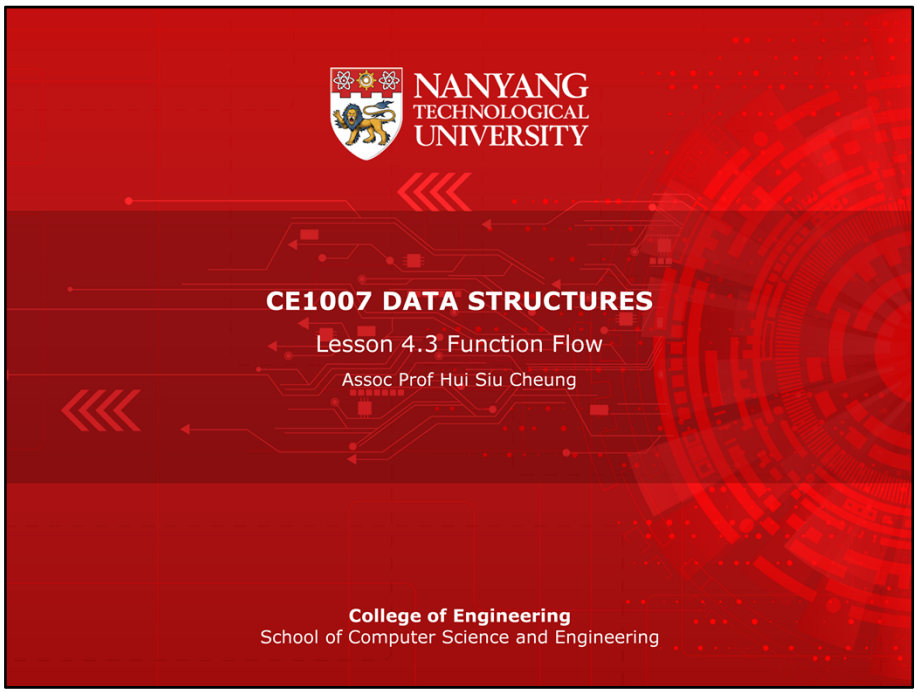
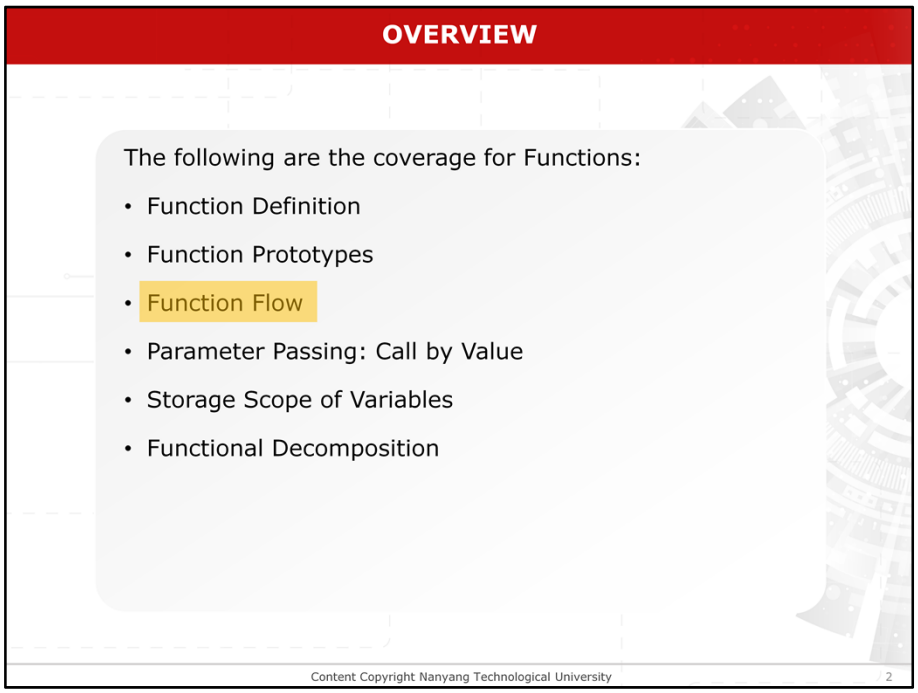
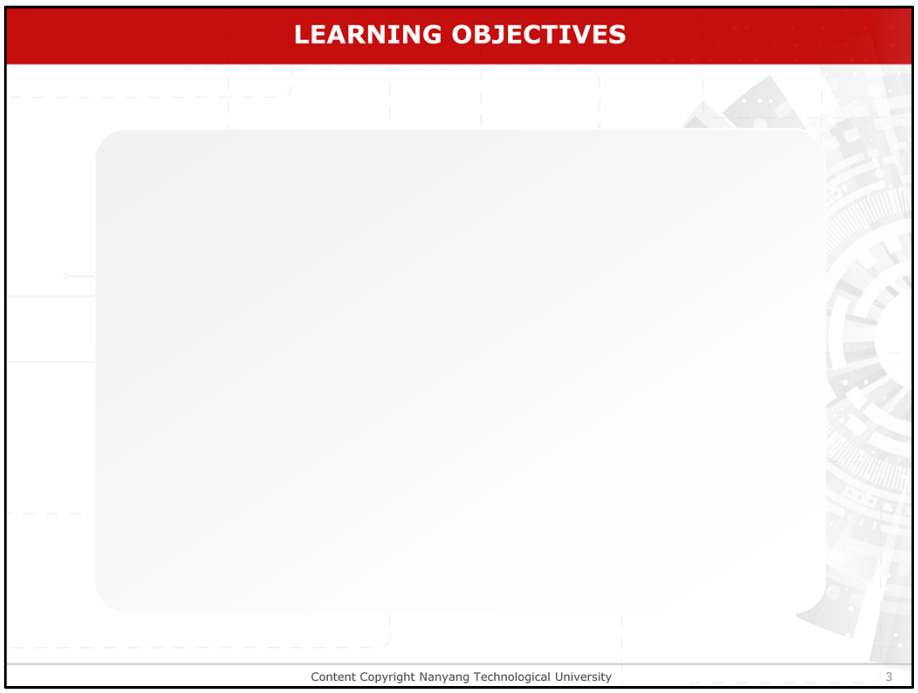


