



User-Defined Functions

FALL 2022 | UNIVERSITY OF MOUNT UNION

Functions in Processing

There are many built-in functions, and we've used quite a few already!

`setup()` and `draw()` run automatically

Other functions we've used include (but are not limited to) `size()`, `line()`, `ellipse()`, `rect()`, and `random()`

Why bother with functions?

Modularity: functions are independent software units that are used to build more complex programs.



Reusability: functions allow you to reuse code without having to retype it.



Defining a function

A function definition includes these parts:

- Return type
- Function name
- Parameters
- Code body

```
returnType functionName(parameters){  
    //code body of function  
}
```

Calling a function

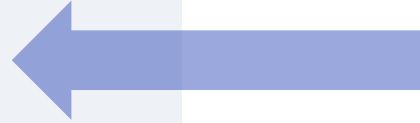
Any user-defined function must be called for it to run!

We call the function using its name and any necessary parameters inside parentheses, followed by a semicolon:

```
functionName(parameter1, parameter2, etc.);
```

Example 9-1

```
void setup() {  
  println("Ready to roll!");  
  rollDice(20);  
  rollDice(20);  
  rollDice(6);  
  println("Finished.");  
}  
  
void rollDice(int numSides) {  
  int d = 1 + int(random(numSides));  
  println("Rolling... " + d);  
}
```



Inside **setup()**, the user-defined function **rollDice()** is called three times. The first two times, the parameter 20 is passed into the function, and the last time, the parameter 6 is passed into the function.

Example 9-1


```
void setup() {  
  println("Ready to roll!");  
  rollDice(20);  
  rollDice(20);  
  rollDice(6);  
  println("Finished.");  
}  
  
void rollDice(int numSides) {  
  int d = 1 + int(random(numSides));  
  println("Rolling... " + d);  
}
```



Here is where the user-defined function **rollDice()** is defined. The word **void** means that no value is returned to the program (more on this later). The word **rollDice** gives us the name of the function. The parameter **int numSides** indicates that the function requires one parameter, an integer.

Example 9-1

```
void setup() {  
    println("Ready to roll!");  
    rollDice(20);  
    rollDice(20);  
    rollDice(6);  
    println("Finished.");  
}  
  
void rollDice(int numSides) {  
    int d = 1 + int(random(numSides));  
    println("Rolling... " + d);  
}
```



Here is the code body of **rollDice()**:

An integer variable **d** is assigned the value of **1+int(random(numSides))**.

With just one parameter, the **random()** function returns a random floating-point value between 0 and up to (but not including) that parameter value.

The function **int()** will convert that value to an integer using rounding.

Adding 1 to that integer will assign a value to **d** that is greater than 0 and less than or equal to the number of sides on the dice.

Then the function prints “Rolling...” and the value of **d** to the console using the **println()** function.

Return values

Remember when I said that'd we'd revisit “**void**”?

And remember that the Processing-defined function `random()` *returns* a floating-point number?

Functions can perform a calculation and then return a value to the main program. (Most of the functions we've used have not returned values, which is why their return type is “void”).

Return values are frequently assigned to a variable or used as parameters in other functions.

```
float r = random(1, 10);
```

```
line(300, 300, random(0, 600), random(0, 600));
```



Returning a value with a user-defined function: Example 9-8

```
void setup() {  
  float yourWeight = 132;  
  float marsWeight = calculateMars(yourWeight);  
  println(marsWeight);  
}  
  
float calculateMars(float w) {  
  float newWeight = w * 0.38;  
  return newWeight;  
}
```



Inside `setup()`, the user-defined function `calculateMars` is called, and its return value is assigned to the floating-point variable `marsWeight`.

Returning a value with a user-defined function: Example 9-8

```
void setup() {  
  float yourWeight = 132;  
  float marsWeight = calculateMars(yourWeight);  
  println(marsWeight);  
}  
  
float calculateMars(float w) {  
  float newWeight = w * 0.38;  
  return newWeight;  
}
```



Here is the definition of `calculateMars()`. Hey, it's a function that *doesn't start with the word **void**!* This function returns a floating-point value, `newWeight`, to the program.