FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011
Integrative Programming and Technologies

EXERCISE

3

String and File Handling

Student Name:	Andrade, Kyla Mae A.
Section:	TC21
Professor:	Sir 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.

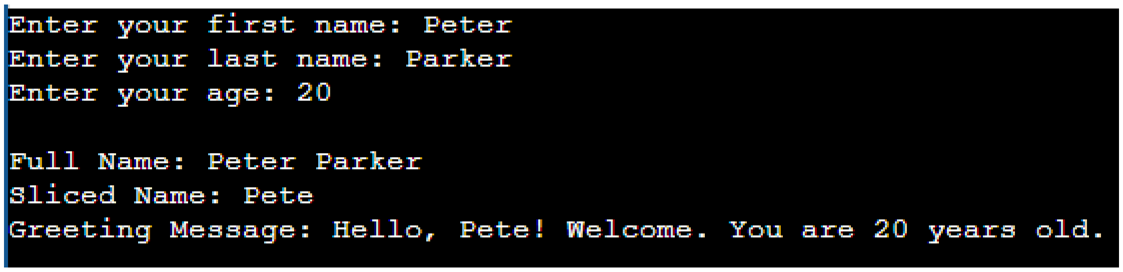
3.1. Activity for Performing String Manipulations

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

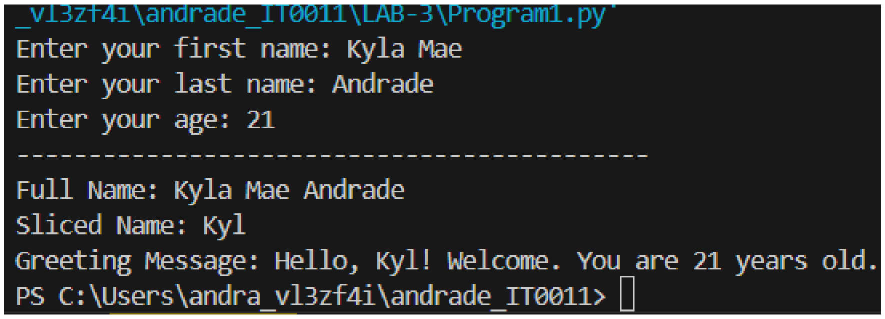


```
Enter your first name: Peter
Enter your last name: Parker
Enter your age: 20

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

Source Code:

```
def string_manip():
   firstn = input("Enter your first name: ")
   lastn = input("Enter your last name: ")
   age = input("Enter your age: ")

   fulln = firstn + " " + lastn
   slicedn = firstn[:3]
   greetingmsg = "Hello, " + slicedn + "! Welcome. You are " + age + " years old."

   print("--------------------------------------------")
   print("Full Name:", fulln)
   print("Sliced Name:", slicedn)
   print("Greeting Message:", greetingmsg)

string_manip()
```

Output:



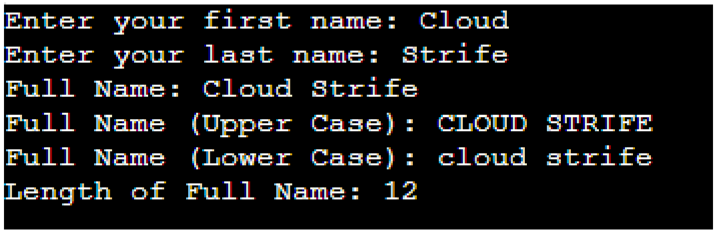
```
_vl3zf4i\andrade_IT0011\LAB-3\Program1.py'
Enter your first name: Kyla Mae
Enter your last name: Andrade
Enter your age: 21
--------------------------------------------
Full Name: Kyla Mae Andrade
Sliced Name: Kyl
Greeting Message: Hello, Kyl! Welcome. You are 21 years old.
PS C:\Users\andra_vl3zf4i\andrade_IT0011>
```

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

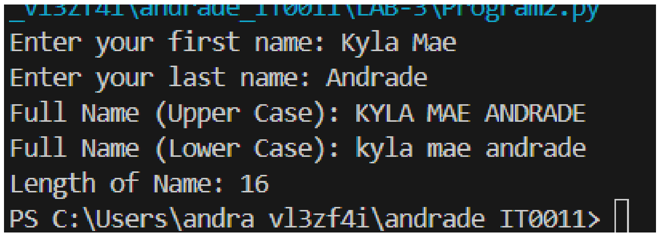


