Functions

Recap

Raw materials

Coffee Machine

Perform a specific Task:
Make Coffee

Product

Input

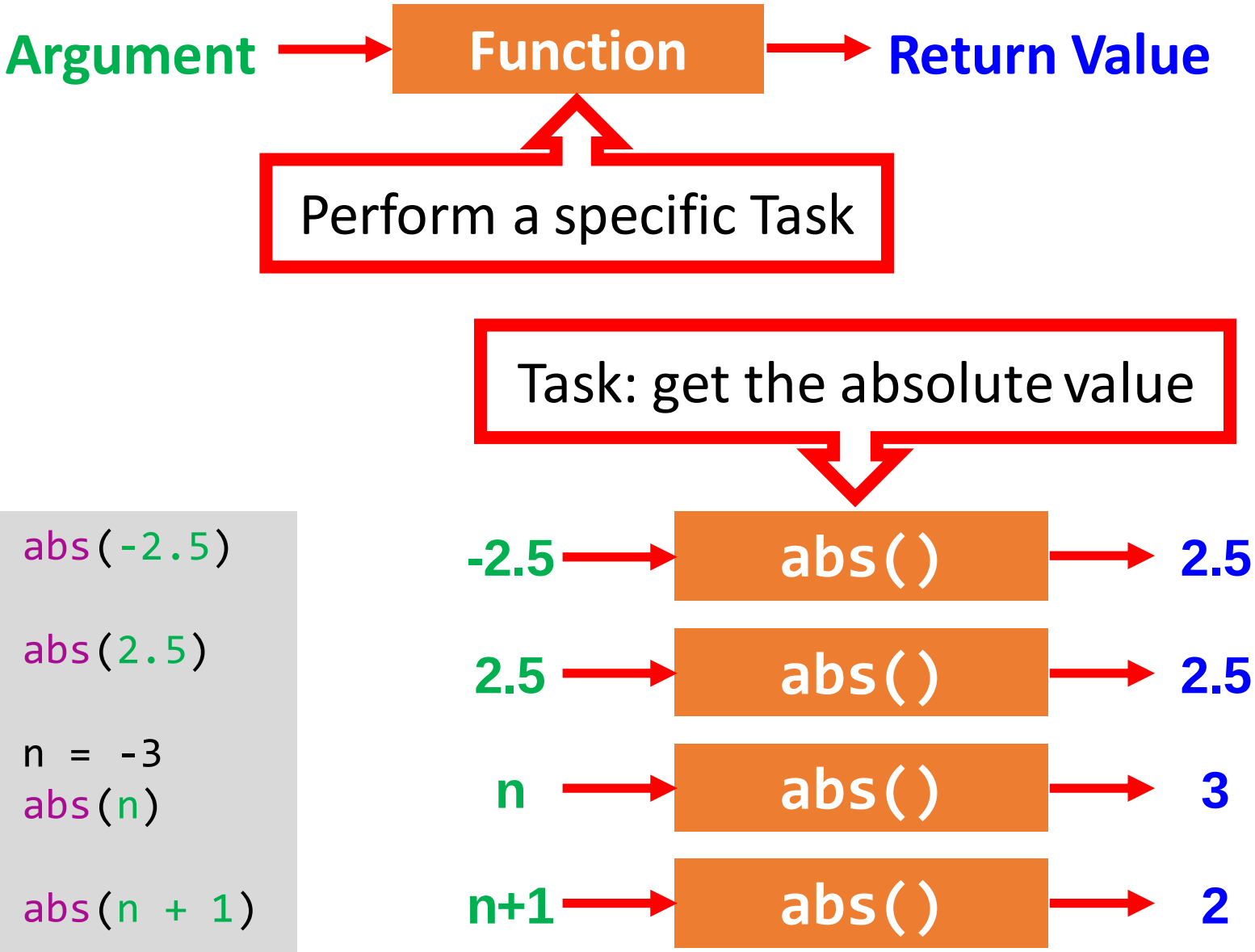
Argument

Function

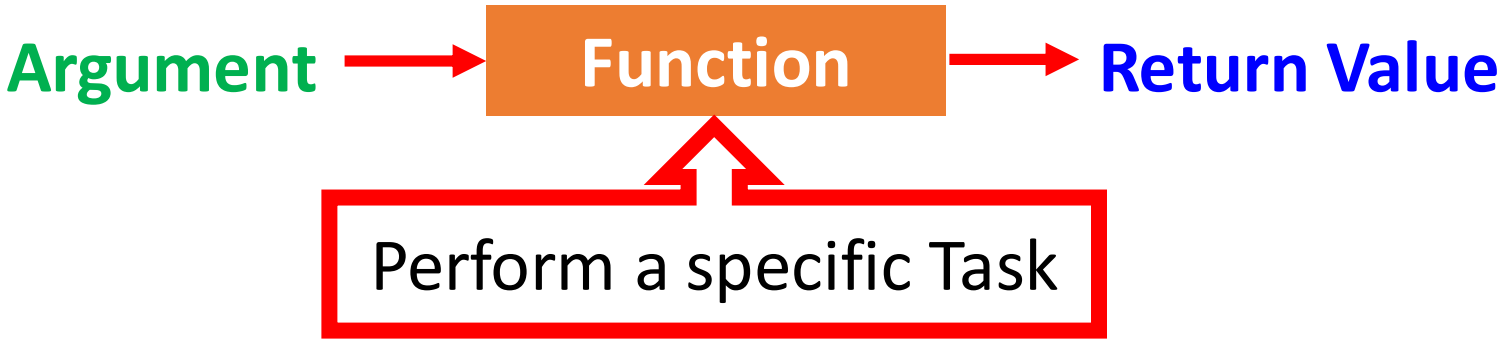
Perform a specific Task

Output

Return Value




```
>>> abs(-2.5)
2.5
>>> abs(2.5)
2.5
>>> n = -3
>>> abs(n)
3
>>> abs(n + 1)
2
```



Examples of calling a function

```
>>> abs(-2.5)
2.5
>>> abs(2.5)
2.5
>>> n = -3
>>> abs(n)
3
>>> abs(n + 1)
2
```

We use a function by **calling** it, how?

`function_name(argument(s))`

When calling a function:

Only need to know what it does

e.g. **`abs()`** returns absolute value of a number

No need to know *HOW* it does the job!

More Examples

```
>>> m = abs(-2)
>>> m
2
>>> n = abs(-3) + m
>>> n
5
>>> t = abs(-8 + n)
>>> t
3
```

