technologies

EXERCISE

3

String and File Handling

Student Name:	Neil Vincent Sioson
Section:	TB22
Professor:	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)
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

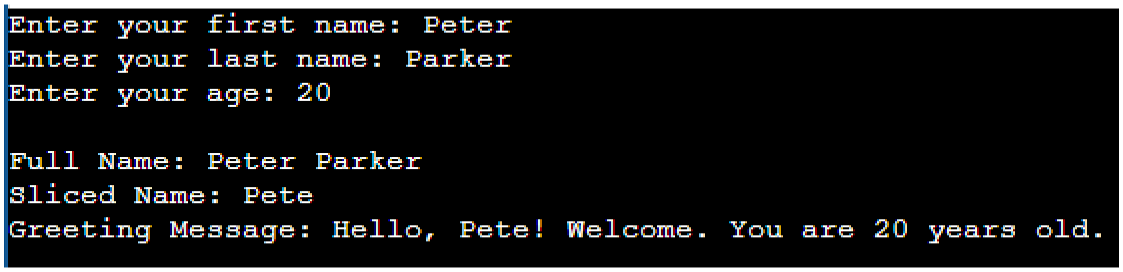
INSTRUCTIONS:
Copy your source codes to be pasted in this document as well as a screen shot of your running output.

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

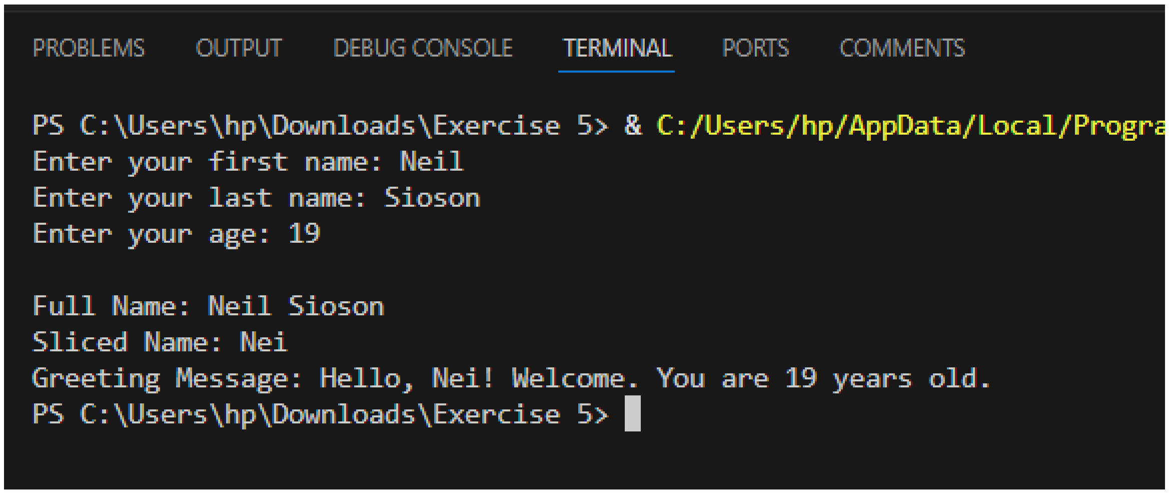


```
Enter your first name: Peter
Enter your last name: Parker
Enter your age: 20

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

Source Code:

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

# link first name and last name
full_name = first_name + " " + last_name

# Slice the first three characters of the first name
sliced_name = first_name[:3]

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

# Display output
print("\nFull Name:", full_name)
print("Sliced Name:", sliced_name)
print("Greeting Message:", greeting_message)
```

Output:



