COLLEGE OF COMPUTER STUDIES

IT0011
Integrative Programming and Technologies

EXERCISE

3

String and File Handling

Student Name:	Franco Jr. Eduardo E.
Section:	TW24

Professor:	Sir. 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	Shows effort and creativity in addressing most requirements.	Adequate effort but lacks creativity or exploration	Minimal effort and creativity evident.	Little to no effort or creativity apparent.

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (2)	Unsatisfactory (1)
	beyond basic requirements.		beyond the basics.		

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.
```

```python
fname = input("Enter your first name: ")
lname = input("Enter your last name: ")
age = input("Enter your age: ")

print()
fullname = fname + " " + lname
print("Full name: ", fullname)
slicname = fullname[0:3]
print("Sliced Name: ", slicname)

greet = "Hello, {}! Welcome. You are {} years old."
print(greet.format(slicname, age))
```

```
Enter your first name: Eduardo
Enter your last name: Franco Jr.
Enter your age: 20

Full name:  Eduardo Franco Jr.
Sliced Name:  Edu
Hello, Edu! 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
```

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

print()
fullname = fname + " " + lname
print("Full name: ", fullname)

print("Full name (Upper Case): ", fullname.upper())
print("Full name (Lower Case): ", fullname.lower())
print("Length of Full Name: ", len(fullname))
```

```
Full name:  Eduardo Franco Jr.
Full name (Upper Case):  EDUARDO FRANCO JR.
Full name (Lower Case):  eduardo franco jr.
Length of Full Name:  18
```
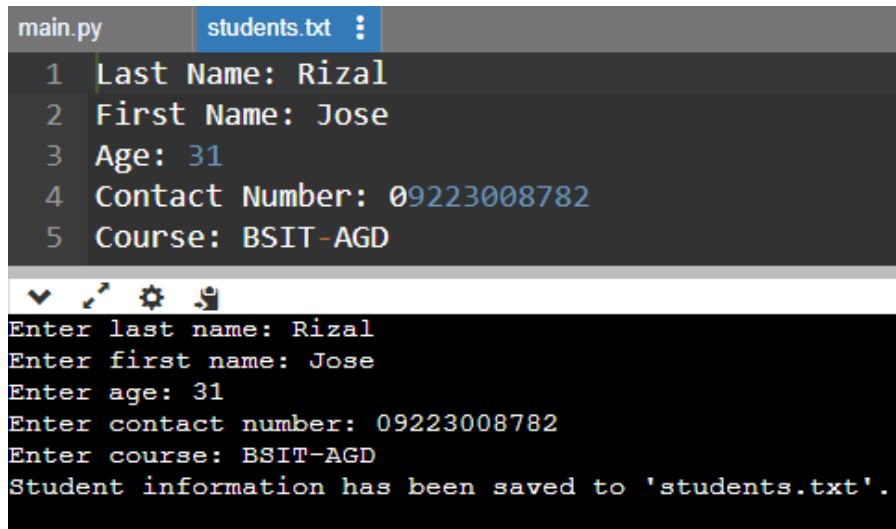
3.3. Practical Problem Solving with String Manipulation and File Handling

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.
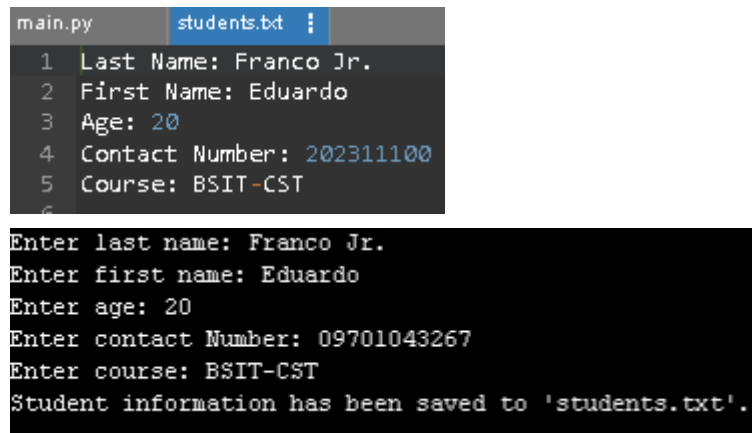
Sample Output


```
lname = input("Enter last name: ")
fname = input("Enter first name: ")
age = input("Enter age: ")
cont = input("Enter contact Number: ")
cour = input("Enter course: ")

data = "Last Name: {}\n" + "First Name: {}\n" + "Age: {}\n" + "Contact Number: {}\n" +
"Course: {}\n"
filedata = data.format(lname, fname, age, cont, cour)

filename = "students.txt"

file = open(filename, 'a')
file.write(filedata)
file.close()
```

print("Student information has been saved to 'students.txt'. ")

```
main.py        students.txt  ⋮
 1   Last Name: Franco Jr.
 2   First Name: Eduardo
 3   Age: 20
 4   Contact Number: 202311100
 5   Course: BSIT-CST
```

```
Enter last name: Franco Jr.
Enter first name: Eduardo
Enter age: 20
Enter contact Number: 09701043267
Enter course: BSIT-CST
Student information has been saved to 'students.txt'.
```
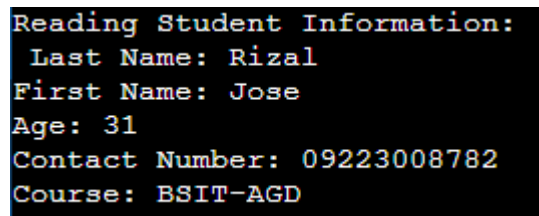
3.4 Activity for Reading File Contents and Display
Objective: Apply file handling techniques to read and display student information from a file.

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

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

lname = input("Enter last name: ")
fname = input("Enter first name: ")
age = input("Enter age: ")
cont = input("Enter contact Number: ")
cour = input("Enter course: ")

data = "Last Name: {}\n" + "First Name: {}\n" + "Age: {}\n" + "Contact Number: {}\n" +
"Course: {}\n"
filedata = data.format(lname, fname, age, cont, cour)

filename = "students.txt"

```
file = open(filename, 'a')
file.write(filedata)
file.close()

print("Student information has been saved to 'students.txt'. ")
print()
print("Reading Student Information:\n")
file = open(filename, 'r')
for line in file:
    print(line, end ="")
file.close()
```

```
Enter last name: Franco Jr.
Enter first name: Eduardo
Enter age: 20
Enter contact Number: 09701043267
Enter course: BSIT-CST
Student information has been saved to 'students.txt'.

Reading Student Information:

Last Name: Franco Jr.
First Name: Eduardo
Age: 20
Contact Number: 09701043267
Course: BSIT-CST
```

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 helps a lot in combining variables as if you have variables that are changing values, and it is easier to rearrange and gives you the option for a fill-in-the-blank template where format comes in handy for a templated message and just have to switch around the variables

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 difference of read mode and write mode is that read mode lets you see the contents of an existing file (it won't work if the file doesn't exist) and you cannot modify it at this mode, whilst write mode creates or replaces the file with the inputs and can be modified

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

Slicing is where you get a portion of a string, where if you have a variable that holds the string "Hello, World!" and you do print(str[0:5]) it will 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 purpose of 'a' or append mode is that instead of erasing the previous file and replacing it with the new one like write mode, append mode just connects the new file to the existing file. It is good for if you have a list and each addition of new information gets added/appended to the file.

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?

filename = "data.txt"

try:

   file = open(filename, "w")
   file.write(filedata)
   file.close()

except FileNotFoundError:

   print("Error file does not exist")