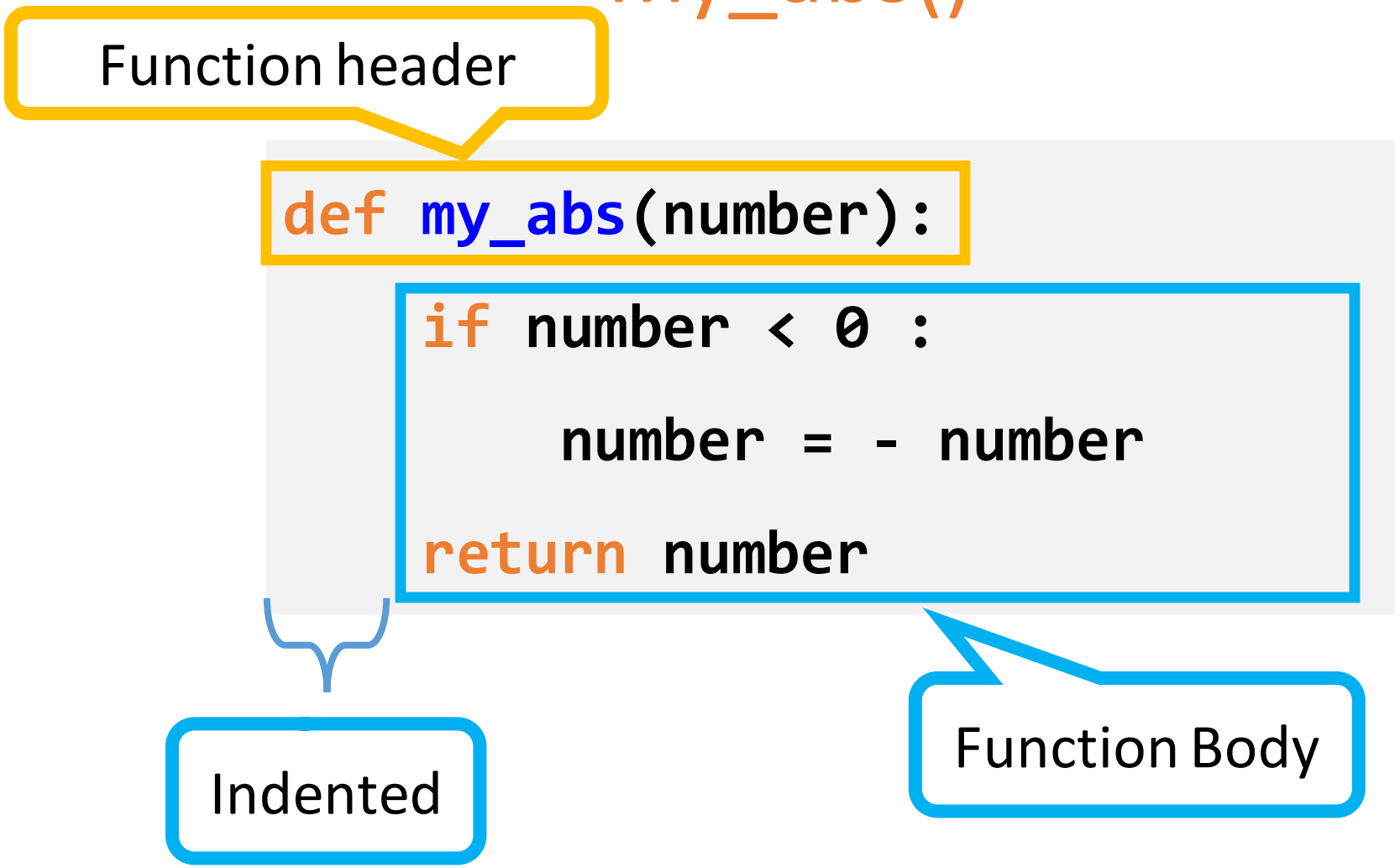
Assigning the return value to a variable

Using the return value for other operations

Argument can be an expression, **evaluate** first, then use value as the argument

Define our own Function

my_abs()


Function header
def my_abs(number):
if number < 0 :
number = - number
return number
Indented
Function Body

Function Header

Always begin with **def**

Name of the function

Parameter(s)

```
def my_abs(number):
    if number < 0 :
        number = - number
    return number
```

