SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

python

19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

- A readable, dynamic, pleasant, flexible, fast and powerful language

Recap

- Iterative statements are used for repeated execution
- ‘for’ and ‘while’ are two looping statements used in python
- ‘for loop’ is definite loop whereas ‘while loop’ is indefinite loop
- State is the change in the behaviour of the objects

Agenda

- Loop control statements
  - break
  - continue
- pass statement
- Functions
  - Definition and use
  - Flow of Execution

3.2 Iterations

- Sometimes there may be a need to exit the loop completely when an external condition is triggered or there may be a situation to skip a part of the code and start the next execution.
- Python provide the following statements
  1. Break
  2. Continue
  3. Pass
- In Python, break and continue statements can alter the flow of a normal loop.
- Loops iterate over a block of code until test expression is false, to terminate the current iteration or even the whole loop without checking test expression.
- The break and continue statements are used in these cases.

3.2 Iterations

3.2.4. break Statement

- The break statement terminates the loop containing it.
- Control of the program is transferred to the statement which is present immediately after the body of the loop.
- If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.

3.2 Iterations

3.2.4. break Statement

Syntax:

```
break
```

Example 1:

```python
for val in "string":
    if val == "i":
        break
    print(val)
print("The end")
```

Output:

```
s
t
r
The end
>>> 
```

3.2 Iterations

3.2.4. break Statement

Example 2:

```
i=1
while i<=10:
        print(i)
        if(i==5):
                break
        i = i + 1
print("completed")
```

Output:

```
1
2
3
4
5
completed
```

3.2 Iterations

3.2.5. continue Statement

- The continue statement is used to skip the rest of the code inside a loop for the current iteration only.
- Loop does not terminate but continues on with the next iteration.

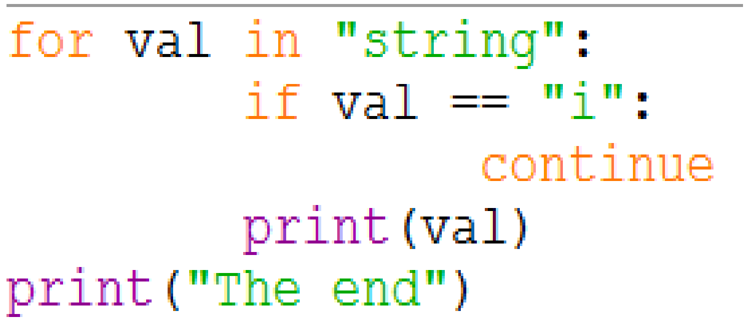
Syntax:

```
continue
```

3.2 Iterations

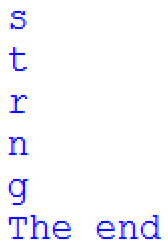
3.2.5. continue Statement

Example 1:



```
for val in "string":
        if val == "i":
                continue
        print(val)
print("The end")
```

Output:



```
s
t
r
n
g
The end
```

3.2 Iterations

3.2.5. pass Statement

- pass is used when a statement is required syntactically but you do not want any command or code to execute.
- The pass statement is a null operation; nothing happens when it executes.
- The pass is also useful in places where your code will eventually go, but has not been written yet.

3.2 Iterations

3.2.5. pass Statement

Syntax

```
pass
```

Example 1:

```
for letter in "Python":
        if letter == 'h':
                pass
                print("This is pass block")
                continue
        print("Current Letter :",letter)
print("Good bye!")
```

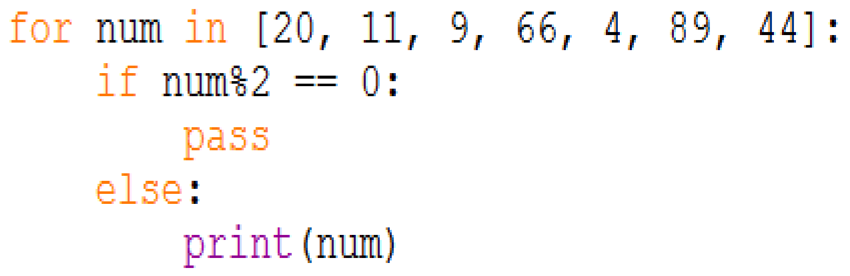
Output:

