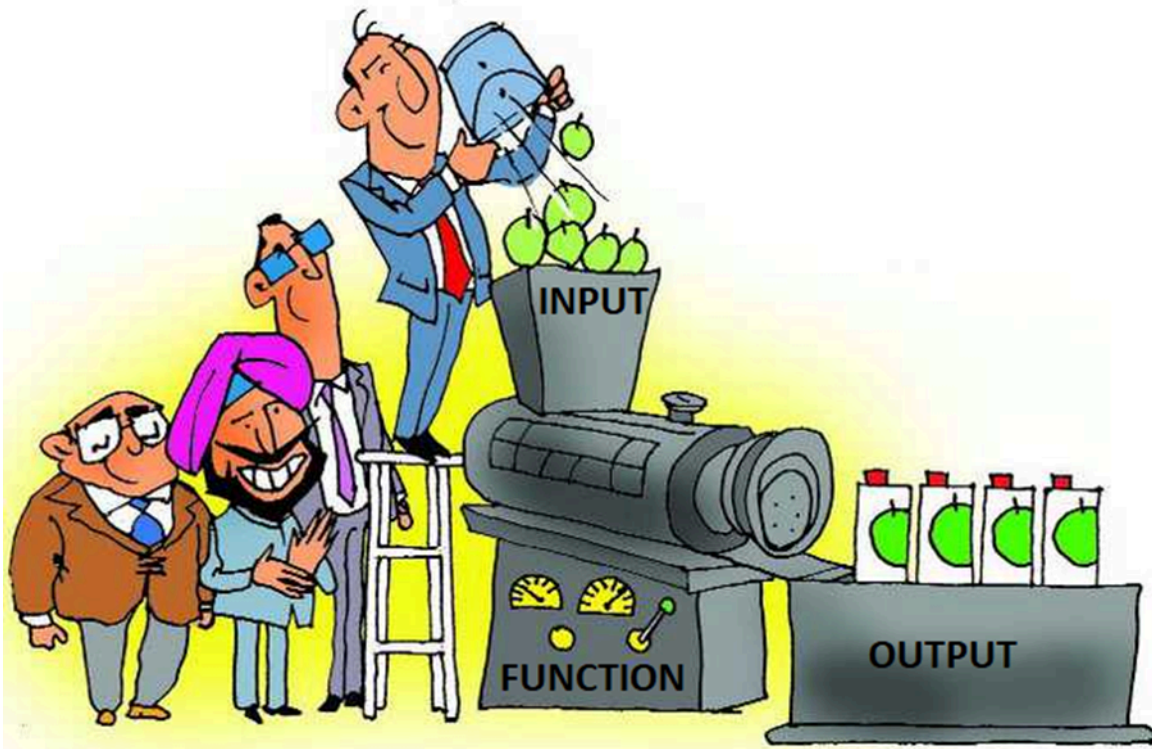


FUNCTION

A function is a **named sequence of statement(s)** that performs a computation that can be invoked from other parts of the program. Function acts on data and often returns value.



we need to define a function once like we purchase machines\appliances once and use whenever required.

functions are like machines / appliances that can be used anywhere in the program(home).

function may take parameters like machines may require things to work upon them.

function may return values like machine returns final product.

example:- popcorn maker purchased once and can be used anywhere in home any number of times . It takes corn and returns popcorn.



Functions in Python provide organized, reusable and modular code to perform a set of specific actions.

