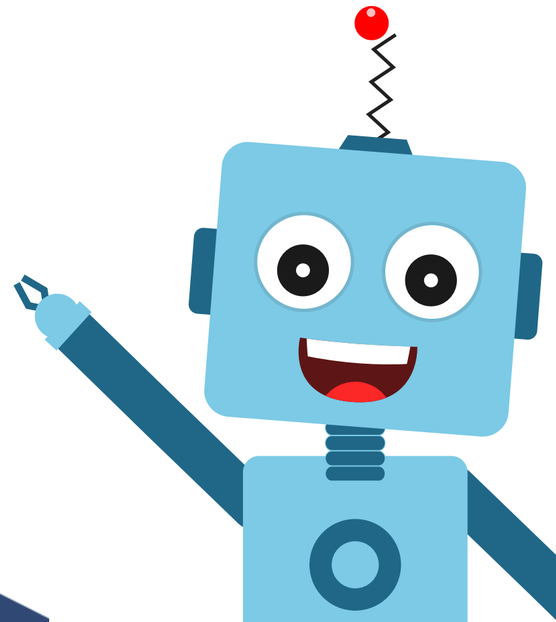
POWERED BY STEMpedia

# Functions in Python

## Session 5



# Topics covered

POWERED BY STEMpedia

1. Mathematical functions in Python
2. Built in functions
3. User defined functions
4. Activity : Properties of cube

# Mathematical functions in python

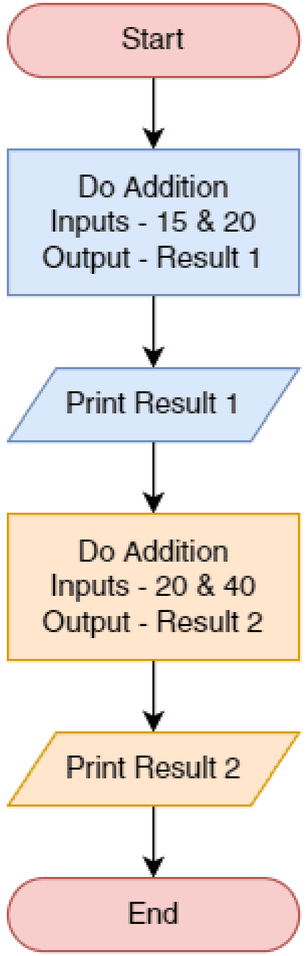
To use mathematical functions, we use math library which can be imported like this: import math.

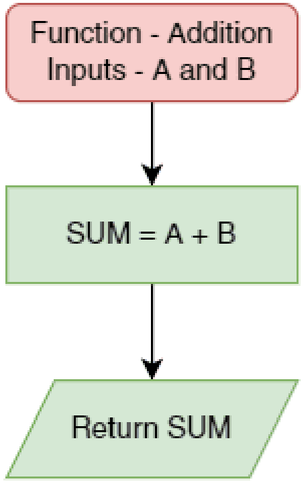
# Functions

A function is a block of code made up of a set of steps that results in a single specific action.

Functions can be categorized as belonging to :

- Modules
- Built-in
- User-Defined



# Module

POWERED BY STEMpedia

- A module is a file containing Python definitions (i.e. functions) and statements.
- Definitions from the module can be used within the code of a program.
- Once you import a module, you can reference (use), any of its functions or variables in your code.
- Import is the simplest and most common way to use modules in our code. Its syntax is:

  **import modulename1 [,modulename2, ———]**

# Module

POWERED BY STEMpedia

**Example :**

```
import math
```

On execution of this statement, Python will

- Search for the file “math.py“.
- Create space where modules definition & variable will be created,
- then execute the statements in the module.

