



FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

IT0011

**Integrative Programming and
Technologies**

EXERCISE

3

String and File Handling

Student Name:	Eadward Andrei D. Lladoc
Section:	TB22
Professor:	Prof. 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 screenshot 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.
```

ANSWER:

SOURCE CODE - Copy and Paste

```
first_name = input("Enter your first name: ")# This will ask the user to input values
last_name = input("Enter your last name: ")
age = input("Enter your age: ")

full_name = first_name + " " + last_name # The concatenated names will be assigned to this variable

sliced_name = first_name[:3] # This function will then slice the first name and get the first three characters

# The message will be assigned to this variable so that when it is called out, it will display it.
# The age will also display depending on the value inputted by the user
greeting_message = f"Hello, {sliced_name}! Welcome. You are {age} years old."

# Output will be displayed with these prompts
print("\nFull Name:", full_name)
print("Sliced Name:", sliced_name)
print("Greeting Message:", greeting_message)
```

SOURCE CODE - Screenshot

```

first_name = input("Enter your first name: ")# This will ask the user to input values
last_name = input("Enter your last name: ")
age = input("Enter your age: ")

full_name = first_name + " " + last_name # The concatenated names will be assigned to this variable

sliced_name = first_name[:3] # This function will then slice the first name and get the first three characters

# The message will be assigned to this variable so that when it is called out, it will display it.
# The age will also display depending on the value inputted by the user
greeting_message = f"Hello, {sliced_name}! Welcome. You are {age} years old."

# Output will be displayed with these prompts
print("\nFull Name:", full_name)
print("Sliced Name:", sliced_name)
print("Greeting Message:", greeting_message)

```

OUTPUT

```

===== RESTART: C:/Users/EDMOND LLADOC/Lladoc_IT0011_Exercise3a.py =====
Enter your first name: Eadward
Enter your last name: Lladoc
Enter your age: 21

Full Name: Eadward Lladoc
Sliced Name: Ead
Greeting Message: Hello, Ead! Welcome. You are 21 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

```

ANSWER:

SOURCE CODE - Copy and Paste

```

first_name = input("Enter your first name: ") # this will prompt the user to input values

```

```
last_name = input("Enter your last name: ") # whatever string that is inputted here will be assigned to these variables
```

```
full_name = first_name + " " + last_name # this will concatenate the values assigned to the variables
```

```
upper_case_name = full_name.upper() # this function converts letters to uppercase
```

```
lower_case_name = full_name.lower() # this function converts letters to lowercase
```

```
name_length = len(full_name) # this function calculates and counts the length of the string
```

```
# Displaying the output together with the values
```

```
print("\nFull Name:", full_name)
```

```
print("Full Name (Upper Case):", upper_case_name)
```

```
print("Full Name (Lower Case):", lower_case_name)
```

```
print("Length of Full Name:", name_length)
```

SOURCE CODE - Screenshot

```
first_name = input("Enter your first name: ") # this will prompt the user to input values
last_name = input("Enter your last name: ") # whatever string that is inputted here will be assigned to these variables

full_name = first_name + " " + last_name # this will concatenate the values assigned to the variables

upper_case_name = full_name.upper() # this function converts letters to uppercase
lower_case_name = full_name.lower() # this function converts letters to lowercase

name_length = len(full_name) # this function calculates and counts the length of the string

# Displaying the output together with the values
print("\nFull Name:", full_name)
print("Full Name (Upper Case):", upper_case_name)
print("Full Name (Lower Case):", lower_case_name)
print("Length of Full Name:", name_length)
```

OUTPUT

```
===== RESTART: C:/Users/EDMOND LLADOC/Lladoc_IT0011_LabExercise3b.py =====
Enter your first name: Eadward
Enter your last name: Lladoc

Full Name: Eadward Lladoc
Full Name (Upper Case): EADWARD LLADOC
Full Name (Lower Case): eadward lladoc
Length of Full Name: 14
|
```

3.3. Practical Problem Solving with String Manipulation and File Handling

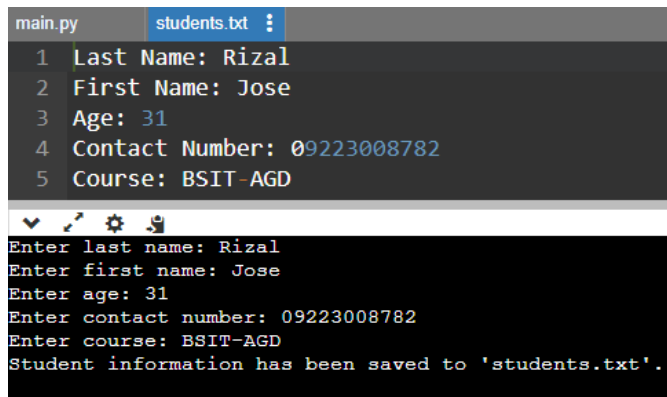
Objective: Apply string manipulation and file handling techniques to store student information in a file.

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 'students.txt' tab is active, displaying the following text:

```
1 Last Name: Rizal
2 First Name: Jose
3 Age: 31
4 Contact Number: 09223008782
5 Course: BSIT-AGD
```

Below the code editor is a terminal window showing the program's execution:

```
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'.
```

ANSWER:

SOURCE CODE - Copy and Paste

```
last_name = input("Enter last name: ") # this will get the last name of the user
first_name = input("Enter first name: ") # this will get the first name of the user
age = input("Enter age: ") # this will get the age of the user
contact_number = input("Enter contact number: ") # this will get the contact number of the user
course = input("Enter program: ") # this will get the program of the user

