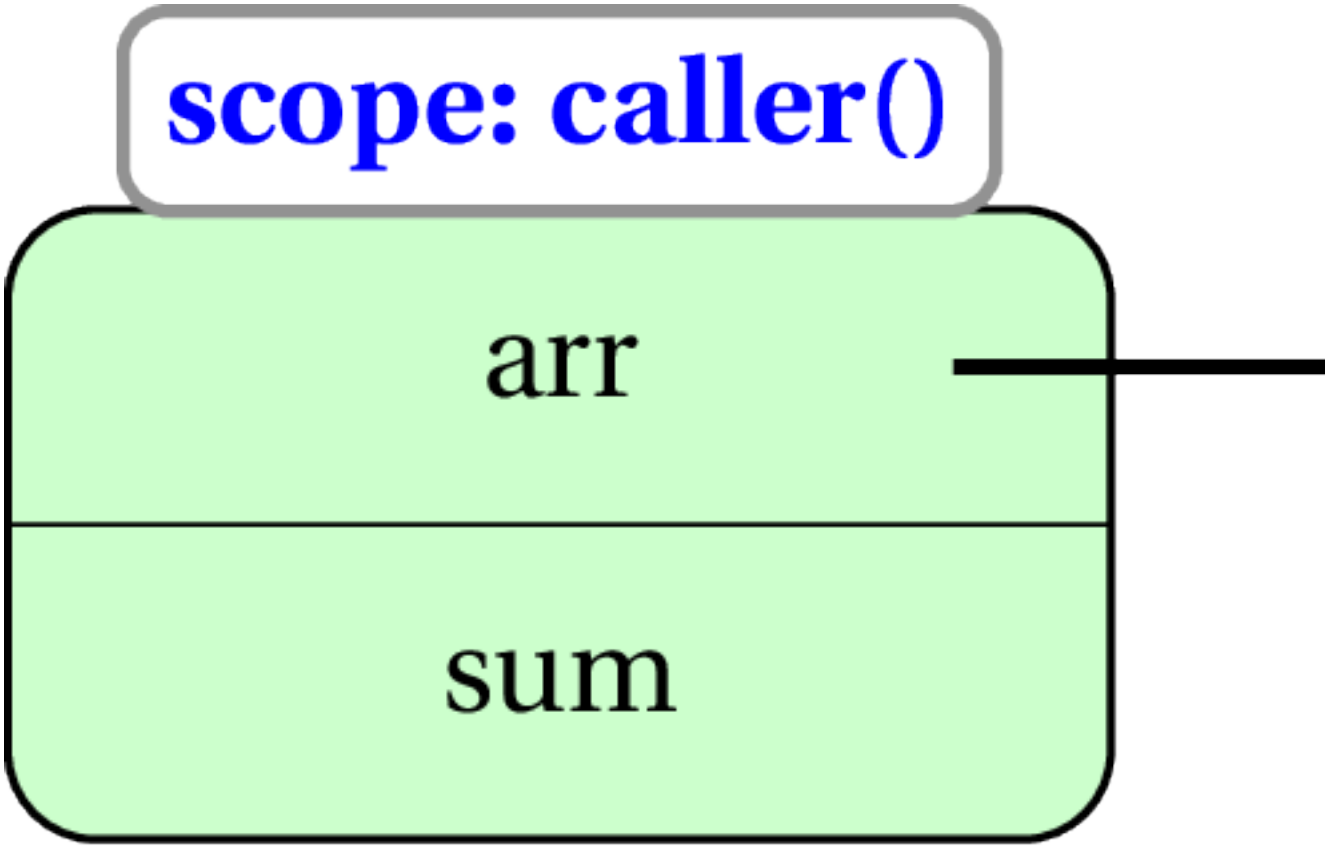
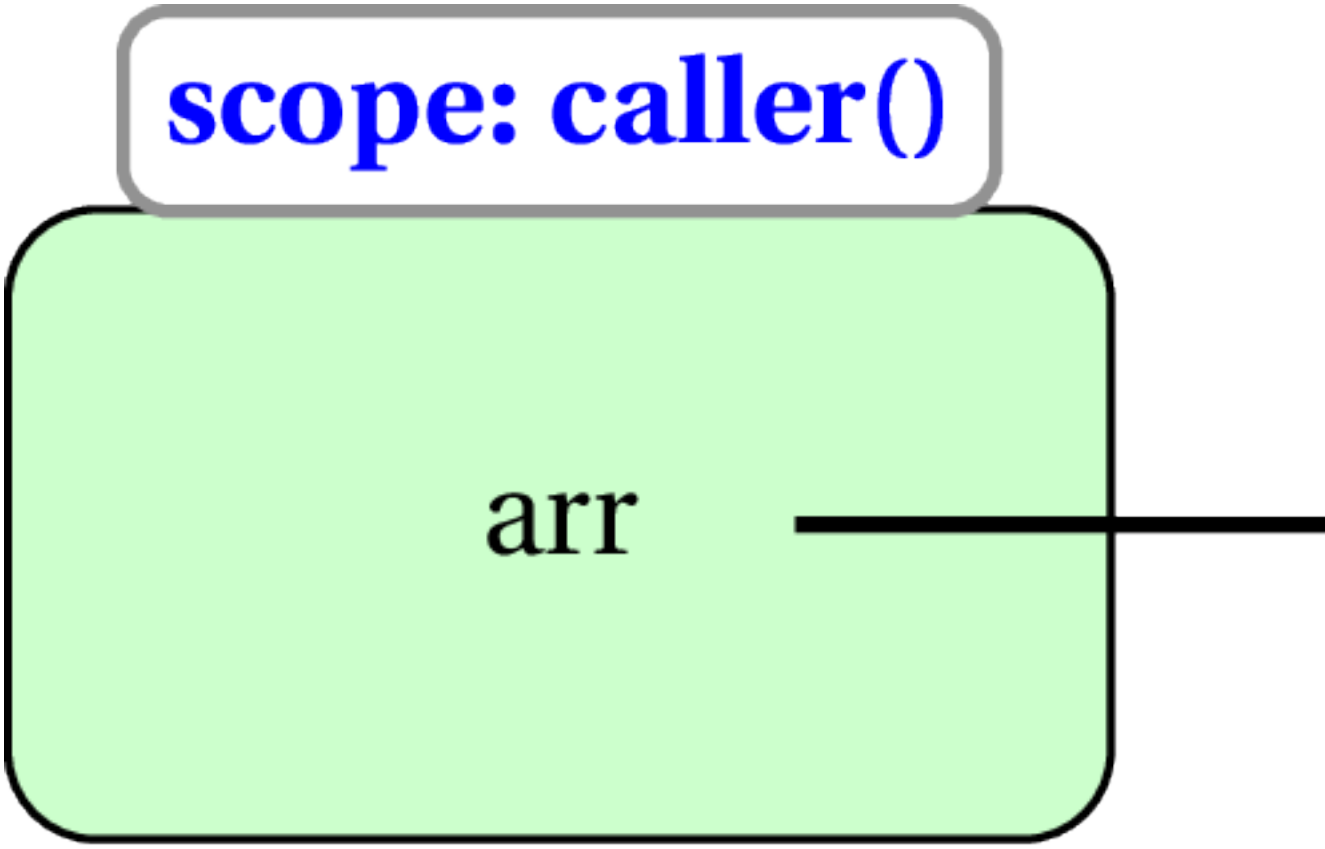
When you pass an array to a function, a shallow copy of the actual parameter is made into the formal parameter. For example, in the following code, **data** becomes a shallow copy of **arr** .

```
void caller() {
        int[] arr = {10, 70, 50};
        int sum = total(arr);
}

int total(int[] data) {
        int result = 0;
        for(int i=0; i < data.length; i++) {
                result+=data[i];
        }
        return result;
}
```