To use/ access/invoke a function, you will specify the module name and name of the function- separated by a dot (.). This format is also known as dot notation.

```
value= math.sqrt (25) # dot notation
```

# Math Module

POWERED BY STEMpedia

| Name of the function | Description | Example |
|---|---|---|
| ceil( x ) | It returns the smallest integer not less than x, where x is a numeric expression. | math.ceil(-45.17)<br>>>-45.0<br>math.ceil(100.12)<br>>>101.0<br>math.ceil(100.75)<br>>>101.0 |
| floor( x ) | It returns the largest integer not greater than x, where x is a numeric expression. | math.floor(-45.17)<br>>>-46.0<br>math.floor(100.12)<br>>>100.0<br>math.floor(100.75)<br>>>100.0 |
| fabs( x ) | It returns the absolute value of x, where x is a numeric value. | math.fabs(-45.17)<br>>>45.17<br>math.fabs(100.12)<br>>>100.12<br>math.fabs(100.75)<br>>>100.75 |

# Math module

POWERED BY STEMpedia

| | | |
|---|---|---|
| exp( x ) | It returns an exponential of x: ex, where x is a numeric expression. | math.fabs(-45.17)<br>>>2.41500621326e-20<br>math.fabs(100.12)<br>>>3.03084361407e+43<br>math.fabs(100.72)<br>>>5.52255713025e+43 |
| log( x ) | It returns the natural logarithm of x, for x > 0, where x is a numeric expression. | math.log(100.12)<br>>>4.60636946656<br>math.log(100.72)<br>>>4.61234438974 |
| log10( x ) | It returns base-10 logarithm of x for x > 0, where x is a numeric expression. | math.log(100.12)<br>>>2.00052084094<br>math.log(100.72)<br>>>2.0031157171 |

# Math Module



| | | |
|---|---|---|
| pow( x, y ) | It returns the value of x to the power y, where x and y are numeric expressions. | math.pow(100, 2)<br>>>10000.0<br>math.pow(100, -2)<br>>>0.0001<br>math.pow(2, 4)<br>>>16.0<br>math.pow(3, 0)<br>>>1.0 |
| sqrt( x ) | It returns the square root of x for x > 0, where x is a numeric expression. | math.sqrt(100)<br>>>10.0<br>math.sqrt(7)<br>>>2.64575131106 |
| cos( x ) | It returns the cosine of x in radians, where x is a numeric expression | math.cos(3)<br>>>-0.9899924966<br>math.cos(-3)<br>>>-0.9899924966<br>math.cos(0)<br>>>1.0<br>math.cos(math.pi)<br>>>-1.0 |

# Math Module



| | | |
|---|---|---|
| sin( x ) | It returns the sine of x, in radians, where x must be a numeric value. | math.sin(3)<br>>>0.14112000806<br>math.sin(-3)<br>>>-0.14112000806<br>math.sin(0)<br>>>0.0 |
| tan( x ) | It returns the tangent of x in radians, where x must be a numeric value. | math.tan(3)<br>>>-0.142546543074<br>math.tan(-3)<br>>>0.142546543074<br>math.tan(0)<br>>>0.0 |
| degrees( x ) | It converts angle x from radians to degrees, where x must be a numeric value. | math.degrees(3)<br>>>171.887338539<br>math.degrees(-3)<br>>>-171.887338539<br>math.degrees(0)<br>>>0.0 |

# Math Module

POWERED BY STEMpedia

| | | |
|---|---|---|
| radians( x ) | It converts angle x from degrees to radians, where x must be a numeric value. | math.radians(3)<br>>>0.0523598775598<br>math.radians(-3)<br>>>-0.0523598775598<br>math.radians(0)<br>>>0.0 |

# Built in Functions

**Built-in functions are the function(s) that are built into Python and can be accessed by a programmer**

# Built in functions



Built-in functions are the function(s) that are built into Python and can be accessed by a programmer.

| Name | Description | Example |
|---|---|---|
| abs( x ) | It returns the distance between x and zero, where x is a numeric expression. | abs(-45)<br>>>45<br>abs(119)<br>>>119 |
| max( x, y, z, .... ) | It returns the largest of its arguments: where x, y, and z are numeric variables/expressions. | max(80, 100, 1000)<br>>>1000<br>max(-80, -20, -10)<br>>>-10 |
| min( x, y, z, .... ) | It returns the smallest of its arguments; where x, y, and z are numeric variables/expressions. | min(80, 100, 1000)<br>>>80<br>min(-80, -20, -10)<br>>>-80 |

