Assignment 1: Exercises on Operators, Strings, and Lists

Arpit Sharma

PROV/MCA/7/24/056

MCA 'A'

Assignment Link: https://github.com/Arpit-Sharma777/python_assignment1.git

Part 1: Operators

Exercise 1: Arithmetic Operators

Write a Python program to perform the following operations:

1. Add, subtract, multiply, and divide two numbers (input by the user).
2. Use the modulus operator to find the remainder of their division.
3. Use the exponentiation operator to raise the first number to the power of the second number.
4. Perform floor division on the two numbers.

Solution:

```
# Get input from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
# Perform arithmetic operations
addition = num1 + num2
subtraction = num1 - num2
multiplication = num1 * num2
division = num1 / num2
modulus = num1 % num2
exponentiation = num1 ** num2
floor_division = num1 // num2
# Print the results
print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
```

```
print("Division:", division)
print("Modulus:", modulus)
print("Exponentiation:", exponentiation)
print("Floor Division:", floor_division)
```

input:

Enter the first number: 1

Enter the second number: 3

output:

Addition: 13.0

Subtraction: 7.0

Multiplication: 30.0

Division: 3.3333333333333335

Modulus: 1.0

Exponentiation: 1000.0

Floor Division: 3.0

Explanation

The code takes two floating-point numbers from the user, performs a series of arithmetic operations (addition, subtraction, multiplication, division, modulus, exponentiation, and floor division), and prints the results of each operation.

-> Input for First Number:

num1 = float(input("Enter the first number: "))

- Prompts the user to enter the first number.
- Converts the input to a float and stores it in the variable num1.

->Input for Second Number:

num2 = float(input("Enter the second number: "))

- Prompts the user to enter the second number.
- Converts the input to a float and stores it in the variable num2.

-> Arithmetic Operations: The following lines perform various arithmetic calculations:

addition = num1 + num2

- **Addition**: Adds num1 and num2.

subtraction = num1 - num2

- **Subtraction**: Subtracts num2 from num1.

*multiplication = num1 * num2*

- **Multiplication**: Multiplies num1 by num2.

division = num1 / num2

- **Division**: Divides num1 by num2. Note: This will raise an error if num2 is zero.

modulus = num1 % num2

- **Modulus**: Computes the remainder of the division of num1 by num2.

*exponentiation = num1 ** num2*

- **Exponentiation**: Raises num1 to the power of num2.

floor_division = num1 // num2

- **Floor Division**: Divides num1 by num2 and returns the largest integer less than or equal to the result.

-> Print the Results: The following lines output the results of the operations:

print("Addition:", addition)

print("Subtraction:", subtraction)

print("Multiplication:", multiplication)

print("Division:", division)

print("Modulus:", modulus)

print("Exponentiation:", exponentiation)

print("Floor Division:", floor_division)

- Each line prints the name of the operation followed by its result.

Exercise 2: Comparison Operators

Write a Python program that asks for two numbers and checks:

1. If the first number is greater than the second.
2. If the first number is equal to the second.
3. If the first number is less than or equal to the second.

Print the results.

Solution:

```
# Get input from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
# Check if the first number is greater than the second
greater_than = num1 > num2
print("Is the first number greater than the second?", greater_than)
# Check if the first number is equal to the second
equal_to = num1 == num2
print("Is the first number equal to the second?", equal_to)
# Check if the first number is less than or equal to the second
less_than_or_equal_to = num1 <= num2
print("Is the first number less than or equal to the second?", less_than_or_equal_to)
```

Input:

Enter the first number: 16

Enter the second number: 15

Output:

Is the first number greater than the second? True

Is the first number equal to the second? False

Is the first number less than or equal to the second? False

Explanation:

The code takes two floating-point numbers from the user and checks three conditions: whether the first number is greater than, equal to, or less than or equal to the second number. It then prints the results of these comparisons.

-> Input for First Number:

num1 = float(input("Enter the first number: "))

- Prompts the user to enter the first number.
- Converts the input to a float and stores it in the variable num1.

-> Input for Second Number:

num2 = float(input("Enter the second number: "))

- Prompts the user to enter the second number.
- Converts the input to a float and stores it in the variable num2.

