**Default arguments**:

* At the time of function definition we will give input arguments.
* Example for default arguments is range() function,it is a builtin function.
* Range function has default argument i.e,starting value=1,ending value and step value.
* If we doesn’t provide input to the arguments through function call which are declared at the function definition default arguments are used for performing operations.

**Example for default arguments by taking user input:**

def cake(flav,wt,shape):

print("you have ordered cake of flavour",flav,"of weight",wt,"of shape",shape)

flav=input('enter flavour')

wt=input('enter wt')

shape=input('enter shape')

cake(flav,wt,shape)

output:enter flavour chocolate

enter wt 2kg

enter shape round

you have ordered cake of flavour chocolate of weight 2kg of shape round

**Example for default arguments by passing values at the time of function definition:**

def cake(flav=’vanilla’,wt=’3kgs’,shape=’rect’):

print("you have ordered cake of flavour",flav,"of weight",wt,"of shape",shape)

cake(flav,wt,shape)

the output is:you have ordered cake of flavour vanilla of 3kgs of shape rect.

**Variable arguments:**

* It doesnot require method overloading.
* It works for 1 to n arguments.
* These are the extension of normal arguments.
* In this the arguments are stored in list manner.
* For this python uses

\*args

\*vars

\*params

**Example for variable arguments:**

**def fun(\*args):**

**sum=0**

**for i in k:**

**sum=sum+i**

**avg=sum//(len(args))**

**print(avg)**

**fun(3,3,4,6)**

**output for this program is 4.**

**Eg-2:**

**def fun(a,b,\*args):**

**sum=a+b**

**for i in k:**

**sum=sum+i**

**avg=sum//(len(args)+2)**

**print(avg)**

**fun(a=3,b=3,4,6)**

**output for this program is 4.**

**Keyword arguments:**

* The double form of the astrick \*\*kwargs is used to pass dictionaries.
* Dictionaries contain keys and values.
* Kwvars is also used to pass values.

**Eg:**

**def fun(a,b,\*\*kwargs):**

**sum=a+b**

**k=kwargs.values()**

**for i in k:**

**sum=sum+i**

**avg=sum//(len(kwargs)+2)**

**print(avg)**

**fun(a=2,b=3,c=4,d=5)**

**:modules:**

* Modules are 2 types.

1.inbuilt modules

2.userdefined modules

* Inbuilt modules are

1. Sys
2. Os
3. Random
4. Math
5. Time
6. Calculaor

* Inbuilt modules are accessed by importing corresponding module.
* There are 143 inbuilt modules in python 3.

**Math module:**

* It provides access to the mathematical functions defined by the C standard.**.**
* This can be done by importing math module.
* These functions cannot be used with complex numbers.

**Eg:import math**

Print(math.factorial(*x*))

Return *x* factorial. Raises [ValueError](https://docs.python.org/3/library/exceptions.html#ValueError) if *x* is not integral or is negative.

**Random module:**

This module implements pseudo-random number generators for various distributions.

Eg:

import random

a=random.randint(1,10)

print('random integer no:',a)

**os module:**

* Programs that import and use os module stand a better chance of being portable between different platforms.
* they must the only use functions that are defined by all platforms (e.g., unlink and opendir), and leave all pathname manipulation to os.path (e.g., split and join).

**Eg:**

Import os

Os.mkdir(‘dummy’)

**Time module:**

* There are two standard representations of time.
* One is the number of seconds since the Epoch, in UTC (a.k.a. GMT).
* It may be an integer or a floating point number (to represent fractions of seconds).
* The Epoch is system-defined; on Unix, it is generally January 1st, 1970.
* The actual value can be retrieved by calling gmtime(0)

Eg:

Import time

From time import \*

Print(time.time())

Print(time.localtime())

**Calendar module:**

* when comparing these calendars to the ones printed by cal(1): By default, these calendars have Monday as the first day of the week, and Sunday as the last (the European convention).
* Use setfirstweekday() to set the first day of the week (0=Monday,6=Sunday).

Eg:

Import calendar

From calendar import \*

Print(calendar.calendar(2018))

**Sys module**:

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

**User defined Modules:**

* Modules refer to a file containing Python statements and definitions.
* We use modules to break down large programs into small manageable and organized files.
* modules provide reusability of code.
* We can define our most used functions in a module and import it, instead of copying their definitions into different programs.
* We use the import keyword to do this.
* To import our previously defined module.
* We can import specific names from a module without importing the module as a whole.
* By using ‘from’ keyword we can do this.
* For example if we implement code code adding of numbers in one module it can be used in another module by importing main module and then call function.

Eg:

def fun(l,char):

for i in l:

if(i==char):

print(l.index(i))

* this is the main module code for printing index of a specified character.
* This program canbe saved as mainmod.py.
* This code can be used in different modules just by passing list and calling function.
* While using this code import mainmod.

import mainmod

from mainmod import \*

l=[1,2,3,4]

fun(l,3)