# Built in functions

POWERED BY STEMpedia

| | | |
|---|---|---|
| cmp( x, y ) | It returns the sign of the difference of two numbers: -1 if x < y, 0 if x == y, or 1 if x > y, where x and y are numeric variable/expression. | cmp(80, 100)<br>>>-1<br>cmp(180, 100)<br>>>1 |
| round( x [, n] ) | It returns float x rounded to n digits from the decimal point, where x and n are numeric expressions. If n is not provided then x is rounded to 0 decimal digits. | round(80.23456, 2)<br>>>80.23<br>round(-100.000056, 3)<br>>>-100.0<br>round (80.23456)<br>>>80.0 |

# Composition



- Composition is the art of combining simple function(s) to build more complicated ones, i.e., the result of one function is used as the input to another.

**Example :**

Suppose we have two functions fn1 & fn2, such that

a= fn2 (x)

b= fn1 (a)

then call to the two functions can be combined as

b= fn1 (fn2 (x))

# User Defined Functions

POWERED BY STEMpedia

- To define a function keyword **def** is used.
- After the keyword comes an identifier i.e. name of the function, followed by a parenthesized list of parameters and the colon which ends up the line.
- Next follows the block of the statement(s) that are part of the function.

**Block of statements:**

A block is one or more lines of code, grouped together so that they are treated as one big sequence of statements while executing.

# Syntax of function



**def NAME ([PARAMETER1, PARAMETER2, …..]):** #Square brackets include

**statement(s)** #optional part of statement

**Example:**

```
def sayHello (): # Line No. 1
  print “Hello World!” # Line No. 2
```

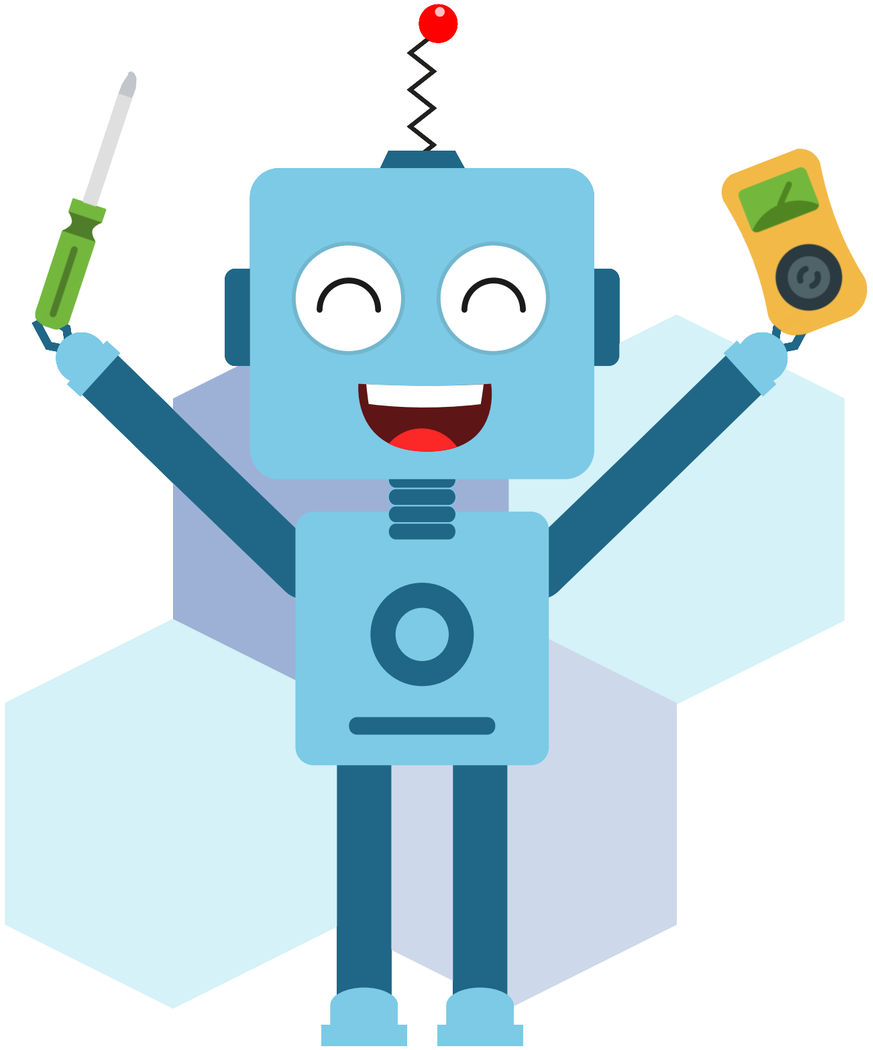
- The first line of the function definition, i.e., Line No. 1 is called the **header**, and the rest, i.e. Line No. 2 in our example, is known as the body.
- The name of the function is **sayHello**, and empty parenthesis indicates no parameters.
- The body of the function contains one Python statement, which displays a string constant on the screen.