-> Check if First Number is Greater:

greater_than = num1 > num2

print("Is the first number greater than the second?", greater_than)

- num1 > num2: Checks if the first number is greater than the second. The result (either True or False) is stored in greater_than.
- Prints whether num1 is greater than num2.

-> Check if First Number is Equal:

equal_to = num1 == num2

print("Is the first number equal to the second?", equal_to)

- num1 == num2: Checks if the two numbers are equal. The result is stored in equal_to.
- Prints whether num1 is equal to num2.

-> Check if First Number is Less Than or Equal:

less_than_or_equal_to = num1 <= num2

print("Is the first number less than or equal to the second?", less_than_or_equal_to)

- num1 <= num2: Checks if the first number is less than or equal to the second. The result is stored in less_than_or_equal_to.
- Prints whether num1 is less than or equal to num2.

Exercise 3: Logical Operators

Write a Python program that:

1. Takes three boolean values (True or False) as input.
2. Uses and, or, and not operators to return the result of combining them.

Solution:

```
 # Get input from the user
bool1 = input("Enter the first boolean value (True or False): ").lower() == "true"
bool2 = input("Enter the second boolean value (True or False): ").lower() == "true"
bool3 = input("Enter the third boolean value (True or False): ").lower() == "true"
# Use logical operators to combine the boolean values
and_result = bool1 and bool2 and bool3
or_result = bool1 or bool2 or bool3
not_result = not bool1
# Print the results
print("Result of AND operation:", and_result)
print("Result of OR operation:", or_result)
print("Result of NOT operation on the first value:", not_result)
```

Input:

```
Enter the first boolean value (True or False): true
Enter the second boolean value (True or False): false
Enter the third boolean value (True or False): true
```

Output:

```
Result of AND operation: False
Result of OR operation: True
Result of NOT operation on the first value: False
```

Explanation:

The code prompts the user for three boolean values, processes them using logical operations (AND, OR, NOT), and prints the results of these operations. The use of .lower() allows for case-insensitive input, ensuring that both "True" and "true" are interpreted correctly.

-> Input for Boolean Values:

bool1 = input("Enter the first boolean value (True or False): ").lower() == "true"

- Prompts the user to enter a boolean value (either "True" or "False").
- Converts the input to lowercase using .lower() to handle case insensitivity.
- Compares the input to the string "true". If the input matches, bool1 is set to True; otherwise, it is set to False.

The same logic is applied to bool2 and bool3:

bool2 = input("Enter the second boolean value (True or False): ").lower() == "true"

bool3 = input("Enter the third boolean value (True or False): ").lower() == "true"

-> Logical Operations:

- **AND Operation**:

and_result = bool1 and bool2 and bool3

  - Combines the three boolean values using the logical AND operator (and).
  - The result (True if all three are True, otherwise False) is stored in and_result.

- **OR Operation**:

or_result = bool1 or bool2 or bool3

  - Combines the three boolean values using the logical OR operator (or).
  - The result (True if at least one is True, otherwise False) is stored in or_result.

- **NOT Operation**:

not_result = not bool1

  - Applies the logical NOT operator (not) to bool1.
  - The result (True if bool1 is False, and vice versa) is stored in not_result.

-> Print the Results:

print("Result of AND operation:", and_result)

print("Result of OR operation:", or_result)

print("Result of NOT operation on the first value:", not_result)

- Outputs the results of the AND, OR, and NOT operations to the user.

Part 2: Strings

Exercise 4: String Manipulation

1. Take a string input from the user.
2. Display the following:
    - The length of the string.
    - The first and last character.
    - The string in reverse order.
    - The string in uppercase and lowercase.

Solution:

```
# Get input from the user
user_string = input("Enter a string: ")
# Calculate the length of the string
string_length = len(user_string)
# Get the first and last characters
first_char = user_string[0]
last_char = user_string[-1]
# Reverse the string
reversed_string = user_string[::-1]
# Convert the string to uppercase and lowercase
uppercase_string = user_string.upper()
lowercase_string = user_string.lower()
# Print the results
print("Length of the string:", string_length)
print("First character:", first_char)
print("Last character:", last_char)
print("Reversed string:", reversed_string)
print("Uppercase string:", uppercase_string)
print("Lowercase string:", lowercase_string)
```

Input

Enter a string: Arpit

Output

Length of the string: 5

First character: A

Last character: t

Reversed string: tiprA

Uppercase string: ARPIT

Lowercase string: arpit

Explanation:

The code takes a string input from the user and performs several operations: it calculates the length of the string, retrieves the first and last characters, reverses the string, and converts it to both uppercase and lowercase. Finally, it prints all these results for the user.

