Introduction to Computer Programming
Lecture 4.1:

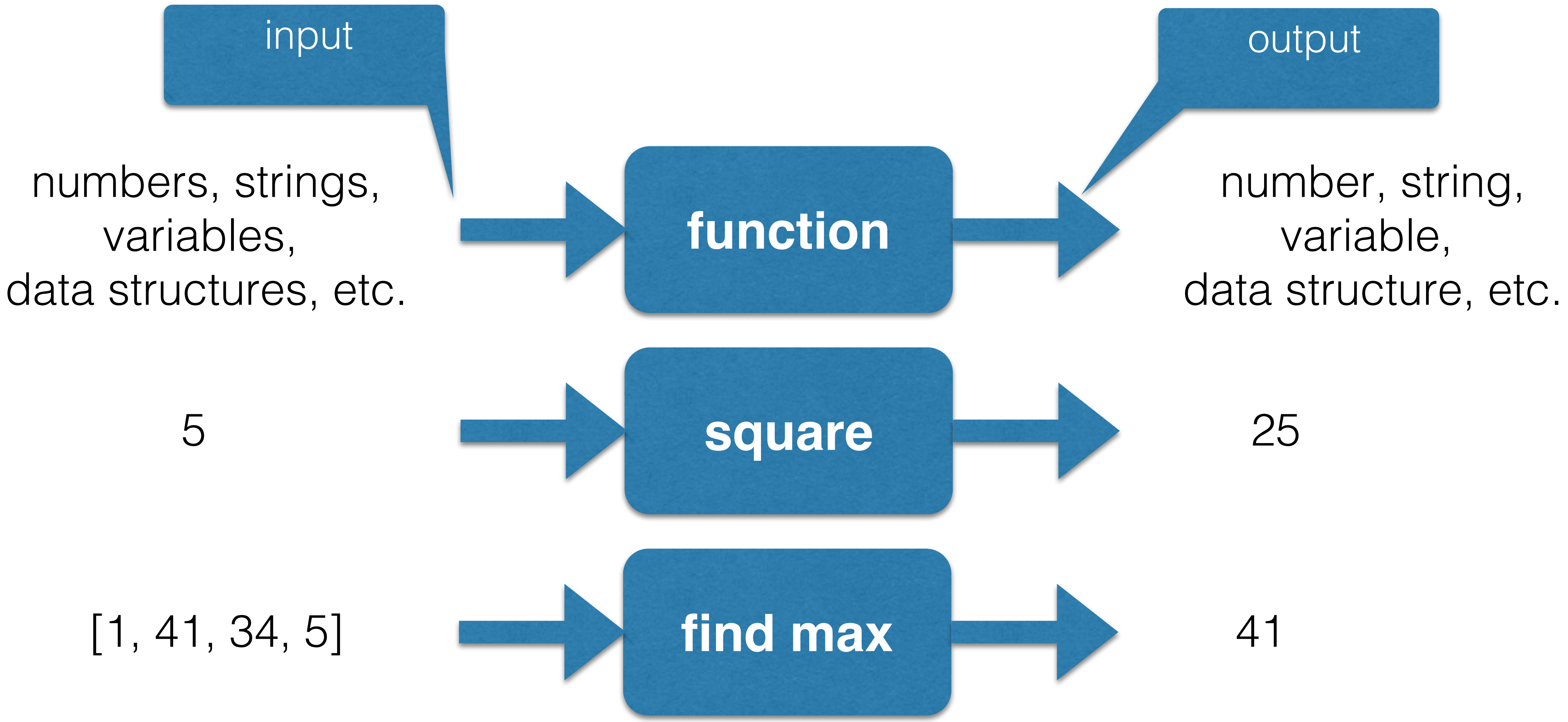
Functions in Python

Hemma Philamore

Department of Engineering Mathematics

Functions

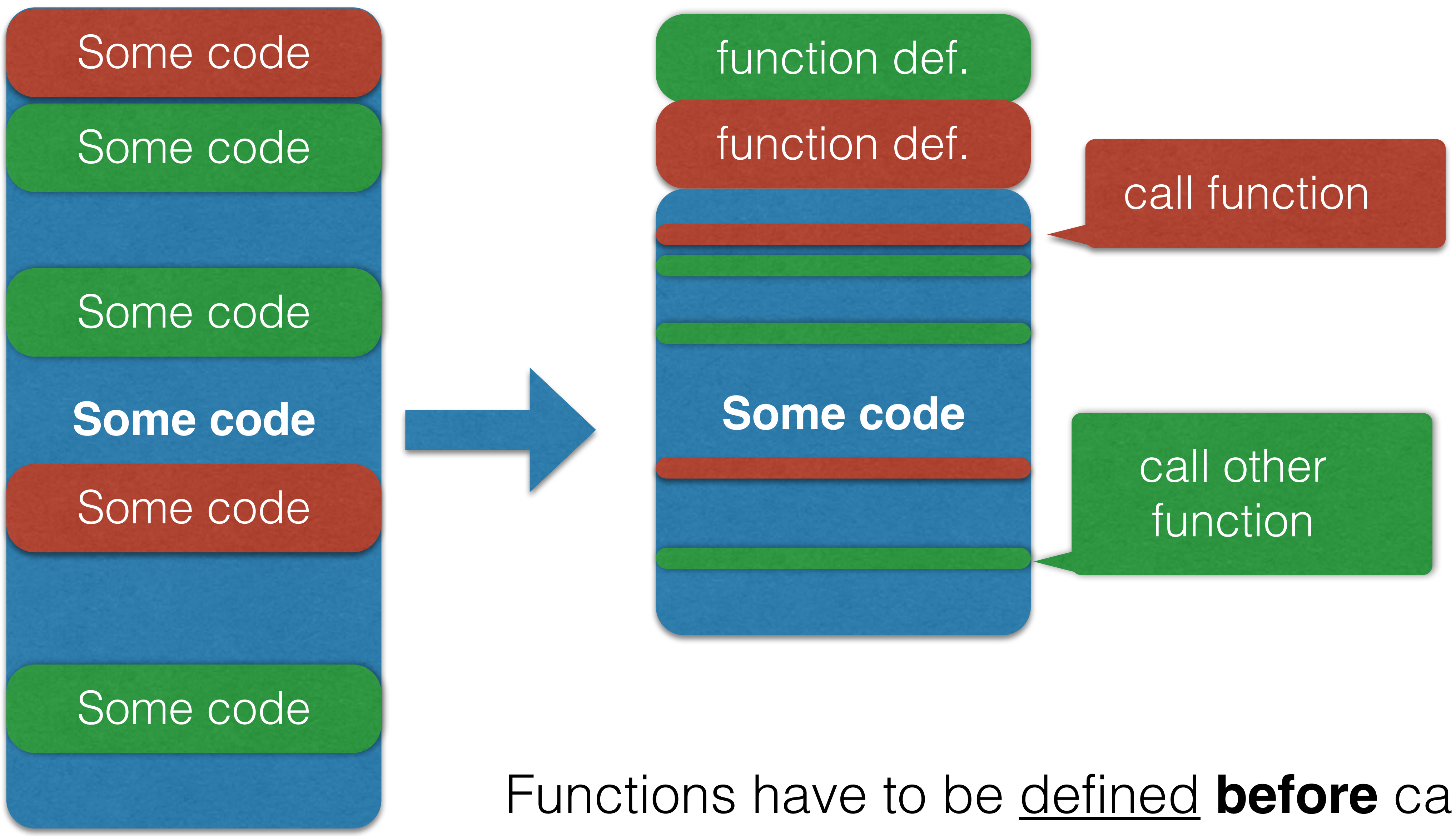
Functions are blocks of code that can be reused throughout your program.


input
output
numbers, strings, variables, data structures, etc.
function
number, string, variable, data structure, etc.
5
square
25
[1, 41, 34, 5]
find max
41

Functions

Functions often lead to **cleaner code** that is easier to understand and maintain.

Code is **typically shorter/smaller** and **more reusable**.