# ACTIVITY

POWERED BY STEMpedia

## Properties of a cube

**Let’s create user define functions to find out surface area and volume of cube.**



# Parameters and Arguments

POWERED BY STEMpedia

- **Parameters** are the **value(s) provided in the parenthesis when we write the function header**. These are the values required by the function to work.

- **Arguments** are the value(s) provided in function call/invoke statement. A list of arguments should be supplied in the same way as parameters are listed.

- Bounding of parameters to arguments is done 1:1, and so there should be the same number and type of arguments as mentioned in the parameter list.

# Let's Code

POWERED BY STEMpedia

- Define the sprite object.

```
sprite = Sprite('Tobi')
```

- Define the Surface Area function. The function will have 1 parameter as the length.

```
def surfaceAreaCube(length):
  # Calculates the surface area of the cube
  surfaceArea = 6*length*length
  return surfaceArea
```

# Let's Code

POWERED BY STEMpedia

- Define the Volume function. The function will have 1 parameter as the length.

```
# Calculates the volume of the cube
def volumeCube(length):
  volume = length*length*length
  return volume
```

- Ask the user to enter the length of the cube.

```
sprite.input("Enter the side length")
l = int(sprite.answer())
```

# Let's Code

POWERED BY STEMpedia

- Display the surface area and the volume

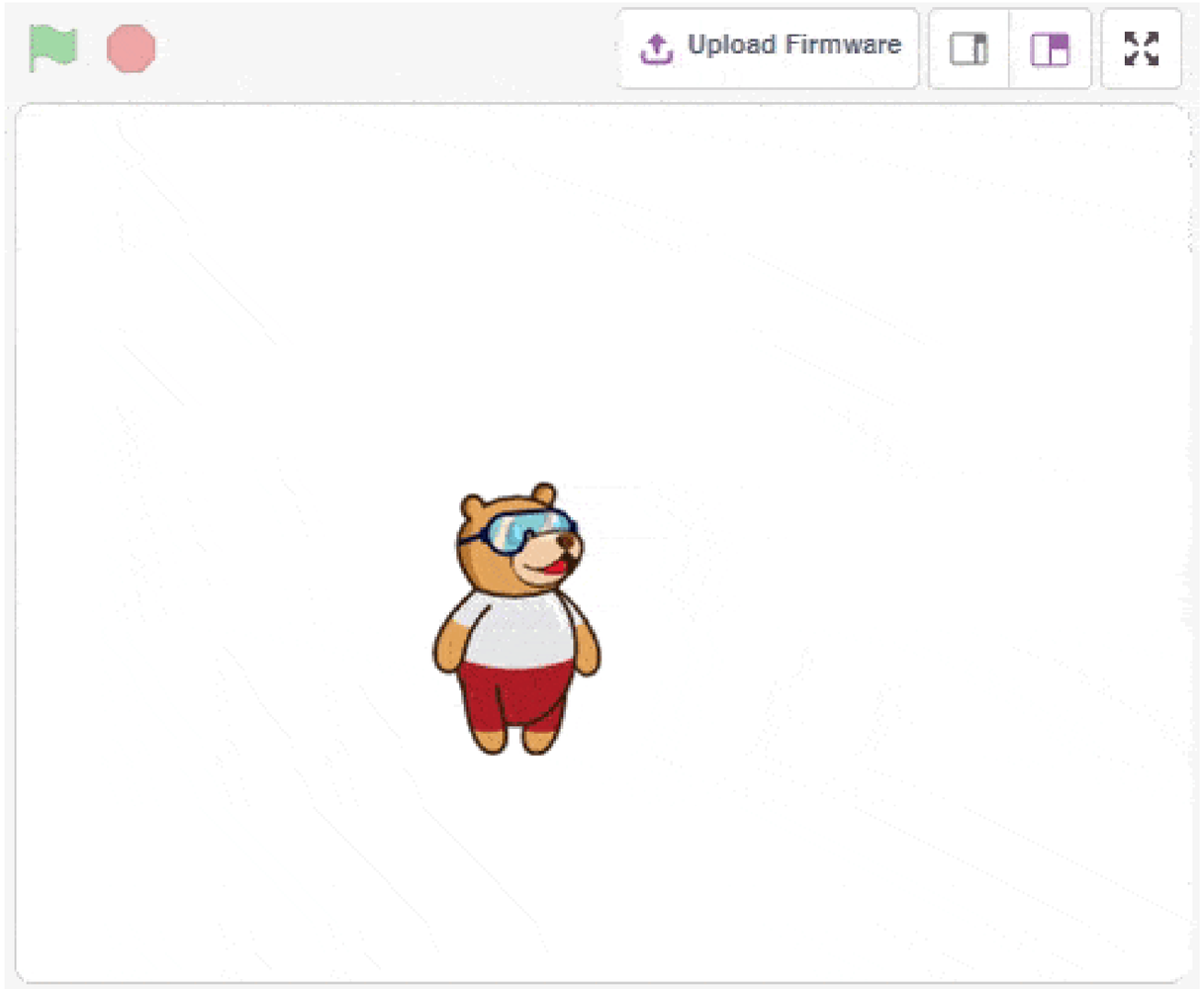
```
sprite.say("Surface Area is " + str(surfaceAreaCube(l)), 2)
sprite.say("Volume is " + str(volumeCube(l)), 2)
```

# Properties of a Cube(Final Code)

POWERED BY STEMpedia

```
sprite = Sprite('Tobi')
def surfaceAreaCube(length):
  # Calculates the surface area of the cube
  surfaceArea = 6*length*length
  return surfaceArea
def volumeCube(length):
  # Calculates the volume of the cube
  volume = length*length*length
