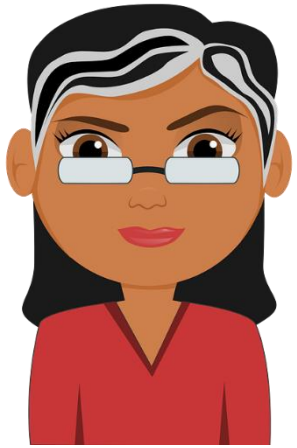
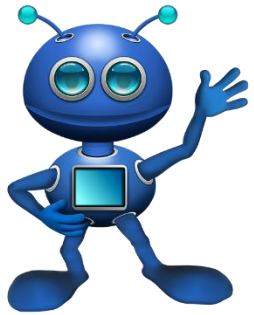
FUNCTIONS

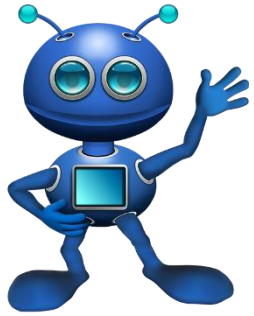
Hey, I want you to write a software that asks the user to find the sum of two numbers, until the user wishes to stop



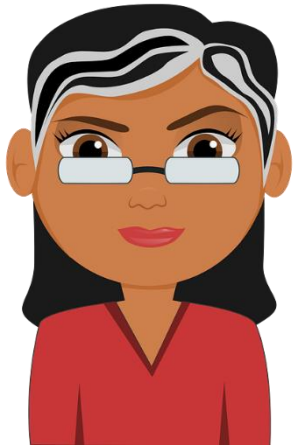


“This is so easy…Here is your code”

FUNCTIONS



"Seeee…what did I tell you?? It's so simple"

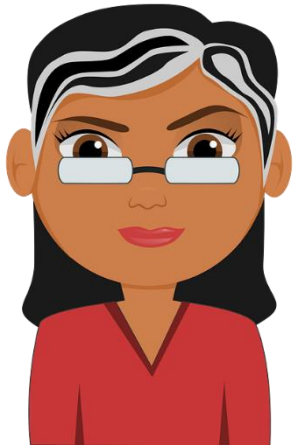


"Hmmmm…Can you read my question again????"

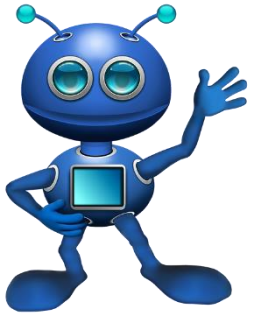
```c
/* Program to add two numbers
   and to display their sum
*/
// Date: 21-08-2018
#include<stdio.h>

int main()
{
   //Declararing and initialising variables
   int first_number=0, second_number=0, sum=0;
   //Read the first number from the user
   printf("Enter the first number:");
   scanf("%d",&first_number);
   //Read the second number from the user
   printf("Enter the second number:");
   scanf("%d",&second_number);
   //Find the sum
   sum=first_number+second_number;
   //Display the sum
   printf("Sum is %d", sum);
   return 0;
}
```

FUNCTIONS

Hey, I want you to write a software that asks the user to find the sum of two numbers, **until the user wishes to stop**



What about this part, **until the user wishes to stop**

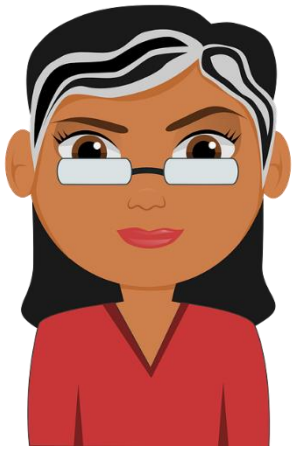


"Oh!!! My program will run only one time"

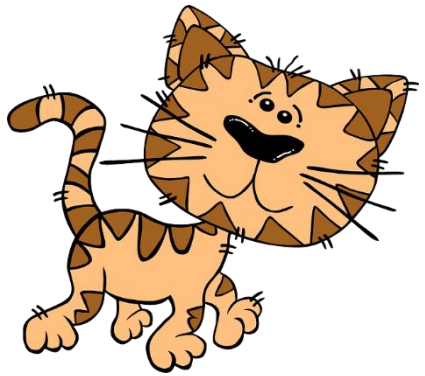
"So what should I do??"