scope: caller()
arr
sum

If you have a function that modifies the passed array, the contents of the actual parameter will also change.

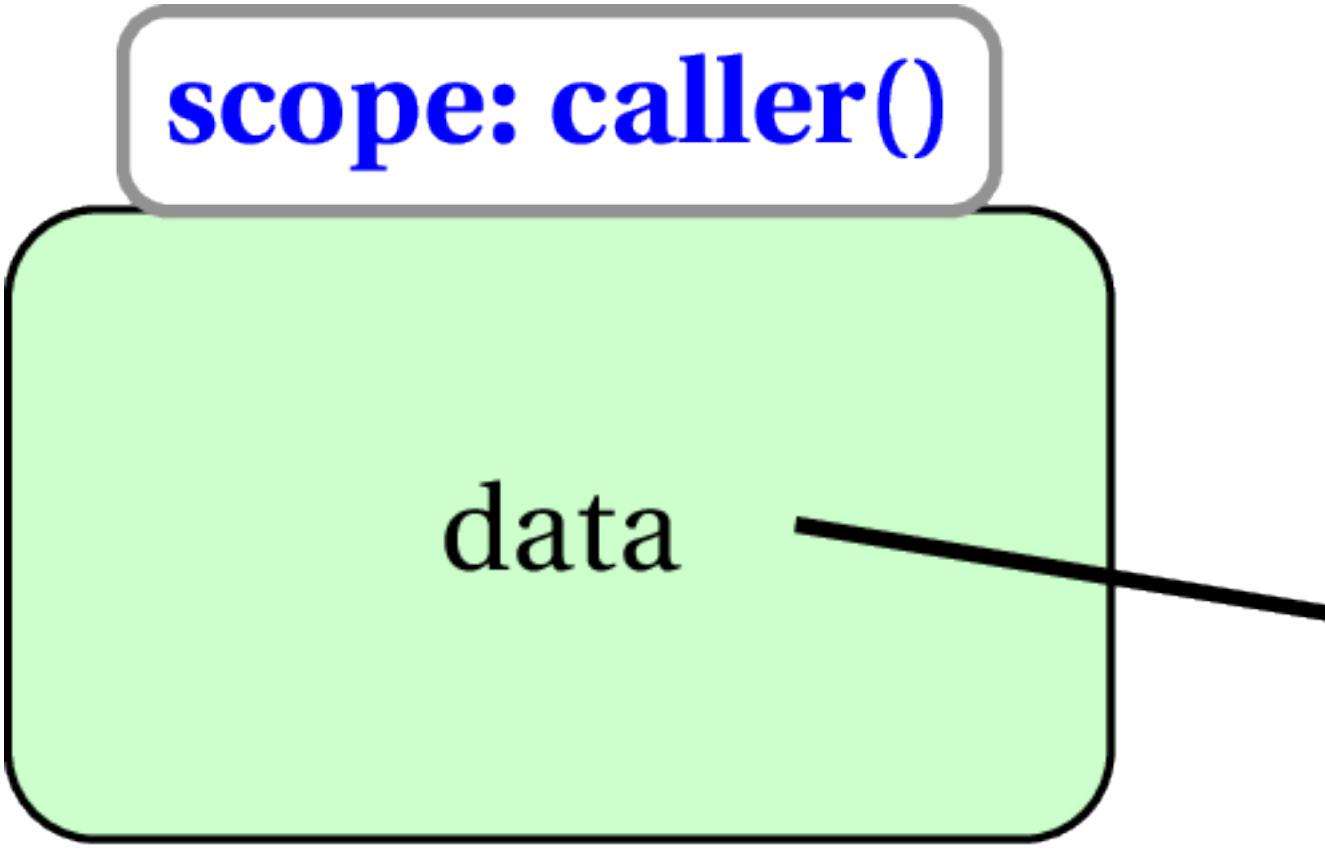
```
void caller() {
        int[] arr = {10, -70, 0};
        negate(arr);
}

void negate(int[] data) {
        for(int i=0; i < data.length; i++) {
                data[i]=data[i]*-1;
        }
}
```


scope: caller()
arr

If a function returns an array (source) and the caller copies it into an array (destination), it's a shallow copy as demonstrated in the following example.

If you have a function that modifies the passed array, the contents of the actual parameter will also change.

```
void caller() {
        int[] data = getDiceOutcomes(5);
}

int[] getDiceOutcomes(int n) {
        int[] outcomes = new int[n];
        for(int i=0; i < outcomes.length; i++) {
                outcomes[i] = (int)random(1, 7);
        }
        return outcomes;
}
```


scope: caller()
data