FUNCTION

(subprogram --> set of instructions)

Function are subprograms which are used to compute a value or perform a task.

TYPES OF FUNCTIONS --> ( )

BUILD - IN FUNCTION

printf( ) , scanf( ) , pow( ) , strcpy( ), sqrt( ),.... .

.

USER – DEFINED FUNCTION :- main( )

BENEFIT OF FUNCTIONS --> ()

1. reuse code

2. reduce main() size

3. debugging (error handling)

4. extend(add)

---

main() --> 2000 -->

fact --> 5 * 25 = 125 lines

---

function --> fact -> 5 (create)
-> 25 (fun. call)

= 30 lines

---

1. programmer :- function // calculation

2. user (run) :- main() // input , print

---

using (calling) pow() function

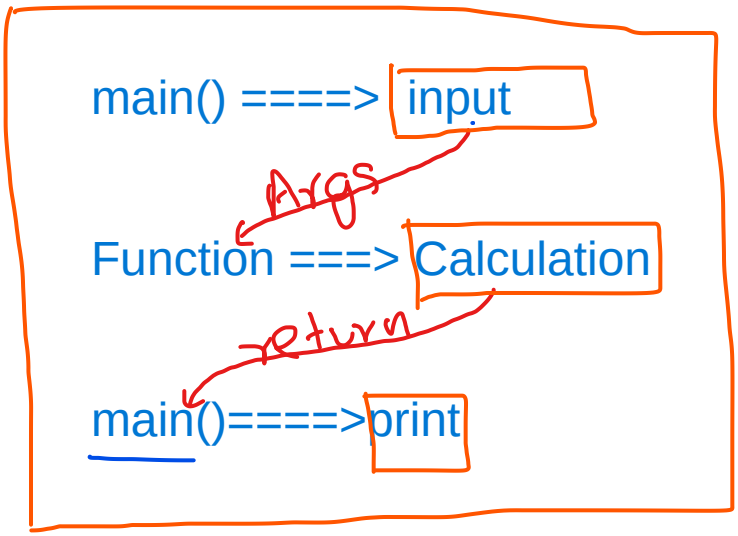
```
int main() // user
{ int x = 2 , n = 3 , t ; // input
t = pow( x , n ); // fun. calling
printf(" %d\n " , t); // print
}
```

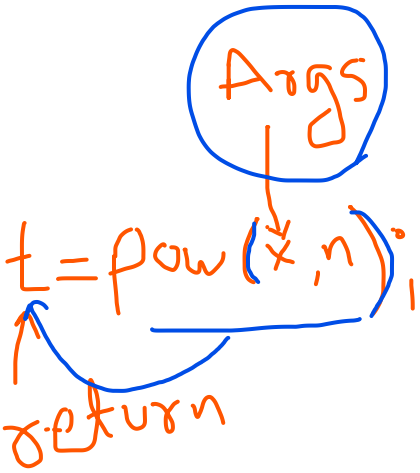
---

1. ARGUMENT (parameter)

2. RETURN

SUMMARY OF FUNCTION





CATEGORY OF FUNCTION

CASE 1. NO AGRUMENT AND NO RETURN VALUE

CASE 2. NO AGRUMENT AND RETURN VALUE

CASE 3. AGRUMENT AND NO RETURN VALUE

CASE 4. AGRUMENT AND RETURN VALUE // imp

1 . NO ARGUMENT AND NO RETURN VALUE

FUN( ) MAIN( )
INPUT
CAL. FUN. CALLING
PRINT

CASE 1. NO AGRUMENT AND NO RETURN VALUE

```
//  ADDITION OF TWO NOS
    #include<stdio.h>

    void sum()
    {
        int a , b , c;

        printf(" ENTER TWO NOS \n ");
        scanf("%d%d", &a , &b);  // INPUT
```

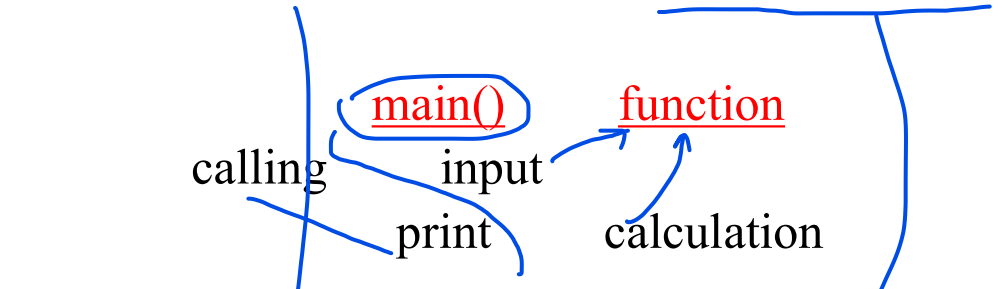
```
        c = a + b ;  // CALCULATION
        printf(" SUM = %d\n ", c);  // PRINT
}
int    main()
{
        sum();  // fun. calling
}
```