# this is so that the details will be formatted properly
student_info = f"Last Name: {last_name}\nFirst Name: {first_name}\nAge: {age}\nContact Number: {contact_number}\nCourse: {course}\n\n"

with open("students.txt", "a") as file: # this function will make the file in append mode to input the information
    file.write(student_info)

# this will display to confirm if data inputted by the user has been saved
print("Student information has been saved to 'students.txt'.")
```

SOURCE CODE - Screenshot

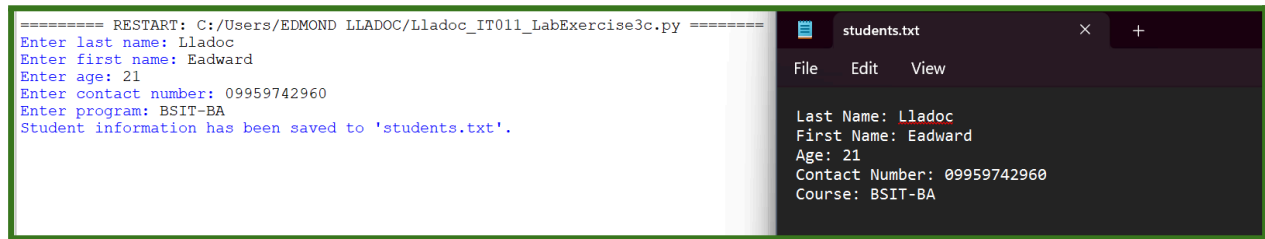
```
last_name = input("Enter last name: ") # this will get the last name of the user
first_name = input("Enter first name: ") # this will get the first name of the user
age = input("Enter age: ") # this will get the age of the user
contact_number = input("Enter contact number: ") # this will get the contact number of the user
course = input("Enter program: ") # this will get the program of the user

# this is so that the details will be formatted properly
student_info = f"Last Name: {last_name}\nFirst Name: {first_name}\nAge: {age}\nContact Number: {contact_number}\nCourse: {course}\n\n"

with open("students.txt", "a") as file: # this function will make the file in append mode to input the information
    file.write(student_info)

# this will display to confirm if data inputted by the user has been saved
print("Student information has been saved to 'students.txt'.")
```

OUTPUT



The screenshot shows a Python program running in a terminal window. The program prompts the user to enter student information, which is then saved to a file named 'students.txt'. The text editor window shows the contents of 'students.txt'.

```
===== RESTART: C:/Users/EDMOND LLADOC/Lladoc_IT011_LabExercise3c.py =====
Enter last name: Lladoc
Enter first name: Eadward
Enter age: 21
Enter contact number: 09959742960
Enter program: BSIT-BA
Student information has been saved to 'students.txt'.
```

students.txt

```
Last Name: Lladoc
First Name: Eadward
Age: 21
Contact Number: 09959742960
Course: BSIT-BA
```

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
```

ANSWER:

SOURCE CODE - Copy and Paste

```
try: # this will open the file 'students.txt' in read mode
    with open("students.txt", "r") as file: # used the 'r' function to read the file
        student_info = file.read()

    # this will display the information in the file
    print("Reading Student Information...\n")
    print(student_info)

except FileNotFoundError: # in cases where the file does not exist, this condition will be executed
    print("Error: The file 'students.txt' was not found.")
```


SOURCE CODE - Screenshot

```
try: # this will open the file 'students.txt' in read mode
    with open("students.txt", "r") as file: # used the 'r' function to read the file
        student_info = file.read()

    # this will display the information in the file
    print("Reading Student Information...\n")
    print(student_info)

except FileNotFoundError: # in cases where the file does not exist, this condition will be executed
    print("Error: The file 'students.txt' was not found.")
```

OUTPUT

```
===== RESTART: C:/Users/EDMOND LLADOC/Lladoc_IT011_LabExercise3d.py =====
Reading Student Information...

Last Name: Lladoc
First Name: Eadward
Age: 21
Contact Number: 09959742960
Course: BSIT-BA
```

QUESTION AND ANSWER:

1. How does the format() function help in combining variables with text in Python? Can you provide a simple example?

In Python, the format() function allows the user to combine variables with text within a string by using curly braces as placeholder, where the values assigned to said variables will be inserted. This makes it convenient to dynamically create formatted strings with multiple 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?

It's simple! The 'r' function indicates that the file will be open for reading only. You would use this when you want to read and/or display file content, and when you don't need to change said content or create a new file. The 'w' function indicates that the file will be open for writing only. You would use this when you want to create a new file or overwrite an existing one.

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

String slicing in Python is the process of getting only specific parts of a string by using start, end, and step values. Basically, you would do string slicing when you want to only extract a portion of a string. For example, I declared a string "Happy Birthday" and assigned it to the variable "greet", and I would only need the first word, I can execute it through the use of the following statements:

```
greet = "Happy Birthday"
greetsubstring = greet[0:4]

print(greetsubstring)
```

This is a great example of extracting a substring (Happy) from a larger string (Happy Birthday).

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.

Well, the 'w' mode overwrites files, and it can erase all existing data before writing new content. The 'a' mode adds new content to the end of an existing file without deleting its current contents — 'a' mode which means append.

If I were to open 'students.txt' using 'w' mode, and write 'Eadward' and 'Lladoc' consecutively, the content that will be saved in the file will only be 'Lladoc'. But if I were to open 'students.txt' using 'a' mode, and write 'Eadward' and 'Lladoc' consecutively, both of these strings will be saved in the file as contents.

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?

In such a case, I will use exception handling! This function will catch the error of the file not existing, and will then print an error message indicating that the action was invalid.

```
try:
    with open("data.txt", "r") as file:
        content = file.read()
        print("File Contents:\n", content)
except FileNotFoundError:
    print("Invalid action! File not found.")
```