```
Enter your first name: Cloud
Enter your last name: Strife
Full Name: Cloud Strife
Full Name (Upper Case): CLOUD STRIFE
Full Name (Lower Case): cloud strife
Length of Full Name: 12
```

Source Code:

```
def string_manip():
   firstn = input("Enter your first name: ")
   lastn = input("Enter your last name: ")

   fulln = firstn + " " + lastn
   uppern = fulln.upper()
   lowern = fulln.lower()
   namelength = len(fulln)

   print("Full Name (Upper Case):", uppern)
   print("Full Name (Lower Case):", lowern)
   print("Length of Name:", namelength)

string_manip()
```

Output:



```
_vl3zf4i\andrade_IT0011\LAB-3\Program2.py
Enter your first name: Kyla Mae
Enter your last name: Andrade
Full Name (Upper Case): KYLA MAE ANDRADE
Full Name (Lower Case): kyla mae andrade
Length of Name: 16
PS C:\Users\andra_vl3zf4i\andrade_IT0011>
```

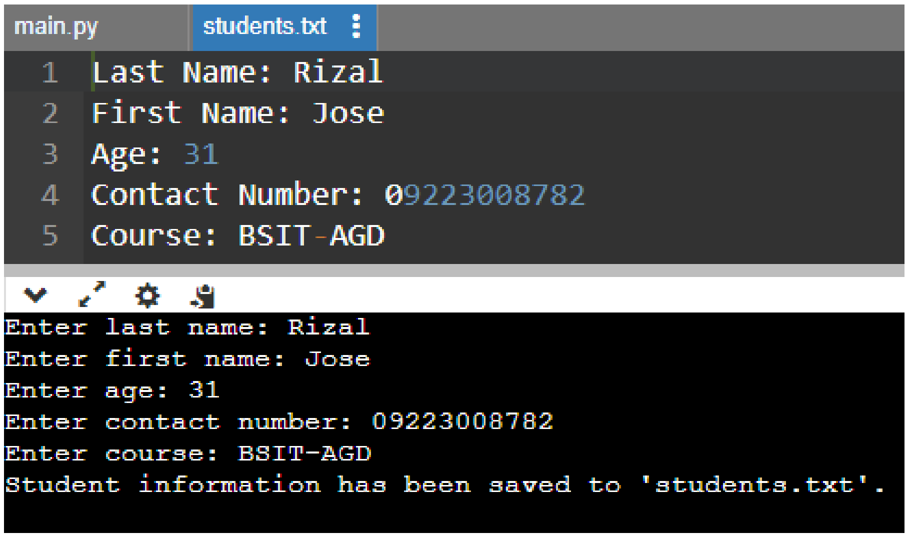
3.3. Practical Problem Solving with String Manipulation and File Handling

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



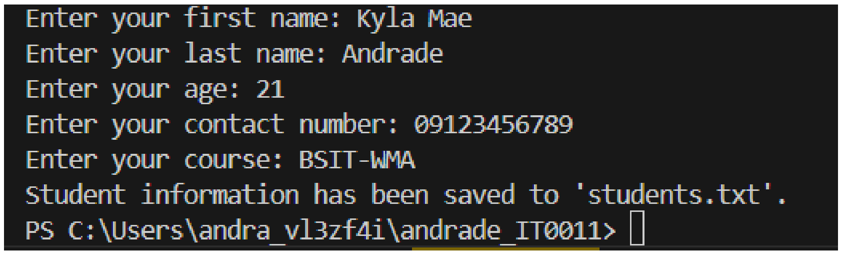
```
Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
Course: BSIT-AGD
```

```
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'.
```

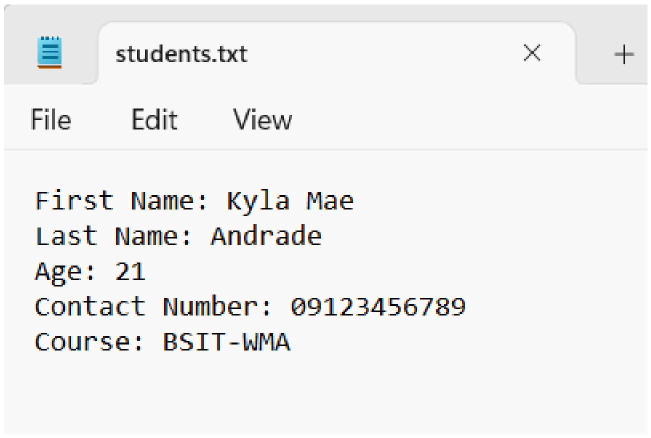
Source Code:

