



# NYU K12 STEM Internet of Things 2019

Lecture – 2



# TUTOR DETAILS

**Contact:**

Monish N. Kapadia

[mnk337@nyu.edu](mailto:mnk337@nyu.edu)

Milind Singh

[ms10997@nyu.edu](mailto:ms10997@nyu.edu)

Ishita Choudhary

[ivc211@nyu.edu](mailto:ivc211@nyu.edu)

Tara Nicole Umesh

[tnu201@nyu.edu](mailto:tnu201@nyu.edu)

**Program Coordinator:**

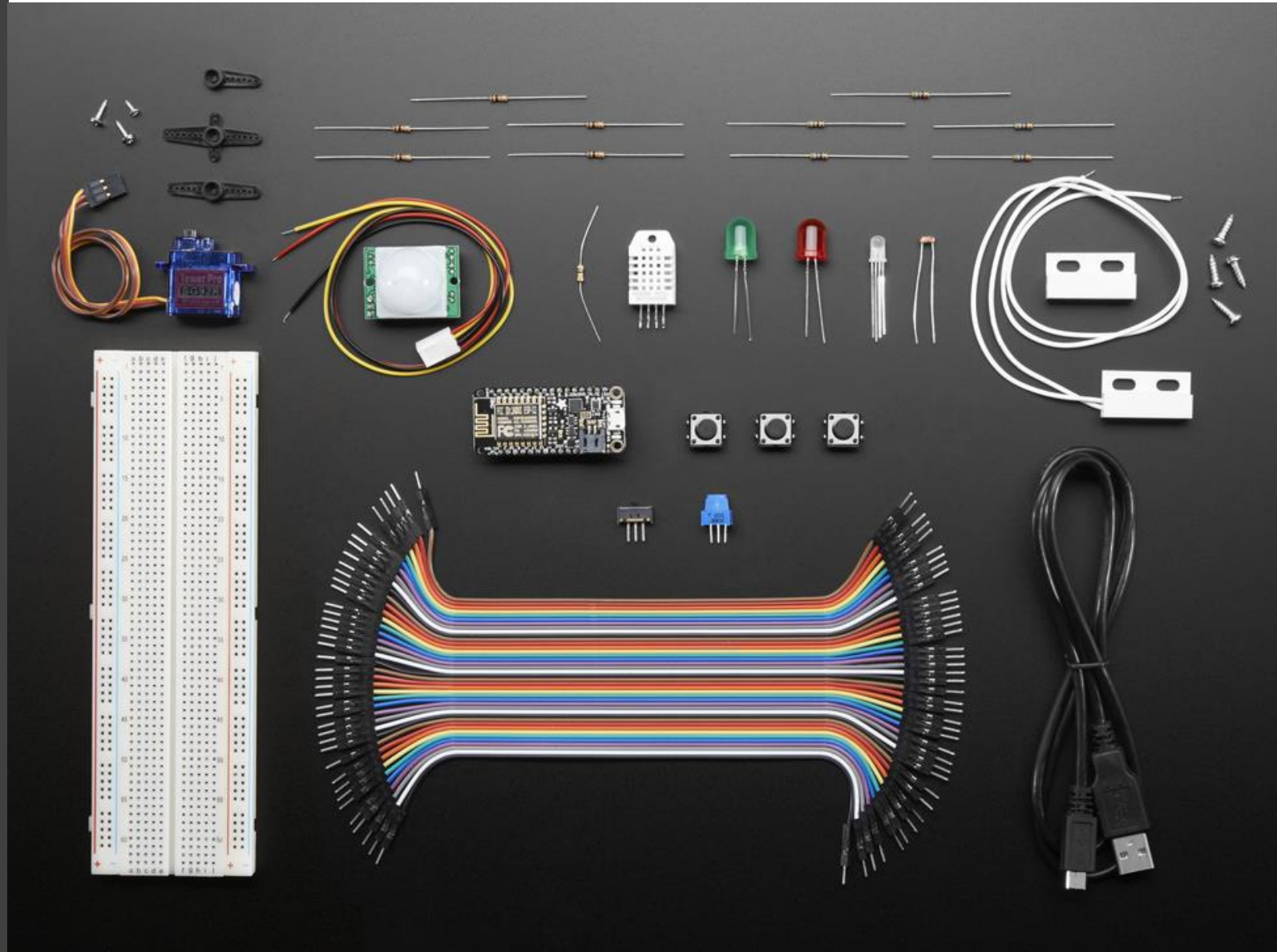
Mathhew S Campisi

[msc398@nyu.edu](mailto:msc398@nyu.edu)