-> Input from the User:

user_string = input("Enter a string: ")

- Prompts the user to enter a string and stores it in the variable user_string.

-> Calculate the Length of the String:

string_length = len(user_string)

- Uses the len() function to calculate the number of characters in user_string and stores the result in string_length.

-> Get the First and Last Characters:

first_char = user_string[0]

last_char = user_string[-1]

- user_string[0]: Accesses the first character of the string.
- user_string[-1]: Accesses the last character of the string (using negative indexing).
- The first and last characters are stored in first_char and last_char, respectively.

-> Reverse the String:

reversed_string = user_string[::-1]

- This uses slicing to reverse the string. The syntax [::-1] means to take the string from start to end but with a step of -1, effectively reversing it.

-> Convert the String to Uppercase and Lowercase:

uppercase_string = user_string.upper()

lowercase_string = user_string.lower()

- user_string.upper(): Converts all characters in the string to uppercase and stores the result in uppercase_string.
- user_string.lower(): Converts all characters in the string to lowercase and stores the result in lowercase_string.

-> Print the Results:

print("Length of the string:", string_length)

print("First character:", first_char)

print("Last character:", last_char)

print("Reversed string:", reversed_string)

print("Uppercase string:", uppercase_string)

print("Lowercase string:", lowercase_string)

- Outputs the results of the operations: the length of the string, the first and last characters, the reversed string, and the uppercase and lowercase versions of the string.

Exercise 5: String Formatting

Write a program that asks for the user's name and age, and displays the message in this format:

Solution:

```
# Get input from the user
name = input("Enter your name: ")
age = input("Enter your age: ")
# Display the formatted message
print(f"Hello {name}, you are {age} years old.")
```

Input:

Enter your name: Arpit

Enter your age: 21

Output:

Hello Arpit, you are 21 years old.

Explanation:

The code prompts the user for their name and age, then prints a personalized message that includes both pieces of information. The use of an f-string makes it easy to format the output in a readable way.

-> Input for Name:

name = input("Enter your name: ")

- Prompts the user to enter their name.
- The input is stored as a string in the variable name.

-> Input for Age:

age = input("Enter your age: ")

- Prompts the user to enter their age.
- The input is also stored as a string in the variable age.

-> Display the Formatted Message:

print(f"Hello {name}, you are {age} years old.")

- Uses an f-string (formatted string literal) to create a message that includes the user's name and age.
- The placeholders {name} and {age} are replaced with the values stored in the name and age variables.
- Finally, the message is printed to the console.

Exercise 6: Substring Search

Write a Python program that:

1. Asks for a sentence input from the user.
2. Asks for a word to search in the sentence.
3. Outputs whether the word exists in the sentence and, if it does, at which position (index).

Solution:

```
# Get input from the user
sentence = input("Enter a sentence: ")
word_to_search = input("Enter a word to search for: ")
# Check if the word exists in the sentence
if word_to_search in sentence:
```

```
  # Find the index of the word
    index = sentence.find(word_to_search)
    print(f"The word '{word_to_search}' exists in the sentence at index {index}.")
else:
    print(f"The word '{word_to_search}' does not exist in the sentence.")
```

Input:

Enter a sentence: *I'm not fooled by this fame game This twitter game, it's so lame 'Cause I'm verified on the inside I don't need a blue check to be satisfied*

Enter a word to search for: twitter

Output:

The word 'twitter' exists in the sentence at index 38.

Explanation:

The code takes a sentence and a word as input, checks if the word exists in the sentence, finds its index if it does, and provides feedback to the user based on whether the word was found or not.

-> Input from the User:

sentence = input("Enter a sentence: ")

word_to_search = input("Enter a word to search for: ")

- This part prompts the user to enter a sentence and a word they want to search for in that sentence. The inputs are stored in the variables sentence and word_to_search.

-> Check if the Word Exists:

if word_to_search in sentence:

- This line checks if the word specified by the user (word_to_search) is present in the sentence. The in keyword is used to perform this check.