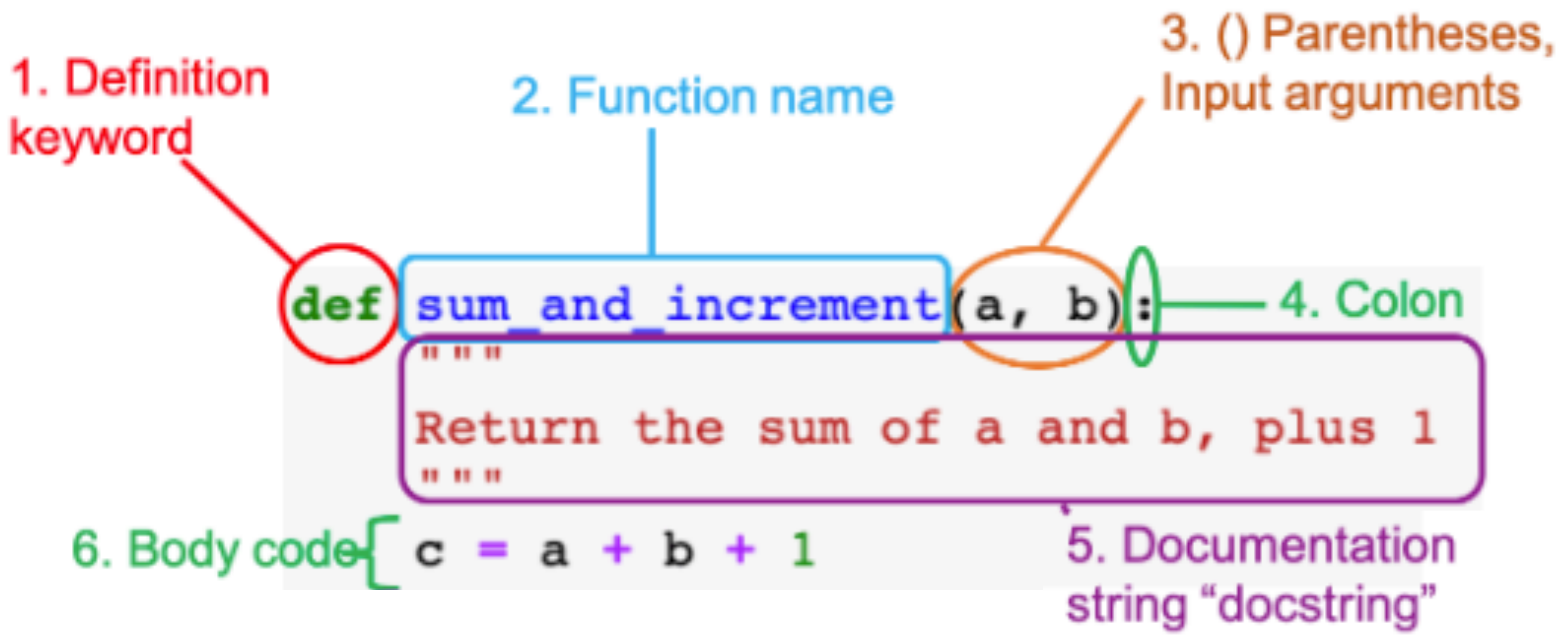


Functions have to be defined **before** called!

