Functions

A function is a **“Block of code”** that performs a specific task that only runs when it is called.
It reduces the code as our program grows larger, and functions make it more organized and manageable. It avoids repetition and makes the code reusable.

Syntax of Function:

```
def Function_name():            #function Definition
        print(“Hi,I’m function”)  #function body
        - - statements - -
        - - statements - -
Function_name()                 #function call
```

Single Parameter function: Parameters are mentioned in the function definition. Actual parameters (arguments)are passed during a function call.

Syntax:

```
def Single_para(a):          #parameter
        print(“This is single parameter:”a)
single_para(100)             #argument passed
```

Multiple parameter functions: We can pass multiple parameters inside a function definition and can give multiple arguments in the function call.

Syntax:

```
def Multi_para(a,b,c):            #multiple parameter
        print(“This is multiple parameter function:”,a,b,c)
Multi_para(100,200,300)           #multiple arguments
```

Keyword Function:

```
def key(name):
        print(“hi”, name)
n=input(“Enter the name”)
key(n)
```

Arbitrary Function:

```
def Multi_argument(*a):
        print(“this is multi-argument function”,a)
Multi_argument(1,2,3,4,5,6)

def Key(**a):
        print(“this is keyword function”)
Key(name=”bob”,age=28)
```

Nested Function:

```
def Outer_func():
    print("I'm Outer function")
    def inner_func():
        print("I'm inner function")
    inner_func()
Outer_func()
```

Advanced Functions

Lambda function: A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

```
f=lambda a,b:a+b
f(2,3)
```

Map Function: Map() returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable(list, tuple).

Syntax: map(function,iterator)

```
def calculate(n):
    return n+n
numbers=(1,2,3,4)
result=map(calculate,number)
print(list(result))
```

Filter function: filter() filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

Syntax: filter(function,sequence)

```
ages=[5,12,17,18,24,32]
def func(x):
    If x<18:
       return False
    else:
       return True
adults=filter(func,ages)
print(list(adults))
```

Reduce function: A reduce() as the name describes applies a given function to the iterable and return a single value.

Syntax: reduce(function,sequence)

```
From functools import reduce
d=reduce(lambda a,b:a+b,[23,31,45,98])
print(d)
```

Generator(): The generator function is defined like a normal function but whenever it needs to generate a value, it does so with the yield keyword rather than return. if the body of a def contains yield, the function automatically becomes a generator function.

```
def simplegenerator():
        yield 1
        yield 2
        yield 3
x=simplegenerator()
print(x.__next__())
print(x.__next__())
print(x.__next__())
```