FUNCTIONS

"Alright, before I answer your question, what is lacking in your code"



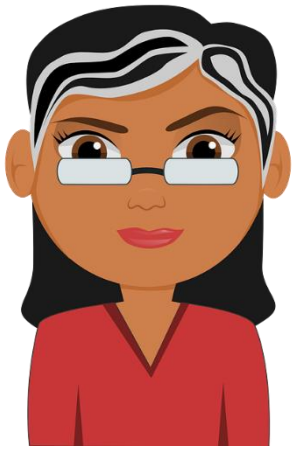
"It asks the user to enter only two numbers and displays the sum. **But it does not ask the user to enter another set of numbers a second time or a third time** "



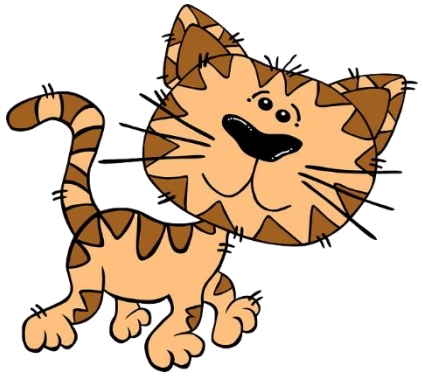
```c
/* Program to add two numbers
   and to display their sum
*/
// Date: 21-08-2018
#include<stdio.h>

int main()
{
   //Declararing and initialising variables
   int first_number=0, second_number=0, sum=0;
   //Read the first number from the user
   printf("Enter the first number:");
   scanf("%d",&first_number);
   //Read the second number from the user
   printf("Enter the second number:");
   scanf("%d",&second_number);
   //Find the sum
   sum=first_number+second_number;
   //Display the sum
   printf("Sum is %d", sum);
   return 0;
}
```

FUNCTIONS

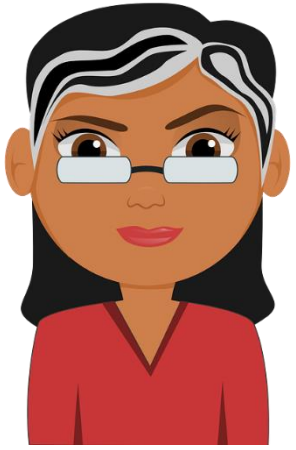
"So, when you have to do a task repeatedly you have two options – **Loops** OR **Functions**"



"Yes…I was about to say we can use LOOPS for the solution"

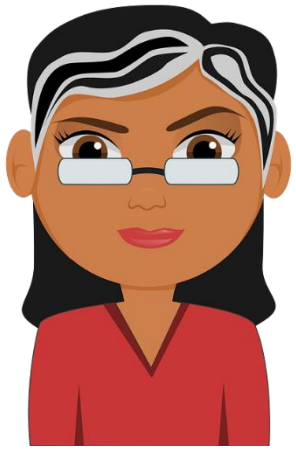


"Yea..i know you can..so do it as **HOMEWORK** and upload it today!!!"

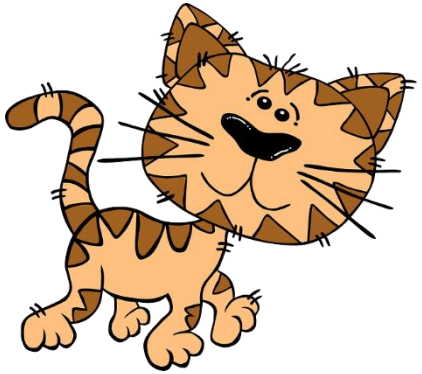


FUNCTIONS

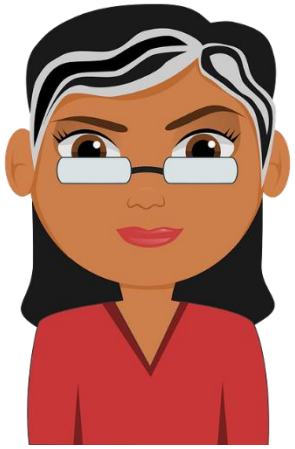
"So, when you have to do a task repeatedly you have two options – **Loops** OR **Functions**"



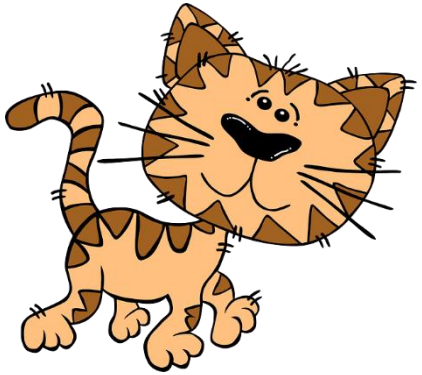
"Yes…I was about to say we can use LOOPS for the solution"



"Yea..i know you can..so do it as **HOMEWORK** and upload it today!!!"

