COLLEGE OF COMPUTER STUDIES

IT0011
Integrative Programming and Technologies

EXERCISE

3

String and File Handling

Student Name:	Rhence Lordien P. Abellar
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.
```

TASK 1:

```
TFA2 > 3.1.py > ...
  1   firstname = input ('Enter your first name: ')
  2   lastname = input ('Enter your last name: ')
  3   age = input ('Enter your age: ')
  4
  5   print ('===============================================================')
  6   print ('Full Name: ', firstname, lastname)
  7
  8   sliced_name = firstname [0:4]
  9   print ('Sliced Name: ', sliced_name)
 10
 11   greeting_message = 'Greeting Message: Hello, {}! Welcome. You are {} years old.'
 12   print (greeting_message.format(sliced_name, age))
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

PS C:\Users\202311305\Documents\GitHub\it0011_abellar> & "C:/Program Files/Python312/python.exe"
ellar/TFA2/3.1.py
Enter your first name: Rhence
Enter your last name: Abellar
Enter your age: 20
===============================================================
Full Name:  Rhence Abellar
Sliced Name:  Rhen
Greeting Message: Hello, Rhen! Welcome. You are 20 years old.
PS C:\Users\202311305\Documents\GitHub\it0011_abellar> █
```

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
```

TASK 2:

3.1.py U   3.2.py U ✕   3.3.py U   3.4.py U

TFA2 > 3.2.py > ...

```python
1   firstname = input ('Enter your first name: ')
2   lastname = input ('Enter your last name: ')
3
4   fullname = firstname + ' ' + lastname
5
6   print ('===============================================================')
7   print ('Full Name: ', fullname)
8
9   print ('Full Name (UPPER CASE): ', (fullname.upper()))
10  print ('Full Name (lower case): ', (fullname.lower()))
11
12  print ('Length of Full Name: ', (len(fullname)) )
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS

```
PS C:\Users\202311305\Documents\GitHub\it0011_abellar> & "C:/Program Files/Python312/pyt
ellar/TFA2/3.2.py
Enter your first name: Rhence
Enter your last name: Abellar
===============================================================
Full Name:  Rhence Abellar
Full Name (UPPER CASE):  RHENCE ABELLAR
Full Name (lower case):  rhence abellar
Length of Full Name:  14
PS C:\Users\202311305\Documents\GitHub\it0011_abellar>
```
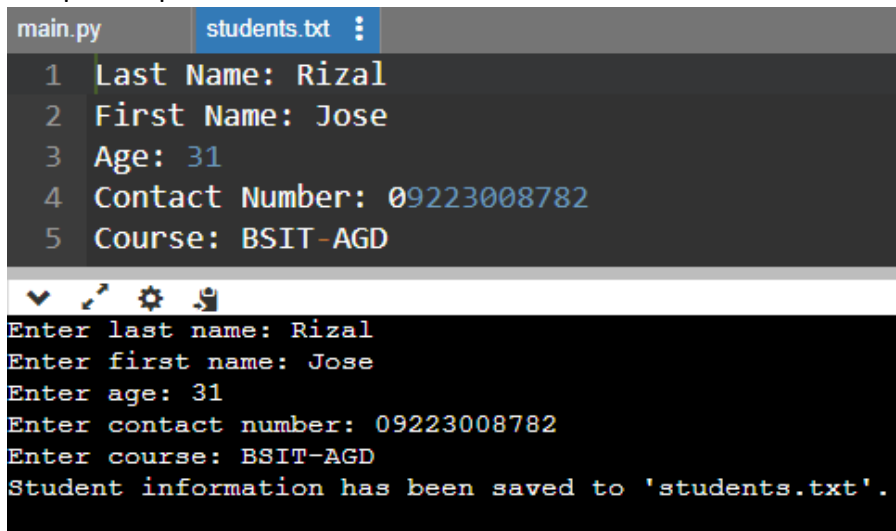
3.3. Practical Problem Solving with String Manipulation and File Handling

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  ⋮
  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'.
```

TASK 3:

```
3.1.py  U      3.2.py  U      3.3.py  U ×     3.4.py  U

TFA2 >  3.3.py > ...
    1   firstname = input ('Enter your first name: ')
    2   lastname = input ('Enter your last name: ')
    3   age = input ('Enter your age: ')
    4   contact_num = input ('Enter contact number: ')
    5   course = input ('Enter course: ')
    6
    7   output_text = "Last Name: {} \nFirst Name: {} \nAge: {} \nContact Number: {} \nCourse: {}"
    8   f=open('TFA2/students.txt', 'a+')
    9   f.write (output_text.format(lastname, firstname, age, contact_num, course))
   10
   11   line = f.readline ()
   12   f.close ()
   13
   14   print ("\nStudent information has been saved to 'students.txt' file.")
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

PS C:\Users\202311305\Documents\GitHub\it0011_abellar> & "C:/Program Files/Python312/python.exe" c:/Users/2(
ellar/TFA2/3.3.py
Enter your first name: Rhence
Enter your last name: Abellar
Enter your age: 20
Enter contact number: 09292144809
Enter course: BSITWMA

Student information has been saved to 'students.txt' file.
PS C:\Users\202311305\Documents\GitHub\it0011_abellar> ▌
```

```
3.1.py  U      3.2.py  U      3.3.py  U      ≡ students.txt U ×

TFA2 >  ≡ students.txt
    1   Last Name: Abellar
    2   First Name: Rhence
    3   Age: 20
    4   Contact Number: 09292144809
    5   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
```

TASK 4:

3.1.py U   3.2.py U   3.3.py U   ≡ students.txt U

TFA2 > 3.4.py > ...

```python
1   f=open('TFA2/students.txt', 'r')
2   print ('Reading Student Information:\n')
3   while True:
4       try:
5           line=next(f)
6           print(line)
7       except StopIteration:
8           break
9   f.close ()
10
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS

```
PS C:\Users\202311305\Documents\GitHub\it0011_abellar> & "C:/Progr
ellar/TFA2/3.4.py
Reading Student Information:

Last Name: Abellar

First Name: Rhence

Age: 20

Contact Number: 09292144809

Course: BSITWMA
PS C:\Users\202311305\Documents\GitHub\it0011_abellar>
```

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 in Python helps combine variables with text by inserting values into placeholders {} within a string.
   ```
   name = "Rhence"
   age = 20
   print("My name is {} and I am {} years old.".format(name, 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) is used to read an existing file. It will raise an error if the file does not exist. 'w' (write mode) creates a new file or overwrites an existing one. Use 'r' when reading data and 'w' when writing new data, erasing previous content.

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

   String slicing extracts parts of a string using the format string[start:end].
   ```
   text = "Hello, world!"
   print(text[0:5])  # 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.

   'a' (append mode) adds new content to a file without deleting existing data, unlike 'w', which overwrites.
   ```
   with open("example.txt", "a") as file:
       file.write("New data added!\n")
   ```

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?
   Use a try-except block to handle missing files:
   ```
   try:
       with open("data.txt", "r") as file:
           content = file.read()
           print(content)
   ```

```python
except FileNotFoundError:
    print("File not found!")
```