



FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011
Integrative Programming and
Technologies

EXERCISE

3

String and File Handling

Student Name:	Ferrence Alagad
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

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.
```

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
```

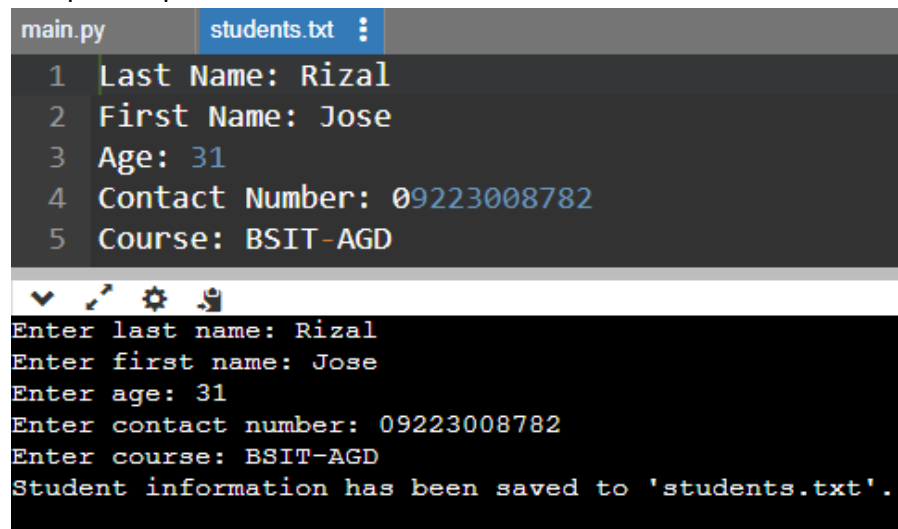
3.3. Practical Problem Solving with String Manipulation and File Handling

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 is a terminal window showing the program's execution. The user has entered 'Rizal' for the last name, 'Jose' for the first name, '31' for age, '09223008782' for the contact number, and 'BSIT-AGD' for the course. The program has successfully written this information to 'students.txt' and displayed a confirmation message.

```
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'.
```

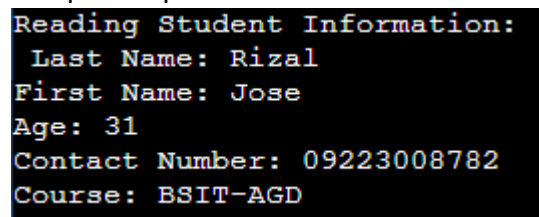
3.4 Activity for Reading File Contents and Display

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



The screenshot shows a terminal window with the output of a Python program. The program has read the contents of 'students.txt' and displayed the student information in a formatted way.

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

3.1. Activity for Performing String Manipulations

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

full_name = first_name + " " + last_name

sliced_first_name = first_name[:3]

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

print("Full Name:", full_name)
print("Sliced First Name:", sliced_first_name)
print("Greeting Message:", greeting_message)

Enter your first name: Ferrence
Enter your last name: Alagad
Enter your age: 20
Full Name: Ferrence Alagad
Sliced First Name: Fer
Greeting Message: Hello, Fer! You are 20 years old.
```

3.2 Activity for Performing String Manipulations

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

full_name = first_name + " " + last_name

full_name_upper = full_name.upper()
full_name_lower = full_name.lower()
full_name_length = len(full_name)

print("Full Name:", full_name)
print("Full Name in Upper Case:", full_name_upper)
print("Full Name in Lower Case:", full_name_lower)
print("Length of Full Name:", full_name_length)
```

```
Enter your first name: Ferrence
Enter your last name: Alagad
Full Name: Ferrence Alagad
Full Name in Upper Case: FERRENCE ALAGAD
Full Name in Lower Case: ferrence alagad
Length of Full Name: 15
```

