Assignment 1: Exercises on Operators, Strings, and Lists

Name: Jyotirmayee Sahu

Date: 20/10/2024

GitHub Repository: https://github.com/sjyotii31/python

Introduction:

This report provides a comprehensive overview of the solutions to a series of exercises focused on key Python concepts, including operators, strings, and lists. These exercises explore fundamental programming principles, offering practical applications of arithmetic, comparison, and logical operators, as well as string manipulation and list handling.

Throughout this assignment, I applied various Python constructs to solve problems, structured the code to ensure clarity, and incorporated meaningful comments to explain each step. Additionally, I reflected on the techniques used in each exercise, which deepened my understanding of Python's core functionality. This report outlines not only the technical approach to solving each exercise but also highlights key insights and takeaways that I encountered while working through the tasks.

Part 1: Operators:

Exercise 1: Arithmetic Operators

Objective:

Write a Python program to perform arithmetic operations (addition, subtraction, multiplication, division, modulus, exponentiation, and floor division) on two user-input numbers.

Code :

```
# Arithmetic Operations

# Taking user input

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

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


# Performing operations

addition = num1 + num2

subtraction = num1 - num2

multiplication = num1 * num2
```

```
division = num1 / num2
modulus = num1 % num2
exponentiation = num1 ** num2
floor_division = num1 // num2

# Displaying the results
print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
print("Division:", round(division, 2))
print("Modulus:", modulus)
print("Exponentiation:", exponentiation)
print("Floor Division:", floor division)
```

output:

```
Enter first number:  10
Enter second number:  3
Addition: 13.0
Subtraction: 7.0
Multiplication: 30.0
Division: 3.33
Modulus: 1.0
Exponentiation: 1000.0
Floor Division: 3.0
```

Approach:

I have taken two numbers from the user and used Python's built-in operators (+, -, *, /, %, **, and //) to perform the required calculations. The results are then displayed to the user.

Key Learnings:

This exercise helped me practice how to handle user inputs and perform basic arithmetic operations in Python. I also learned about floor division and exponentiation.

Exercise 2: Comparison Operators

Objective:

Check if the first number is greater than, equal to, or less than or equal to the second number**.**

Code:

```
# Comparison Operators

# Taking user input

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

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


# Comparing the numbers

greater = num1 > num2

equal = num1 == num2

less_equal = num1 <= num2


# Displaying the results

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

print("Are both numbers equal?", equal)

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

Output:

```
Enter first number:  80

Enter second number:  77

Is the first number greater than the second? True

Are both numbers equal? False

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

Approach:

I have used the comparison operators (>, ==, <=) to compare the two user-input numbers and have printed the results.

Key Learnings:

This task familiarized me with conditional comparisons and Python interprets them when comparison are done between two values.

Exercise 3: Logical Operators

Objective:

Take three boolean values and combine them using and, or, and not operators.

Code:

```
# Logical Operators

# Taking boolean inputs

val1 = bool(int(input("Enter first boolean (1 for True, 0 for False): ")))

val2 = bool(int(input("Enter second boolean (1 for True, 0 for False): ")))

val3 = bool(int(input("Enter third boolean (1 for True, 0 for False): ")))


# Applying logical operators

and_result = val1 and val2 and val3

or_result = val1 or val2 or val3

not_result = not val1


# Displaying the results

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

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

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

output:

```
Enter first boolean (1 for True, 0 for False):  3

Enter second boolean (1 for True, 0 for False):  5

Enter third boolean (1 for True, 0 for False):  7

Result of AND operation: True

Result of OR operation: True

Result of NOT operation on first value: False
```

Approach:

I have used boolean values as input and applied logical operators to compute the combined result. This program prints the results of and, or, and not operations.

Key Learnings:

This exercise is useful as it made me understand who logical operations does work in Python and how to use them to evaluate conditions.

Part 2: Strings

Exercise 4: String Manipulation

Objective:

Perform various operations on a string, such as finding its length, reversing it, and changing the case.

Code:

```
# String Manipulation

user_string = input("Enter a string: ")


# Display results

print("Length of the string:", len(user_string))

print("First character:", user_string[0])

print("Last character:", user_string[-1])

print("String in reverse:", user_string[::-1])

print("Uppercase:", user_string.upper())

print("Lowercase:", user_string.lower())
```

output:

```
Enter a string:  33

Length of the string: 2

First character: 3

Last character: 3

String in reverse: 33

Uppercase: 33

Lowercase: 33
```

Approach:

I used Python's string methods like len(), slicing ([::-1]), upper(), and lower() to perform the necessary operations on the string.

Key Learnings:

I learned how to manipulate strings and retrieve specific information like length, first and last characters, and reverse strings in Python.

Exercise 5: String Formatting

Objective:

Take the user's name and age and format it into a sentence.

Code:

```
# String Formatting

name = input("Enter your name: ")

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


# Formatting output

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

Output:

Enter your name: jyoti

Enter your age: 22

Hello jyoti, you are 22 years old.

Approach:

I have used the Python's f-string formatting to display the name and age in the required format.

Key Learnings:

This exercise introduced me to string formatting techniques, which are useful for creating formatted output.

Exercise 6: Substring Search

Objective:

Find whether a word exists in a sentence and, if it does, print its index.

