



FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011
Integrative Programming and
Technologies

EXERCISE

3

String and File Handling

Student Name:	Julius Francis G. De Leon
Section:	TW24
Professor:	Prof. 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.
```

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
```

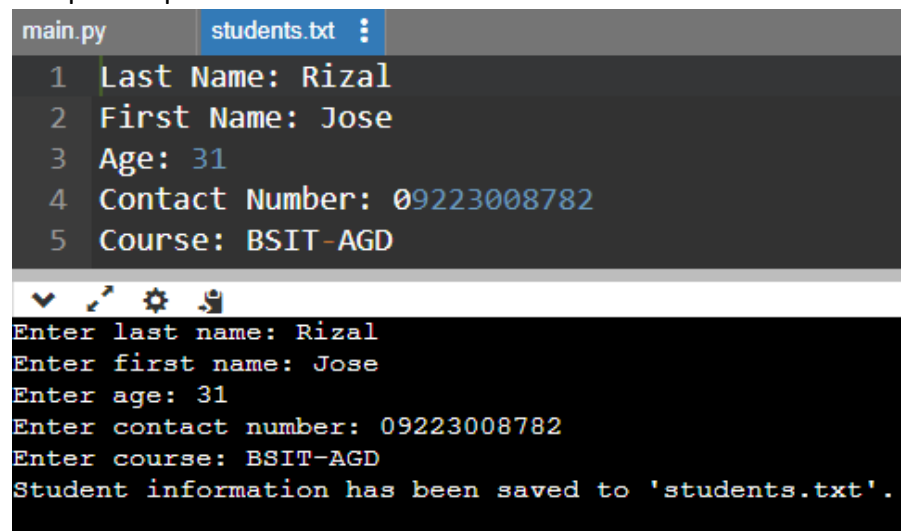
3.3. Practical Problem Solving with String Manipulation and File Handling

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 program with five lines of code. Below the code editor, the output of the program is shown in a terminal window. The program prompts the user for last name, first name, age, contact number, and course, and then displays 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'.
```

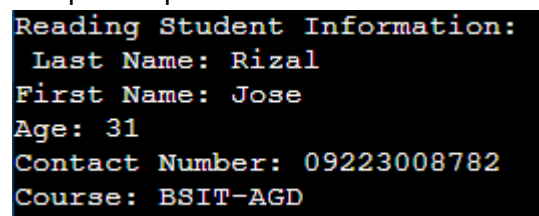
3.4 Activity for Reading File Contents and Display

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 reads the contents of the 'students.txt' file and displays the student information to the user.

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

INPUT AND OUTPUTS:

3.1. Activity for Performing String Manipulations

INPUT

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

age = int(input("Enter your age: "))

#slice the first three characters of the first name using string
manipulation
sliced_name = first_name[:3]

print('=====')
print("        STRING MANIPULATIONS (Program 1)        ")
print('=====')
print("Full Name           :", full_name)
print("Sliced Name          :", sliced_name)
# used the FORMAT() function
print("Greeting Message    : Greetings, {}! You are {} years
old.".format(sliced_name, age))
print('=====')
```

OUTPUT

```
Enter your first name: Julius Francis
Enter your last name: De Leon
Enter your age: 20

=====
        STRING MANIPULATIONS (Program 1)
=====
Full Name           : Julius Francis De Leon
Sliced Name          : Jul
Greeting Message    : Greetings, Jul! You are 20 years old.
=====
```

3.2 Activity for Performing String Manipulations

INPUT

```
print('=====')
print("      STRING MANIPULATIONS (Program 2)      ")
print('=====')
first_name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
full_name = first_name + " " + last_name
print('=====')
print(' ')

print('=====')
print("      STRING MANIPULATIONS OUTPUTS      ")
print('=====')
print("Full Name      :", full_name)
print("Full Name (Upper) :", full_name.upper())
print("Full Name (Lower) :", full_name.lower())
print("Length of Full Name:", len(full_name))
print('=====')
print(' ')
```

