**FUNCTION**

A function is a block of code or statements that performs a special task.

The main advantage of functions is code Reusability. Once we define the statements as single unit and we can call that unit any number of times based on our requirement without rewriting.

**ADVANTAGES OF FUNCTIONS**

1. Avoids rewriting the code again and again.
2. Length of the source code is reduced.
3. Saves memory space.
4. Easy to test and debug.
5. Clearly understands.

Python supports 2 types of functions:

1. Built in Functions
2. User Defined functions

**Built-in Functions:** The functions which are coming along with Python software automatically are called built in functions.

**Example:** id( ), type( ), input( ), eval( ).. etc..

**User Defined Functions:** The functions which are developed by programmer explicitly according to business requirements are called user defined functions.

**Syntax:**

**def function\_name(parameters):**

**documentation string**

**statement block**

**return value**

To define the function, we follow simple rules

* Function blocks begin with the keyword **def** followed by the function name and parentheses ( ( ) ).
* After the parentheses colon is placed.
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
* The code block within every function starts with a colon (:) and is indented.
* A function may have return statement or not.

While creating functions we use 2 keywords

1. def (mandatory)
2. return (optional)

**Example:**

|  |  |
| --- | --- |
| def wish():  print("hello cnu reddy")  wish()  wish() | def fun():  for i in range(1,21,2):  print(i,end=" ")  fun() |

## Calling a Function

Defining a function gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt

**Syntax:**

**Function\_name(parameters)**

**Parameters:**

Parameters are inputs to the function. If a function contains parameters then, at the time of

calling, compulsory we should provide values, otherwise, we get error.

**Example:**

|  |  |
| --- | --- |
| def wish(name):  print(" Hello ",name,"Good Morning ")  wish("cnu ")  wish("Ashwika ")  wish("Nitiksha ") | def swap(a,b):  a,b=b,a  print(a,b)  a=10  b=20  swap(a,b) |

**Return statement:**

Return statement is used to return a value from a user-defined function to its calling point.

**Syntax:**

return(10)

return(a+b)

**Example:**

|  |  |
| --- | --- |
| def fun():  return(20)  r=fun()  print(r) | def fun():  return("cnu reddy")  print(fun()) |

**Returning multiple values from a function:**

In other languages like C, C++, Java, functions can return one value. But in Python a function can return any number of values.

def sum\_sub(a,b):

sum=a+b

sub=a-b

mul=a\*b

div=a/b

return sum,sub,mul,div

t=sum\_sub(24,3)

for i in t:

print(i)

**Example:**

def sum\_sub(a,b):

sum=a+b

sub=a-b

return sum,sub

x,y=sum\_sub(24,3)

print(" The summ is ",x)

print(" The sub is ",y)

**FUNCTION ARGUMENTS**

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.

**Example:**

def sub(a,b):

print(a-b)

sub(100,200)

sub(200,100)

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.

**Example:**

def wish(name,msg):

print("Hello ",name, msg)

wish(name="sreenu ",msg="Good morning ")

wish(name="Good morning", msg=" sreenu ")

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

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

**Example:**

def wish(name,msg):

print("Hello ",name,msg)

wish("cnu","Good morning")

wish("cnu",msg="good morning")

**Default arguments:**

Sometimes we can provide default values for our positional arguments. The default value to an argument is provided by using the assignment operator(=).

**Example:**

def details(name, place, course="B.Sc"):

print("name:",name," Place: ",place," course: ",course)

details("sreenu reddy","badvel")

details(name="ashwika",place="kadapa",course="B.A")

**Example:**

def wish(name="Guest"):

print(" Hello ",name,"Good morning")

wish("Durga")

wish()

**Variable length arguments:**

In some situations, it is not known in advance how many arguments will be passed to a function. In such cases, Python allows programmers to make function calls with arbitrary (or any) number of arguments. When we use arbitrary arguments or variable length arguments, then the function definition uses an asterisk (\*) before the parameter name.

**Example:**

def fun(name,\*sub):

print(name,"likes to read")

for subject in sub:

print(subject)

fun("sreenu reddy ","C","DATA STRUCTURES","C++","JAVA","PYTHON")

fun("Nitiksha Reddy")

**Example:**

def fun(n,\*s):

print(n)

for n1 in s:

print(n1)

fun(10)

fun(20,"A","B","C")

fun(30,"!","@","%")

def sum(\*n):

total=0

for n1 in n:

total=total+n1

print(" The sum is ",total)

sum()

sum(1,3)

sum(3,5,8,9)

sum(1,2,3,4,5,6,7,8)

**TYPES OF VARIABLES**

Python supports 2 types of variables

1. Global variables
2. Local variables

**Global Variable:**

The variables which are declared outside of function are called global variables. These variables can be accessed in all functions of that module. As a good programming habit, try to avoid the use of global variables because they may get altered by mistake and then result in erroneous output.