3.3. Practical Problem Solving with String Manipulation and File Handling

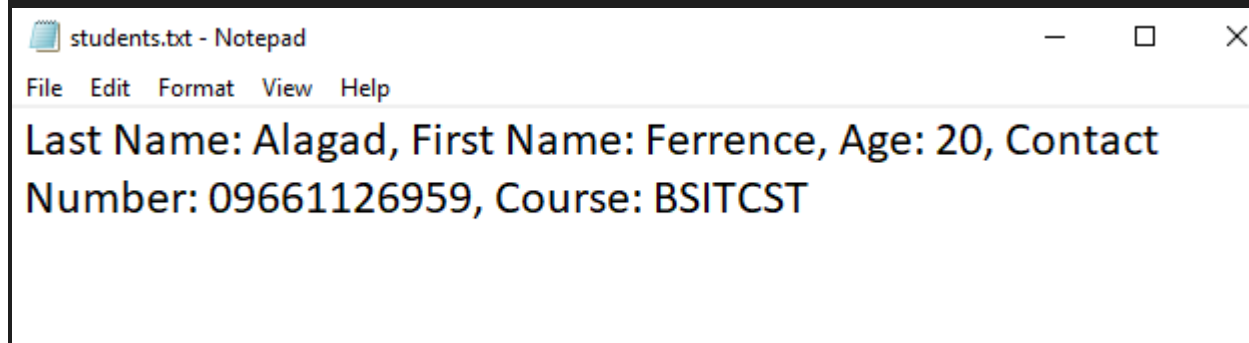
```
last_name = input("Enter your last name: ")
first_name = input("Enter your first name: ")
age = input("Enter your age: ")
contact_number = input("Enter your contact number: ")
course = input("Enter your course: ")

student_info = f"Last Name: {last_name}, First Name: {first_name}, Age: {age},
Contact Number: {contact_number}, Course: {course}\n"

with open("students.txt", "a") as file:
    file.write(student_info)

print("Information has been saved successfully.")
```

```
Enter your last name: Alagad
Enter your first name: Ferrence
Enter your age: 20
Enter your contact number: 09661126959
Enter your course: BSITCST
Information has been saved successfully.
```



3.4 Activity for Reading File Contents and Display

```

try:
    with open("students.txt", "r") as file:
        contents = file.readlines()

    if contents:
        print("Student Information:")
        for line in contents:
            print(line.strip())
    else:
        print("The file is empty.")

except FileNotFoundError:
    print("The file 'students.txt' does not exist.")

```

```

Student Information:
Last Name: Alagad, First Name: Ferrence, Age: 20, Contact Number: 09661126959, 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?

`format()` function in Python helps combine variables with text by inserting values into placeholders `{}` within a string. It makes it easier to create dynamic sentences.

Example:

```

python
CopyEdit
name = "fer"
age = 20
message = "My name is {} and I am {} years old.".format(name, age)
print(message)

```

Output:

```

pgsql
CopyEdit
My name is fer and I am 20 years old.

```

Here, `{}` acts as a placeholder, and the `format()` function replaces them with the values of `name` and `age`.

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):

- Used to read the contents of an existing file.
- The file must already exist; otherwise, an error occurs.
- Example:

```
python
CopyEdit
file = open("example.txt", "r")
content = file.read()
print(content)
file.close()
```

- Use this mode when you only need to access and read a file without modifying it.

❓ 'w' (Write Mode):

- Used to write to a file.
- If the file already exists, it **erases** all previous content.
- If the file does not exist, a new file is created.
- Example:

```
python
CopyEdit
file = open("example.txt", "w")
file.write("Hello, World!")
file.close()
```

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 extract a portion (substring) of a string using **indexing**

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**) in Python is used to **add new content** to the end of a file without deleting its existing data.

```
file = open("example.txt", "a") # Open file in append mode
file.write("\nThis is new data.") # Adds text at the end
file.close()
```

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: # Open file in read mode
    content = file.read() # Read the file contents
    print(content) # Print the content except FileNotFoundError:
    print("Error: The file 'data.txt' does not exist.")
```

- If the file does **not exist**, the `FileNotFoundException` is caught, and an error message is displayed instead of crashing the program.