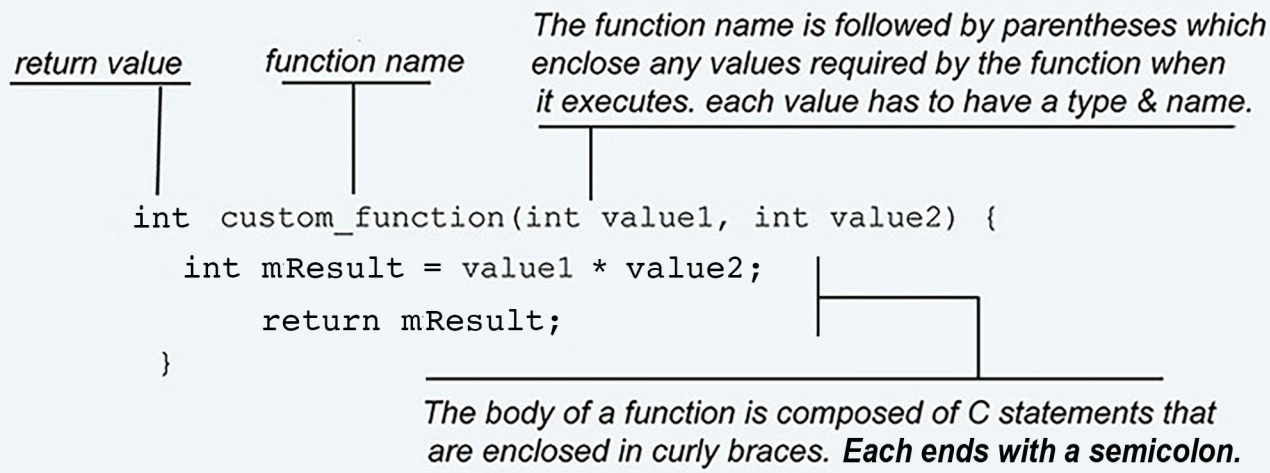
C functions have a structure and syntax as follows:



Note: In this example, the definition of "custom_function()" states that it requires two integers to execute and that it will output an integer. If it was actually used in a program it would look something like this: `storedValue = custom_function(7, 34);`

Arduino and Processing environments have two built-in functions for handling initialization and then a main performance loop. With the Arduino these are: "`setup()`" and "`loop()`". For Processing they are "`setup()`" and "`draw()`". These functions don't return any values (numbers, strings, etc.) so their return values are "void":

With the Arduino, the setup is usually used to set pin modes (input or output) and begin communication with the Serial port or other special sensors you are using.
Any variables that need to be accessed by the various functions your program uses need to be defined as "global variables" that are not part of a function. Their variable definitions are placed at the begining of your code, before the setup function.

example:

```
// (Global variables first)
int a_in;                // create variable for storing analog readings
int timesTen;            // variable for result of math operation
int sensorPin = A5;      // variable for analog pin as A5

void setup()  {          // only do once
    Serial.begin(9600);
    pinMode(sensorPin, INPUT);
}
void loop() {                              // repeat forever
  a_in = analogRead(sensorPin);            // get 10 bit analog value
  a_in = map(a_in, 0, 1023, 0, 255);       // scale it to 8 bits

  timesTen = custom_function(a_in, 10);
  Serial.println(timesTen);
  delay(50);                              // make text less flickery
}


// (Here's the defining of the custom function:)

int custom_function(int r, int d){
        return (r * d);
}
```