  return volume
sprite.input("Enter the side length")
l = int(sprite.answer())
sprite.say("Surface Area is " + str(surfaceAreaCube(l)), 2)
sprite.say("Volume is " + str(volumeCube(l)), 2)
```

# Final Output



# Right Angle Tringle

POWERED BY STEMpedia

- First, we import the math module, which contains mathematical functions and constants that can be used in the code.

```
import math
```


- First, we import the math module, which contains mathematical functions and constants that can be used in the code.


- Then we define a function named **calarea** that takes two parameters: base and height. This function calculates the area of a right-angled triangle using the formula 0.5*base*height and returns the result.

```
def calarea(base,height):
    return(0.5*base*height)
```

- Further we define a function named hyp that takes two parameters: base and height. This function calculates the hypotenuse of a right-angled triangle using the Pythagorean theorem, which states that the square of the hypotenuse is equal to the sum of the squares of the other two sides.

# Right Angle Tringle



- Then we calculate the hypotenuse using the Pythagorean theorem. It first multiplies the values of base and height by themselves using the * symbol, adds the two resulting values together using the + symbol, and then takes the square root of the sum using the function. The result is stored in the variable var.
- Then return is write to specifies the value that the hyp function will return. It returns the value of the var variable, which represents the hypotenuse of the triangle.
- Further the user to enter a value for the length of the base of the triangle using the input() function, which takes the user's input as a string. The int() function is used to convert the user's input from a string to an integer data type, and the resulting value is assigned to the variable base.

