Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Spring 2024, B.Sc. in CSE (DAY)

Lab Report NO # 01

Course Title: Artificial Intelligence Lab
Course Code: CSE 316 **Section: CSE 213 – D1**

Problems / Tasks / Domains:

- Write a Python program to get the 4th element from the beginning and the 4th element from the last of a tuple.

```
Sample Input:
tuplex = ("w", 3, "r", "e", "s", "o", "u", "r", "c", "e")
Sample Output:
e,u
```

- Write a Python Program to Count Even and Odd Numbers in a list.

Student Details

Name	ID
Md. Shahidul Islam Prodhan	**213902017**

Lab Assigned Date: 15th March 2024

Submission Date: 17th March 2024

Course Teacher's Name: Fairuz Shaiara, Lecturer

[For Teacher's use only: Don't write anything inside this box]

Lab Report Status

Marks:	**Signature:**
Comments:	**Date:**

1. TITLE OF THE LAB REPORT EXPERIMENT

Introduction to Basic Operations on Python (Part-1, 2)

2. OBJECTIVES/AIM

The primary objectives of this lab is to learn about basic operations, conditions, loops, variables, functions and statements in python. Basic Operations on Python such as Lists, Tuple, Dictionary, Numpy, Pandas, Matplotlib.

3. PROCEDURE / ANALYSIS / DESIGN

Problem 1	Problem 2
Purpose: The purpose of the code is to extract the 4th element from both the beginning and end of a given tuple. **Design and Approach**: The design involves a simple function that takes a tuple (tuplex) as input. It employs indexing to directly access the 4th element from both ends of the tuple. **Implementation**: The get_elements function accesses the 4th element from the beginning using tuplex[3]. It also accesses the 4th element from the end using negative indexing, tuplex[-4]. The function then returns both elements as a tuple.	**Purpose**: The purpose of the code is to count the number of even and odd numbers in a given list. **Design and Approach**: The code utilizes a straightforward iterative approach to traverse the list. It employs conditional statements to check whether each number is even or odd. **Implementation**: The count_even_odd function iterates through each element in the numbers list. It increments the even_count variable if the number is even (num % 2 == 0), otherwise increments odd_count. The function returns a tuple containing the counts of even and odd numbers.

Complexity:
Code 1 is simpler as it involves basic tuple manipulation and indexing.
Code 2 is slightly more complex due to iteration and conditional statements.

Functionality:
Code 1 performs a specific task of extracting elements from a tuple.
Code 2 is more versatile as it counts even and odd numbers, applicable to various scenarios.

Python Features:
Both codes utilize fundamental Python features such as indexing, iteration, and conditional statements.
Code 2 demonstrates the usage of functions and returning multiple values as a tuple.

4. IMPLEMENTATION

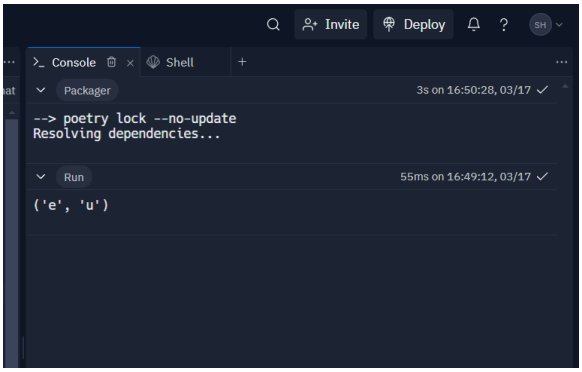
```python
def get_elements(tuplex):

  fourth_from_beginning = tuplex[3]
  fourth_from_end = tuplex[-4]
  return fourth_from_beginning, fourth_from_end

tuplex = ("w", 3, "r", "e", "s", "o", "u", "r", "c", "e")

result = get_elements(tuplex)

print(result)
```

Code 1

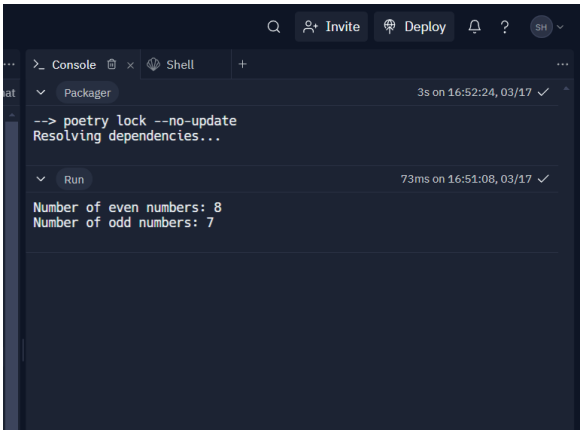
```python
def count_even_odd(numbers):
  even_count = 0
  odd_count = 0
  for num in numbers:
      if num % 2 == 0:
          even_count += 1
      else:
          odd_count += 1
  return even_count, odd_count

numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
even, odd = count_even_odd(numbers)

print("Number of even numbers:", even)
print("Number of odd numbers:", odd)
```

Code 2

5. TEST RESULT / OUTPUT



Code 1 Output



Code 2 Output

6. ANALYSIS AND DISCUSSION

Code 1

It aims to efficiently extract specific elements from a tuple. It provides a concise solution for accessing elements based on their positions in the tuple. The code demonstrates an effective design by directly accessing elements using indexing. It showcases the simplicity and elegance of python syntax for tuple manipulation. This approach ensures straightforward implementation and readability. It's particularly useful when dealing with tuples where positional information is crucial.

Code 2

It focuses on counting even and odd numbers within a list. It provides a practical solution for analyzing numerical data distribution. The code employs a loop to iterate through each element, utilizing conditional statements for classification. It demonstrates a scalable and adaptable design for analyzing different datasets. This approach showcases the versatility of Python in data analysis tasks. It highlights the importance of iteration and conditional logic in processing data efficiently.

7. SUMMARY

In summary, the first problem's solution efficiently extracts specific elements from a tuple using straightforward indexing, showcasing Python's simplicity and elegance in tuple manipulation. Conversely, the second problem's solution focuses on analyzing numerical data distribution by counting even and odd numbers within a list. Leveraging iteration and conditional statements, the solution highlights Python's versatility in handling data analysis tasks with adaptability and scalability. Together, these solutions exemplify Python's prowess in tackling diverse computational challenges with clarity and efficiency.