# LAB KITS

## Microsoft Azure IoT Starter Kit w/ Adafruit Feather HUZZAH

PRODUCT ID: 3032



## LAB KITS

- You will be given kits for the duration of the course.
- Please, ensure you do not lose components , they are not replaceable.
- Clean-Up your workspace before you leave for the day.
- No Food or Drinks are permitted during Lab Hours for safety reasons.

This kit includes:

- 1x Assembled Adafruit Feather HUZZAH ESP8266 WiFi
- 1x Micro Servo
- 1x PIR (motion) Sensor
- 1x USB Cable - A/Micro B
- 1x Fast Vibration Switch
- 1x Magnetic Contact Switch (door sensor)
- 1x Half-sized Breadboard
- 1x Premium Male/Male Jumper Wires - 40 x 6"
- 1x DHT22 Temperature-humidity Sensor + Extras

Component bag containing:

- 3x 12mm Tactile Switches
- 1x Breadboard Trim Potentiometer 10K
- 1x Diffused 10mm Green LED
- 1x Diffused 10mm Red LED
- 5x 10K 5% 1/4W Resistor
- 5x 560 Ohm 5% 1/4W Resistor
- 1x Piezo Buzzer
- 1x Photo Cell Light Sensor
- 1x Diffused RGB (tri-color) LED
- 1x Breadboard-friendly SPDT Slide Switch



# INTRODUCTION TO PROGRAMMING

Operations	Operator	Syntax
Addition	+	number1 + number2
Subtraction	-	number1 - number2
Multiplication	*	number1 * number2
Division	/	number1 / number2
Remainder	%	number1 % number2

**Important point:**

- In division, the result of the operation will be the quotient only i.e. it will not give the decimal value.

Eg. `int number1 = 15;`  
`int number2 = 10;`  
`printf("%f", (number1 / number2));`  
Output: 1.000000 instead of 1.5

- To obtain decimal value one of the variable has to be a float.

Eg. `float number1 = 15;`  
`int number2 = 10;`  
`printf("%f", (number1 / number2));`  
Output: 1.50000

# Mathematical Operations

- **Syntax: switch(variable)**

```
{  
    case value:  
        // do something  
        break;  
    case value2:  
        // do something else  
        break;  
    default:  
        // if none of the above case satisfy then do something else  
        break;  
}
```

# Switch case

- **Syntax: switch(variable)**

```
{
    case value:
        // do something
        break;
    case value2:
        // do something else
        break;
    default:
        // if none of the above case satisfy then do something else
        break;
}
```

- **Eg. int number = 1;**

```
switch(number)
{
    case 0:
        printf("Case 0 selected");
        break;
    case 1:
        printf("Case 1 selected");
        break;
    default:
        printf("Wrong selection");
        break;
}
```

# Switch case



```
#include <stdio.h>
#include <stdlib.h>
```

```
int main()
{
    char alphabet;
    printf("Select a character: ");
    scanf("%c", &alphabet);
    switch(alphabet)
    {
        case 'A':
            printf("You selected apples\n");
            break;
        case 'B':
            printf("You selected bananas\n");
            break;
        case 'G':
            printf("You selected grapes\n");
            break;
        case 'O':
            printf("You selected oranges\n");
            break;
        default:
            printf("No fruit selected\n");
            break;
    }
    return 0;
}
```

# Switch case

# Calculator

Design a calculator that performs the following operations:

Enter first number: 11

Enter second number: 25

```
-----  
|           1.| Addition |  
|           2.| Subtraction |  
|           3.| Multiplication |  
|           4.| Division |  
|-----|
```

