F# - MODULES

As per MSDN library, an F# module is a grouping of F# code constructs, such as types, values, function values, and code in do bindings. It is implemented as a common language runtime *CLR* class that has only static members.

Depending upon the situation whether the whole file is included in the module, there are two types of module declarations −

- Top-level module declaration
- Local module declaration

In a top-level module declaration the whole file is included in the module. In this case, the first declaration in the file is the module declaration. You do not have to indent declarations in a top-level module.

In a local module declaration, only the declarations that are indented under that module declaration are part of the module.

Syntax

Syntax for module declaration is as follows −

```
// Top-level module declaration.
module [accessibility-modifier] [qualified-namespace.]module-name
   declarations
// Local module declaration.
module [accessibility-modifier] module-name =
   declarations
```

Please note that the accessibility-modifier can be one of the following − public, private, internal. The default is **public**.

The following examples will demonstrate the concepts −

Example 1

The module file Arithmetic.fs −

```
module Arithmetic
let add x y =
   x + y

let sub x y =
   x - y

let mult x y =
   x * y

let div x y =
   x / y
```

The program file main.fs −

```
// Fully qualify the function name.
open Arithmetic
let addRes = Arithmetic.add 25 9
let subRes = Arithmetic.sub 25 9
let multRes = Arithmetic.mult 25 9
let divRes = Arithmetic.div 25 9

printfn "%d" addRes
```

```
printfn "%d" subRes
printfn "%d" multRes
printfn "%d" divRes
```

When you compile and execute the program, it yields the following output −

```
34
16
225
2
110
90
1000
10
```

Example 2

```
// Module1
module module1 =
   // Indent all program elements within modules that are declared with an equal sign.
   let value1 = 100
   let module1Function x =
      x + value1

// Module2
module module2 =
   let value2 = 200

   // Use a qualified name to access the function.
   // from module1.
   let module2Function x =
      x + (module1.module1Function value2)

let result = module1.module1Function 25
printfn "%d" result

let result2 = module2.module2Function 25
printfn "%d" result2
```

When you compile and execute the program, it yields the following output −

```
125
325
```