# FUNCTIONS

If a group of statements is repeatedly required then it is not recommended to write these statements everytime seperately.We have to define these statements as a single unit and we can call that unit any number of times based on our requirement without rewriting.This unit is nothing but function.

**Python supports 2 types of functions**

1. **Built in Functions**
2. **User Defined Functions**

## Built in Functions:

**Ex: id(),type(),input(), etc**

## User Defined Functions:

The functions which are developed by programmer explicitly according to businessrequirements ,are called user defined functions.

### Syntax: to create user defined function

```
def function_name(parameters) :
----
-----

return value
```

# Parameters

**Parameters are inputs to the function. If a function contains parameters,then at the time of calling,compulsory we should provide values otherwise,otherwise we will get error.**

**Eg:**

```
def  f1(name):
      Print("hello",name)
f1("sumayya")
f1("nahid")
```

**********Output**********

```
hello sumayya
hello nahid
```

**Eg 2: Write a function to take number as input and print its square value**

```
def  square(number):
      print("the square of",number,"is",number* number)
square(10)
square(5)
```

o/p:

```
the square of 10 is 100
the square of 5 is 25
```

## Return Statement:

**Function can take input values as parameters and executes business logic, and returns output to the caller with return statement.**

**Ex:**

```
def  func(x):
     return x
r=func(10)
print(r)
```

************************************Output**************************

```
10
```

**Note:If we are not writing return statement then default return value is None**

```
def  f1():
     print(“hello”)
f1()
print(f1())
```

************************************output************************************

```
hello
hello
None
```

## Types of arguments

```
def  f1(a,b)
     ----
     ----
f1(10,20)
```

**a,b are formal arguments where 10,20 are actual arguments**

**There are 4 types are actual arguments are allowed in Python.**

1. **positional arguments**
2. **keyword arguments**
3. **default arguments**
4. **Variable length arguments**

## positional arguments:

**__These are the arguments passed to function in correct positional order.**

**Ex:**

```
def sub(a,b)
     print(a-b)
    sub(100,200)
    sub(200,100)
```

****************************Output*********************

```
-100
 100
```

**Note:**
**The number of arguments and position of arguments must be matched. If we change the order then result may be changed.**

**If we change the number of arguments then we will get error.**

## keyword arguments:

**We can pass argument values by keyword i.e by parameter name.**
**Eg:**

```
def  f1(name,msg):
      print("hello",name,msg)

f1(name="sumayya",msg="how are you")

f1(msg="good morning",name="nahid")

**************************************Output**********************************
hello sumayya how are you

hello nahid  good morning
```

**Here the order of arguments is not important but number of arguments must be matched.**

**Important Notes:**

**We can use both positional and keyword arguments simultaneously. But first we have to**
**take positional arguments and then keyword arguments,otherwise we will get syntaxerror.**

```
def  f1(name,msg):
       print("hello",name,msg)

f1("sumayya","how are you")   # valid
f1("nahid", msg="good morning") # valid

f1(name="sumayya","how are you")   # invalid
```

## Default Arguments:

**Sometimes we can provide default values for our positional arguments.**

**Eg:**

```
def wish(name="Guest"):
      print("Hello",name,"Good Morning")

 wish("sumaa")
 wish()
```

**************************************Output************************

**Hello sumaa good morning**
**Hello Guest Good Morning**

**If we are not passing any name then only default value will be considered.**

**Important Notes**

**After default arguments we should not take non default arguments**

**def wish(name="Guest",msg="Good Morning"): ===>Valid**
**def wish(name,msg="Good Morning"): ===>Valid**
**def wish(name="Guest",msg): ==>Invalid**

## Variable length arguments:

**Sometimes we can pass variable number of arguments to our function,such type of arguments are called variable length arguments.**

**We can declare a variable length argument with * symbol as follows**

**def f1(*n):**

**We can call this function by passing any number of arguments including zero number. Internally all these values represented in the form of tuple.**

**Eg:**

```
 def sum(*n):
      total=0
      for n1 in n:
          total=total+n1
      print("The Sum=",total)

sum()
sum(10)
sum(10,20)
sum(10,20,30,40)
```