Enter operation number: 1

| 11 + 25 = 36

# Calculator

---

```
int main(void) {
    setvbuf(stdout, NULL, _IONBF, 0);
    int firstNumber, secondNumber, operation;
    printf("Enter first number: ");
    scanf("%d", &firstNumber);
    printf("Enter second number: ");
    scanf("%d", &secondNumber);
    printf("-----\n");
    printf("| 1. | Addition      |\n");
    printf("| 2. | Subtraction   |\n");
    printf("| 3. | Multiplication |\n");
    printf("| 4. | Division      |\n");
    printf("-----\n");
    printf("Enter operation number: ");
    scanf("%d", &operation);
    // printf("\n");
    switch(operation)
    {
        case 1:
            printf("%d + %d = %d", firstNumber, secondNumber, (firstNumber + secondNumber));
            break;
        case 2:
            printf("%d - %d = %d", firstNumber, secondNumber, (firstNumber - secondNumber));
            break;
        case 3:
            printf("%d * %d = %d", firstNumber, secondNumber, (firstNumber * secondNumber));
            break;
        case 4:
            printf("%d / %d = %d", firstNumber, secondNumber, (firstNumber / secondNumber));
            break;
        default:
            printf("Invalid operation number");
            break;
    }
    printf("\n");
    return 0;
}
```

# Pointers

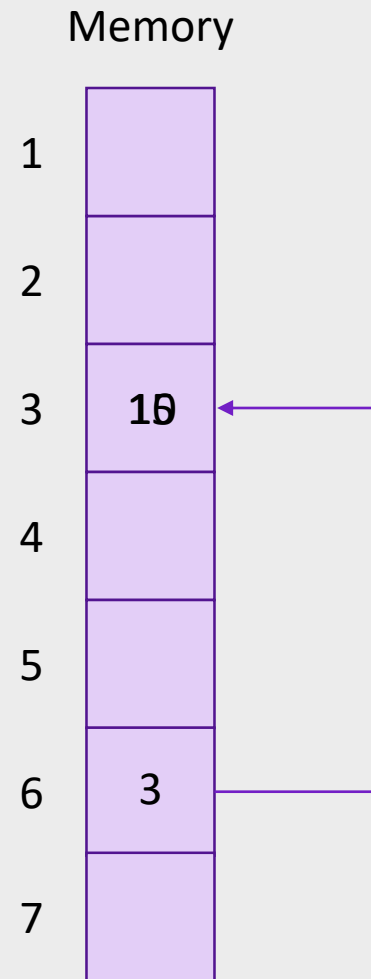
- Points to where a variable is stored in memory



# Pointer Syntax

- To declare a pointer:
  - `int* pointer;`
  - `char* pointer;`
- Pointers must be set to an address
- To get the address of a variable use `&`:
  - `int* pointer = &data;`
- To get the data that the pointer is pointing to use `*`:
  - `int more_data = *pointer;`

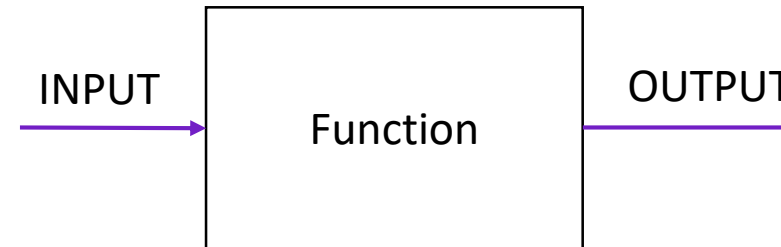
Lets talk about  
memory!



```
int y = 15;  
int* pointer = &y;  
y = 10;  
printf("%d\n", *pointer);
```

# Functions

- Functions are essentially blocks of code which have their own designated identifiers (names).
- Think of a function as a black-box , you don't what's inside the box.
- A function takes an input and gives a certain **desired** output.

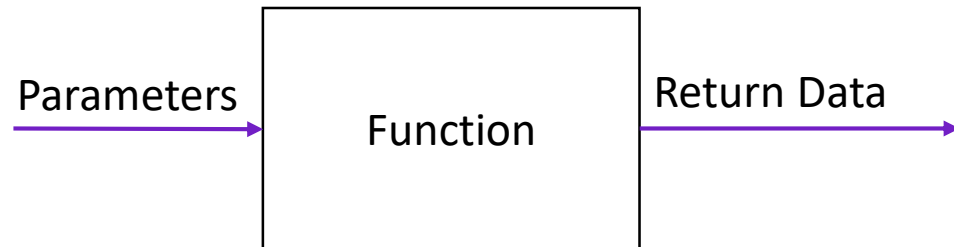


# Functions

So how do we give input and get output?