```
def studentinfo():
    firstn = input("Enter your first name: ")
    lastn = input("Enter your last name: ")
    age = input("Enter your age: ")
    contactnmbr = input("Enter your contact number: ")
    course = input("Enter your course: ")
    student = f"First Name: {firstn}\nLast Name: {lastn}\nAge: {age}\nContact Number:
{contactnmbr}\nCourse: {course}\n"
    with open("students.txt", "a") as file:
        file.write(student)

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

studentinfo()
```

Output:



```
Enter your first name: Kyla Mae
Enter your last name: Andrade
Enter your age: 21
Enter your contact number: 09123456789
Enter your course: BSIT-WMA
Student information has been saved to 'students.txt'.
PS C:\Users\andra_vl3zf4i\andrade_IT0011>
```



```
First Name: Kyla Mae
Last Name: Andrade
Age: 21
Contact Number: 09123456789
Course: BSIT-WMA
```

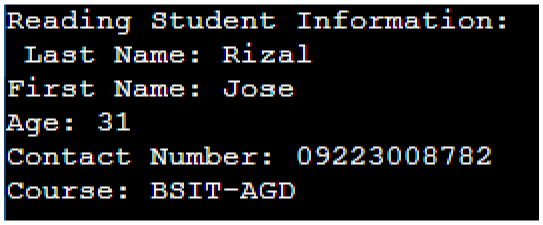
3.4 Activity for Reading File Contents and Display

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

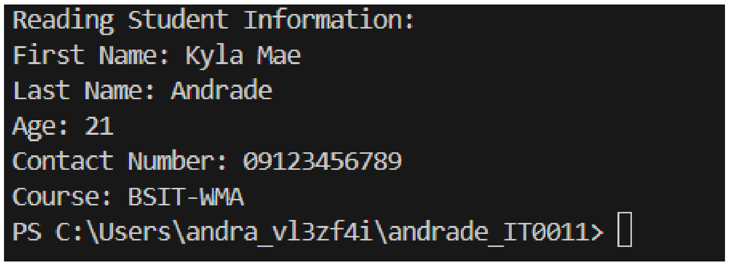


```
Reading Student Information:
 Last Name: Rizal
First Name: Jose
Age: 31
Contact Number: 09223008782
Course: BSIT-AGD
```

Source Code:

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

        print("Reading Student Information:")
        for line in contents:
            print(line.strip())

readstudentinf()
```

Output:



```
Reading Student Information:
First Name: Kyla Mae
Last Name: Andrade
Age: 21
Contact Number: 09123456789
Course: BSIT-WMA
PS C:\Users\andra_vl3zf4i\andrade_IT0011>
```

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 basically helps in combining text and variables easily as you use curly braces {} as placeholders in your string, and then call format() with the variables you want to insert. Let’s say for example, if you have name = "Mae" and age = 21, you can create a message like "My name is {} and I am {} years old." Using format(name, age) will fill in the placeholders, which in turn will result in "My name is Mae and I am 21 years old." Overall, this makes it straightforward to create sentences without making use of many + signs.

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

In Python, opening a file in read mode simply lets you look at the contents of the file without changing anything; and you would typically use this mode when you only want to read data from a file. While on the other hand, opening a file in write mode allows you to create a new file or even overwrite an existing one, which basically means you can change its contents so one can use this mode when you want to save new information or update what is already there.

3. Describe what string slicing is in Python. Provide a basic example of extracting a substring from a larger string.
String slicing allows you to extract a part of a string by simply specifying the starting and ending positions. With this, you use square brackets [] with a colon to indicate which part you want. Let’s say for example, if you have the string text = "FEU TECH", you can get "FEU" by using text[0:3], which starts at index 0 and goes up to, but does not include, index 3. In this way, string slicing basically helps you to easily grab specific pieces of text from a larger string.

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.
When saving information to a file in Python, using 'a' mode or append allows you to add new data to the end of the file without removing what is already there. While on the other hand, the 'w' mode or write will erase the existing content and basically start fresh. For example, if you have a file containing "Integrative Programing" and you open it in 'a' mode to add " and Technologies" the file will then contain "Integrative Programming and Technologies" This is particularly useful when you want to keep the original text and just add more.

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:
        content = file.read()
        print(content)
except FileNotFoundError:
    print("The file 'data.txt' does not exist.")
```

In this code, the try attempts to open and read the file, while except is the one that catches the FileNotFoundError if the file is not found, which then displays a message that the file does not exist.