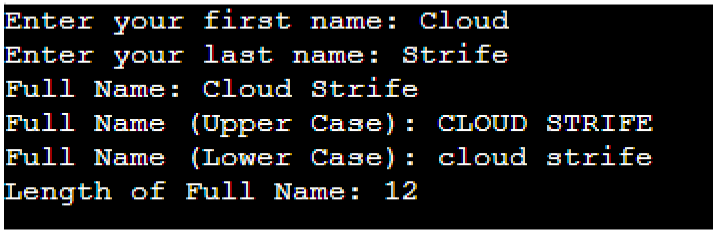
```
PS C:\Users\hp\Downloads\Exercise 5> & C:/Users/hp/AppData/Local/Progra
Enter your first name: Neil
Enter your last name: Sioson
Enter your age: 19

Full Name: Neil Sioson
Sliced Name: Nei
Greeting Message: Hello, Nei! Welcome. You are 19 years old.
PS C:\Users\hp\Downloads\Exercise 5>
```

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



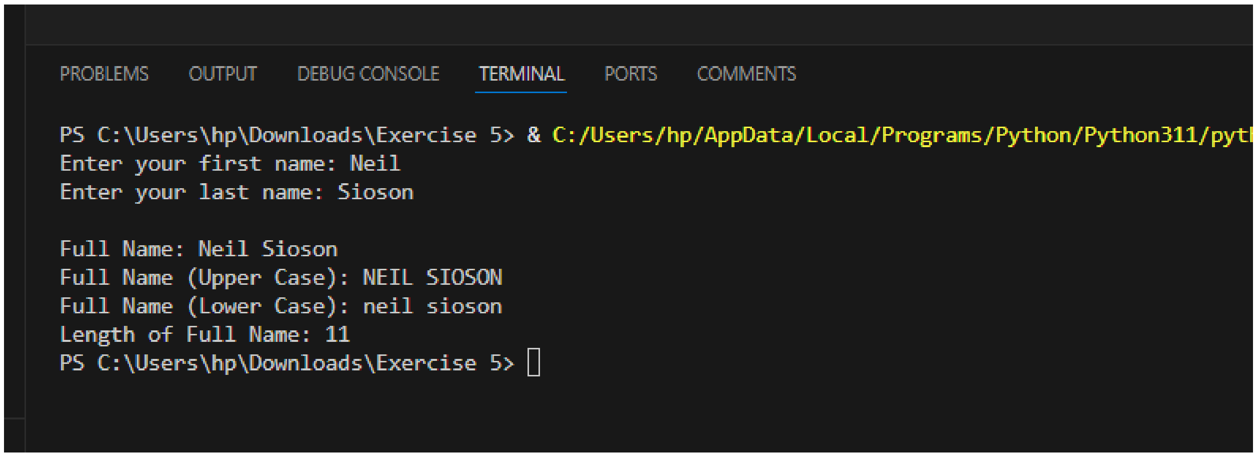
```
Enter your first name: Cloud
Enter your last name: Strife
Full Name: Cloud Strife
Full Name (Upper Case): CLOUD STRIFE
Full Name (Lower Case): cloud strife
Length of Full Name: 12
```

Source code:

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

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

# Display the full name in upper and lower case
full_name_upper = full_name.upper()
```

```
full_name_lower = full_name.lower()

# Count and display the length of the full name
full_name_length = len(full_name)

# Display output
print("\nFull Name:", full_name)
print("Full Name (Upper Case):", full_name_upper)
print("Full Name (Lower Case):", full_name_lower)
print("Length of Full Name:", full_name_length)
```

Output:



```
PS C:\Users\hp\Downloads\Exercise 5> & C:/Users/hp/AppData/Local/Programs/Python/Python311/pyt
Enter your first name: Neil
Enter your last name: Sioson