USE OF FUNCTION

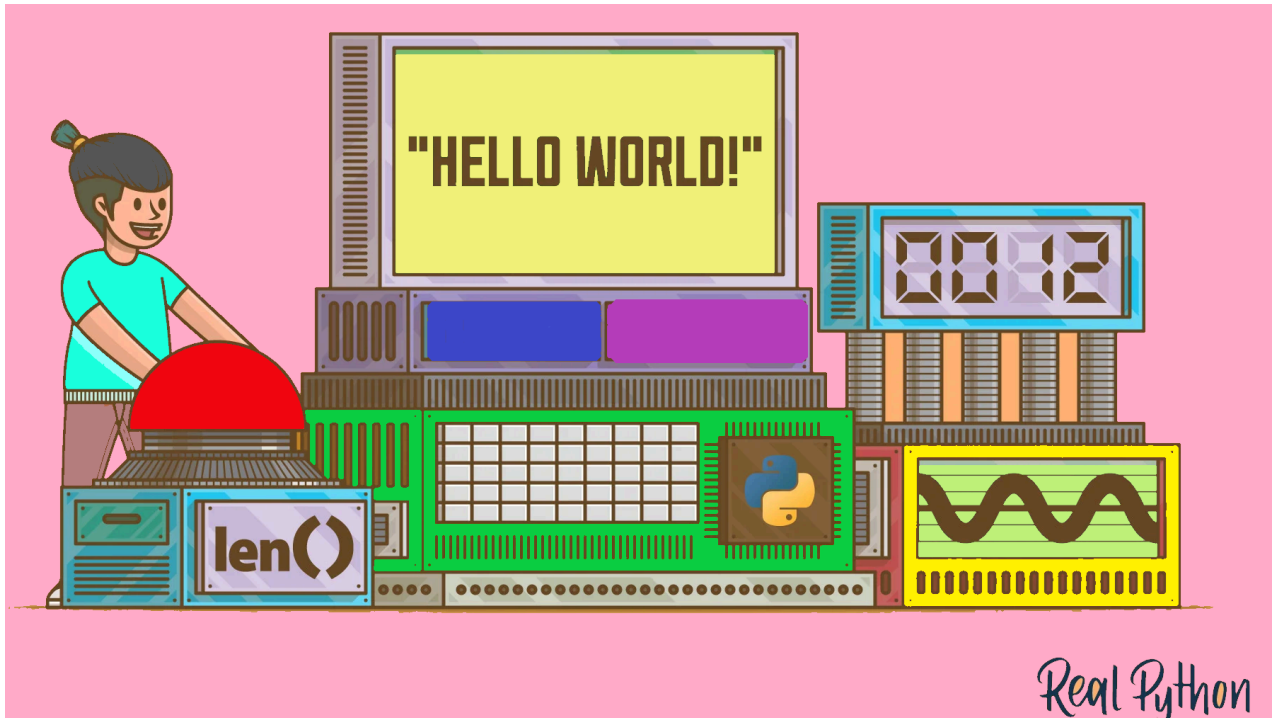
- more efficient by reducing program size
- improve program readability and clarity
- handling and managing a program become easier



Vegetable Cutter

TYPE OF FUNCTION

- Built-in function – predefined and always available for use ex:- `len()` , `type()`, `int()`



- Function defined module- pre-defined in particular module and can use after importing the module ex:- `pow()` of math module, `randint()` of random module



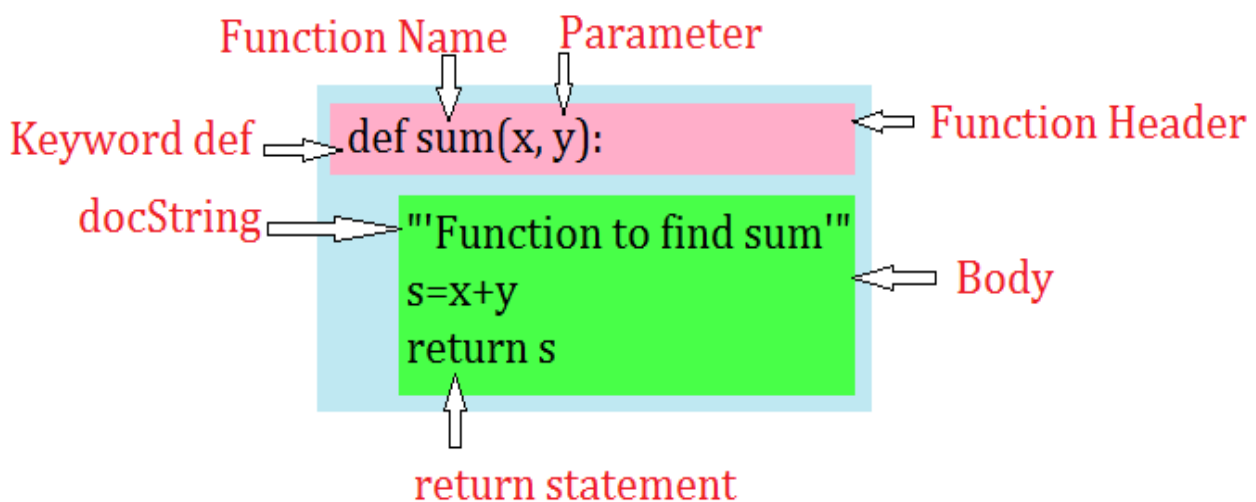
- User defined function- defined by programmer

