Variable Scope

Further increase your knowledge about function behavior in C by learning about variable scope inside and outside of a function.

Any variables declared inside a function, are **local to that function**, and are not accessible outside of the function. Similarly, code within a function doesn't have access to variables that have been declared outside of that function (for example in another function, or in `main()`). If you want this functionality, then you can specify that a variable be **global**. Any variable declared outside of **any** function (it also has to be outside of `main()`) is global, and can be seen by every function. In C, global variables are known as **external** variables (they are external to any function).

For example in the following code, the variable `myGlob` is declared outside of `main()` and outside of `myFunc()` and thus can be access by code within both. The variable `myInt` is declared within the function `myFunc()` and is thus **local** to `myFunc()` and cannot be accessed outside of `myFunc()` (for example within `main()`). Similarly, the variable `myChar` is declared within `main()` and so cannot be seen within `myFunc()`.

```c
#include <stdio.h>

float myGlob = 3.14;

void myFunc(void) {
    int myInt = 8;
    printf("my favourite number is %d\n", myInt);
    printf("my favourite float is %.2f\n", myGlob);
//  printf("my favourite letter is $c\n", myChar); // THIS WOULD NOT WORK
}

int main() {
    char myChar = 'x';
    printf("my favourite letter is %c\n", myChar);
    myFunc();
    printf("my favourite float is %.2f\n", myGlob);
//  printf("my favourite number is %d", myInt); // THIS WOULD NOT WORK
    return 0;
}
```

Next, we'll see that the variables in our function can either be deleted from
memory, or persist with their last updated value.

memory, or persist with their last updated value.