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : o
Current Letter : n
Good bye!
```

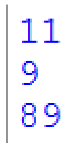
3.2 Iterations

3.2.5. pass Statement

Example:



```
for num in [20, 11, 9, 66, 4, 89, 44]:
    if num%2 == 0:
        pass
    else:
        print(num)
```



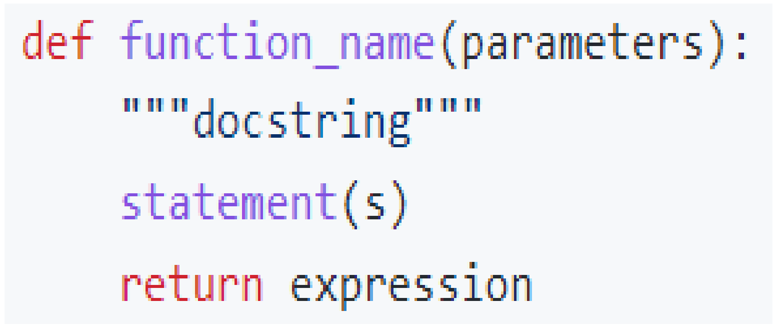
```
11
9
89
```

3.4 Functions

- Python Functions is a block of related statements designed to perform a computational, logical, or evaluative task.
- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- 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.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.4 Functions

Syntax:

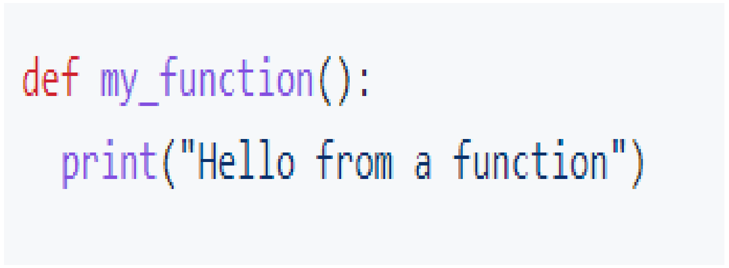


```
def function_name(parameters):
    """docstring"""
    statement(s)
    return expression
```

3.4 Functions

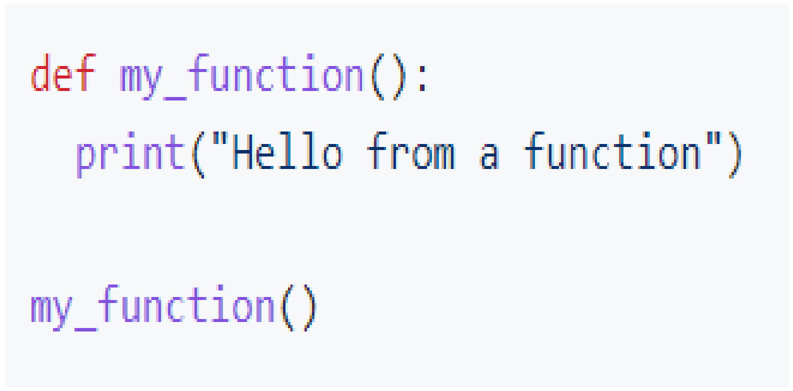
Function Definition and Use

- In Python a function is defined using the def keyword:



