Chapter-Functions and Modules

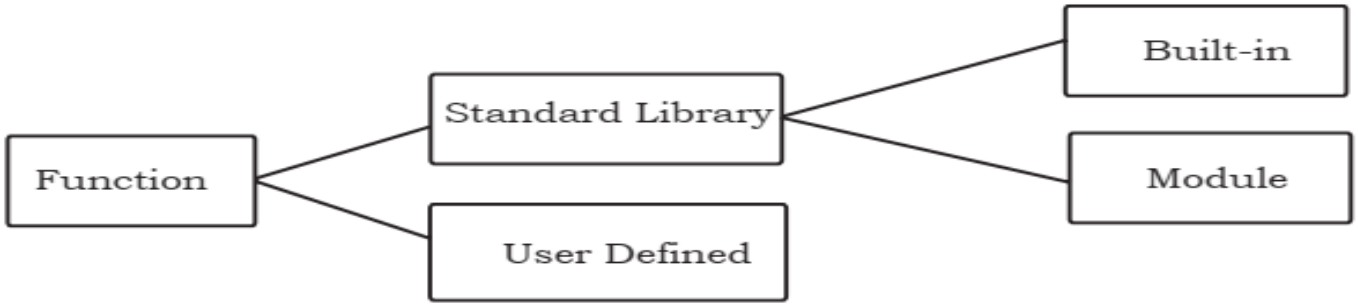
Introduction

- Function can be defined as a named group of instructions that accomplish a specific task when it is invoked. Once defined, a function can be called repeatedly.

Advantages of using Functions

- Increases readability, particularly for longer code as by using functions, the program is better organised and easy to understand.
- Reduces code length as same code is not required to be written at multiple places in a program. This also makes debugging easier.
- Increases reusability, as function can be called from another function or another program. Thus, we can reuse or build upon already defined functions and avoid repetitions of writing the same piece of code.
- Work can be easily divided among team members and completed in parallel.

Types of Functions



- **Build-in Functions:** Python has many predefined functions called built-in functions. These functions can be easily used in a Python program by importing the module using import command. Use of built-in functions makes programming faster and efficient.

Build-in Functions

Input/ Output	Datatype Conversion	Mathematical Functions	Other Functions
input()	bool()	abs()	__import__()
print()	chr()	divmod()	len()
	dict()	max()	range()
	float()	min()	type()
	int()	pow()	
	list()	sum()	
	ord()		
	set()		
	str()		
	tuple()		

User-defined Functions

- In addition to the standard library functions, we can define our own functions while writing the program. Such functions are called user defined functions.
- A function definition begins with def (short for define). The syntax for creating a user defined function is as follows:

```
def<Function name> ([parameter 1, parameter 2,....]):
    set of instructions to be executed
    [return <value>]
```

Function Header

Function Body (Should be ndented within the function header)

User- defined Functions (contd.)

- Example: Write a user defined function to add 2 numbers and display their sum.

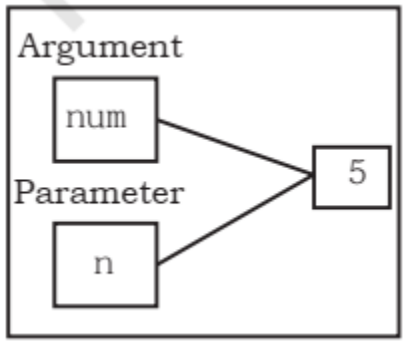
```
#function definition

def addnum():
fnum = int(input("Enter first number: "))
snum = int(input("Enter second number: "))
sum = fnum + snum
print("The sum of ",fnum,"and ",snum,"is ",sum)

#function call
addnum()
```

Arguments and Parameters

- An argument is a value passed to the function during the function call which is received in corresponding parameter defined in function header.
- Example: Write a program using a user defined function that displays sum of first *n* natural numbers, where *n* is passed as an argument.




```
def sumSquares(n): #n is the parameter
sum = 0
for i in range(1,n+1):
sum = sum + i
print("The sum of first",n,"natural numbers is: ",sum)
num = int(input("Enter the value for n: "))
#num is an argument referring to the value input by the user
sumSquares(num) #function call
```