Function definition checklist

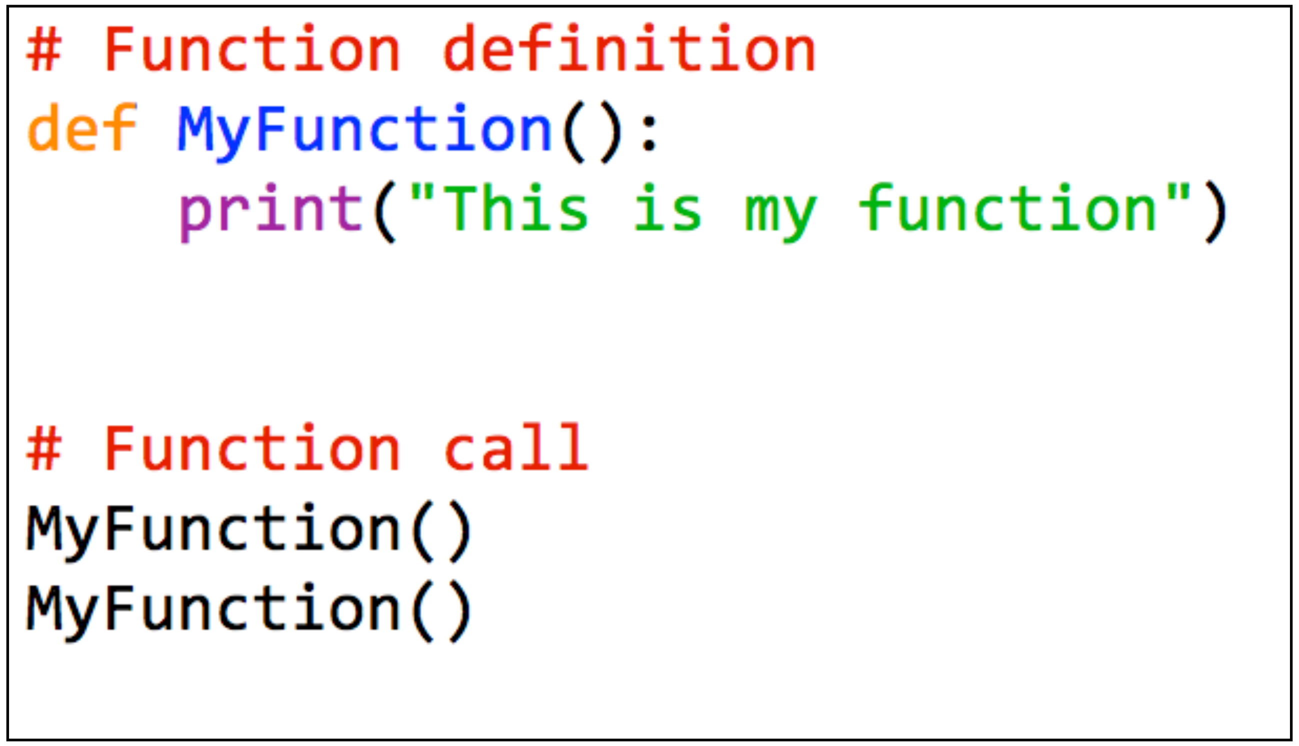
A function is **declared** using:

1. The definition keyword, `def`.
2. A **function name** of your choice.
3. **() parentheses** which optionally contain **arguments** (the *inputs* to the function)
4. **: a colon** character
5. A **documentation string** that says what the function does.
6. The **body code** to be executed when the function is *called*.


1. Definition keyword
2. Function name
3. () Parentheses, Input arguments
4. Colon
5. Documentation string "docstring"
6. Body code

```
def sum_and_increment(a, b):
    """
    Return the sum of a and b, plus 1
    """
    c = a + b + 1
```

Function Example



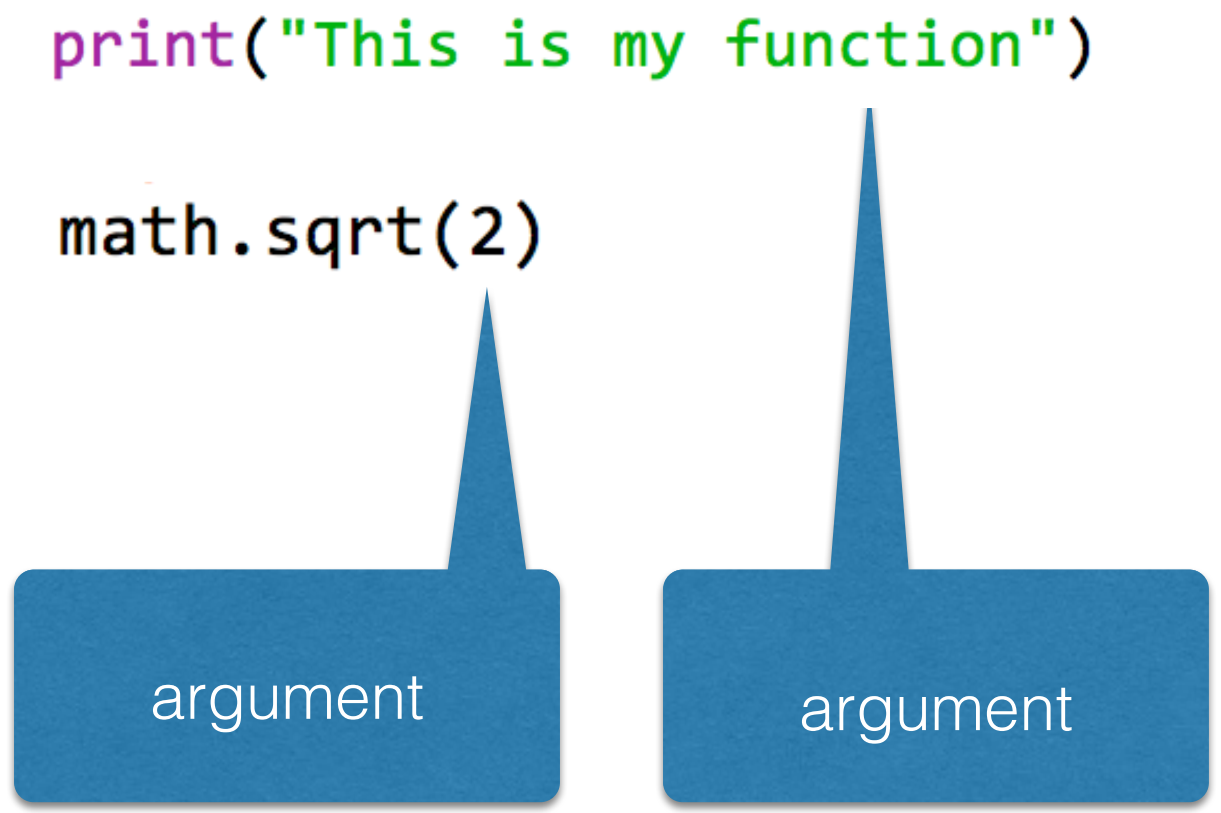
```python
# Function definition
def MyFunction():
    print("This is my function")


# Function call
MyFunction()
MyFunction()
```

```
This is my function
This is my function
```