->Finding the Index:

index = sentence.find(word_to_search)

- If the word is found in the sentence, the code uses the find() method to get the index of the first occurrence of that word. This index is stored in the variable index.

->Output the Result:

print(f"The word '{word_to_search}' exists in the sentence at index {index}.")

- If the word is found, it prints a message indicating the word exists and provides its index.

-> Handling the Case Where the Word is Not Found:

else:

print(f"The word '{word_to_search}' does not exist in the sentence.")

- If the word is not found in the sentence, the code goes to the else block and prints a message stating that the word does not exist in the sentence.

Part 3: Lists

Exercise 7: List Operations

Write a Python program that:

1. Creates a list of 5 numbers (input from the user).
2. Displays the sum of all the numbers in the list.
3. Finds the largest and smallest number in the list.

Solution:

```
# Create an empty list to store the numbers
numbers = []
# Get input from the user for 5 numbers
for i in range(5):
    number = float(input(f"Enter number {i+1}: "))
    numbers.append(number)
# Calculate the sum of the numbers
sum_of_numbers = sum(numbers)
# Find the largest and smallest numbers
largest_number = max(numbers)
smallest_number = min(numbers)
# Print the results
print("List of numbers:", numbers)
print("Sum of the numbers:", sum_of_numbers)
print("Largest number:", largest_number)
print("Smallest number:", smallest_number)
```

Input:

Enter number 1: 4

Enter number 2: 6

Enter number 3: 8

Enter number 4: 4

Enter number 5: 7

Output:

List of numbers: [4.0, 6.0, 8.0, 4.0, 7.0]

Sum of the numbers: 29.0

Largest number: 8.0

Smallest number: 4.0

Explanation:

This code collects five floating-point numbers from the user, stores them in a list, calculates their sum, and identifies the largest and smallest numbers among them. It then outputs these results to the user.

->Create an Empty List:

numbers = []

- This line initializes an empty list called numbers that will be used to store the numbers input by the user.

-> Get Input from the User:

for i in range(5):

number = float(input(f"Enter number {i+1}: "))

numbers.append(number)

- This loop iterates five times (from 0 to 4) to collect five numbers from the user.
- During each iteration, it prompts the user to enter a number, converts the input to a floating-point number using float(), and stores it in the variable number.
- The append() method adds each entered number to the numbers list.

-> Calculate the Sum of the Numbers:

sum_of_numbers = sum(numbers)

- This line calculates the sum of all numbers in the numbers list using the built-in sum() function and stores the result in sum_of_numbers.

-> Find the Largest and Smallest Numbers:

largest_number = max(numbers)

smallest_number = min(numbers)

- The max() function is used to find the largest number in the numbers list and store it in largest_number.
- The min() function finds the smallest number in the list and stores it in smallest_number.

-> Print the Results:

print("List of numbers:", numbers)

print("Sum of the numbers:", sum_of_numbers)

print("Largest number:", largest_number)

print("Smallest number:", smallest_number)

- Finally, the code prints the list of numbers, the sum of those numbers, the largest number, and the smallest number.

Exercise 8: List Manipulation

1. Create a list of 5 of your favorite fruits.
2. Perform the following:
   - o Add one more fruit to the list.
   - o Remove the second fruit from the list.
   - o Print the updated list.

Solution:

```
# Take input from the user
fruits = input("Enter your favorite fruits separated by commas: ").split(',')
# Strip any extra whitespace
fruits = [fruit.strip() for fruit in fruits]
# Add one more fruit
new_fruit = input("Enter one more fruit to add: ")
fruits.append(new_fruit.strip())
# Remove the second fruit (if it exists)
if len(fruits) > 1:
```

```
    fruits.pop(1)
# Print the updated list
print("Updated list of fruits:")
print(fruits)
```

Input:

Enter your favorite fruits separated by commas: Mango,Apple ,Banana, Kiwi,Pineapple

Enter one more fruit to add: Custured apple

Output:

Updated list of fruits:

['Mango', 'Banana', 'Kiwi', 'Pineapple', 'Custured apple']

Explanation:

This code collects a list of favorite fruits from the user, processes the input to clean it up, adds another fruit, potentially removes the second fruit, and then displays the updated list of fruits.

->Take Input from the User:

fruits = input("Enter your favorite fruits separated by commas: ").split(',')

- This line prompts the user to enter their favorite fruits as a single string, with each fruit separated by a comma.
- The split(',') method splits this string into a list of fruits, using the comma as the delimiter.

