Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

Creating a Function

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

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

Calling a Function

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

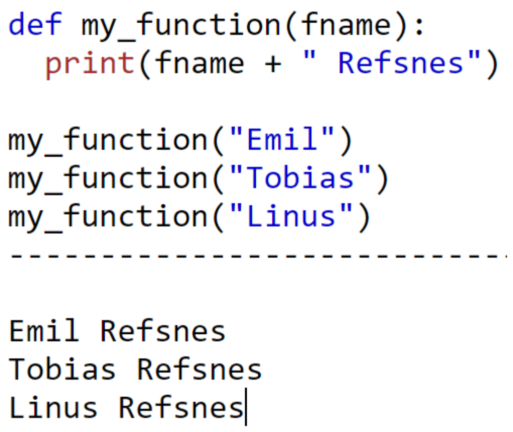
```python
def my_function():
  print("Hello from a function")

my_function()

------------------------------
Hello from a function
```

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

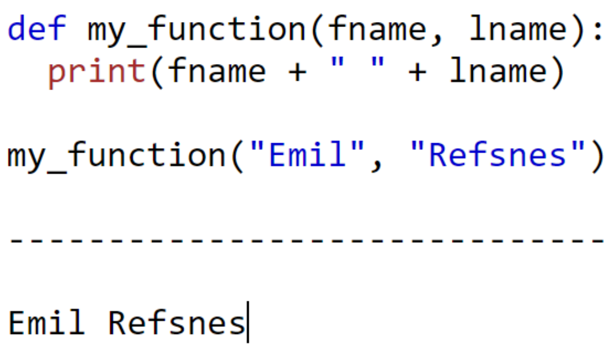


```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
----------------------------

Emil Refsnes
Tobias Refsnes
Linus Refsnes
```

Arguments are often shortened to args in Python documentations.

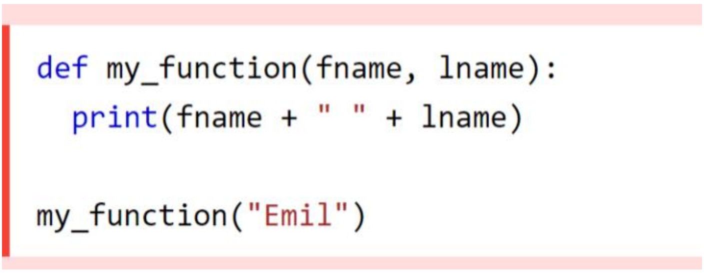
Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.



```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")

------------------------------

Emil Refsnes
```

If you try to call the function with 1 or 3 arguments, you will get an error:

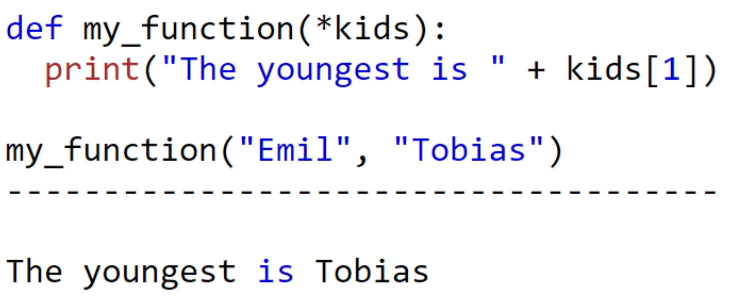


```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil")
```

Arbitrary Arguments, *args

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

This way the function will receive a *tuple* of arguments, and can access the items accordingly:

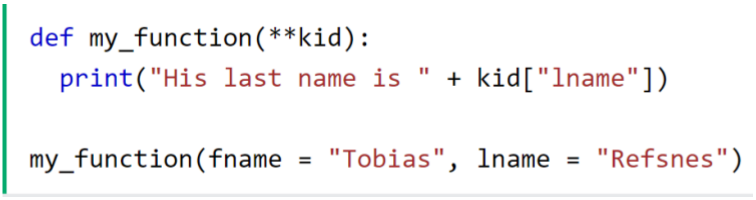


```
def my_function(*kids):
  print("The youngest is " + kids[1])

my_function("Emil", "Tobias")
--------------------------------------

The youngest is Tobias
```

Arbitrary Keyword Arguments, **kwargs

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: ** before the parameter name in the function definition.

This way the function will receive a *dictionary* of arguments, and can access the items accordingly:

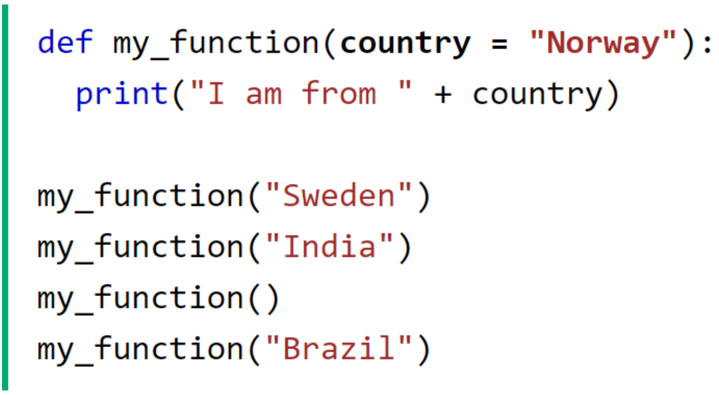


```
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

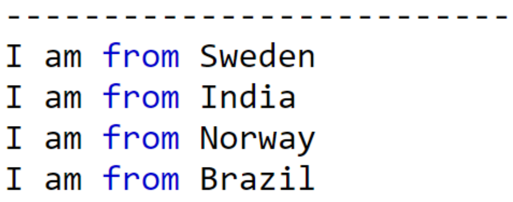
Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without argument, it uses the default value:



```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```



```
--------------------------
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

Return Values

To let a function return a value, use the `return` statement:

```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
----------------------
15
25
45
```

The pass Statement

`function` definitions cannot be empty, but if you for some reason have a `function` definition with no content, put in the `pass` statement to avoid getting an error.

```
def myfunction():
  pass
```

# having an empty function definition like this, would raise an error without the pass statement