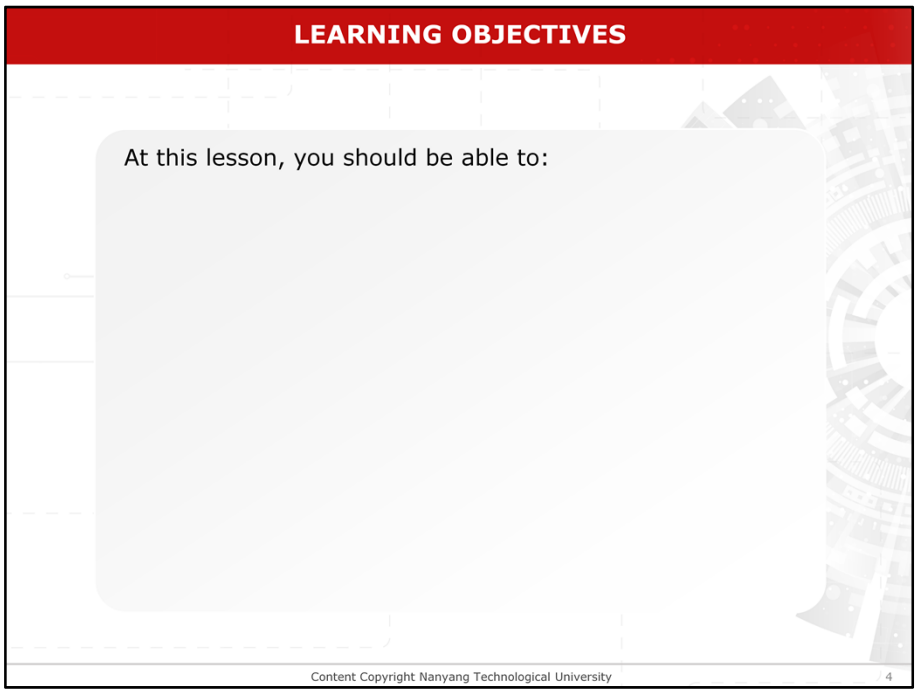
This lesson is on Functions



There are 6 main sections to cover for Functions. This video focused on the 3rd topic: Function flow



Learning objectives



At this lesson, you should be able to:

