PYTHON

BY OVIYA

(RA2411030010004)

Element-wise Operations :

Write a program to perform various **element-wise operations** on two input arrays using **NumPy**. The operations should include arithmetic and the program should be able to handle arrays of equal lengths.
PROGRAM:

```
import numpy as np
# Function to perform element-wise operations
def elementwise_operations(A, B):
 # 1. Add corresponding elements of A and B
 A_add_B = A + B
 # 2. Multiply each element of array B by 2
  B_multiply_2 = B * 2
  # 3. Add arrays A and B element-wise
   A_plus_B = A + B
 # 4. Subtract array B from array A element-wise
  A_minus_B = A - B


  # 5. Square each element of array A
  A_square = A ** 2
  # Print results
  print("Array A + Array B:", A_add_B)
  print("Array B after multiplying by 2:", B_multiply_2)
  print("Element-wise addition of A and B:",
A_plus_B)
  print("Element-wise subtraction of B from A:",
A_minus_B)
  print("Element-wise square of A:", A_square)
  # Main program to input arrays A and B
   def main():
  # Input arrays from the user
   A = np.array(list(map(int, input("Enter elements
for array A separated by space: ").split())))
   B = np.array(list(map(int, input("Enter elements
for array B separated by space: ").split())))
```

```
    # Perform element-wise operations
    elementwise_operations(A, B)


# Run the program
if __name__ == "__main__":
    main()
```

OUTPUT :

```
Enter elements for array A separated by space:
6344832
Enter elements for array B separated by space:
5356442432
Array A + Array B: [5362787264]
Array B after multiplying by 2: [10712884864]
Element-wise addition of A and B: [5362787264]
Element-wise subtraction of B from A: [-
5350097600]
Element-wise square of A: [40256893108224]
```

CORRECT TEST CASE :

Test Case 1: Valid Input

Array A: [1, 2, 3, 4, 5]

Array B: [6, 7, 8, 9, 10]

Output:

Array A + Array B: [7 9 11 13 15]

Array B after multiplying by 2: [12 14 16 18 20]

Element-wise addition of A and B: [7 9 11 13 15]

Element-wise subtraction of B from A: [-5 -5 -5 -5 -5]

Element-wise square of A: [1 4 9 16 25]

Explanation:

Element-wise addition and subtraction between A and B should give corresponding results, and multiplying B by 2 gives the doubled values.

Test Case 2: Another Valid Input

Array A: [3, 4, 5]

Array B: [2, 1, 3]

Output:

Array A + Array B: [5 5 8]

Array B after multiplying by 2: [4 2 6]

Element-wise addition of A and B: [5 5 8]

Element-wise subtraction of B from A: [1 3 2]

Element-wise square of A: [9 16 25]

REFUTE TEST CASE:

These test cases are invalid due to incorrect input, such as mismatched array sizes or non-numeric inputs.

Test Case 1: Mismatched Array Lengths

- **Array A**: [1, 2, 3]
- **Array B**: [4, 5]

OUTPUT :

ValueError: operands could not be broadcast together with shapes (3,) (2,)

Explanation:

- **Refute**: NumPy requires that the arrays have the same length to perform element-wise operations. The length mismatch will raise an error.

Test Case 2: Non-Numeric Input

- **Array A**: ["a", "b", "c"]
- **Array B**: [1, 2, 3]

Output (Error):

ValueError: invalid literal for int() with base 10: 'a'

Explanation:

- **Refute**: If the input contains non-numeric values, the program will throw a ValueError while trying to convert the input to integers.

Test Case 3: Empty Arrays

- **Array A**: [] (empty array)
- **Array B**: [] (empty array)

Expected Output:

less

Copy code

```
Array A + Array B: []

Array B after multiplying by 2: []

Element-wise addition of A and B: []

Element-wise subtraction of B from A: []

Element-wise square of A: []
```

Explanation:

- **Refute**: While NumPy allows empty arrays, the operations on empty arrays will result in empty results. This is an edge case where no output is meaningful, but it's still handled without errors.

DISCOUNT CALCULATOR :

PROBLEM STATEMENT :

Discount Calculator that calculates the final price of an item after applying a discount. The calculator should allow users to input the original price of the item and the discount percentage, then calculate and display the discounted price.

PROGRAM :

```
# Function to calculate discount and final price
def calculate_discount():
 # Get user input for original price and discount percentage
original_price = float(input("Enter the original price of the item: "))
discount_percentage = float(input("Enter the discount percentage: "))
# Calculate discount amount
```

```
discount_amount = (original_price *
discount_percentage) / 100
# Calculate the final price after applying
discount
 final_price = original_price - discount_amount
  # Display the results
  print("Original Price: ", original_price)
   print("Discount Amount: ", discount_amount)
   print("Final Price: ", final_price)
# Run the discount
calculation calculate_discount()
```

OUTPUT :

```
Enter the original price of the item: 829342
Enter the discount percentage: 22
Original Price:  829342.0
Discount Amount:  182455.24
Final Price:  646886.76
```

CORRECT TEST CASE :

Valid Inputs:

- Original Price: 100
- Discount Percentage: 20

Output:

```
Enter the original price of the item: 100
Enter the discount percentage: 20
Original Price:  100.0
Discount Amount:  20.0
Final Price:  80.0
```

Explanation:

- The original price is 100 and the discount is 20%.
- The discount amount is calculated as (100 * 20) / 100 = 20.
- The final price is 100 - 20 = 80

REFUTE TEST CASE :

Test Case 1: Invalid Input (Negative Price)

- Original Price: -50 (invalid, price cannot be negative)
- Discount Percentage: 20

Expected Output :

```
Enter the original price of the item: -50
Enter the discount percentage: 20
Original Price:  -50.0
Discount Amount:  -10.0
Final Price:  -60.0
```

Explanation:

- While the program doesn't handle errors explicitly in the current version, this is an invalid scenario. A negative price should not be accepted, as prices can't be negative in real-world scenarios. To handle this, you could add a check to ensure that the price is positive.