Functions

Functions make large programs possible - reduce complexity
Break problem into subproblem - divide and conquer
Simplify task
Smaller modules are easier to write, understand, verify, test, debug, maintain
Modules can be tested independently then put together
Different programmers can write different modules
Repeated code can be put in a function call (execute from multipe points)
Can/should write reusable modules

Functions should perform a well-defined task - choose and appropriate name
Data into function -- arguments (a,b)
Function can return zero or one thing -- return type

Two types of functions:
Standard library, math.h, stdio.h

Function has it own local variables
Function has parameters pased in
Function can access global variables

Function main() is a required function

Function declaration ---- function prototype
Function definition

E.g., Function declaration
```
int printf(char* s, ...);
```

Function definition:

```
int square(int y) {
        int result;             // local variable
        result = y*y;
        return result;
}
```

```
int square( int y) {    // Alternate version of previous
        return y*y;
}
```

How to call the function

```
int square(int y);  // Forward declaration -- assuming the function is defined below this

int x, x2;

x2 = square(x);
x2 = square(3);
```

Compiler will cast input arguments to the type in the prototype

```
x2 = square(3.9);
```

Function prototype has effect from the point of declaration to the end of the file
Can use void to mean no arguments or no return value

```
int buttonpushed(void);
int buttonpushed();     // Same

void blinkLED();
```

Function should be limited to 1/2 to 1 page
Program is written as a collection of small functions