DEFINE A FUNCTION AND CALL A FUNCTION

Syntax:

```
def <function_name>([comma separated list of parameter]):  
    [“Docstring”]  
    <statements>
```

EXAMPLE :-



EXAMPLE 1 :-

```
def sum(x, y):    # function header  
    "First Function"  
    s=x+y  
    return(s)  
print("docstring of function test:-",sum.__doc__)  
print("sum of 5 and 10=",sum(5,10))    # called function sum
```

OUTPUT:-

```
docstring of function test:- 'First Function'  
sum of 5 and 10= 15
```

NOTE:-

1. **Function Header**:- first line of function definition that begins with keyword **def** and ends at colon(:) . it specifies the **name** of the function and its **parameter**.

ex:- **def sum(x, y):**

2. **Parameter**:- **Variable** that is listed **in parenthesis** in function header. It is used to **receive value** in function from calling function. ex:- **x, y**

3. **docstring** :- Docstrings are - unlike regular **comments** - stored as an **attribute** of the function they document, meaning that can be accessed programmatically.

ex:- `"""First Function"""`
`print(sum.__doc__)`

4. **Function Body**: - The block of **statement** beneath the function header that defines the **action to be performed**. May or may not have a return statement.

ex:- `"""First Function"""`
`s=x+y`
`return(s)`

5. **return Statement**:- it is used to send one or more values from calling function to called function for further use. ex:- `return(s)`

6. **Function call statement**:-it is used to execute the function. ex:- `sum(5,10)`

FLOW OF EXECUTION IN A FUNCTION CALL

Python treats a function body as a block. In Python, a block is executed in an execution block. An execution block contains:-

- i) Some internal information used for debugging
- ii) Name of Function
- iii) Value passed to function
- iv) Variables that are created within function
- v) Information about the next instruction to be executed

Whenever a function call statement is encountered , an execution frame for the called function is created and control is transferred to it. Within the function's execution frame , the statement of body is executed and with return statement or last statement of function control is returned to the statement from which the function is called.

Assignment:-

Q 1: Consider the following Python code. Note that line numbers are included on the left.

```
1  def pow(b, p):
2      y = b ** p
3      return y
4
5  def square(x):
6      a = pow(x, 2)
7      return a
8
9  n = 5
10 result = square(n)
11 print(result)
```

Which of the following best reflects the order in which these lines of code are processed in Python?

- ☐ (A) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
- ☐ (B) 1, 2, 3, 5, 6, 7, 9, 10, 11
- ☐ (C) 9, 10, 11, 1, 2, 3, 5, 6, 7
- ☐ (D) 9, 10, 5, 6, 1, 2, 3, 6, 7, 10, 11
- ☐ (E) 1, 5, 9, 10, 5, 6, 1, 2, 3, 6, 7, 10, 11

Q -2: Consider the following Python code. Note that line numbers are included on the left.

```
1  def pow(b, p):  
2      y = b ** p  
3      return y  
4  
5  def square(x):  
6      a = pow(x, 2)  
7      return a  
8  
9  n = 5  
10 result = square(n)  
11 print(result)
```

What does this function print?

- ☐ (A) 25
- ☐ (B) 5
- ☐ (C) 125
- ☐ (D) 32

can refer :-

- 1) <https://www.guru99.com/python-tutorials.html>
- 2) <https://docs.python.org/3/tutorial/controlflow.html#defining-functions>