// CASE 1 . NO AGRUMENT AND NO RETURN VALUE

```
#include<stdio.h>

void   fact()
{
        int  n , i , f = 1;

        printf(" ENTER NO \n ");

        scanf("%d", &n);

        for( i = 1 ; i <= n ; i++)
        {
                f = f * i ;
        }
```

```
        printf(" FACT = %d\n " , f);
}

int    main()
{
        fact();  // fun. calling

}
```

CASE 2 : NO AGRUMENT AND RETURN VALUE



```
#include<stdio.h>

int        sum()
{
        int  a , b , c;

         printf(" ENTER TWO NOS \n ");
         scanf("%d%d", &a , &b); // input
```

```
        c = a + b;     // calculation
        return ( c );
}
int     main()
{
        int  t ;

        t = sum(); // fun. Calling

        printf(" SUM = %d\n " , t); // print
}
```

CASE 2 . NO AGRUMENT AND RETURN VALUE

```
#include<stdio.h>

    int   fact()
    {
          int  n , i , f = 1 ;

           printf(" ENTER NO \n ");

           scanf("%d" , &n);
```

```
        for( i = 1 ; i <= n ; i++)
        {
              f = f * i;
        }
        return(f);
}
int   main()
{
        printf(" FACT = %d\n " , fact( ) );  // print

}
```

---

1. POWER
2. EXPONENTIOAL SERIES (E^X)
3. SUM OF N NOS
4. LENTH OF NUMBER

CASE 3. ARGUMENT AND NO RETURN VALUE

main()
input
calling

function
calculation
print

```
//  ADDITION OF TWO NOS

#include<stdio.h>
 void  sum ( int  p , int  q ) ;      // forward  declaration

int     main()
{
        int   a , b;

         printf(" ENTER TWO NOS \n ");
         scanf("%d%d" , &a , &b);  //  input

        sum (a,b); // fun.  calling   // sum(3,6);
}
void      sum ( int  p , int q )
{
             int  c;
              c = p + q ;   // calculation
               printf(" sum = %d\n " , c ); // print
}

//  CASE 3. NO  AGRUMENT  AND  RETURN VALUE
#include<stdio.h>
```

```
void   fact ( int  p );

int    main()
{
        int  n;

         printf(" ENTER NO \n ");

         scanf("%d", &n);

        fact( n );
}

void     fact ( int  p )
{
           int  i , f = 1;

           for( i = 1 ; i <= p ; i++)
           {
                  f = f * i;
           }
             printf(" FACT = %d\n " , f);
}
```

CASE 4. ARGUMENT AND RETURN VALUE

main() function

INPUT
CALLING CALCULATION
PRINT

```
#include<stdio.h>
      int   sum ( int  p , int  q );

      int   main()
      {
            int  a , b , t ;

             printf( " ENTER TWO NOS \n ");
             scanf("%d%d" , &a , &b ); // INPUT

            t = sum ( a,b ); // CALLING

             printf(" SUM = %d\n " , t ); // PRINT

      }
      int      sum ( int  p , int  q )
      {        int   c ;
```

```
            c = p + q;   // calcultion
            return(c);
      }
//  CASE 4.  ARGUMENT AND  RETURN VALUE

#include<stdio.h>

int     fact ( int  p  );  // FUNCTION DECLARATION OR PROTOTYPE

int      main()
{
         int  n , t;

          printf(" ENTER NO \n ");
          scanf("%d", &n);   // input

         t = fact( n );  // FUNCTION CALLING

          printf(" FACT = %d \n " , t);   // print
 }

 int          fact ( int  p )  // FUNCTION  DEFINITION {
int  i , f = 1;
```

```
        for( i = 1 ; i <= p ; i++)
        {
            f = f * i;
        }
        return( f );
}
```

```
        t   =  fact( n );  // ACTUAL ARGUMENT

        int  fact ( int p ) // FORMAL ARGUMENT
```

CASE 1. sum( )

CASE 2. t = sum();

CASE 3. sum(a , b);

CASE 4. t = sum(a , b); // IMP

SUMMARY OF FUNCTIONS

CASE 1. NO ARGUMENT AND NO RETURN VALUE

a. FUN. DEFINITION :- void sum( )

b. FUN. CALLING :- sum( )

c.

main() fun()

CALLING INPUT
CALCULATION
PRINT

CASE 2. NO AGRUMENT AND RETURN VALUE

fun. definition :- int sum()

fun. calling :- t = sum()

main()	fun()
calling	input
print	calculation

CASE 3. AGRUMENT AND NO RETURN VALUE

fun. definition :- void sum (int p , int q)

fun. calling :- sum(a,b)

main() fun()

input calculation
calling print

CASE 4. AGRUMENT AND RETURN VALUE

fun. definition :- int sum (int p , int q)

fun. calling :- t = sum (a , b)

main() fun()

input
calling calculation
print