Code:

```
# Substring Search

sentence = input("Enter a sentence: ")

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


# Checking if the word is in the sentence

if word in sentence:

   index = sentence.index(word)

   print(f"The word '{word}' is present at index {index}.")

else:

   print(f"The word '{word}' is not in the sentence.")
```

Output:

Enter a sentence: english is a popular language

Enter a word to search: is

The word 'is' is present at index 4.

Approach:

I have used Python's in operator to check for the presence of the word and index() method to find its position.

Key Learnings:

This exercise taught me how to search for substrings in a string and how to handle user inputs in a search operation.

Part 3: Lists

Exercise 7: List Operations

Objective:

Create a list of numbers, then find the sum, largest, and smallest values.

Code:

```
# List Operations
numbers = [float(input(f"Enter number {i+1}: ")) for i in range(5)]

# Performing operations
total_sum = sum(numbers)
largest = max(numbers)
smallest = min(numbers)

# Displaying results
print("Sum of numbers:", total_sum)
print("Largest number:", largest)
print("Smallest number:", smallest)
```

Output:

```
Enter number 1:  1
Enter number 2:  44
Enter number 3:  78
Enter number 4:  88
Enter number 5:  10
Sum of numbers: 221.0
Largest number: 88.0
Smallest number: 1.0
```

Approach:

I used Python's sum (), max (), and min () functions to perform these operations on a list of user-input numbers.

Key Learnings:

I became familiar with manipulating lists and applying basic list operations in Python.

Exercise 8: List Manipulation

Objective:

Add and remove elements from a list of fruits.

Code:

```
# List Manipulation
fruits = [input(f"Enter your favorite fruit {i+1}: ") for i in range(5)]

# Adding and removing elements
fruits.append(input("Enter one more fruit: "))
fruits.pop(1)  # Removing second fruit

# Displaying updated list
print("Updated list of fruits:", fruits)
```

Output:

Enter your favorite fruit 1: mango

Enter your favorite fruit 2: jack fruit

Enter your favorite fruit 3: custurd apple

Enter your favorite fruit 4: grapes

Enter your favorite fruit 5: orange

Enter one more fruit: apple

Updated list of fruits: ['mango', 'custurd apple', 'grapes', 'orange', 'apple']

Approach:

I used append () to add a fruit to the list and pop () to remove the second fruit.

Key Learnings:

This exercise introduced me to modifying lists by adding and removing elements dynamically.

Exercise 9: Sorting a List

Objective:

Sort a list of numbers in both ascending and descending order.

Code:

```
# Sorting a List
numbers = [float(input(f"Enter number {i+1}: ")) for i in range(5)]

# Sorting
ascending = sorted(numbers)
descending = sorted(numbers, reverse=True)

# Displaying results
print("List in ascending order:", ascending)
print("List in descending order:", descending)
```

Output:

```
Enter number 1:  79
Enter number 2:  22
Enter number 3:  56
Enter number 4:  31
Enter number 5:  09
List in ascending order: [9.0, 22.0, 31.0, 56.0, 79.0]
List in descending order: [79.0, 56.0, 31.0, 22.0, 9.0]
```

Approach:

I used Python's sorted() function to sort the list in both orders.

Key Learnings:

I learned how to sort lists using the sorted() function and how to reverse the sorting order using the reverse argument.

Exercise 10: List Slicing

Objective:

Slice a list to retrieve specific subsets of elements.

Code:

List Slicing

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


# Slicing the list
print("First 5 elements:", numbers[:5])
print("Last 5 elements:", numbers[-5:])
print("Elements from index 2 to 7:", numbers[2:8])
```

Output:

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

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

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

Approach:

I used Python's list slicing ([:5], [-5:], and [2:8]) to extract specific portions of the list.

Key Learnings:

This task helped me understand the concept of slicing lists to retrieve specific elements or ranges.

Bonus Challenge: Nested List

Exercise 11: Nested List

Objective:

Store and process a nested list containing the names and scores of three students, and calculate their average scores.

Code:

```
# Nested List
students = []
for i in range(3):
    name = input(f"Enter student {i+1}'s name: ")
    scores = [float(input(f"Enter {subject} score: ")) for subject in ['Math', 'Science', 'English']]
    students.append([name, scores])
```

```
# Displaying each student's average score
for student in students:
    name, scores = student
    average = sum(scores) / len(scores)
    print(f"{name}'s average score: {average:.2f}")
```

Output:

```
Enter student 1's name:  jyoti
Enter Math score:  31
Enter Science score:  35
Enter English score:  40
Enter student 2's name:  hanshika
Enter Math score:  32
Enter Science score:  32
Enter English score:  39
Enter student 3's name:  sruti
Enter Math score:  35
Enter Science score:  36
Enter English score:  30
jyoti's average score: 35.33
hanshika's average score: 34.33
sruti's average score: 33.67
```

Approach:

I created a nested list structure to store students' names and scores. Then, I used a loop to calculate the average score for each student and display it.

Key Learnings:

This exercise taught me how to work with nested data structures and how to manipulate lists within lists.

Conclusion

This assignment reinforced my understanding of Python operators, strings, and lists. It allowed me to explore essential Python concepts, such as arithmetic and logical operations, string manipulations, and list processing. Additionally, I gained practical experience in writing clean, readable code with comments explaining each step. This will be beneficial for more complex projects and collaborative work in the future.