Functions

You already know a number of predefined functions such as

- print(…)
- len(…)
- type(…)
- sqrt(…)
- etc.



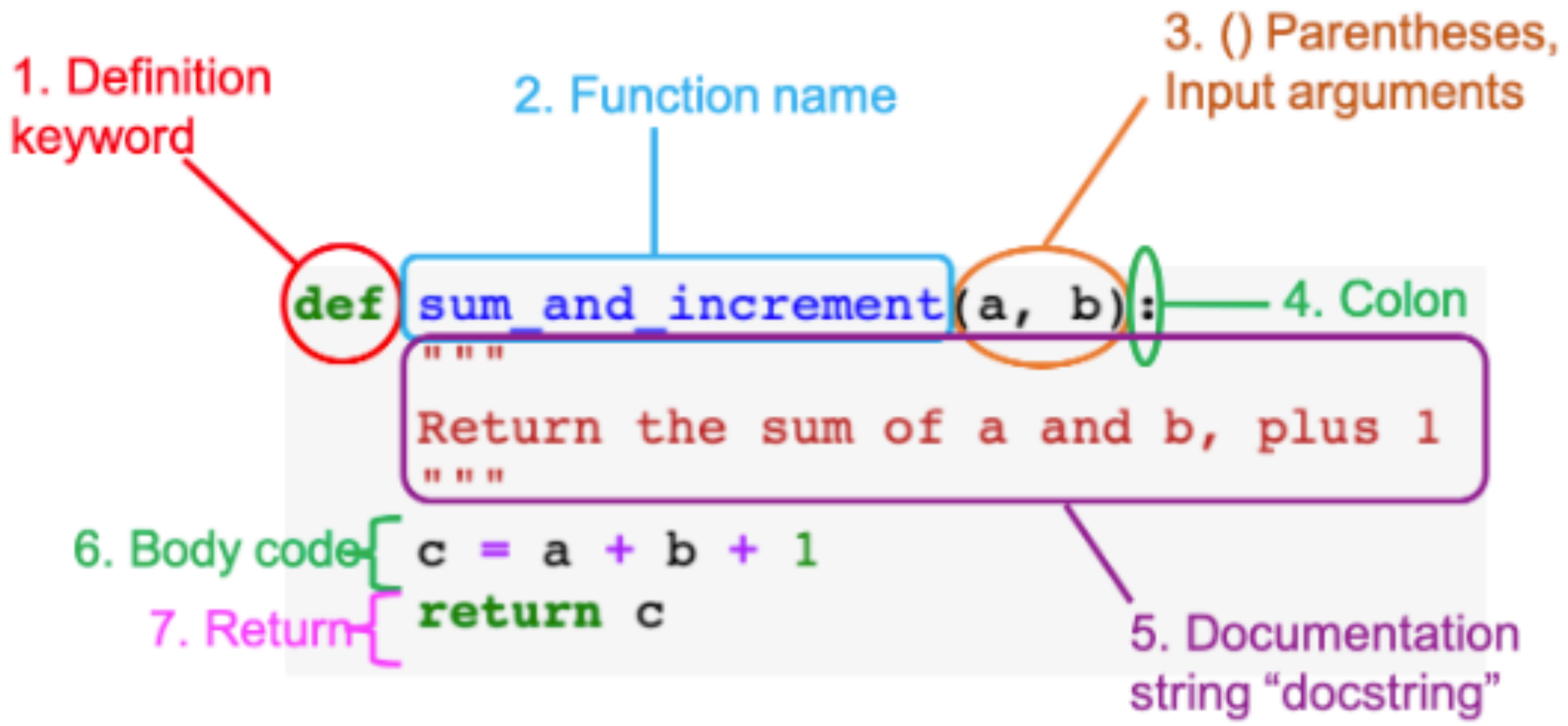
```
print("This is my function")

math.sqrt(2)
```

