Practical 1 : Python Basics

Name : $Deepak\ Kumar\ Bharti$

Roll No. : 25056765013

Group : A

Problem Statement

Solve the following problems using Python. Write clear, well-commented code for each part. Whenever required, display the output clearly.

Problem 1 : Basics Mathematical Operations

Consider two numbers :

$a = 15$ and $b = 4$

Compute and print the following :

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Integer division
6. Remainder
7. Power of a raised to b

In [1]:

```
# Given values
a = 15
b = 4
```

In [2]:

```
# 1. Addition
print("Addition :", a + b)
```

```
Addition : 19
```

In [3]:

```
# 2. Subtraction
print("Subtraction :", a - b)
```

```
Subtraction : 11
```

In [4]:

```
# 3. Multiplication
print("Multiplication :", a * b)
```

```
Multiplication : 60
```

In [5]:

```
# 4. Division
print("Division :", a / b)
```

```
Division : 3.75
```

In [6]:

```
# 5. Integer Division
print("Integer Division :", a // b)
```

```
Integer Division : 3
```

In [7]:

```
# 6. Remainder
print("Remainder :", a % b)
```

```
Remainder : 3
```

In [8]:

```
# 7. Power of a raised to b
print("Power of a raised to b :", a ** b)
```

```
Power of a raised to b : 50625
```

Problem 2 : Creating and Inspecting Lists

(a) Create a list x containing the first five natural numbers.

(b) Create another list y containing the values $[10, 20, 30, 40, 50]$.

(c) Display both lists.

(d) Find:Length of list x and First and last elements of x

In [9]:

```
# (a) Creating list x containing first five natural numbers
x = [1, 2, 3, 4, 5]
```

In [10]:

```
# (b) Creating list y
y = [10, 20, 30, 40, 50]
```

In [11]:

```
# (c) Displaying both lists
print("List x :", x)
print("List y :", y)
```

```
List x : [1, 2, 3, 4, 5]
List y : [10, 20, 30, 40, 50]
```

In [12]:

```
# (d) Finding required values
print("Length of list x :", len(x))
print("First element of x :", x[0])
print("Last element of x :", x[-1])
```

```
Length of list x : 5
First element of x : 1
Last element of x : 5
```

Problem 3 : Modification and Deletion of List Elements

Using the list x created earlier:

(a) Replace the third element of x with the value 100.

(b) Create a new list z by concatenating lists x and y.

(c) Delete the second element of list z.

(d) Remove the value 40 from list z.

(e) Delete the entire list y.

In [13]:

```
# (a) Replacing the third element of x with 100
x[2] = 100
print("After replacing third element of x :", x)
```

```
After replacing third element of x : [1, 2, 100, 4, 5]
```

In [14]:

```
# (b) Creating a new list z by concatenating x and y
z = x + y
print("Concatenated list z = x + y :", z)
```

```
Concatenated list z = x + y : [1, 2, 100, 4, 5, 10, 20, 30, 40, 50]
```

In [15]:

```
# (c) Deleting the second element of list z
del z[1]
print("After deleting second element of z :", z)
```

```
After deleting second element of z : [1, 100, 4, 5, 10, 20, 30, 40, 50]
```

In [16]:

```
# (d) Removing the value 40 from list z
z.remove(40)
print("After removing 40 from z :", z)
```

```
After removing 40 from z : [1, 100, 4, 5, 10, 20, 30, 50]
```

In [17]:

```
# (e) Deleting the entire list y
del y
print("List y deleted successfully")
```

```
List y deleted successfully
```

Problem 4 : Element-wise Operations on Lists (Without NumPy)

Consider the lists:

$x = [1, 2, 3, 4, 5]$

$y = [10, 20, 30, 40, 50]$

(a) Perform element-wise addition of x and y using a for loop.

(b) Perform the same operation using zip().

(c) Store the result in a new list and display it.

$Note$: Explain briefly why x + y does not give element-wise addition for Python lists.

In [18]:

```
# Given lists
x = [1, 2, 3, 4, 5]
y = [10, 20, 30, 40, 50]
```

In [19]:

```
# (a) Element-wise addition using for loop
result_loop = []
for i in range(len(x)):
    result_loop.append(x[i] + y[i])
```

```python
print("Element-wise addition using for loop:", result_loop)
```

```
Element-wise addition using for loop: [11, 22, 33, 44, 55]
```

In [20]:

```python
# (b) Element-wise addition using zip()
result_zip = []
for a, b in zip(x, y):
    result_zip.append(a + b)

print("Element-wise addition using zip():", result_zip)
```

```
Element-wise addition using zip(): [11, 22, 33, 44, 55]
```

In [21]:

```python
# (c) Store result in a new list and display it
result = result_zip
print("Final result list:", result)
```

```
Final result list: [11, 22, 33, 44, 55]
```

In [22]:

```python
# Explanation
print("x + y does not give element-wise addition for Python lists because x and y a
print("and in Python '+' for lists concatenates lists instead of performing element
```

```
x + y does not give element-wise addition for Python lists because x and y are lists
and in Python '+' for lists concatenates lists instead of performing element-wise ad
dition.
```

Problem 5 : Sum of Elements in a List

(a) Create a list data = $[5, 10, 15, 20, 25]$.

