**Lists and Strings**

Splitting

str\_var.split(separator)

Splits a string into a list at every specified separator. If no separator is specified, default separator is whitespace.

**Code**



1

2

3

nums = "1 2 3 4"

num\_list = nums.split()

print(num\_list)

PYTHON

**Output**



['1', '2', '3', '4']

Multiple WhiteSpaces

Multiple whitespaces are considered as single when splitting.

**Code**



1

2

3

nums = "1 2 3 4 "

num\_list = nums.split()

print(num\_list)

PYTHON

**Output**



['1', '2', '3', '4']

New line

\n

and tab space

\t

are also whitespace.

**Code**



1

2

3

nums = "1\n2\t3 4"

num\_list = nums.split()

print(num\_list)

PYTHON

**Output**



['1', '2', '3', '4']

Using Separator

Breaks up a string at the specified separator.

*Example -1*

**Code**



1

2

3

nums = "1,2,3,4"

num\_list = nums.split(',')

print(num\_list)

PYTHON

**Output**



['1', '2', '3', '4']

*Example -2*

**Code**



1

2

3

nums = "1,2,,3,4,"

num\_list = nums.split(',')

print(num\_list)

PYTHON

**Output**



['1', '2', '', '3', '4', '']

Space as Separator

**Code**



1

2

3

nums = "1 2 3 4 "

num\_list = nums.split(" ")

print(num\_list)

PYTHON

**Output**



['1', '', '2', '3', '4', '']

String as Separator

*Example - 1*

**Code**



1

2

3

string\_a = "Python is a programming language"

list\_a = string\_a.split('a')

print(list\_a)

PYTHON

**Output**



['Python is ', ' progr', 'mming l', 'ngu', 'ge']

*Example - 2*



1

2

3

string\_a = "step-by-step execution of code"

list\_a = string\_a.split('step')

print(list\_a)

PYTHON

**Output**



['', '-by-', ' execution of code']

Joining

str.join(sequence)

Takes all the items in a sequence of strings and joins them into one string.

**Code**



1

2

3

list\_a = ['Python is ', ' progr', 'mming l', 'ngu', 'ge']

string\_a = "a".join(list\_a)

print(string\_a)

PYTHON

**Output**



Python is a programming language

Joining Non String Values

Sequence should not contain any non-string values.

**Code**



1

2

3

list\_a = list(range(4))

string\_a = ",".join(list\_a)

print(string\_a)

PYTHON

**Output**



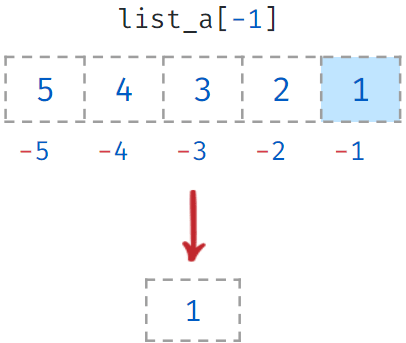
TypeError: sequence item 0: expected str instance, int found

**Negative Indexing**

Using a negative index returns the nth item from the end of list.

Last item in the list can be accessed with index

-1



Reversing a List

-1

for step will reverse the order of items in the list.

**Code**



1

2

3

list\_a = [5, 4, 3, 2, 1]

list\_b = list\_a[::-1]

print(list\_b)

PYTHON

**Output**



[1, 2, 3, 4, 5]

Accessing List Items

*Example-1*

**Code**



1

2

3

list\_a = [5, 4, 3, 2, 1]

item = list\_a[-1]

print(item)

PYTHON

**Output**



1

*Example-2*

**Code**



1

2

3

list\_a = [5, 4, 3, 2, 1]

item = list\_a[-4]

print(item)

PYTHON

**Output**



4

Slicing With Negative Index

You can also specify negative indices while slicing a List.

**Code**



1

2

3

list\_a = [5, 4, 3, 2, 1]

list\_b = list\_a[-3:-1]

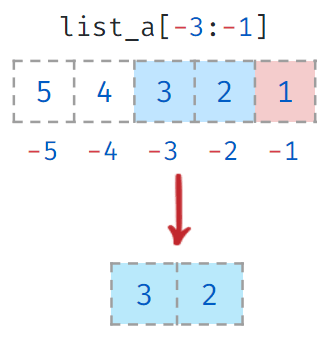
print(list\_b)

PYTHON

**Output**



[3, 2]



Out of Bounds Index

While slicing, Index can go out of bounds.