Function Header

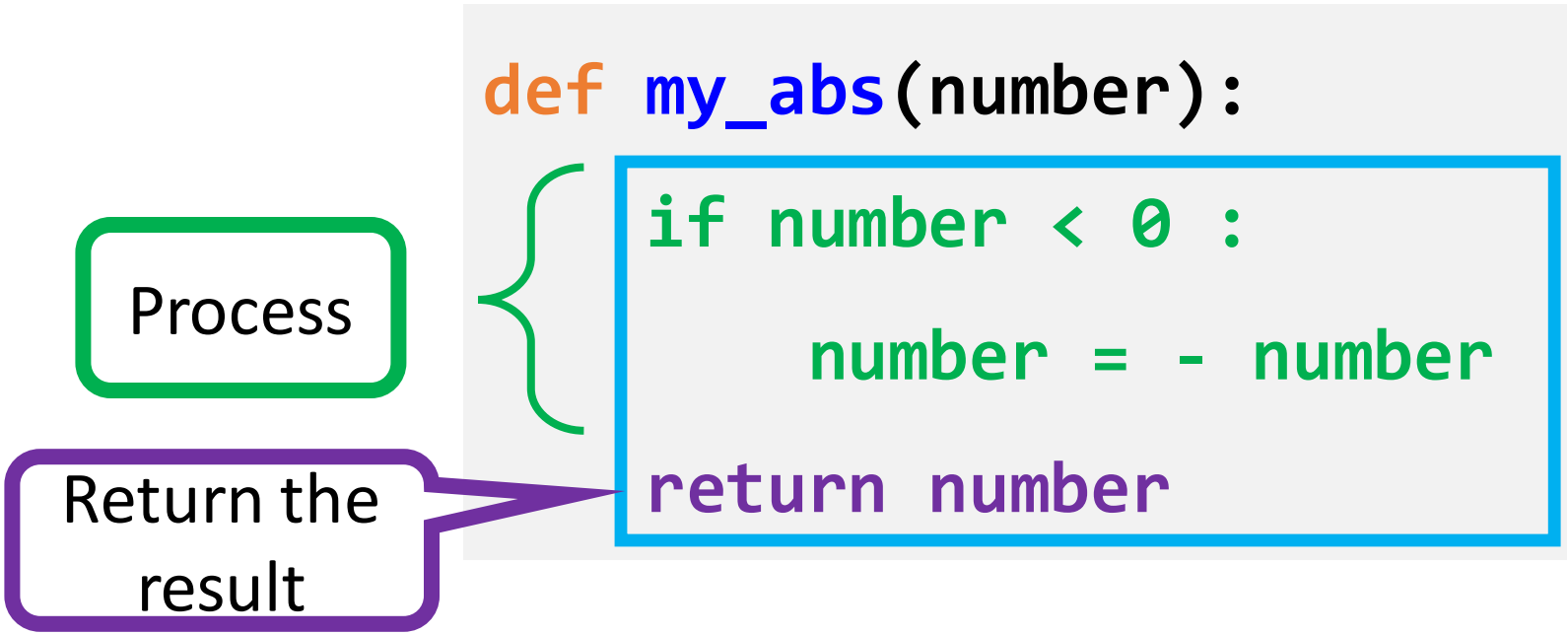
Always begin with **def**

Name of the function

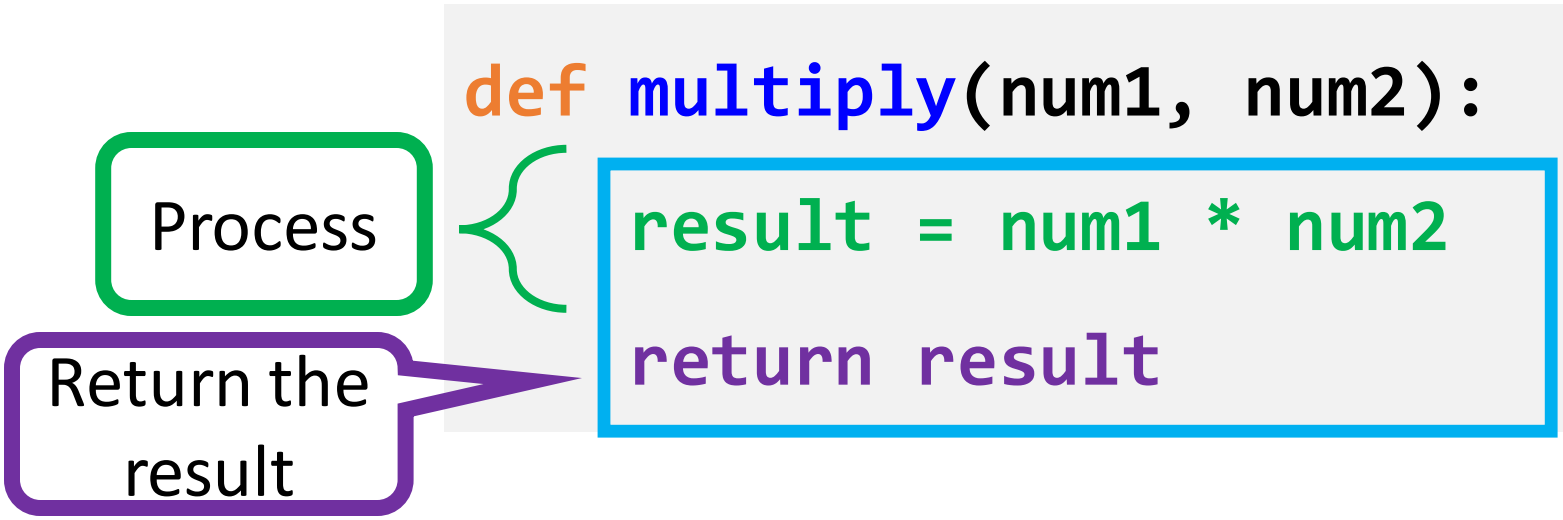
Parameter(s)

```
def multiply(num1, num2):
    result = num1 * num2
    return result
```