**Local Variable:**

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.

**Example:**

num1=10

print("Global :",num1)

def fun1():

num2=20

print("Local value",num2)

print("Global value",num1)

fun1()

**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 perform required modification.

**Example:**

b=10

def fun():

global a

a=32

print(b)

print(a)

def fun1():

print(b)

print(a)

fun()

fun1()

**RECURSIVE FUNCTIONS**

A function that calls itself is known as Recursive functions

Ex:

Fact(3)=3\*fact(2)

=3\*2\*fact(1)

=3\*2\*1\*fact(0)

=3\*2\*1\*1

=6

Advantages of recursive functions are

* We can reduce length of the code and improve readability
* We can solve complex problems very easily

**RECURSIVE PROGRAM**

def fact(n):

if(n==0):

result=1

else:

result=n\*fact(n-1)

return result

print("Factorial of 4 is",fact(4))

print("Factorial of 7 is",fact(7))

**ANONYMOUS FUNCTIONS/ LAMBDA**

Sometimes we declare a function without any name. Such type of nameless functions is called anonymous functions or lambda functions.

Main purpose of anonymous function is just for instant use.(i.e for one time usage)

**Lambda function**

We can define by using lambda keyword

Lambda n:n\*n

**Syntax:**

Lambda argument: expression

**Examples:**

s=lambda n:n\*n

print("Square of 4 is",s(4))

print("Square of 4 is",s(5))

s=lambda a,b:a+b

print("The sum of 4,5 is",s(4,5))

print("The sum of 7,3 is",s(7,3))

s=lambda a,b:a if a>b else b

print("The biggest of 34,43 is",s(34,43))

We use lambda with functions like filter(), map(), and reduce() functions because these functions expect function as argument.

**Filter function()**

We use filter function to filter values from the given sequence based on some condition.

filter(function,sequence)

**Program to filter only even numbers from the list by using filter() function?**

**Without lambda function**

def iseven(x):

if x%2==0:

return True

else:

return False

L=[1,2,3,4,5,6,7,8,9,10]

L1=list(filter(iseven,L))

print(L1)

**With Lambda Function**

L=[1,2,3,4,5,6,7,8,9,10]

L1=list(filter(lambda x:x%2==0,L))

print(L1)

**Map Function()**

For every element present in the given sequence apply some functionality and generate new element with the required modification. For this requirement we should go for map() function.

**Syntax:**

**map(function, sequence)**

**Ex:**

**Without lambda**

L=[1,2,3,4,5]

def doublet(x):

return 2\*x

L1=list(map(doublet,L))

print(L1)

**with lambda**

L=[1,2,3,4,5]

L2=list(map(lambda x:x\*2,L))

print(L2)

**Reduce() function:**

Reduce sequence of elements into a single element by applying the specified function.

**reduce(function,sequence)**

**Example:**

from functools import \*

result=reduce(lambda x,y:x+y,range(1,10))

print(result)

**MODULE**

A group of functions, variables and classes saved to a file, which is nothing but a module. Every Python file(.py) acts as module.

**Example:**

**Create a file testmob.py**

x=789

def add(a,b):

print("The sum=",a+b)

def product(a,b):

print("The product=",a\*b)

testmob module contains one variable and 2 functions.

If we want to use members of module in another program then we should import that module.

**Import modulename**

We can access members by using module name.

**Modulename.variable()**

**Modulename.function()**

**Output**

789

The sum= 7

The product= 15

**testmod1.py**

import testmod

print(testmod.x)

testmod.add(3,4)

testmod.product(3,5)

Renaming a module at the time of import (module aliasing):

**Ex:**

Import testmod as m

Here testmod is original module name and m is alias name. We can access members by using alias name m.

**WORKING WITH MATH MODULE**

Python provides inbuilt module math.

This module defines several functions which can be used for mathematical operations.

Important functions are:

1. sqrt(x)
2. ceil(x)
3. floor(x)
4. fabs(x)
5. log(x)
6. sin(x)
7. tan(x)

**Example:**

from math import \*

print(sqrt(4))

print(ceil(10.4))

print(floor(10.6))

print(fabs(-33))

print(sin(45))

**WORKING WITH RANDOM MODULE**

This module defines several functions to generate random numbers. We use these functions while developing games, in cryptography and to generate random numbers on fly for authentication.

**random() function:**

This function generates some float values between o and 1(not inclusive)

**Ex:**

from random import \*

for i in range(1,10):

print(random())

**randint() function:**

To generate random integer between two given numbers (inclusive)

**Ex:**

from random import \*

for i in range(10):

print(randint(1,100))