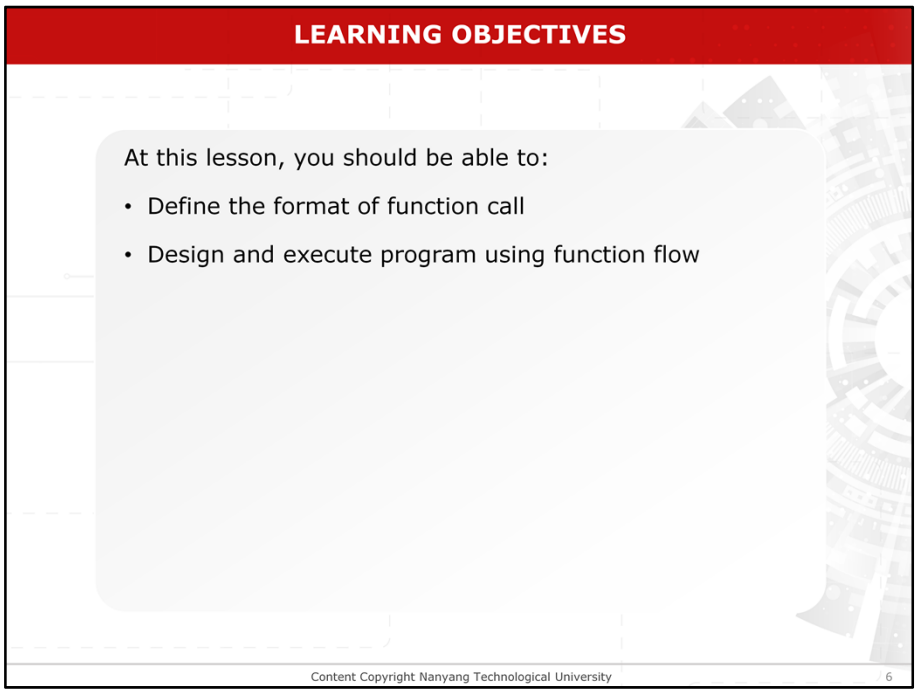
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Define the format of function call

 5

Define the format of function call



- Design and execute program using Function flow

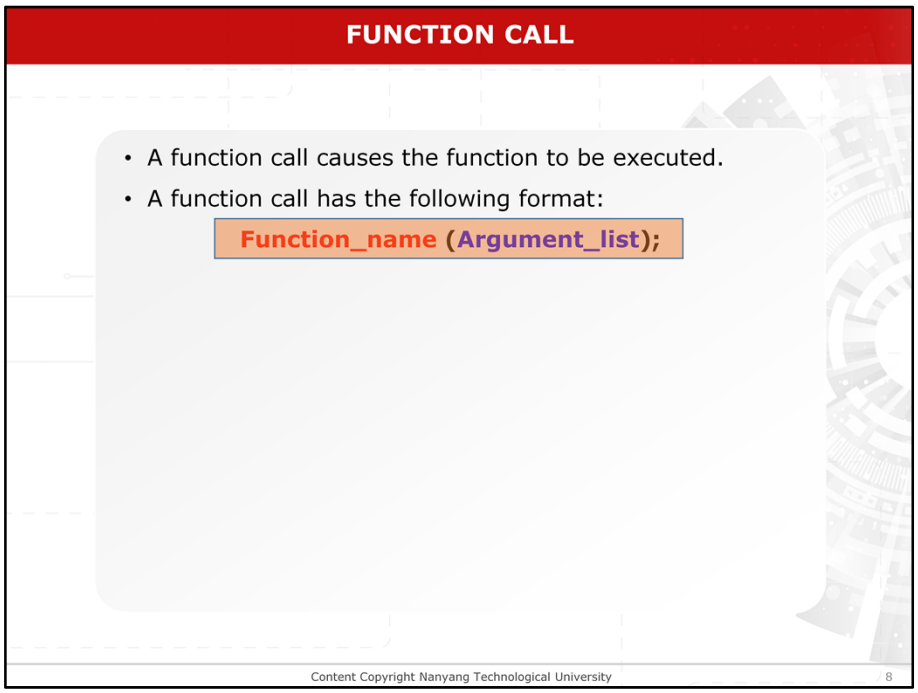
FUNCTION CALL

- A function call causes the function to be executed.

