EXPERIMENT NO: 03

```
class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight
        self.ratio = value / weight  # Value-to-weight ratio

def fractional_knapsack(capacity, items):
    # Sort items by value-to-weight ratio in descending order
    items.sort(key=lambda x: x.ratio, reverse=True)

    total_value = 0.0  # Initialize total value

    for item in items:
        if capacity <= 0:  # No more capacity in the knapsack
            break

        if item.weight <= capacity:  # Item can be added fully
            total_value += item.value
            capacity -= item.weight
        else:  # Take the fractional part of the item
            total_value += item.ratio * capacity  # Add the value of the fraction
            capacity = 0  # The knapsack is now full

    return total_value

# Example usage
if __name__ == "__main__":
    # Sample items (value, weight)
```

```
    items = [
        Item(60, 10),  # value=60, weight=10
        Item(100, 20), # value=100, weight=20
        Item(120, 30)  # value=120, weight=30
    ]

    knapsack_capacity = 50  # Maximum weight capacity of the knapsack
    max_value = fractional_knapsack(knapsack_capacity, items)

    print(f"Maximum value in the knapsack: {max_value:.2f}")
```

OUTPUT:-

Maximum value in the knapsack: 240.00