- Inputs to Functions are called Parameters or Arguments.
- A function is always declared with a Data Type , the function gives output by returning a value back to the caller.

You have already seen a function repeatedly





# The Main Function

1. The main function is the entry point for any program.
2. The main function is of type int by convention .
3. The main function returns a Zero upon successful completion of program.

```
int main(void){  
  
    statement1;  
    statement2;  
    return 0;  
}
```

# The Main Function

Return Data Type      Function Name      Arguments

```
int main(void){  
    statement1;  
    statement2;  
    return 0;  
}
```

← Code Block

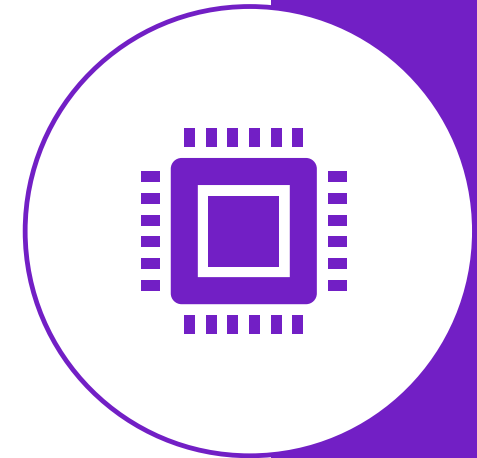
← Returning Explicitly

A diagram illustrating the components of the C main function. The code 'int main(void){ statement1; statement2; return 0; }' is shown. Above the code, three labels with arrows point to their respective parts: 'Return Data Type' points to 'int', 'Function Name' points to 'main', and 'Arguments' points to '(void)'. To the right of the code, two more labels with arrows point to specific lines: 'Code Block' points to the line 'statement1;', and 'Returning Explicitly' points to the line 'return 0;'. A vertical line is positioned to the left of the code block.

# What's inside the Black Box?

We now know that Functions can take inputs and return outputs , we will now see what is inside a function body and how does it work:

```
Eg    void calcSquare(){  
        int a , sqr;  
        printf("Enter Number:");  
        scanf("%d",&a);  
        sqr = a* a ;  
        printf("\nSquare:%d",&sqr);  
    }  
        // Function needs no parameters , why?  
        // Statements inside function are executed  
        // consecutively .  
        // Input is taken here  
        //  
        // No Return needed for Void Data Type
```



# How are functions declared?

**Declaration:** In Programming terms , declaration means to declare that a function or a variable exists somewhere.

A declaration is done by simply writing only the function name and its arguments.

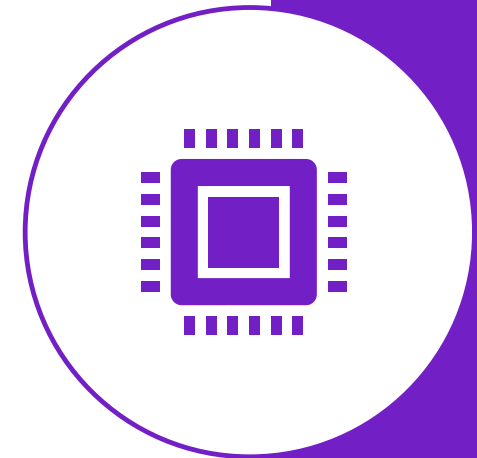
```
int x;  
char name[20];  
int funct(int a , int b);
```

Notice , that we are only giving the names of the variables/functions , we are not assigning them values or defining their implementation.

**Definition:** Definition Refers to the actual implementation of the program.

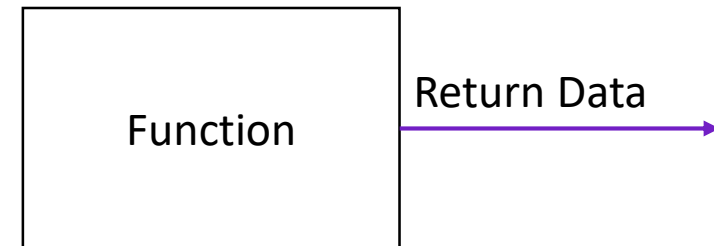
```
int add(int a , int b){  
    return (a+b);  
}
```

Notice here , we are telling what the function will do i.e. return the sum of integers a & b.