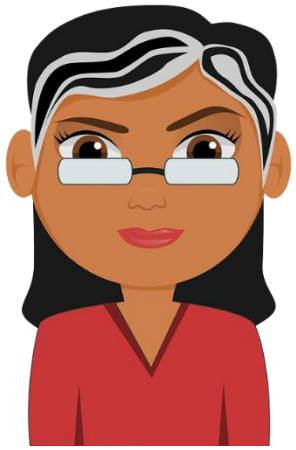


"But if we can get the solution via LOOPS then why should we learn FUNCTIONS"

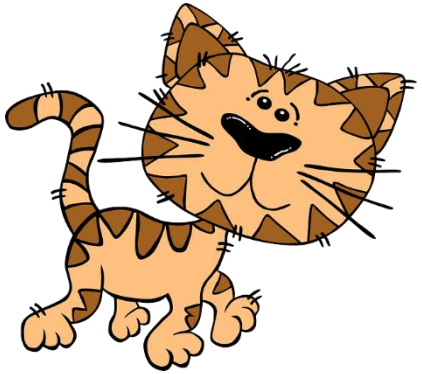


FUNCTIONS

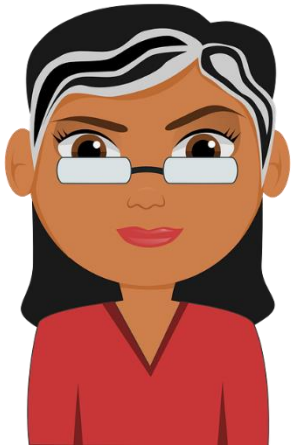
“Because, you will find that **FUNCTIONS have more advantages than LOOPS** as we proceed further in the course”



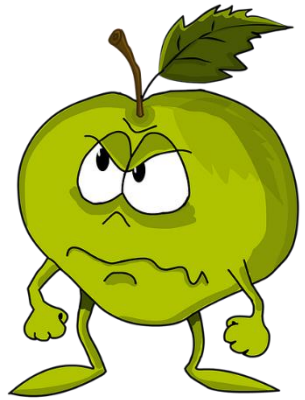
“Why don’t you say it now? What those advantages are?”



“I would rather say it by showing examples. BUT UNLESS YOU READ THE TEXTBOOK, AND TRY OUT THE CODE ON YOUR OWN, YOU CANNOT PASS THIS COURSE!”



“Yes, Yes I know.. You have been saying the same story since the beginning.”



OBJECTIVE

WHAT

WHY

HOW

FUNCTION

Objective

WHAT

"A **function** is a self-contained **block of statements that perform a coherent task of some kind.** Every C program can be thought of as a collection of these functions." – Let Us C by Yashwant Kanetkar

Using a function is something like hiring a person to do a specific job for you. **Sometimes the interaction with this person is very simple, sometimes it's complex** - Let Us C by Yashwant Kanetkar

Objective

WHAT

"A **function** is a self-contained **block of statements that perform a coherent task of some kind.** Every C program can be thought of as a collection of these functions." – Let Us C by Yashwant Kanetkar

```c
/*
    Demo of a function
*/
// Date: 21-08-2018
#include<stdio.h>
void message();
int main()
{
   message() ;
   printf ( "This statement is in the main function" ) ;
   return 0;
}
void message()
{
printf ( "This is in the function called MESSAGE()" ) ;
printf("\n");
}
```

Objective

WHAT

"A **function** is a self-contained **block of statements that perform a coherent task of some kind" -** Let Us C by Yashwant Kanetkar

Block of statements that make up the function **message**

```
1  /*
2      Demo of a function
3  */
4  // Date: 21-08-2018
5  #include<stdio.h>
6  void message();
7  int main()
8  {
9     message() ;
10    printf ( "This statement is in the main function" ) ;
11    return 0;
12 }
13 void message()
14 {
15 printf ( "This is in the function called MESSAGE()" ) ;
16 printf("\n");
17 }
```

Objective

WHAT

COMPONENTS

- Function Prototype
- Function Invocation
- Function definition

```
1  /*
2      Demo of a function
3  */
4  // Date: 21-08-2018
5  #include<stdio.h>
6  void message();
7  int main()
8  {
9     message() ;
10    printf ( "This statement is in the main function" ) ;
11    return 0;
12 }
13 void message()
14 {
15 printf ( "This is in the function called MESSAGE()" ) ;
16 printf("\n");
17 }
```

Function Invocation / Calling the function

Function Definition

Summary of functions

Brief Summary

- Any C program contains at least one function.
- If a program contains only one function, it must be **main( )**.
- If a C program contains more than one function, then one (and only one) of these functions must be **main( )**, because program execution always begins with **main( )**.
- There is no limit on the number of functions that might be present in a C program.
- Each function in a program is called in the sequence specified by the function calls in **main( )**.
- After each function has done its thing, control returns to **main( )**.When **main( )** runs out of function calls, the program ends.