 7

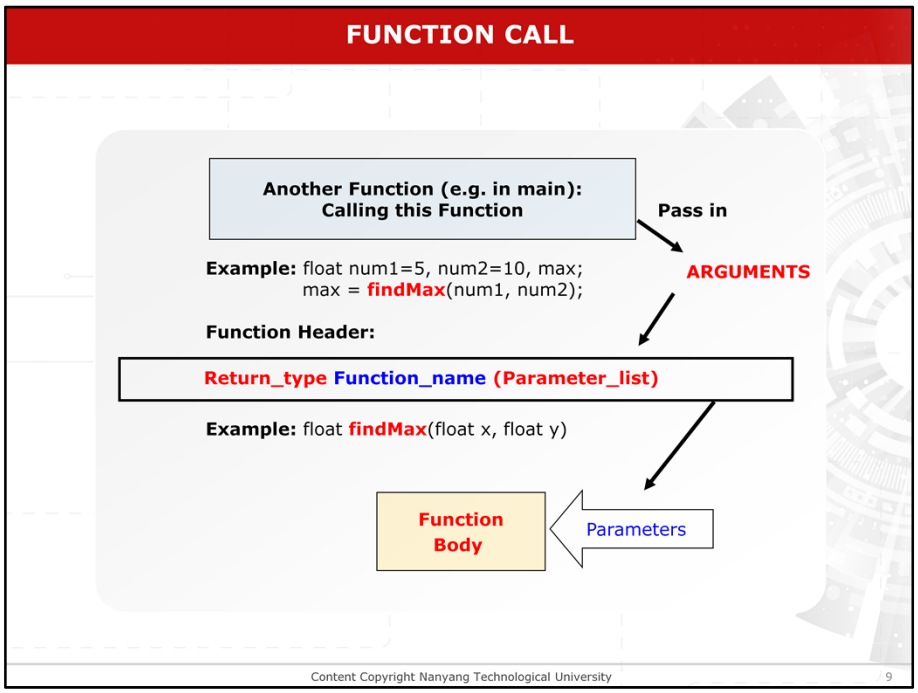
Function Call

A function is executed when it is called.

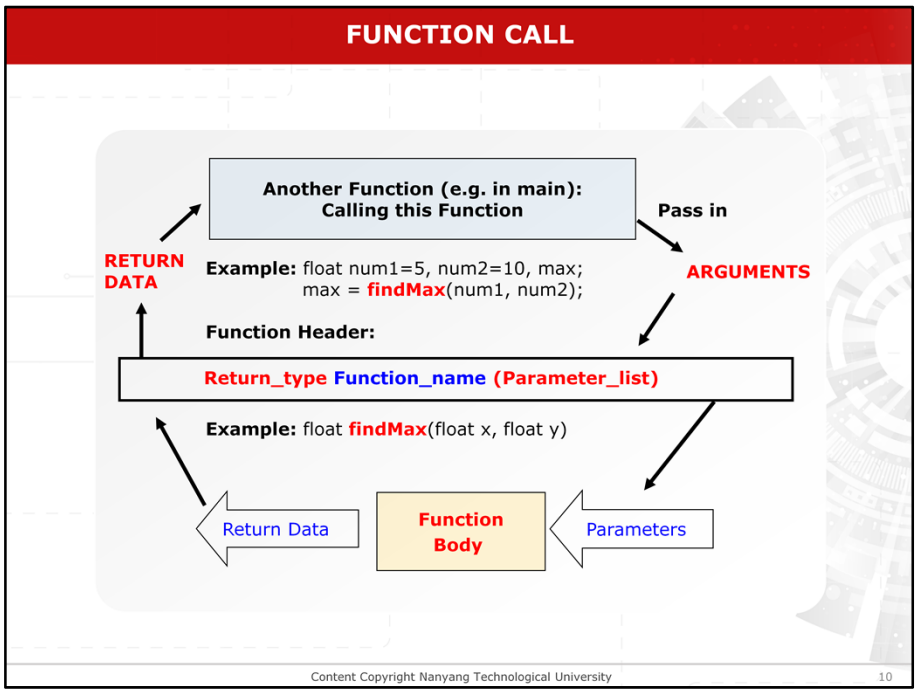


Function Call

A function is executed when it is called. A function call has the format as shown:



The function can be called by using the function name followed by a list of *arguments as shown here*


FUNCTION CALL
Another Function (e.g. in main):
Calling this Function
Pass in
ARGUMENTS
RETURN DATA
Example: float num1=5, num2=10, max;
max = findMax(num1, num2);
Function Header:
Return_type Function_name (Parameter_list)
Example: float findMax(float x, float y)
Return Data
Function Body
Parameters
Content Copyright Nanyang Technological University
10

FUNCTION CALL

The statement

```
hello_n_times(4);
```

calls the function **hello_n_times()**.

