Function Parameters

This lesson covers function parameters and explains why parameters are always copied.

WE'LL COVER THE FOLLOWING

- Function parameters
 - Parameters are always copied

Function parameters

Some of the concepts of this chapter have already appeared earlier in the course. For example, the ref keyword that we saw in the foreach loop lesson was making actual elements available in foreach loops as opposed to copies of those elements.

Additionally, we covered the **const** and **immutable** keywords and the differences between value types and reference types in previous chapters. We have written functions that produced results by making use of their parameters. For example, the following function uses its parameters in a calculation:

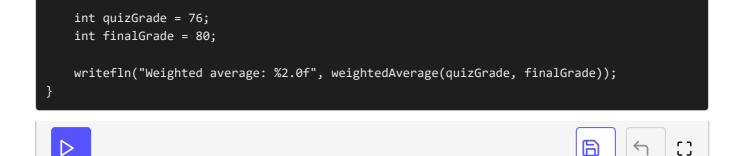
```
double weightedAverage(double quizGrade, double finalGrade) {
    return quizGrade * 0.4 + finalGrade * 0.6;
}
```

That function calculates the average grade by taking 40% of the quiz grade and 60% of the final grade. Here is how it may be used:

```
import std.stdio;

double weightedAverage(double quizGrade, double finalGrade) {
    return quizGrade * 0.4 + finalGrade * 0.6;
}

void main() {
```





Parameters are always copied

In the code above, the two variables are passed as arguments to weightedAverage). The function uses its parameters. This fact may give a false impression that the function uses the actual variables that have been passed as arguments. In reality, what the function uses are copies of those variables.

This distinction is important because modifying a parameter changes only the copy. This can be seen in the following function, which is trying to modify its parameter (i.e. making a side effect). Let's assume that the following function is written for reducing the energy of a game character:

```
void reduceEnergy(double energy) {
  energy /= 4;
}
```

Here is a program that tests reduceEnergy():

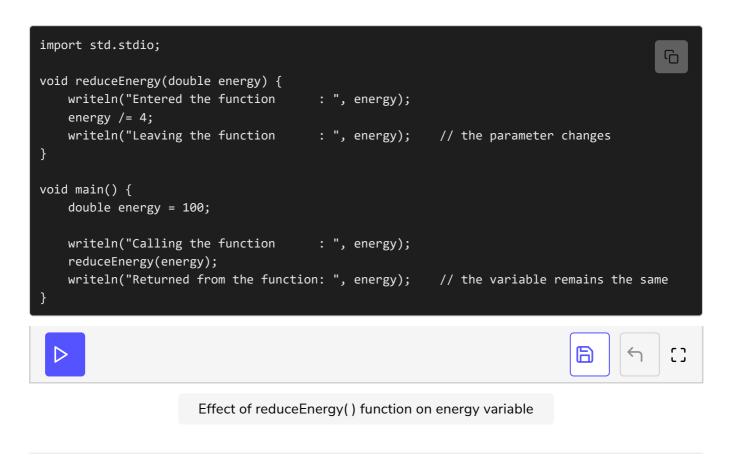
```
import std.stdio;

void reduceEnergy(double energy) {
    energy /= 4;
}

void main() {
    double energy = 100;
    reduceEnergy(energy);
    writeln("New energy: ", energy);
}
```

Although reduceEnergy() drops the value of its parameter to a quarter of its original value, the variable energy in main() does not change. The reason for this is that the energy variable in main() and the energy parameter of reduceEnergy() are separate; the parameter is a copy of the variable in main().

To observe this more closely, let's insert some writeln() expressions:



In the next lesson, we will see that referenced variables are not copied.