-> Strip Any Extra Whitespace:

fruits = [fruit.strip() for fruit in fruits]

- This line uses a list comprehension to iterate through the fruits list.
- For each fruit, it calls the strip() method, which removes any leading or trailing whitespace from the string. This ensures that each fruit is cleanly formatted.

-> Add One More Fruit:

new_fruit = input("Enter one more fruit to add: ")

fruits.append(new_fruit.strip())

- Here, the code prompts the user to enter another fruit.
- The strip() method is again used to remove any extra whitespace, and the new fruit is added to the fruits list using the append() method.

-> Remove the Second Fruit (if it exists):

if len(fruits) > 1:

fruits.pop(1)

- This conditional checks if the fruits list has more than one item (to ensure there is a second fruit to remove).
- If true, it uses the pop(1) method to remove the fruit at index 1 (the second fruit) from the list.

-> Print the Updated List:

print("Updated list of fruits:")

print(fruits)

- Finally, the code prints a message indicating that the list of fruits has been updated, followed by the current contents of the fruits list.

Exercise 9: Sorting a List

Write a Python program that:

1. Asks the user to input a list of 5 numbers.
2. Sorts the list in ascending order and displays it.
3. Sorts the list in descending order and displays it.

Solution:

```
# input a list of 5 numbers
numbers = []
for i in range(5):
    while True:
        num = float(input(f"Enter number {i + 1}: "))
        numbers.append(num)
        break
# Sort the list in ascending order and display it
ascending = sorted(numbers)
print("Sorted in ascending order:", ascending)
```

#Sort the list in descending order and display it

descending = sorted(numbers, reverse=True)

print("Sorted in descending order:", descending)

Input:

Enter number 1: 4

Enter number 2: 5

Enter number 3: 1

Enter number 4: 3

Enter number 5: 9

Output:

Sorted in ascending order: [1.0, 3.0, 4.0, 5.0, 9.0]

Sorted in descending order: [9.0, 5.0, 4.0, 3.0, 1.0]

Explanation:

The code collects 5 numbers from the user and stores them in a list.

It then sorts the list twice: once in ascending order and once in descending order, displaying both results.

-> **Initialization**:

numbers = []

- An empty list named numbers is created to store the user inputs.

->**Input Collection**:

for i in range(5):

while True:

num = float(input(f"Enter number {i + 1}: "))

numbers.append(num)

break

- A for loop runs 5 times, allowing the user to input 5 numbers.

- Inside the loop, a while True statement is used, but it immediately breaks after appending the number. This loop is redundant here since you only need to collect one number per iteration.
- The input() function prompts the user to enter a number, which is then converted to a float and appended to the numbers list.

-> **Sorting in Ascending Order**:

ascending = sorted(numbers)

print("Sorted in ascending order:", ascending)

- The sorted() function sorts the numbers list in ascending order and stores it in the variable ascending.
- The sorted list is printed.

-> **Sorting in Descending Order**:

descending = sorted(numbers, reverse=True)

print("Sorted in descending order:", descending)

- The sorted() function is called again with reverse=True, sorting the list in descending order and storing it in descending.
- This sorted list is also printed.

Exercise 10: List Slicing

Given the list numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], perform the following:

1. Print the first 5 elements.
2. Print the last 5 elements.
3. Print the elements from index 2 to index 7.

Solution:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Print the first 5 elements

print("First 5 elements:", numbers[:5])

# Print the last 5 elements

print("Last 5 elements:", numbers[-5:])