**Code**



1

2

3

list\_a = [5, 4, 3, 2, 1]

list\_b = list\_a[-6:-2]

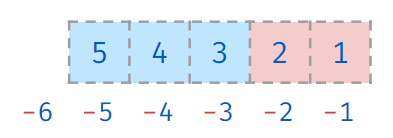
print(list\_b)

PYTHON

**Output**



[5, 4, 3]



Negative Step Size

variable[start:end:negative\_step]

Negative Step determines the decrement between each index for slicing.

Start index should be greater than the end index in this case

* start > end

Negative Step Size Examples

*Example - 1*

**Code**



1

2

3

list\_a = [5, 4, 3, 2, 1]

list\_b = list\_a[4:2:-1]

print(list\_b)

PYTHON

**Output**



[1, 2]

*Example - 2*

Negative step requires the start to be greater than end.

**Code**



1

2

3

list\_a = [5, 4, 3, 2, 1]

list\_b = list\_a[2:4:-1]

print(list\_b)

PYTHON

**Output**



[]

Reversing a List

-1

for step will reverse the order of items in the list.



1

2

3

list\_a = [5, 4, 3, 2, 1]

list\_b = list\_a[::-1]

print(list\_b)

PYTHON

**Output**



[1, 2, 3, 4, 5]

Reversing a String

-1

for step will reverse the order of the characters.

**Code**



1

2

3

string\_1 = "Program"

string\_2 = string\_1[::-1]

print(string\_2)

**Output**



margorP

Negative Step Size - Strings

**Code**



1

2

3

string\_1 = "Program"

string\_2 = string\_1[6:0:-2]

print(string\_2)

PYTHON

**Output**

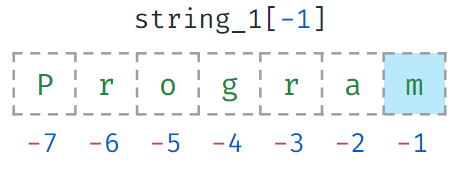


mro

Indexing & Slicing - Strings

*Example - 1*

**Code**





1

2

3

string\_1 = "Program"

string\_2 = string\_1[-1]

print(string\_2)

PYTHON

**Output**



m

*Example - 2*

**Code**



1

2

3

string\_1 = "Program"

string\_2 = string\_1[-4:-1]

print(string\_2)

PYTHON

**Output**



gra

Notes

Discussions

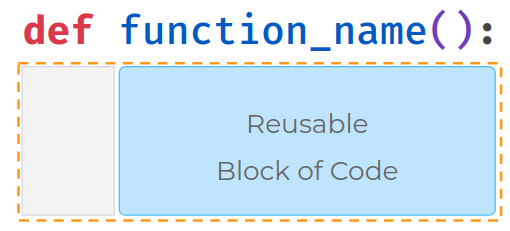
Notes

**Functions**

Block of reusable code to perform a specific action.

Reusing Code

Using an existing code without writing it every time we need.



**Code**



1

2

3

4

5

def greet():

print("Hello")

name = input()

print(name)

PYTHON

**Input**



Teja

**Output**

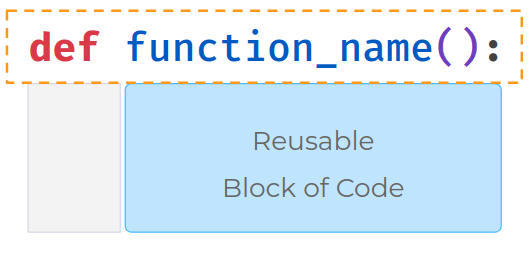


Teja

Defining a Function

Function is uniquely identified by the

function\_name



**Code**



1

2

3

4

5

def greet():

print("Hello")

name = input()

print(name)

PYTHON

**Input**



Teja

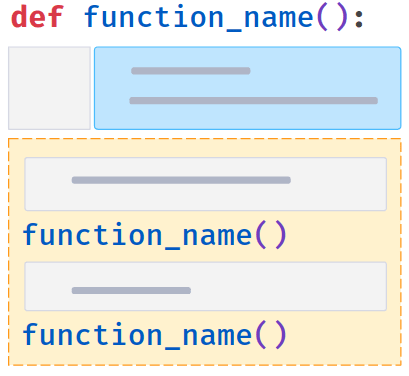
**Output**



Teja

Calling a Function

The functional block of code is executed only when the function is called.