Function body



Function body



Function Definition Syntax

Function header

```
def <function_name>(<parameter1, parameter2, …>):
    <statements>
    return <result>
```

Indented Function Body

Call a Function

Call a function

Step 1: Define a function

```
def my_abs(number):
    if number < 0 :
        number = - number
    return number
```

Step 2: Call the function

Function name

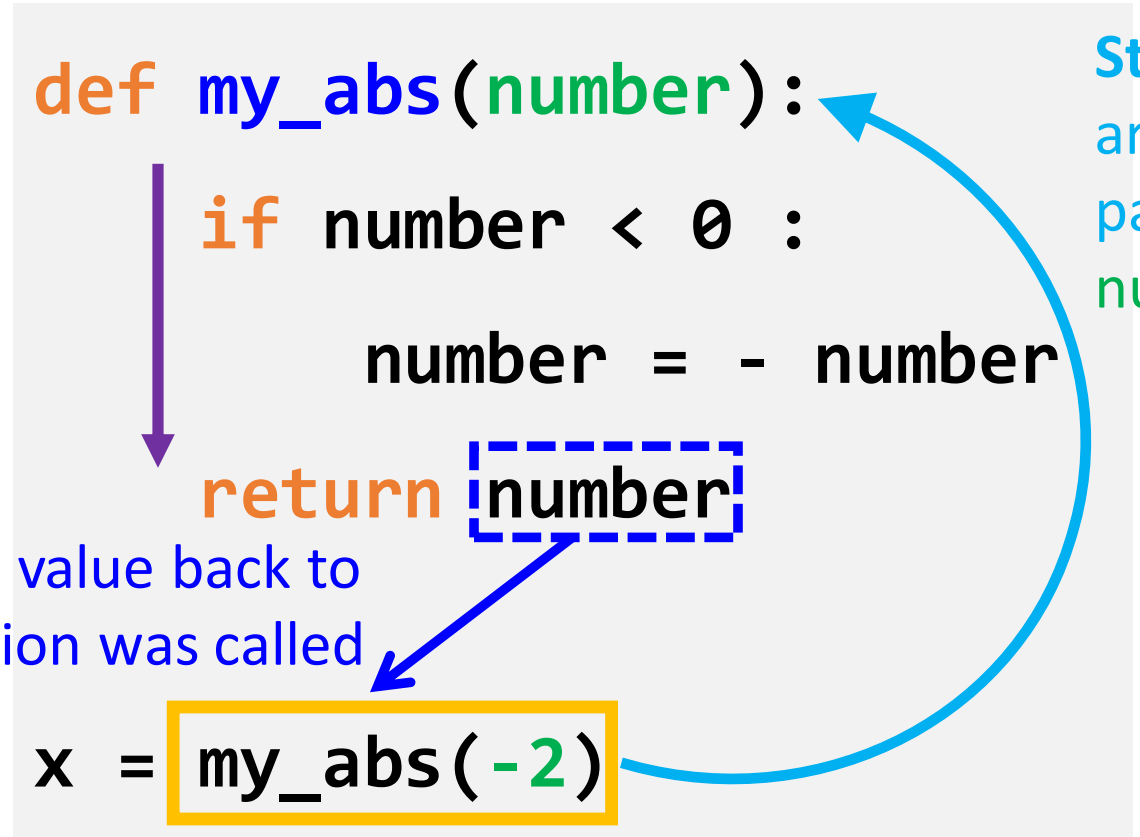
Argument

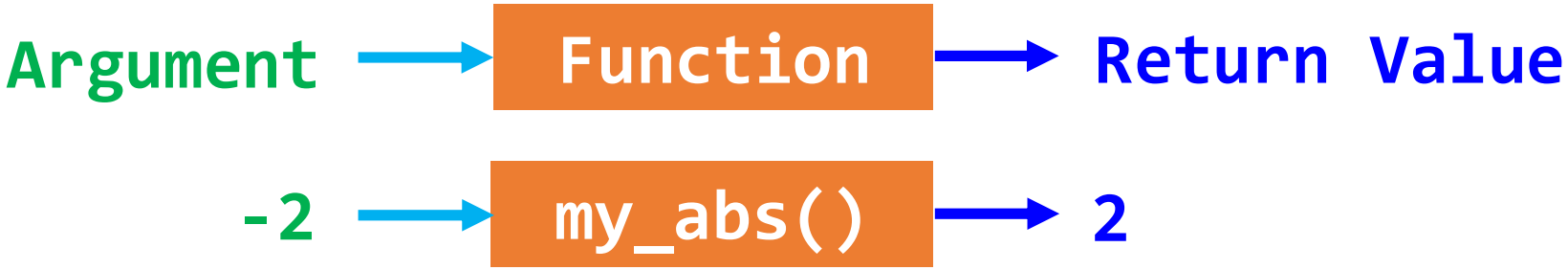
```
x = my_abs(-2)
print(x)
```

Result:

```
2
```

Details of Calling a function





Indentation Matters

Indented: function body
Belongs to the function definition

Non-indented:
Does not belong to the function definition

```
def my_abs(number):
    if number < 0 :
        number = - number
    return number

x = my_abs(-2)
print(x)
```

Combine the code

```
def my_abs(number):
    if number < 0 :
        number = - number
    return number


x = my_abs(-2)
print(x)
```

```
print(my_abs(-2))
```

Call a function multiple times

```
def my_abs(number):
    if number < 0 :
        number = - number
    return number

print(my_abs(-2))
print(my_abs(0))
print(my_abs(5))
```

Result:

```
2
0
5
```

Call a function multiple times

```
def multiply(num1, num2):
    result = num1 * num2
    return result


print(multiply(2,3))
print(multiply(4,5))
print(multiply(-1,2))
```

Result:

```
6
20
-2
```

Important Notes

When a function is **defined**, the code inside the function body is idle
(i.e. code is not ran/not executed)

The code in the function body is only ran/executed when the function is **called**

Run function body only on call

```
def simple_function():
    return "Hello World!"