```python
def my_function():
  print("Hello from a function")
```

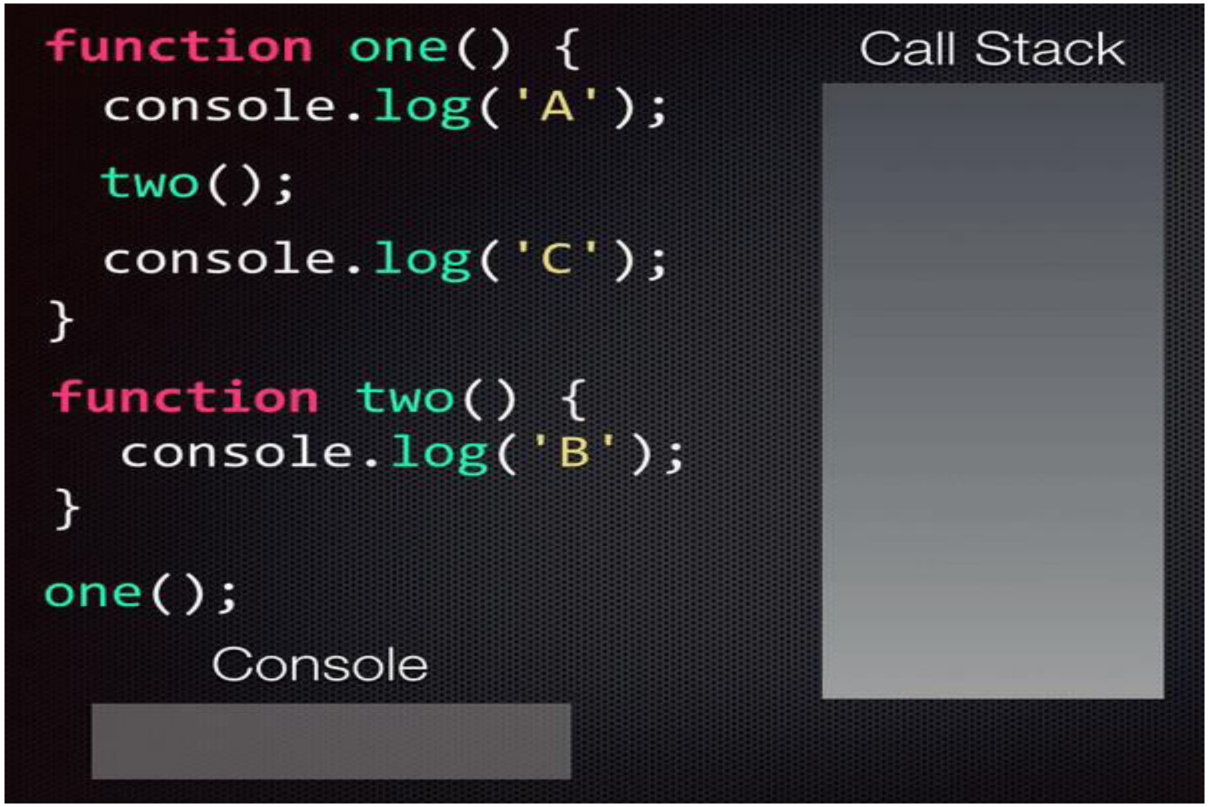
- To call a function, use the function name followed by parenthesis:



```python
def my_function():
  print("Hello from a function")

