Using Python List (in-built array)

```
# ✅ Pass List (Array) to Function and Modify It
def multiply_by_two(arr):
    for i in range(len(arr)):
        arr[i] *= 2
    print("Inside function (doubled):", arr)

# Main program
numbers = [10, 20, 30, 40]
print("Original array:", numbers)

# Pass to function
multiply_by_two(numbers)

print("After function call:", numbers)
#✅ Note: This modifies the original list (passed by reference).
```

```
Original array: [10, 20, 30, 40]
Inside function (doubled): [20, 40, 60, 80]
After function call: [20, 40, 60, 80]
```

Return New Array from Function

```
def square_elements(arr):
    new_arr = []
    for num in arr:
        new_arr.append(num ** 2)
    return new_arr

# Main
original = [2, 3, 4]
squared = square_elements(original)

print("Original array:", original)
print("New squared array:", squared)
#✅ Note: The original list is unchanged; the function returns a new
one.
```

```
Original array: [2, 3, 4]
New squared array: [4, 9, 16]
```

Using NumPy Array in Function

```
import numpy as np

def add_five(arr):
    arr += 5
    print("Inside function (after +5):", arr)

# Main
marks = np.array([70, 80, 90])
print("Original marks:", marks)

add_five(marks)
print("After function call:", marks)
# Note: NumPy arrays support vectorized operations, so you can add
directly: arr + 5.
```

```
Original marks: [70 80 90]
Inside function (after +5): [75 85 95]
After function call: [75 85 95]
```