print("No result yet!")
print(simple_function())
```

Function call

Result:

```
No result yet!
Hello World
```

Function Definition Syntax (Recap)

Function header

```
def <function_name>(<parameter1, parameter2, …>):
    <statements>
    return <result>
```

Indented Function Body

```
def multiply(num1, num2):
    r = num1 * num2
    return r
```

Process

Return the result

Function Calling Syntax (Recap)

```
<function_name>(<argument1, argument2, …>)
```

```
def multiply(num1, num2):
    r = num1 * num2
    return r


x = multiply(2, 3)
print(x)
```

Function Calling Example:

```
def multiply(num1, num2):
    r = num1 * num2
    return r


n1 = input("Enter number 1: ")
n2 = input("Enter number 2: ")
n1 = int(n1)
n2 = int(n2)
x = multiply(n1, n2)
print(x)
```

Function Calling Example:

Is my function correct?

1. Test the function!
2. By calling it with some test data as argument.
3. Display the return value

```
def my_abs(number):
    if number < 0 :
        number = - number
    return number

x = my_abs(-2)
print(x)
```

```
print(my_abs(-2))
```

How many test data is enough?

Function **abc** returns different result for different argument

- If argument is positive, return 1
- If argument is negative, return -1
- if argument is zero, return 0

```
def abc(number):
    ……
    return result
print(abc(-2))
print(abc(3))
print(abc(0))
```

Test at least 3 values!

Test your function

- Test your function thoroughly in Wing IDE before copying the code into mycodeschool.net
- Copy **which part**?

```
def abc(number):
    ……
    return result
print(abc(-2))
print(abc(3))
print(abc(0))
```

Only the function definition, not the testing code!