**Code**



1

2

3

4

5

6

def greet():

print("Hello")

name = input()

greet()

print(name)

PYTHON

**Input**



Teja

**Output**



Hello

Teja

Defining & Calling a Function

A function should be defined before it is called.

**Code**



1

2

3

4

5

6

name = input()

greet()

print(name)

def greet():

print("Hello")

PYTHON

**Input**



Teja

**Output**



NameError: name 'greet' is not defined

Printing a Message

**Code**



1

2

3

4

5

6

def greet():

msg = "Hello " + ?

print(msg)

name = input()

greet()

PYTHON

**Input**



Teja

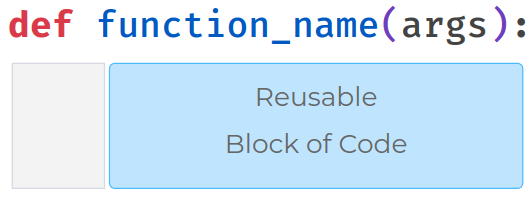
**Output**



Hello Teja

Function With Arguments

We can pass values to a function using Argument.



**Code**



1

2

3

4

5

6

def greet(word):

msg = "Hello " + word

print(msg)

name = input()

greet(word=name)

PYTHON

**Input**



Teja

**Output**



Hello Teja

Variables Inside a Function

A variable created inside a function can only be used in it.

**Code**



1

2

3

4

5

6

def greet(word):

msg = "Hello " + word

name = input()

greet(word=name)

print(msg)

PYTHON

**Input**



Teja

**Output**



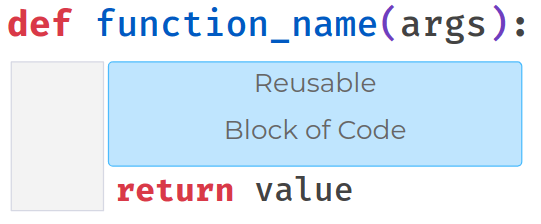
NameError: name 'msg' is not defined

Returning a Value

To return a value from the function use

return

keyword.



Exits from the function when return statement is executed.

**Code**



1

2

3

4

5

6

7

def greet(word):

msg = "Hello " + word

return msg

name = input()

greeting = greet(word=name)

print(greeting)

PYTHON

**Input**



Teja

**Output**



Hello Teja

Code written after

return

statement will not be executed.

**Code**



1

2

3

4

5

6

7

8

def greet(word):

msg = "Hello "+word

return msg

print(msg)

name = input()

greeting = greet(word=name)

print(greeting)

PYTHON

**Input**



Teja

**Output**



Hello Teja

Built-in Functions

We are already using functions which are pre-defined in Python. Built-in functions are readily available for reuse

* print()
* int()
* str()
* len()



[Home](https://learning.ccbp.in/)Code Playground

Nakka Vijay Kumar

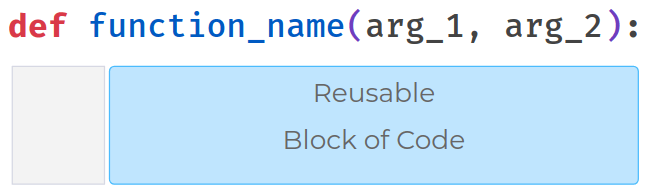
**NK**

**Cheat Sheet**

Cheat Sheet

**Function Arguments**

A function can have more than one argument.



Keyword Arguments

Passing values by their names.

**Code**



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_1=greeting,arg\_2=name)

PYTHON

**Input**



Good Morning

Ram

**Output**



Good Morning Ram

Possible Mistakes - Keyword Arguments

**Code**



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

**Input**



Good Morning

Ram

**Output**



TypeError: greet() missing 1 required positional argument: 'arg\_1'

Positional Arguments

Values can be passed without using argument names.

* These values get assigned according to their position.
* Order of the arguments matters here.