 11

The statement **hello_n_times(4)** calls the function **hello_n_times()**.

FUNCTION CALL

The statement

```
hello_n_times(4);
```

calls the function **hello_n_times()**.

- One argument is passed to the function. The function does not return a value.

 12

One argument is passed to the function

FUNCTION CALL

The statement

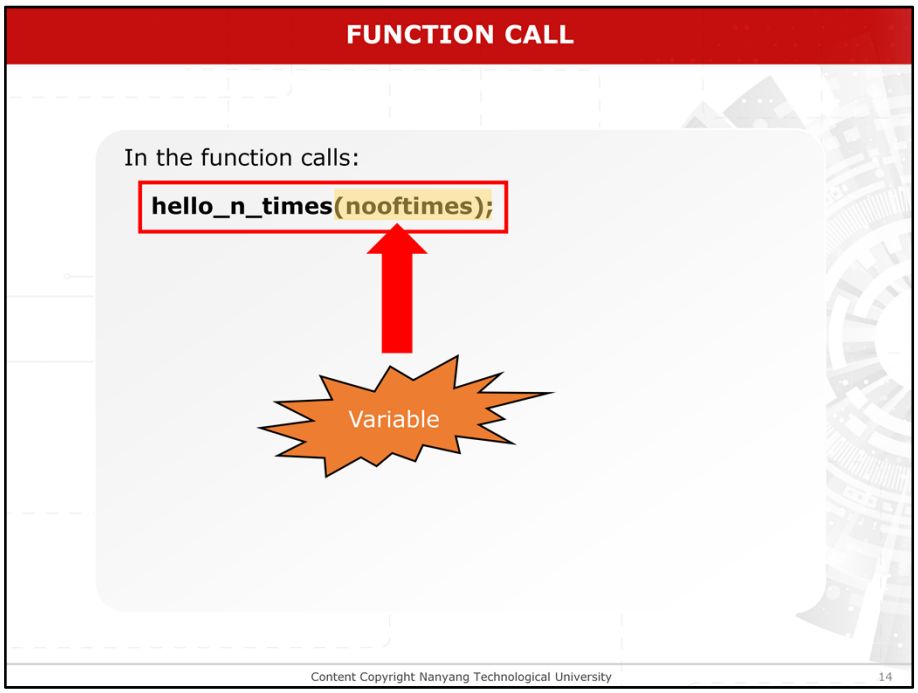
```
hello_n_times(4);
```

calls the function **hello_n_times()**.

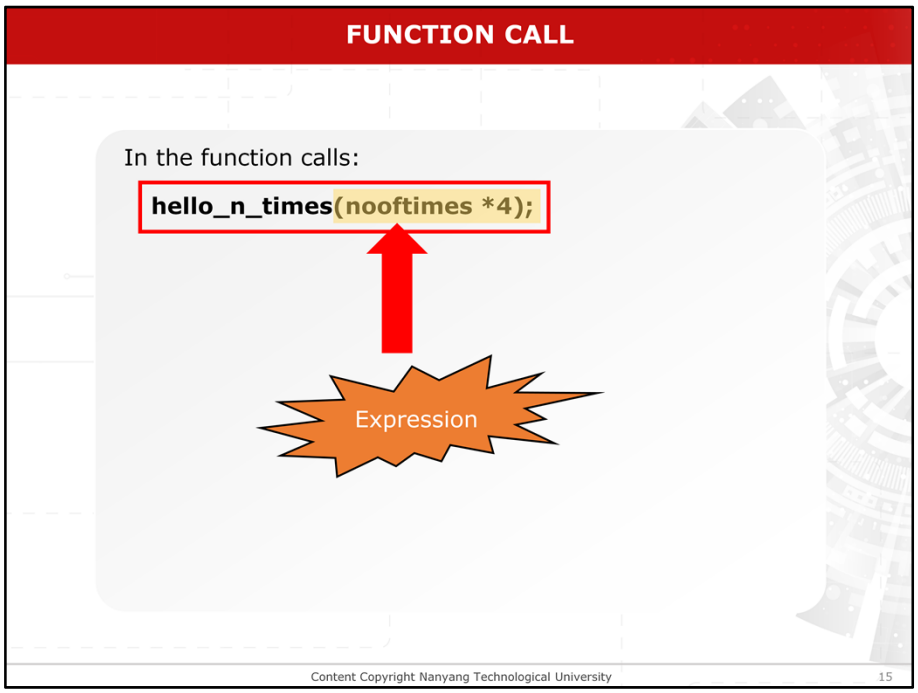
- One argument is passed to the function. The function does not return a value.
- Function arguments can be constants, variables or expressions.