(b) Compute the sum of elements using: Built-in sum() function.

In [23]:

```python
# (a) Creating the list
data = [5, 10, 15, 20, 25]
print("Data list:", data)
```

```
Data list: [5, 10, 15, 20, 25]
```

In [24]:

```python
# (b) Computing the sum using built-in sum() function
total = sum(data)
print("Sum of elements:", total)
```

```
Sum of elements: 75
```

Problem 6 : Handling Missing Values in Lists

Consider the list:

data = $[10, None, 20, 30, None, 40]$

(a) Count the number of missing values.

(b) Remove missing values from the list.

(c) Compute the sum of the cleaned data.

In [25]:

```
# Given list with missing values
data = [10, None, 20, 30, None, 40]
print("Original data:", data)
```

```
Original data: [10, None, 20, 30, None, 40]
```

In [26]:

```
# (a) Counting missing values
missing_count = data.count(None)
print("Number of missing values:", missing_count)
```

```
Number of missing values: 2
```

In [27]:

```
# (b) Removing missing values
cleaned_data = [value for value in data if value is not None]
print("Cleaned data:", cleaned_data)
```

```
Cleaned data: [10, 20, 30, 40]
```

In [28]:

```
# (c) Computing sum of cleaned data
cleaned_sum = sum(cleaned_data)
print("Sum of cleaned data:", cleaned_sum)
```

```
Sum of cleaned data: 100
```

Problem 7 : List vs Tuple (Mutability and Usage)

Python provides different data structures for storing sequences of values. Two commonly used structures are lists and tuples.

(a) Create a list $L = [10, 20, 30, 40]$ and a tuple $T = (10, 20, 30, 40)$. Display both.

(b) Attempt to change the second element of the list L to 200 and display the updated list.

(c) Attempt to change the second element of the tuple T to 200. Observe and note the error message.

(d) Use a simple example to show one situation where a list is preferred and one situation where a tuple is preferred.

(e) Briefly explain the key defference between a list and a tuple in terms of **mutability.**

In [29]:

```
# (a) Creating a list and a tuple
L = [10, 20, 30, 40]
T = (10, 20, 30, 40)
# Displaying the both
print("List L:", L)
print("Tuple T:", T)
```

```
List L: [10, 20, 30, 40]
Tuple T: (10, 20, 30, 40)
```

In [30]:

```
# (b) Attempting Change the second element of the list
L[1] = 200
print("Updated list L:", L)
```

```
Updated list L: [10, 200, 30, 40]
```

In [31]:

```
# (c) Attempting to change the second element of the tuple
try:
    T[1] = 200
```

```
except TypeError as e:
    print("Error while modifying tuple:", e)
```

```
Error while modifying tuple: 'tuple' object does not support item assignment
```

In [32]:

```
# (d) Examples of usage
# List example (modifiable data)
scores = [50, 60, 70]
scores.append(80)
print("List example (modifiable):", scores)
# Tuple example (fixed data)
coordinates = (10.5, 20.3)
print("Tuple example (fixed):", coordinates)
```

```
List example (modifiable): [50, 60, 70, 80]
Tuple example (fixed): (10.5, 20.3)
```

In [33]:

```
# (e) Explanation of mutability
print("\nExplanation:")
print("Lists are mutable, meaning their elements can be changed after creation (you
print("Tuples are immutable, meaning once created, their elements cannot be modifie
```

```
Explanation:
Lists are mutable, meaning their elements can be changed after creation (you can ad
d, remove, or modify items).
Tuples are immutable, meaning once created, their elements cannot be modified.
```

Problem 8 : Element-wise Operations Using NumPy

(a) Convert the lists x and y into NumPy arrays.

(b) Perform element-wise:

- Addition
- Subtraction
- Multiplication
- Division

(c) Comment on difference between Python lists and Numpy arrays for numerical operations

In [34]:

```
# Import NumPy
import numpy as np

# (a) Converting the earlier given lists x and y lists to NumPy arrays
x_arr = np.array(x)
y_arr = np.array(y)

print("NumPy array x:", x_arr)
print("NumPy array y:", y_arr)
```

```
NumPy array x: [1 2 3 4 5]
NumPy array y: [10 20 30 40 50]
```

In [35]:

```
# (b) Element-wise operations
print("Addition:", x_arr + y_arr)
print("Subtraction:", x_arr - y_arr)
print("Multiplication:", x_arr * y_arr)
print("Division:", x_arr / y_arr)
```

```
Addition: [11 22 33 44 55]
Subtraction: [ -9 -18 -27 -36 -45]
Multiplication: [ 10  40  90 160 250]
Division: [0.1 0.1 0.1 0.1 0.1]
```

In [36]:

```
# (c) Comment on the difference
print("\nComment:")
print("NumPy arrays support fast, element-wise numerical operations,")
print("while Python lists do not perform element-wise operations directly.")
```

```
Comment:
NumPy arrays support fast, element-wise numerical operations,
while Python lists do not perform element-wise operations directly.
```