From Let Us C by Yashwant Kanetkar, 5th edition

Try out yourself-1

"Follow these instructions"

"1. Type the code "

"2. Run the code"

"3. Which line of code **invokes** the function?"

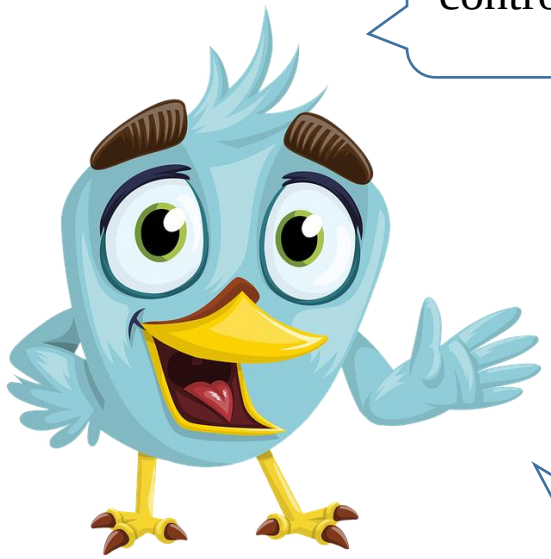
"4. From which line of code does the function **definition** begin?"

```c
/* Program to invoke
   a function called
   'sum'
*/
// Date: 21-08-2018
#include<stdio.h>
void sum(void);
int main()
{
   sum();
   return 0;
}
void sum()
{
    printf("Look this code\n");
    printf("is in a function\n");
    printf("called SUM\n");
}
```

Homework-2

"At line number 10, the **sum()** is **invoked/called**

"4. From line number 10, the control of the program moves to line number 13.



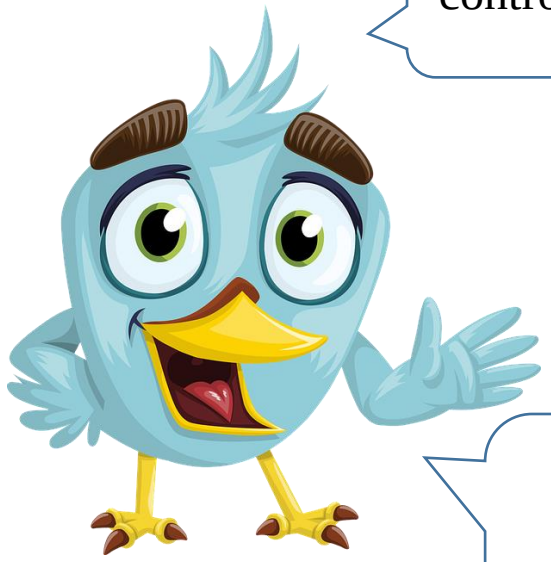
"4. After finishing the block of statements beginning with line number 13-18, the control will go back OR return to line number 14.

```
1  /* Program to invoke
2     a function called
3     'sum'
4  */
5  // Date: 21-08-2018
6  #include<stdio.h>
7  void sum(void);
8  int main()
9  {
10    sum();
11    return 0;
12 }
13 void sum()
14 {
15      printf("Look this code\n");
16      printf("is in a function\n");
17      printf("called SUM\n");
18 }
```

Homework-3

"At line number 13, the **hello()** is **invoked/called**

"4. From line number 13, the control of the program moves to line number 18.



"4. After finishing the block of statements beginning with line number 18-21, the control will go back OR return to line number 14.

```
1  /*
2     Demo program to show the
3     working of a function call
4
5     Date: 24-08-2018
6     Time: 2:30 PM
7  */
8  #include<stdio.h>
9  void hello(void);
10 void helloagain(void);
11 int main()
12 {
13   hello();
14   printf("Hello function has been executed\n");
15   helloagain();
16   return 0;
17 }
18 void hello()
19 {
20     printf("In Hello function\n");
21 }
22 void helloagain()
23 {
24     printf("In Hello Again function\n");
25 }
```

Find the output-1

"What is the output of the code?