 13

Function arguments can be constants, variables or expressions.



In the function call shown here, the argument **nooftimes** is a variable,



Whereas In the function calls shown here, where the argument **nooftimes*4** is a expression

FUNCTION CALL

We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

 16

We can also call a function that returns a value as shown here.

FUNCTION CALL

We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

- The function call has two arguments, separated by a comma.

The function call has two arguments, separated by a comma.

FUNCTION CALL

We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

- The function call has two arguments, separated by a comma.
- The function also returns a value.

The function also returns a value.

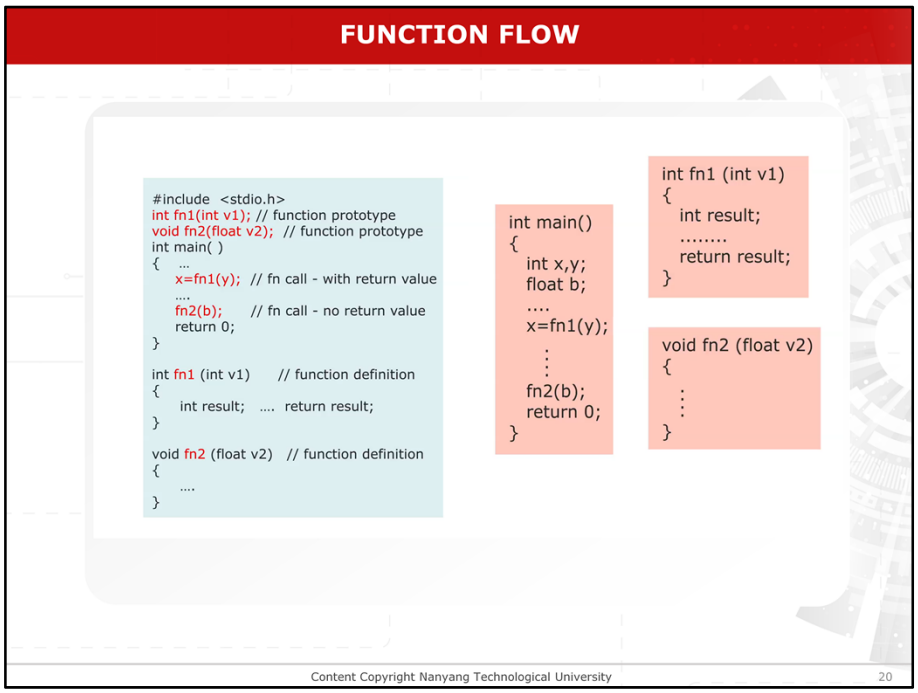
FUNCTION CALL

We can also call a function that returns a value:

```
dist = distance(2.0, 4.5);
```

- The function call has two arguments, separated by a comma.
- The function also returns a value.
- The returned value is assigned to the variable **dist**.

The returned value is assigned to the variable **dist**.



Function Flow

In the program, the **main()** function will start the execution. When the function **fn1()** is called, the program transfers the control to the **fn1()** function which then starts execution. As **fn1()** will return a value back to the calling function, the statements in the function body of **fn1()** will be executed until a **return** statement is encountered. Control is then transferred back to the **main()** function. The value of the variable **result** will be assigned to the variable **x** in **main()**. The next statement after the function call then starts execution. When the second function **fn2()** is called. The control is then transferred to **fn2()**. The function will execute until the end of the function body. Control will then be transferred back to the **main()** function.