Function definition checklist: return

The results can be **returned in a variable**

A function is **declared** using:

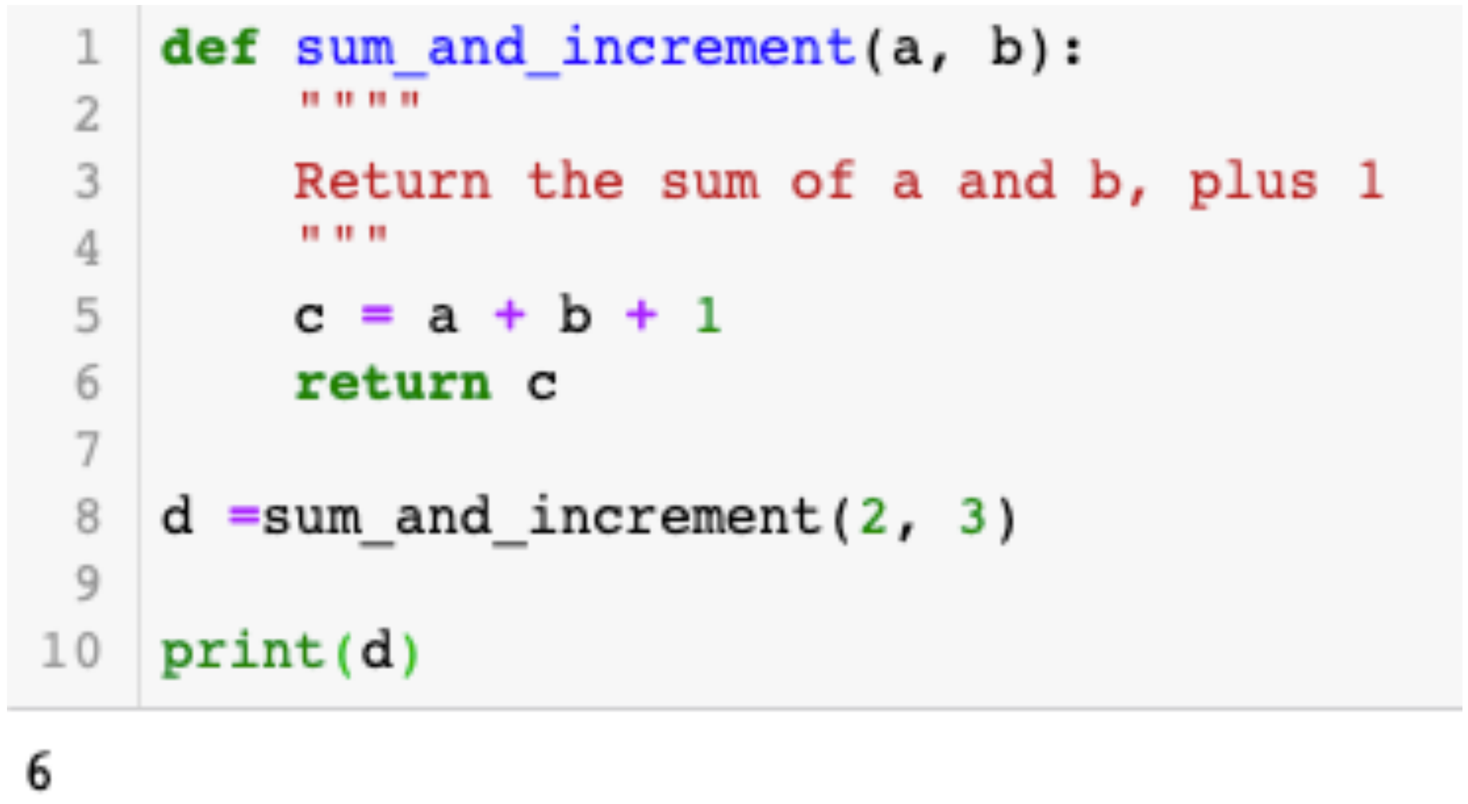
1. The definition keyword, `def`.
2. A **function name** of your choice.
3. **() parentheses** which optionally contain **arguments** (the *inputs* to the function)
4. **: a colon** character
5. A **documentation string** that says what the function does.
6. The **body code** to be executed when the function is *called*.
7. An optional **return** statement (the *output* of the function)



```python
def sum_and_increment(a, b):
    """
    Return the sum of a and b, plus 1
    """
    c = a + b + 1
    return c
```