# Right Angle Tringle



- Then the user enters a value for the height of the triangle using the function, which takes the user's input as a string. The **int()** function is used to convert the user's input from a string to an integer data type, and the resulting value is assigned to the variable height.
- Then we call the calarea function with the values of base and height passed as arguments, and assigns the result to the variable area.

```
#calling the functions
area=calarea(base,height)
hypotenuse=hyp(base,height)
```

- Then we use print statement to print area and hypotenuse of tringle.

```
print("The area of the given triangle is {}cm^2".format(area))
print("The hypotenuse of the given triangle is {}cm".format(hypotenuse))
```

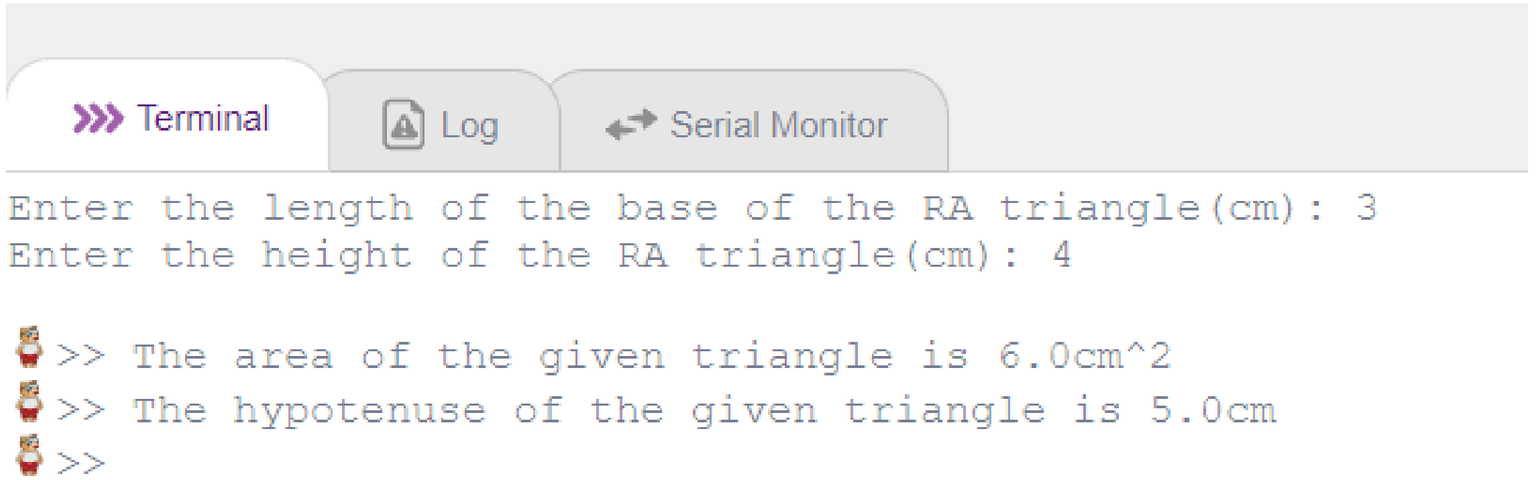
# Right Angle Tringle(Final Code)

POWERED BY STEMpedia

```
import math

def calarea(base,height):     #function to calculate area

    return(0.5*base*height)

def hyp(base,height):      #function to calculate hypotenuse

    var=math.sqrt((base*base)+(height*height))

    return(var)

base=int(input("Enter the length of the base of the RA triangle(cm): ")) #taking user inputs

height=int(input("Enter the height of the RA triangle(cm): "))

area=calarea(base,height)                    #calling the functions

hypotenuse=hyp(base,height)

print("The area of the given triangle is {}cm^2".format(area))

print("The hypotenuse of the given triangle is {}cm".format(hypotenuse))
```

# Right Angle Tringle(Final Output)






POWERED BY
STEMpedia