String as Parameters

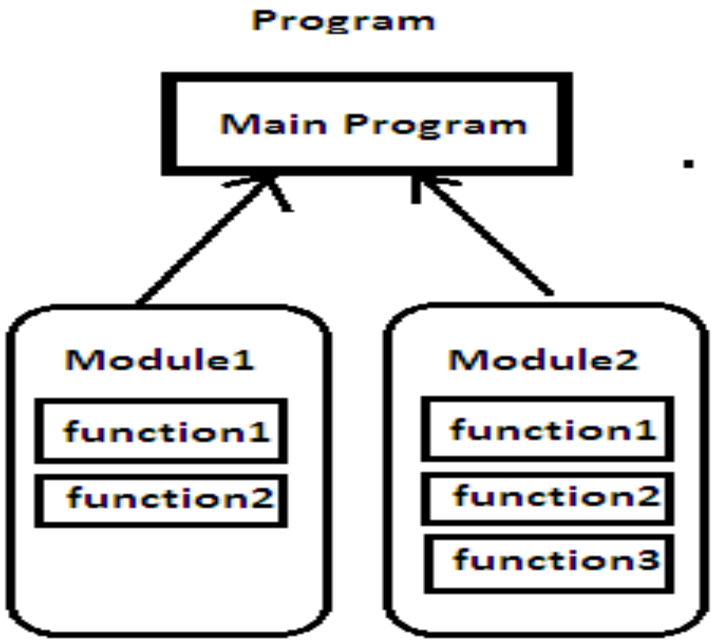
```
def fullname(first,last):
#+ operator is used to concatenate strings
fullname = first + " " + last
print("Hello",fullname)
#function ends here
frst = input("Enter first name: ")
last = input("Enter last name: ")
#function call
fullname(frst,last)
```

Default Parameters

- Python allows assigning a default value to the parameter. A default value is a value that is predefined and assigned to the parameter when the function call does not have its corresponding argument.
- **Function returning values:** A function may or may not return a value when called.
- When function does not return value they are known as void functions. The return statement returns the values from the function.

Module

- A module is a logical organization of Python code. Related code are grouped into a module which makes the code easier to understand and use. Any python module is an object with different attributes which can be bind and referenced.



Build-in Modules

- Python library has many built-in modules that are really handy to programmers.

Commonly used modules:

- Math
- Random
- Statistics
- **Math:** It contains different types of mathematical functions. Most of the functions in this module return a float value.

math.ceil(x)	x may be an integer or floating point number	ceiling value of x	`>>> math.ceil(-9.7)`
			`-9`
			`>>> math.ceil (9.7)`
			`10`
			`>>> math.ceil(9)`
			`9`

math.floor(x)	x may be an integer or floating point number	floor value of x	`>>> math.floor(-4.5)`
			`-5`
			`>>> math.floor(4.5)`
			`4`
			`>>> math.floor(4)`
			`4`

Math module (contd.)

math.fabs(x)	x may be an integer or floating point number	absolute value of x	>>> math.fabs(6.7)
			6.7
			>>> math.fabs(-6.7)
			6.7
			>>> math.fabs(-4)
			4.0

math.factorial(x)	x is a positive integer	factorial of x	>>> math.factorial(5) 120

math.fmod(x,y)	x and y may be an integer or floating point number	x % y with sign of x	>>> math.fmod(4,4.9)
			4.0
			>>> math.fmod(4.9,4.9)
			0.0
			>>> math.fmod(-4.9,2.5)
			-2.4
			>>> math.fmod(4.9,-4.9)
			0.0

math.gcd(x,y)	x, y are positive integers	gcd (greatest common divisor) of x and y	>>> math.gcd(10,2) 2

math.pow(x,y)	x, y may be an integer or floating point number	x^y (x raised to the power y)	>>> math.pow(3,2)
			9.0
			>>> math.pow(4,2.5)
			32.0
			>>> math.pow(6.5,2)
			42.25
			>>> math.pow(5.5,3.2)
			233.97

math.sqrt(x)	x may be a positive integer or floating point number	square root of x	>>> math.sqrt(144)
			12.0
			>>> math.sqrt(.64)
			0.8

math.sin(x)	x may be an integer or floating point number in radians	sine of x in radians	>>> math.sin(0)
			0
			>>> math.sin(6)
			-0.279

Random module

- The random module provides access to functions that support many operations. Perhaps the most important thing is that it allows us to generate random numbers.