OUTPUT

```
=====
      STRING MANIPULATIONS (Program 2)
=====
Enter your first name: Julius Francis
Enter your last name: De Leon
=====

      STRING MANIPULATIONS OUTPUTS
=====
Full Name      : Julius Francis De Leon
Full Name (Upper) : JULIUS FRANCIS DE LEON
Full Name (Lower) : julius francis de leon
Length of Full Name: 22
=====
```

3.3. Practical Problem Solving with String Manipulation and File Handling

INPUT

```
print('=====')
print("          FILE HANDLING (Program 3)          ")
print('=====')

first_name = input("Enter your first name      : ")
last_name = input("Enter your last name       : ")
full_name = first_name + " " + last_name
age = int(input("Enter your age                : "))
contact_number = input("Enter your contact number : ")
course = input("Enter your course              : ")
print('=====')
print(' ')

#Open a file named "students.txt" in append mode
file = open("students.txt", "a")

#Write the formatted information to the file
file.write("Full Name      : " + full_name + "\n")
file.write("Age           : " + str(age) + "\n")
file.write("Contact Number   : " + contact_number + "\n")
file.write("Course            : " + course + "\n")

#display a confirmation message
print("Information has been saved successfully!")
print('File Name: students.txt')
print(' ')

#close the file
file.close()
```

OUTPUT

```
=====
          FILE HANDLING (Program 3)
=====
Enter your first name      : Julius Francis
Enter your last name       : De Leon
Enter your age             : 20
Enter your contact number : 09478998433
Enter your course          : BSITWMA
=====

Information has been saved successfully!
File Name: students.txt
```


3.4 Activity for Reading File Contents and Display

INPUT

```
print('Reading a file...')
print('=====')
print("      READING FILE NAMED: students.txt      ")
print('=====')

#Open a file named "students.txt" in read mode
file = open("students.txt", "r")

#Read the contents of the file
contents = file.read()

#Display the student information to the user
print(contents)

#Close the file
file.close()

print('=====')
print('File has been closed.')
print("Thank you for using the program!")
print()
```

OUTPUT

```
Reading a file...
=====
      READING FILE NAMED: students.txt
=====
Full Name      : Julius Francis De Leon
Age            : 20
Contact Number : 09478998433
Course         : BSITWMA
=====
File has been closed.
Thank you for using the program!
```

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 is very helpful in many ways, the format() function in Python lets you insert variables into a string by placing curly braces {} where you want the values to go. Then, you just use format() to pass in the variables. An example is from my code attached below...

```
print("Greeting Message : Greetings, {}! You are {} years old.".format(sliced_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

Opening a file in read mode, allows you to ONLY read the contents of the file if its already exist. A wrong file name or file that don't exist, will result raising an error. The other one the write mode, allows you to write information or data to the file. The difference between read mode and write mode when the file does not exist, the write mode will create a new one. However, when the file already exists or have the same file name, write mode will overwrite the existing content or information.

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

String slicing can be very intimidating at the start because of its complexity, however the string slicing in Python allows you to extract a part of a string by specifying a range of indices. You can use the syntax string[start:end], where start is the index where the slice begins, and end is the index where the slice ends (but the character at end is not included). This also means that the end was always higher to the start when it comes to positioning of the 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.

The 'a' mode in Python is used when you want to add new content to a file without losing what's already in it. In contrast, 'w' mode will overwrite the file completely. So, you'd use 'a' if you want to keep the existing data and just add more.

**** Answer in question 5 is in the next page****

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?

Code snippet:

```
try:
    with open("data.txt", "r") as file:
        content = file.read()
        print(content)
except FileNotFoundError:
    print("Oh no, 'data.txt' doesn't exist in this directory!")
    print("Please make sure to check the file name.")
```

This code will attempt to open data.txt and print its contents using the try-except block. If the file is missing, it won't crash the program; instead, it'll print a helpful message saying the file name doesn't exist.