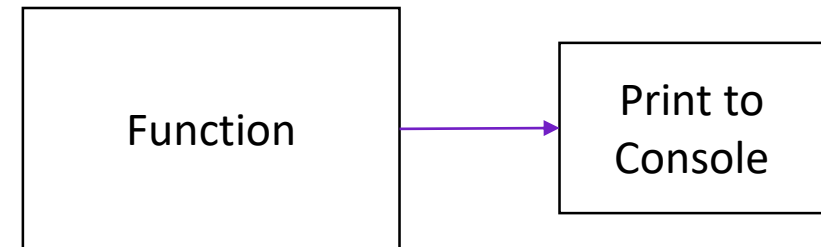
Functions can  
exist without  
inputs

`calcSquare()` a function which asks users to input values into console and uses those to execute further and still returns a valid integer value .



Functions can  
exist without  
return outputs  
too

Void Type Functions do not return any values. But they still can be used to perform certain tasks e.g. printing on console.



# Calling a Function

## Where are functions located?

- Conventionally , above the main program .
- Alternate arrangements are possible using declarations
- You can even move all your functions to a separate file known as a Header file.

## How do functions work in a program?

Functions are implemented inside a code using Function Calls.

## What is a Function Call?

```
int main(void){  
    int sum;  
    int x = 10 ;  
    int y = 20 ;  
    calcSquare();  
    sum = add(x,y);  
    return 0;  
}
```

Function Call to Void type  
(No Arguments)

Function Call to int type  
(Two Arguments)

# Recursion

1. Recursion is when a function calls itself during its execution.
2. What is recursion useful for?
  1. Looping.
  2. Using new arguments , which are obtained through the function itself.
  3. Data Structures and Algorithms.
  4. Polling – Embedded Systems



The background of the slide features a series of thin, curved lines in a light gray color, creating a sense of motion and depth. These lines are more prominent on the left side and fade out towards the right.