random.random()	No argument (void)	Random Real Number (float) in the range 0.0 to 1.0	`>>> random.random()` `0.65333522`

random. randint(x,y)	x, y are integers such that x <= y	Random integer between x and y	`>>> random.randint(3,7)`
			`4`
			`>>> random.randint(-3,5)`
			`1`
			`>>> random.randint(-5,-3)`
			`-5.0`

random. randrange(y)	y is a positive integer signifying the stop value	Random integer between 0 and y	`>>> random.randrange(5)` `4`
random. randrange(x,y)	x and y are positive integers signifying the start and stop value	Random integer between x and y	`>>> random.randrange(2,7)` `2`

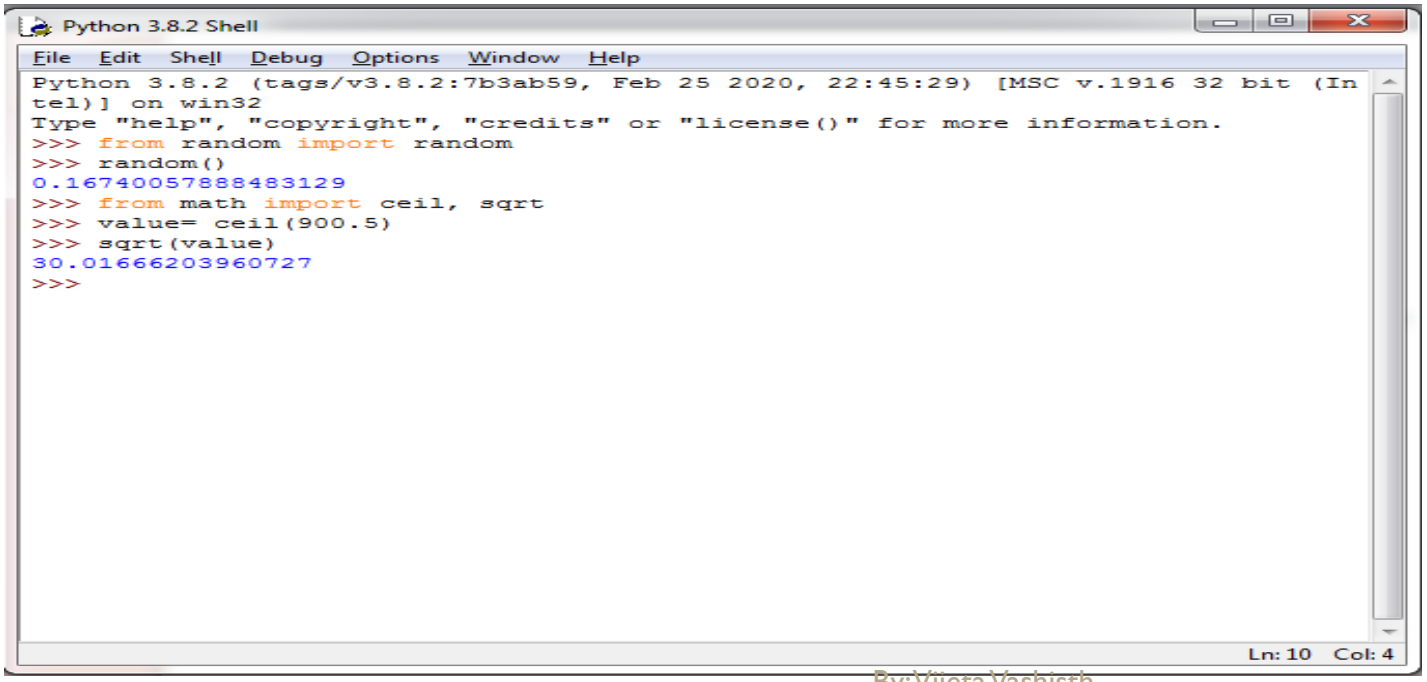
Statistics Module

- This module provides functions for calculating mathematical statistics of numeric (Real-valued) data.

statistics.mean(x)	x is a numeric sequence	arithmetic mean	>>> statistics. mean([11,24,32,45,51]) 32.6
statistics.median(x)	x is a numeric sequence	median (middle value) of x	>>>statistics. median([11,24,32,45,51]) 32
statistics.mode(x)	x is a sequence	mode (the most repeated value)	>>> statistics. mode([11,24,11,45,11]) 11 >>> statistics. mode(("red","blue","red")) 'red'

From Statement

- Instead of loading all the functions into memory by importing a module, from statement can be used to access only the required functions from a module. It
loads only the specified function(s) instead of all the functions in a module. Its syntax is
>>> from modulename import functionname [, functionname, ...]



Question Session