FUNCTION FLOW: EXAMPLES

```
#include <stdio.h>
char findGrade(float marks);
int main( )
{
  char answer;
  answer = findGrade(68.5);
  printf("Grade is %c", answer);
  return 0;
}

char findGrade(float marks){
  char grade;   // variable

  if (marks >= 50)
    grade = 'P';
  else
    grade = 'F';
  return grade;
}
```

Click to play

 21

Function Flow: findGrade()

In the program, the **main()** function calls the function **findGrade()**. When the statement an**swer = findGrade(68.5)** is executed, it calls the function **findGrade()**. Control is then transferred to the function **findGrade()**. Information is passed between the calling function and the called function through the argument. In this case, the function receives one argument with the value of **68.5**. It is assigned to the corresponding parameter in the function definition to compute the grade. When the execution of statements in the function body encounters the **return** statement, the control is then transferred back to the **main()** function, and the statement just after the function call in **main()** will continue to execute. The name for parameter needs not be the same as function argument. However, the number of arguments and the data type of the arguments must be the same as parameters defined in function definition. In the program, the argument **68.5** must correspond to the parameter **marks** in the function call. Notice that the function prototype is declared as: **float findGrade(float marks)** and is placed at the beginning of the program before the **main()** function.

FUNCTION FLOW: EXAMPLES

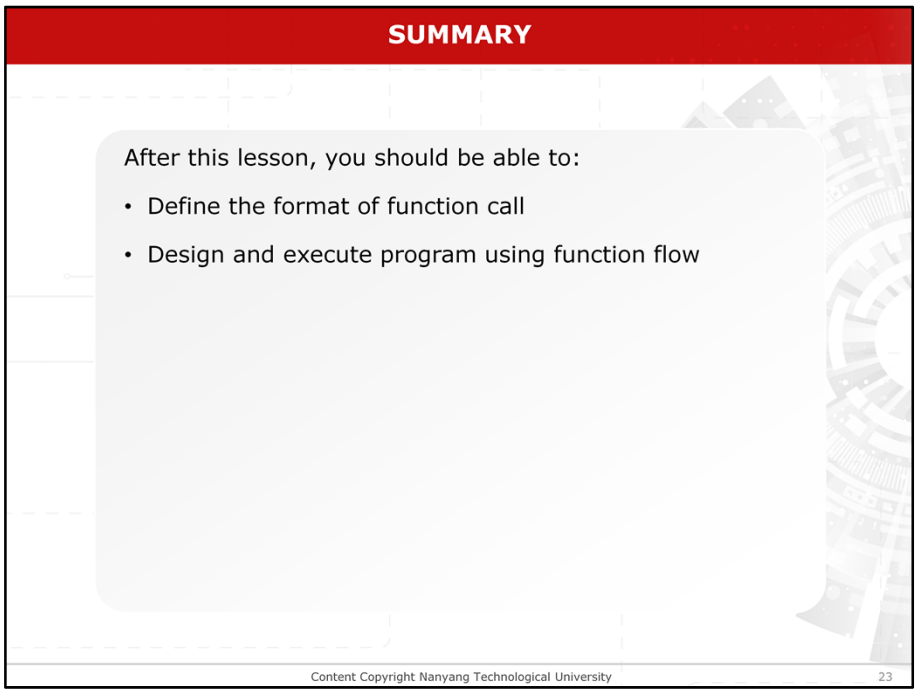
```
#include <stdio.h>
float areaOfCircle(float);
int main( )
{
  float answer;
  answer = areaOfCircle(2.5);
  printf("Area is %.1f", answer);
  return 0;
}

float areaOfCircle(float radius){
  const float pi = 3.14;
  float area;
  area = pi*radius*radius;
  return area;
}
```

Click to play

 22

Function Flow: areaOfCircle()

In the program, the **main()** function calls the function **areaOfCircle()**. When the statement **answer = areaOfCircle(2.5)** is executed, it calls the function **areOfCircle ()**. Control is then transferred to the function **areaOfCircle()**. Information is passed between the calling function and the called function through the argument. In this case, the function receives one argument with the value of **2.5**. It is assigned to the corresponding parameter in the function definition to compute the area of the circle. When the execution of statements in the function body encounters the **return** statement, the control is then transferred back to the **main()** function, and the statement just after the function call in **main()** will continue to execute. Notice that the function prototype is declared as
float areaOfCircle(float) and is placed at the beginning of the program before the **main()** function.



In summary, after viewing this video lesson, your should be able to the listed.