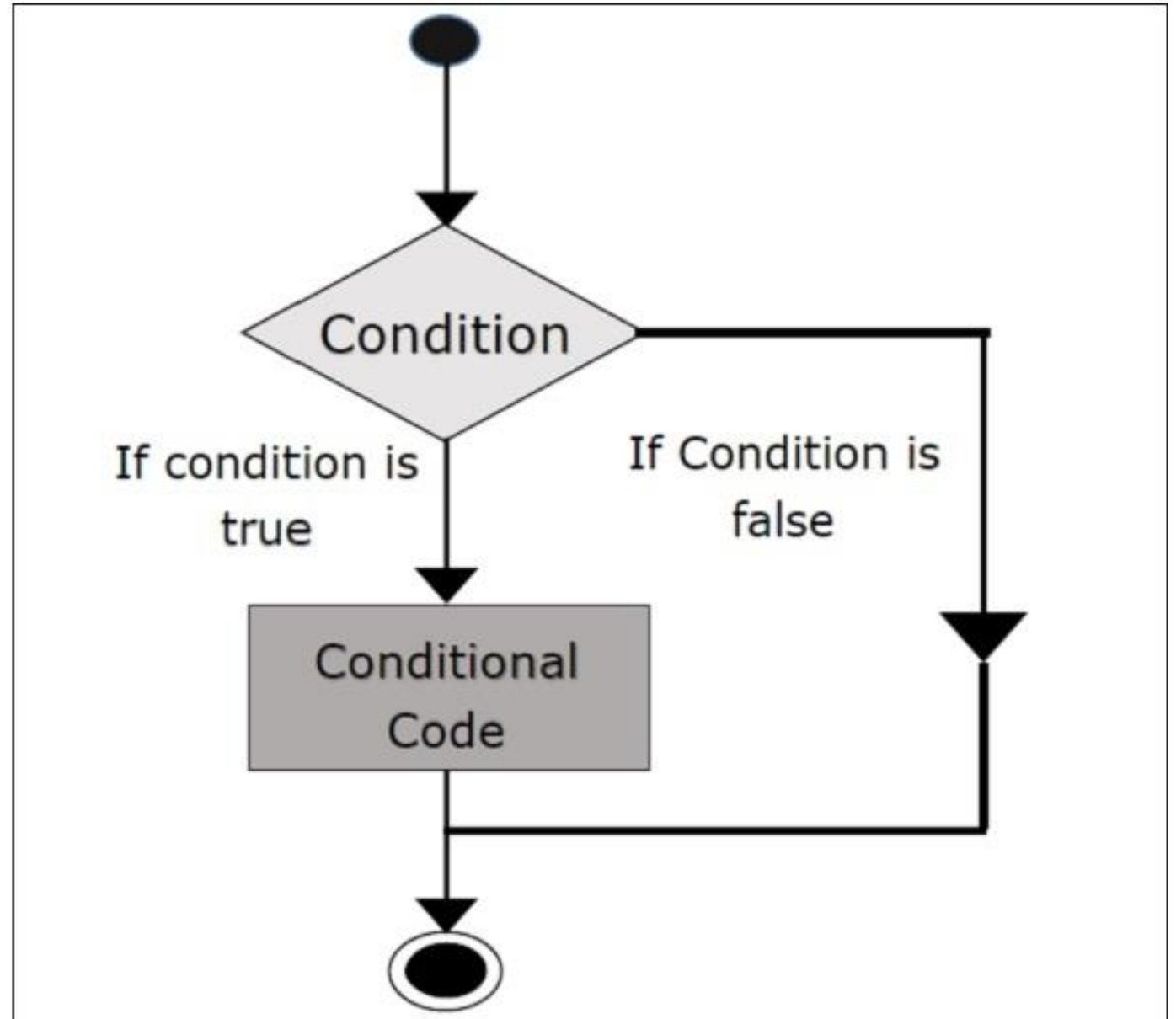
## What are functions useful for?

Using functions programmers can:

1. Make their Code Reusable.
2. Share and Collaborate with other developers.
3. Compile Libraries and Drivers
4. Use Recursion
5. Make their code organized and readable.

# Control Flow If-Else

---



# If-Else Statement

## What is control flow?

- A program always executes in a sequential manner.
- Sometimes , it is required to redirect the flow of program to a different outcome.
- The redirecting of program flow can be done using If-Else Statements.