**********Output**********

```
 The Sum= 0
 The Sum= 10
 The Sum= 30
 The Sum= 100
```

## Note:

**We can mix variable length arguments with positional arguments.**

**Eg:**

```
 def f1(n1,*s):
      print(n1)
      for s1 in s:
           print(s1)

 f1(10)
 f1(10,20,30,40)
 f1(10,"A",30,"B")
```

**********Output**********

```
10
10
20
30
40
10
A
30
B
```

**Note: After variable length argument,if we are taking any other arguments then we should provide values as keyword arguments.**

**Eg:**

```
def f1(*s,n1):
    for s1 in s:
        print(s1)
    print(n1)

f1("A","B",n1=10)
```

**********Output**********

```
A
B
10
```

**f1("A","B",10) ==>Invalid**

**Note: We can declare key word variable length arguments also.**
**For this we have to use **.**

```
def f1(**n):
```

**We can call this function by passing any number of keyword arguments.**
**Internally these keyword arguments will be stored inside a dictionary.**

**Eg:**

```
def display(**kwargs):
    for k,v in kwargs.items():
        print(k,"=",v)
display(n1=10,n2=20,n3=30)
display(rno=100,name="sumayya ",marks=70,subject="python ")
```

**********Output**********

```
n1 = 10
n2 = 20
n3 = 30
rno = 100
name = sumayya
marks = 70
subject = python
```

## Important Points

## Case Study:

```
def f(arg1,arg2,arg3=4,arg4=8):
    print(arg1,arg2,arg3,arg4)
```

f(3,2) ==> 3 2 4 8

f(10,20,30,40) ===>10 20 30 40

f(25,50,arg4=100) ==>25 50 4 100

f(arg4=2,arg1=3,arg2=4)===>3 4 4 2

f()===>Invalid

f(arg3=10,arg4=20,30,40) ==>Invalid

f(4,5,arg2=6)==>Invalid

f(4,5,arg3=5,arg5=6)==>Invalid

**********

# Types of Variables

**Python supports 2 types of variables.**

1. **Global Variables**
2. **Local Variables**

## Global Variables

**The variables which are declared outside of function are called global variables. These variables can be accessed in all functions of that module.**

**Eg:**

**The variables which are declared outside of function are called global variables. These variables can be accessed in all functions of that module.**

**Eg:**

```
a=10 # global variable
def f1():
     print(a)

def f2():
    print(a)

f1()
f2()
```

**********Output**********

```
10
10
```

## Local Variables:

**The variables which are declared inside a function are called local variables. Local variables are available only for the function in which we declared it.i.e from outside of function we cannot access.**

**Eg:**

```
 def f1():
       a=10
       print(a) # valid

def f2():
     print(a) #invalid

f1()
f2()
```

## global keyword:

**We can use global keyword for the following 2 purposes:**
**1. To declare global variable inside function**
**2. To make global variable available to the function so that we can perfor required modifications**

```
 a=10

 def f1():
       a=777
       print(a)

 def f2():
      print(a)

 f1()
f2()
```

**********Output**********

```
777
10
```

**Eg 2:**

```
 a=10
```

```
def f1():
      global a
      a=777
      print(a)

def f2():
      print(a)

 f1()
 f2()
```

**********Output**********

```
777
777
```

**Eg 3:**

```
 def f1():
       a=10
       print(a)

 def f2():
       print(a)

f1()
f2()
```

**********output**********

```
NameError: name 'a' is not defined
```

**Eg 4:**

```
def f1():
      global a
      a=10
     print(a)

def f2():
      print(a)

f1()
f2()
```

**********Output**********

```
10
10
```

**********

# Function Aliasing

**For the existing function we can give another name, which is nothing but function aliasing.**

**Eg:**

```
def wish(name):
    print("Good Morning:",name)

greeting=wish
print(id(wish))
print(id(greeting))

greeting('sumayya')
wish('sumayya')
```

**********Output**********

```
4429336
4429336
Good Morning: sumayya
Good Morning: sumayya
```

**Note: In the above example only one function is available but we can call that function by using either wish name or greeting name.**

**If we delete one name still we can access that function by using alias name**

**Eg:**

```
def wish(name):
    print("Good Morning:",name)

greeting=wish

greeting('sumayya')
wish('sumayya')

del wish

#wish("nahid")➔Name error :name wish is not defind
```

**greeting(‘Arshad’)**

************************************Output************************************

**Good Morning Sumayya**
**Good Morning sumayya**
**Goodd Morning Arshad**