# Print the elements from index 2 to index 7

print("Elements from index 2 to index 7:", numbers[2:8])
```

Input:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Output:

First 5 elements: [1, 2, 3, 4, 5]

Last 5 elements: [6, 7, 8, 9, 10]

Elements from index 2 to index 7: [3, 4, 5, 6, 7, 8]

Explanation:

The code initializes a list of numbers from 1 to 10.

It uses slicing to print:

- The first 5 elements.
- The last 5 elements.
- Elements from index 2 to index 7.

-> **List Initialization**:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- A list named numbers is created, containing the integers from 1 to 10.

->**Printing the First 5 Elements**:

print("First 5 elements:", numbers[:5])

- The expression numbers[:5] uses list slicing to get the first 5 elements of the list.
- This slice starts from the beginning of the list (index 0) and goes up to, but not including, index 5.
- The output will be: First 5 elements: [1, 2, 3, 4, 5].

-> **Printing the Last 5 Elements**:

print("Last 5 elements:", numbers[-5:])

- The expression numbers[-5:] uses negative indexing to access elements from the end of the list.
- Here, -5 refers to the fifth element from the end, and the slice goes from there to the end of the list.
- The output will be: Last 5 elements: [6, 7, 8, 9, 10].

-> **Printing Elements from Index 2 to Index 7**:

print("Elements from index 2 to index 7:", numbers[2:8])

- The expression numbers[2:8] slices the list to get elements starting from index 2 up to, but not including, index 8.
- This means it will include the elements at indices 2, 3, 4, 5, 6, and 7.
- The output will be: Elements from index 2 to index 7: [3, 4, 5, 6, 7, 8].

Bonus Challenge

Exercise 11: Nested List

Write a Python program that:

1. Takes input of 3 students' names and their respective scores in 3 subjects.
2. Stores them in a nested list.
3. Prints each student's name and their average score

Solution:

```
# Initialize an empty list to store student data
student_data = []
# Loop to collect data for 3 students
for i in range(3):
    # Get the name of the student
    student_name = input(f"Enter the name of student {i + 1}: ")
    scores = []  # Initialize an empty list to store scores for the current student
        # Loop to collect scores for 3 subjects
    for j in range(3):
        # Get the score for the current subject
        score = float(input(f"Enter score for subject {j + 1} for {student_name}: "))
        scores.append(score)  # Add the score to the scores list
```

```
    # Add the student's name and scores to the student_data list
    student_data.append([student_name, scores])
# Loop through the collected student data to calculate and display average scores
for student in student_data:
    name = student[0]  # Get the student's name
    scores = student[1]  # Get the student's scores
    average_score = sum(scores) / len(scores)  # Calculate the average score
    # Print the student's name and average score
    print(f"{name}: Average Score = {average_score}")
```

Input:

```
Enter the name of student 1: Arp
Enter score for subject 1 for Arp: 89
Enter score for subject 2 for Arp: 80
Enter score for subject 3 for Arp: 85
Enter the name of student 2: Anl
Enter score for subject 1 for Anl: 89
Enter score for subject 2 for Anl: 96
Enter score for subject 3 for Anl: 56
Enter the name of student 3: shu
Enter score for subject 1 for shu: 82
Enter score for subject 2 for shu: 88
Enter score for subject 3 for shu: 93
```

Output:

```
Arp: Average Score = 84.66666666666667
Anl: Average Score = 80.33333333333333
shu: Average Score = 87.66666666666667
```

Explanation:

The code allows users to enter names and scores for three students across three subjects, and then it calculates and displays each student's average score.

1. Initialize the List

student_data = []

This line creates an empty list called student_data that will store information about each student, including their name and scores.

2. Collect Student Information

for i in range(3):

This loop will iterate three times (for three students). The variable i will take values 0, 1, and 2 in each iteration.

a. Get Student Name

student_name = input(f"Enter the name of student {i + 1}: ")

In each iteration, it prompts the user to enter the name of the student. The i + 1 is used to display a human-friendly count (starting from 1).

b. Collect Scores

scores = []

for j in range(3):

An empty list called scores is initialized to store the scores for the current student. A nested loop runs three times (for three subjects).

i. Get Scores

score = float(input(f"Enter score for subject {j + 1} for {student_name}: "))

scores.append(score)

In each iteration of the inner loop, it prompts the user to enter a score for each subject. The input is converted to a float to allow decimal scores and added to the scores list.

c. Store Student Data

student_data.append([student_name, scores])

After collecting the name and scores for a student, a list containing the student's name and their scores is appended to student_data.

3. Calculate and Print Average Scores

for student in student_data:

This loop iterates through the list of students collected earlier.

a. Extract Name and Scores

name = student[0]

scores = student[1]

For each student, it retrieves their name and scores.

b. Calculate Average Score

average_score = sum(scores) / len(scores)

The average score is calculated by summing the scores and dividing by the number of subjects (3 in this case).

c. Print Results

print(f"{name}: Average Score = {average_score}")

Finally, it prints out the student's name along with their average score.