Full Name: Neil Sioson
Full Name (Upper Case): NEIL SIOSON
Full Name (Lower Case): neil sioson
Length of Full Name: 11
PS C:\Users\hp\Downloads\Exercise 5> 
```

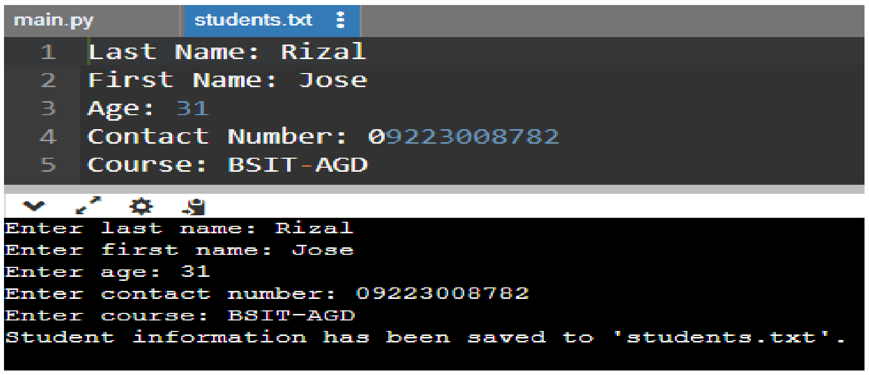
3.3. Practical Problem Solving with String Manipulation and File Handling

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


main.py students.txt

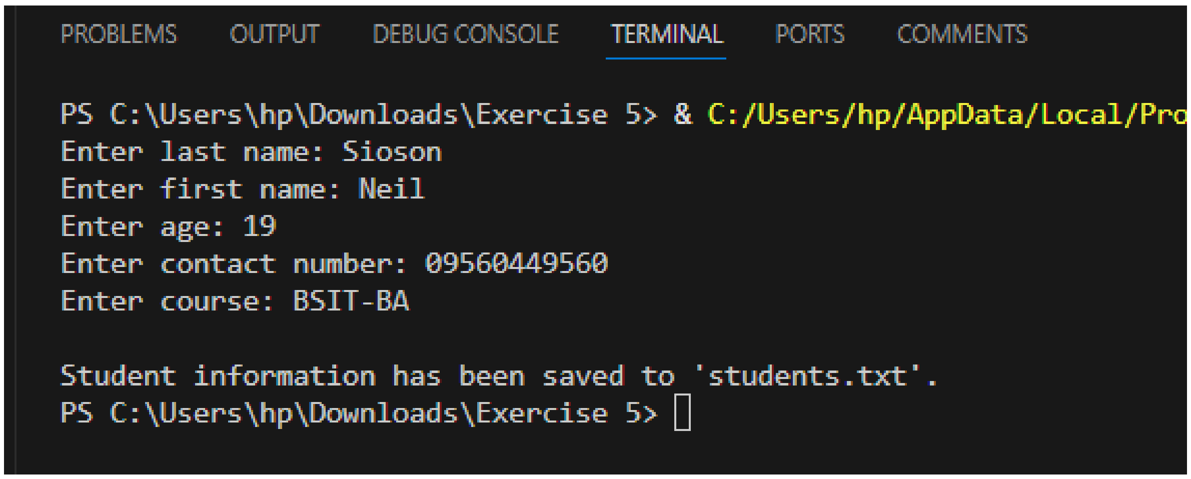
```
Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
Course: BSIT-AGD
```

```
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:

```
# Get user input
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: ")

# Format the collected information
student_info = f"Last Name: {last_name}\nFirst Name: {first_name}\nAge: {age}\nContact
Number: {contact_number}\nCourse: {course}\n\n"

# Open the file in append mode and write the information
with open("students.txt", "a") as file:
  file.write(student_info)

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

Output:



```
PS C:\Users\hp\Downloads\Exercise 5> & C:/Users/hp/AppData/Local/Pro
Enter last name: Sioson
Enter first name: Neil
Enter age: 19
Enter contact number: 09560449560
Enter course: BSIT-BA