**Code**



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(greeting,name)

PYTHON

**Input**



Good Morning

Ram

**Output**



Good Morning Ram

Possible Mistakes - Positional Arguments

**Mistake - 1**

**Code**



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(greeting)

PYTHON

**Input**



Good Morning

Ram

**Output**



TypeError: greet() missing 1 required positional argument: 'arg\_2'

**Mistake - 2**

**Code**



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet()

PYTHON

**Input**



Good Morning

Ram

**Output**



TypeError: greet() missing 2 required positional arguments

Default Values

*Example - 1*

**Code**



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet()

PYTHON

**Input**



Hello

Teja

**Output**



Hi Ram

*Example - 2*

**Code**



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(greeting)

PYTHON

**Input**



Hello

Teja

**Output**



Hello Ram

*Example - 3*

**Code**



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(name)

PYTHON

**Input**



Hello

Teja

**Output**



Teja Ram

*Example - 4*

**Code**



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

**Input**



Hello

Teja

**Output**



Hi Teja

*Example - 5*

**Code**



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

**Input**



Hello

Teja

**Output**



SyntaxError:non-default argument follows default argument

Non-default arguments cannot follow default arguments.

*Example - 6*

**Code**



1

2

3

4

5

6

def greet(arg\_2, arg\_1="Hi"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

**Input**



Hello

Teja

**Output**



Hi Teja

Passing Immutable Objects

**Code**



1

2

3

4

5

6

def increment(a):

a += 1

a = int(input())

increment(a)

print(a)

PYTHON

**Input**



5

**Output**



5

Even though variable names are same, they are referring to two different objects. Changing the value of the variable inside the function will not affect the variable outside.

Notes

Discussions

Notes

NEW NOTE

CANCELSAVE



[Home](https://learning.ccbp.in/)Code Playground

Nakka Vijay Kumar

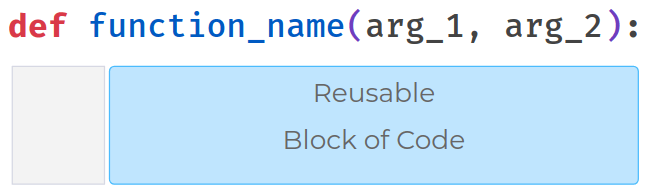
**NK**

**Cheat Sheet**

Cheat Sheet

# Function Arguments

A function can have more than one argument.



## Keyword Arguments

Passing values by their names.

#### Code



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_1=greeting,arg\_2=name)

PYTHON

#### Input



Good Morning

Ram

#### Output



Good Morning Ram

### Possible Mistakes - Keyword Arguments

#### Code



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

#### Input



Good Morning

Ram

#### Output



TypeError: greet() missing 1 required positional argument: 'arg\_1'

## Positional Arguments

Values can be passed without using argument names.

* These values get assigned according to their position.
* Order of the arguments matters here.

#### Code



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(greeting,name)

PYTHON

#### Input



Good Morning

Ram

#### Output



Good Morning Ram

### Possible Mistakes - Positional Arguments

**Mistake - 1**

#### Code



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(greeting)

PYTHON

#### Input



Good Morning

Ram

#### Output



TypeError: greet() missing 1 required positional argument: 'arg\_2'

**Mistake - 2**

#### Code



1

2

3

4

5

6

def greet(arg\_1, arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet()

PYTHON

#### Input



Good Morning

Ram

#### Output



TypeError: greet() missing 2 required positional arguments

### Default Values

*Example - 1*

#### Code



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet()

PYTHON

#### Input



Hello

Teja

#### Output



Hi Ram

*Example - 2*

#### Code



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(greeting)

PYTHON

#### Input



Hello

Teja

#### Output



Hello Ram

*Example - 3*

#### Code



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(name)

PYTHON

#### Input



Hello

Teja

#### Output



Teja Ram

*Example - 4*

#### Code



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2="Ram"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

#### Input



Hello

Teja

#### Output



Hi Teja

*Example - 5*

#### Code



1

2

3

4

5

6

def greet(arg\_1="Hi", arg\_2):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

#### Input



Hello

Teja

#### Output



SyntaxError:non-default argument follows default argument

Non-default arguments cannot follow default arguments.

*Example - 6*

#### Code



1

2

3

4

5

6

def greet(arg\_2, arg\_1="Hi"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

PYTHON

#### Input



Hello

Teja

#### Output



Hi Teja

## Passing Immutable Objects

#### Code



1

2

3

4

5

6

def increment(a):

a += 1

a = int(input())

increment(a)

print(a)

PYTHON

#### Input



5

#### Output



5

Even though variable names are same, they are referring to two different objects. Changing the value of the variable inside the function will not affect the variable outside.

Notes

Discussions

Notes

NEW NOTE

CANCELSAVE

[**Download App**](https://play.google.com/store/apps/details?id=co.ibhubs.chat)

**Personal Manager**



1