## How to Use If-Else?

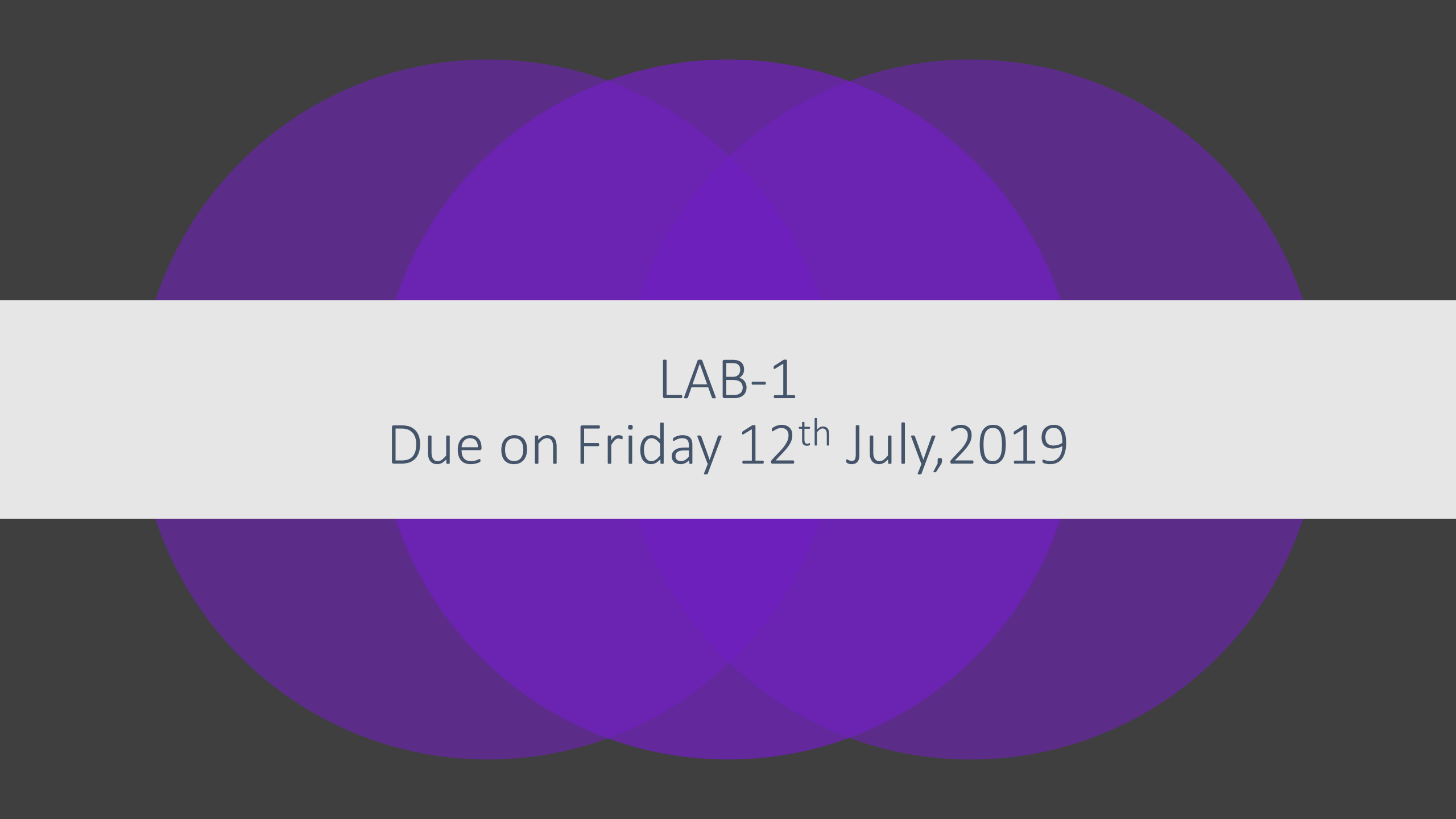
```
if (conditional statement){  
    statement1;  
    statement2;  
    }  
else if (conditional statement 2) {  
    statement 3;  
    }  
else{  
    default_Statement;  
    }
```

# If-Else Example

---

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  /*
4   This Program verifies your age.
5   */
6  int main()
7  {
8      int a;
9      printf("Enter Your Age==>");
10     scanf("%d",&a);
11
12     if (a>18)
13     {
14         printf("You are allowed to enter.");
15     }
16     else
17     {
18         printf("STOP");
19     };
20     return 0;
21 }
```

// Comments can also be in-line  
// Align all comments for readability  
//Variable to store age  
//  
//Scan input  
  
// If age is above 18 , allow entry  
//  
// Else stop , else if can also be used.  
//  
//Return 0 to indicate successful execution

The background of the slide features three large, overlapping circles in a vibrant purple color. These circles are arranged horizontally, with the middle circle slightly offset upwards and downwards relative to the two flanking circles, creating a central triangular intersection. The circles are set against a dark gray background. A horizontal white band cuts across the middle of the image, containing the text.

LAB-1  
Due on Friday 12<sup>th</sup> July, 2019

# CLASS SUMMARY

What we have covered:

1. Intro. to Internet of Things
2. Intro. To Programming
  1. The Sanity Test Program – Hello World
  2. Primitive Data Types
  3. Header Files and Libraries
  4. Variable Definitions and Declarations
  5. Printing To Console
  6. Control Flow
  7. Loops
  8. Recursion

# ADDITIONAL READING

What you can discover on your own:

1. Recommended Reading :
  1. Head First Introduction to Programming
  2. Head First C - David Griffiths , Dawn Griffiths
2. Practice programming on your own
  1. Explore Coding Practice Services like HackerRank(free for everyone) , Codility , CodeChefs .
  2. Get Used to GitHub.
  3. Stack Overflow will be your best friend.
  4. Reach out to us.
3. Explore New Languages- Some good relevant options are Python , Java , C++ .
4. C is a low-level language