my_function()
```

3.4 Functions

Function Definition and Use



```
function one() {
  console.log('A');
  two();
  console.log('C');
}
function two() {
  console.log('B');
}
one();
```

3.4 Functions

Function Definition and Use

Example

- https://replit.com/@ErAmbikaM/functionexample#main.py

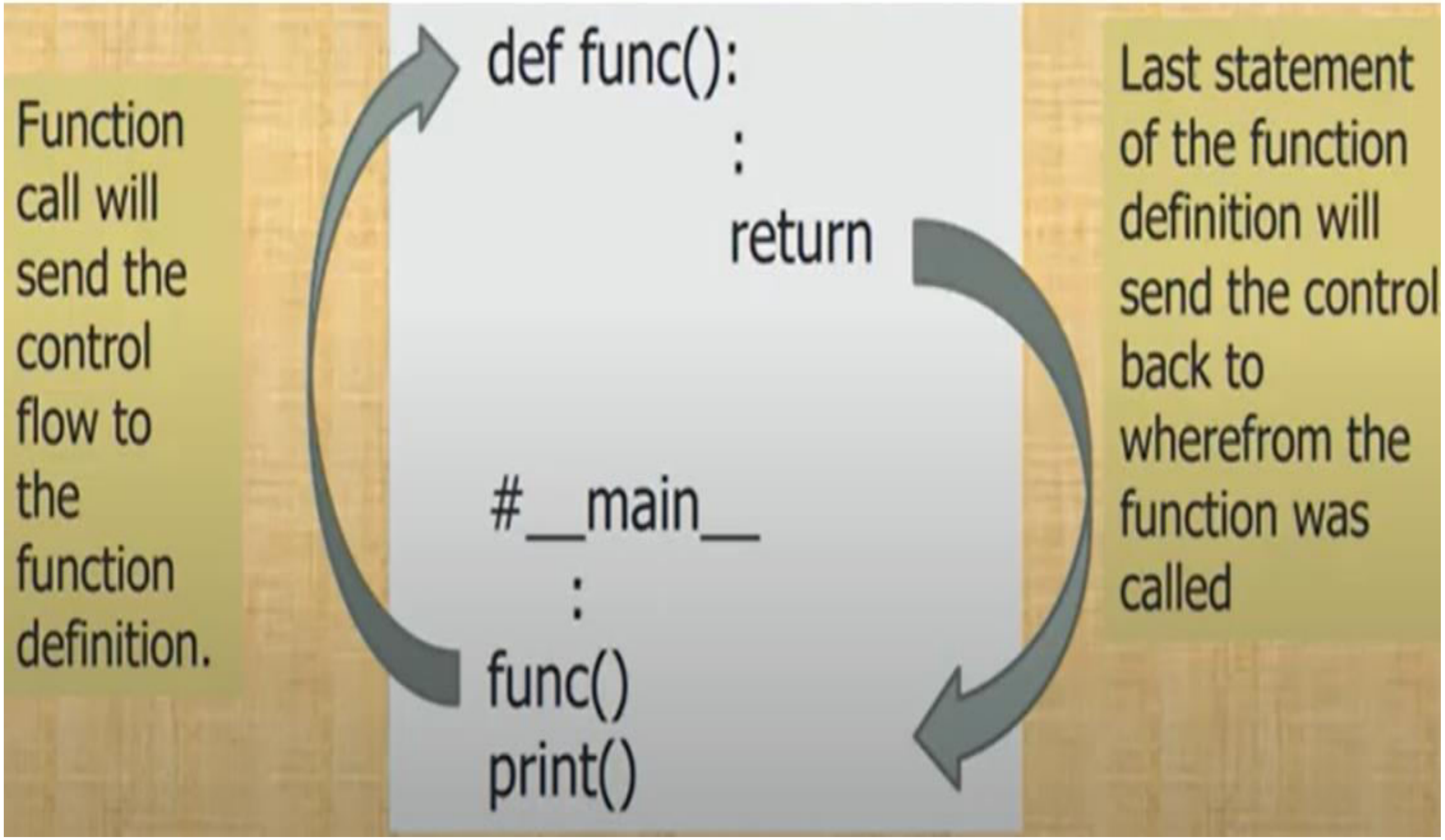
3.4 Functions

Flow of Execution

- Flow of execution - the order in which statements are executed
- Execution always starts at the first statement of the program
- Statements execute one at a time from top to bottom
- Functions definitions do not alter the flow of execution
- When a function is called, the flow control will jump to the first line of the called function
- Then, it will execute all the statements there. After that, it will come back to pick up where it left off.

3.4 Functions

Flow of Execution



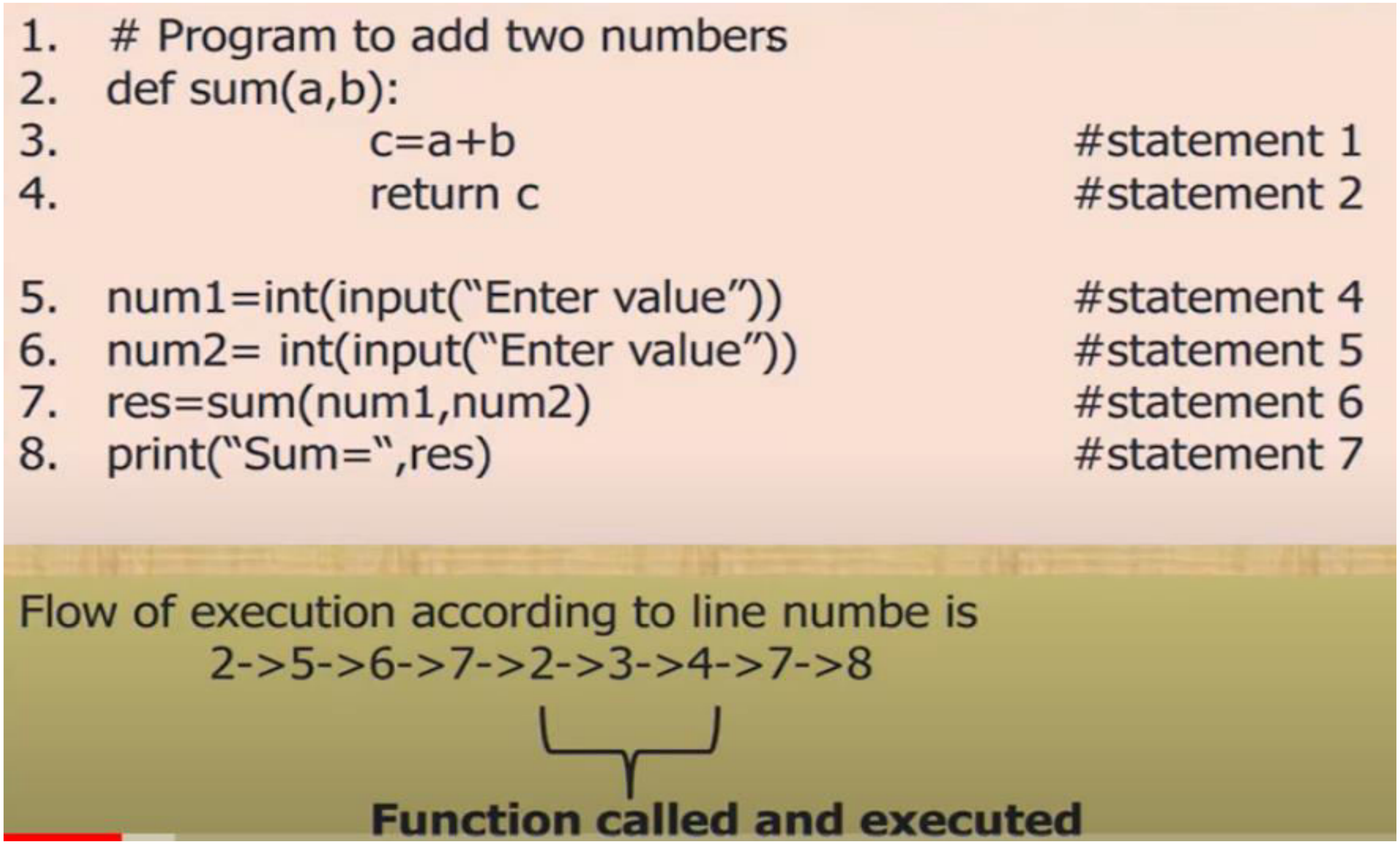
Function call will send the control flow to the function definition.

```
def func():
          :
          return



#__main__
   :
func()
print()
```

Last statement of the function definition will send the control back to wherefrom the function was called

3.4 Functions

Flow of Execution



```
1.  # Program to add two numbers
2.  def sum(a,b):
3.                  c=a+b                                #statement 1
4.                  return c                             #statement 2

5.  num1=int(input("Enter value"))                       #statement 4
6.  num2= int(input("Enter value"))                      #statement 5
7.  res=sum(num1,num2)                                   #statement 6
8.  print("Sum=",res)                                    #statement 7
```

Flow of execution according to line numbe is

$$2\text{->}5\text{->}6\text{->}7\text{->}\underbrace{2\text{->}3\text{->}4}_{\textbf{Function called and executed}}\text{->}7\text{->}8$$

Summary

- “break” statement is used terminate the loop in between the iterations
- “continue” statement is used to skip an iteration
- “pass” statement acts as a placeholders for future code
- Python Functions is a block of related statements designed to perform a computational, logical, or evaluative task.
- Flow of execution is the order in which statements are executed

THANK YOU