Student information has been saved to 'students.txt'.
PS C:\Users\hp\Downloads\Exercise 5>
```

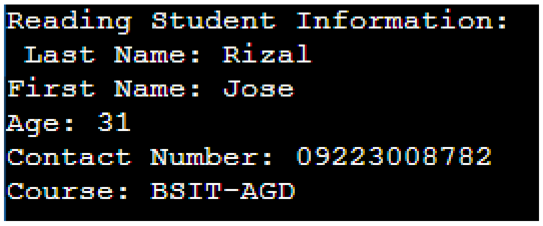
3.4 Activity for Reading File Contents and Display

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

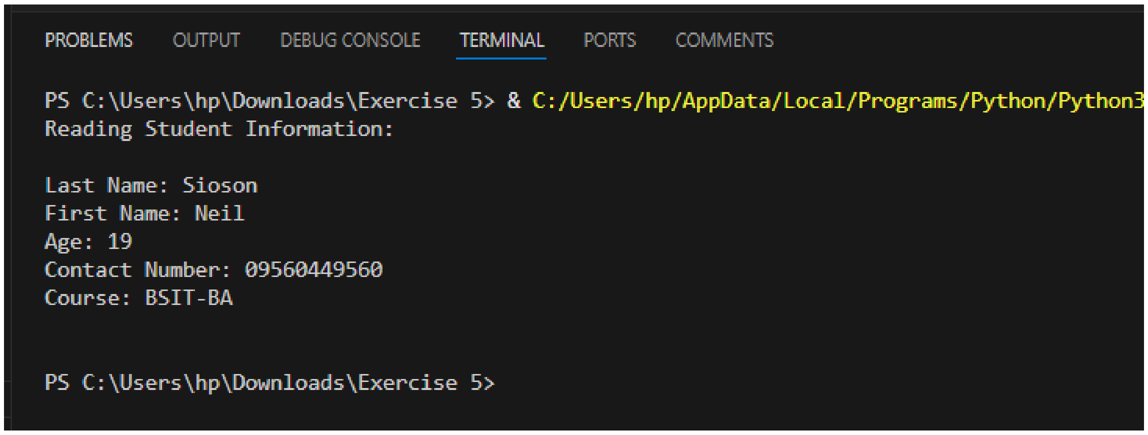


```
Reading Student Information:
 Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
Course: BSIT-AGD
```

Source code:

```
# Open the file in read mode
try:
   with open("students.txt", "r") as file:
      # Read the contents of the file
      student_data = file.read()

      # Display the student information
      print("Reading Student Information:\n")
      print(student_data)

except FileNotFoundError:
   print("Error: 'students.txt' not found. Please ensure the file exists before reading.")
```

Output:



```
PS C:\Users\hp\Downloads\Exercise 5> & C:/Users/hp/AppData/Local/Programs/Python/Python3
Reading Student Information:

Last Name: Sioson
First Name: Neil
Age: 19
Contact Number: 09560449560
Course: BSIT-BA


PS C:\Users\hp\Downloads\Exercise 5>
```

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 makes it easy to insert variables into a string without using messy concatenation. It replaces {} placeholders with the values you provide. For example, "My name is {}".format("Alice") outputs "My name is Alice". You can also use multiple placeholders like "I am {} years old and live in {}".format(25, "Paris"). This keeps your code cleaner and easier to read.

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) lets you open a file to read its content but won't let you change anything. 'w' (write mode) creates a new file or erases everything in an existing file before writing new content. You'd use 'r' when you just need to view or process a file's data, like reading a config file. 'w' is best when you need to completely rewrite a file, like saving new user data. Be careful with 'w' because it erases everything in the file before writing.

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

String slicing is a way to grab part of a string using index positions. You do this with [start:end], where start is where you begin, and end is where you stop (but it won't include the end index). For example, "Hello World"[0:5] gives "Hello", pulling characters from index 0 to 4. If you leave out start, it defaults to the beginning, and leaving out end means it goes to the end. This is super useful for things like extracting file extensions or short parts of text.

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) lets you add new content to a file without deleting what's already there, while 'w' clears everything before writing. You use 'a' when you want to keep old data, like logging messages or saving user history. For example, with open("log.txt", "a") as file: file.write("New entry\n") adds a new line without removing previous content. If you used 'w', everything in "log.txt" would be erased first. So, 'a' is great for keeping records without losing past data.

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?

if a file might not exist, you can handle it using a try-except block to catch the FileNotFoundError. This prevents the program from crashing and allows you to show a helpful message instead.

Simple Code:

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