In Hello function
Hello function has been executed
In Hello Again function

```c
/*
   Demo program to show the
   working of a function call

   Date: 24-08-2018
   Time: 2:30 PM
*/
#include<stdio.h>
void hello(void);
void helloagain(void);
int main()
{
  hello();
  printf("Hello function has been executed\n");
  helloagain();
  return 0;
}
void hello()
{
    printf("In Hello function\n");
}
void helloagain()
{
    printf("In Hello Again function\n");
}
```

Find the output-2

"What is the output of the code?

Till the memory space runs out

```
main( )
{
    printf ( "\nOnly stupids use C?" ) ;
    display( ) ;
}
display( )
{
    printf ( "\nFools too use C!" ) ;
    main( ) ;
}
```

Let us C by Yashwant Kanetkar

WHAT does ‘return’ mean?

The new feature in this program is in line number 12

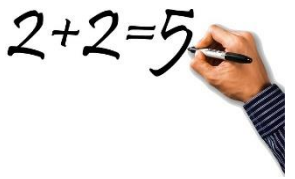
result = demo()
To understand the meaning behind this line of code, let us look at
int a = 10;
What does it mean?

It means that the value 10 is assigned to the integer variable **‘a’**

Go ahead, type and execute the code…

ERROR? Is It?

```
1  /*
2  C program to demonstrate the
3  working of 'return'
4  Date:26-08-2018
5  Time: 11:50 AM
6  */
7  #include<stdio.h>
8  void demo();
9  int main()
10 {
11    int result;
12    result = demo();
13    printf("The value returned by sum is %d",result);
14    return 0;
15 }
16 void demo()
17 {
18    int a;
19    printf("Enter the number: ");
20    scanf("%d",&a);
21    return a;
22 }
```



After the **demo** function executes all of it’s statements, it gives back a value to **‘result’** variable

WHAT does ‘return’ mean (HOMEWORK-3)?

Look at line number 8 and 16

void is replaced with int

Because ‘a’ is returned back and the data type of ‘a’ is int

The type of the value returned has to be reflected

Go ahead, type and execute the code…

```
1  /*
2  C program to demonstrate the
3  working of 'return'
4  Date:26-08-2018
5  Time: 11:50 AM
6  */
7  #include<stdio.h>
8  int demo();
9  int main()
10 {
11    int result;
12    result = demo();
13    printf("The value returned by sum is %d",result);
14    return 0;
15 }
16 int demo()
17 {
18    int a;
19    printf("Enter the number: ");
20    scanf("%d",&a);
21    return a;
22 }
```

After the **demo** function executes all of it’s statements, it gives back a value to **‘result’** variable

CORRECT THE MISTAKE IN THE CODE

Read the comment to understand the objective of the program

Go through the code line by line

Did you find the logical error?

In which line number?

What is to be changed?

Go ahead, type and execute the code…

```c
/*
C program to add 10 to the
number entered by the user
and return it to the main function
Date:26-08-2018
Time: 1:50 AM
*/
#include<stdio.h>
int demo();
int main()
{
   int result;
   result = demo();
   printf("The value returned by sum is %d",result);
   return 0;
}
int demo()
{
   int a,b;
   printf("Enter the number: ");
   scanf("%d",&a);
   b=a+10;
   return a;
}
```

WRITE A C PROGRAM TO FIND THE SUM OF DIGITS OF A NUMBER USING A FUNCTION

Given on the right is the program to find the sum of digits of a number

But that is using the main function

Can you write a code that returns the sum of digits of a number from another function to the main function

```c
#include<stdio.h>
int main()
{
    int a,sum=0;
    printf("Enter the number: ");
    scanf("%d",&a);
    while(a>0)
    {
        sum  = sum + a%10;
        a=a/10;
    }
    printf("The sum of digits is %d",sum);
    return 0;
}
```

WRITE A C PROGRAM TO FIND THE SUM OF DIGITS OF A NUMBER USING A FUNCTION

```c
#include<stdio.h>
int main()
{
    int a,sum=0;
    printf("Enter the number: ");
    scanf("%d",&a);
    while(a>0)
    {
        sum  = sum + a%10;
        a=a/10;
    }
    printf("The sum of digits is %d",sum);
    return 0;
}
```

```c
/*
C program to find the sum of digits
and return it to the main function
Date:26-08-2018
Time: 1:50 AM
*/
#include<stdio.h>
int demo();
int main()
{
   int result;
   result = demo();
   printf("The value returned by sum is %d",result);
   return 0;
}
int demo()
{



        //FILL THE EMPTY SPACE WITH CODE



}
```