1. Definition keyword
2. Function name
3. () Parentheses, Input arguments
4. Colon
5. Documentation string "docstring"
6. Body code
7. Return

Function Example



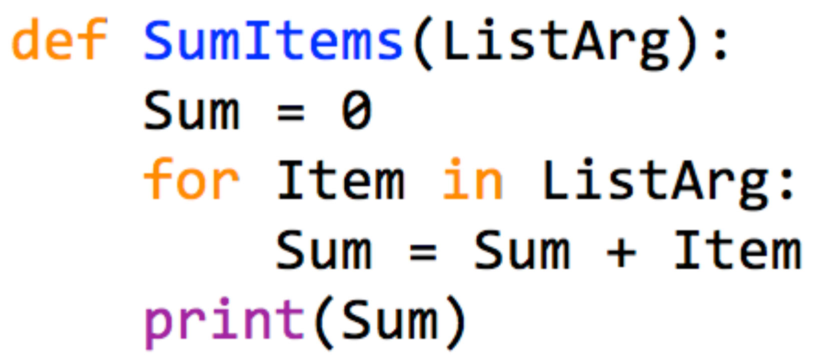
```python
def sum_and_increment(a, b):
    """"
    Return the sum of a and b, plus 1
    """
    c = a + b + 1
    return c

d =sum_and_increment(2, 3)

print(d)
```

```
6
```

Return Statements

The results can be **returned in a variable**

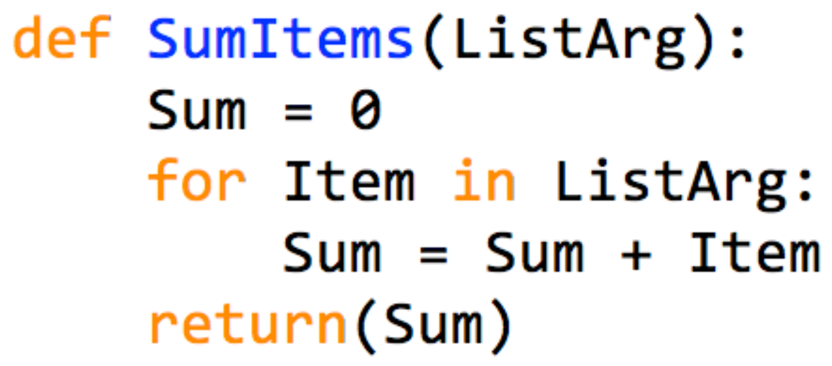
```
ListArg = [1, 2, 3]
```



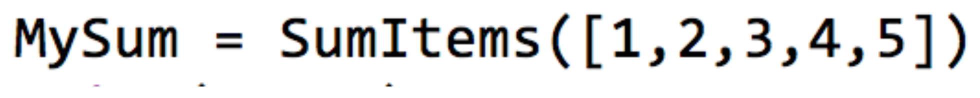
```
def SumItems(ListArg):
    Sum = 0
    for Item in ListArg:
        Sum = Sum + Item
    print(Sum)
```

```
SumItems([1,2,3])
```

```
6
```



```
def SumItems(ListArg):
    Sum = 0
    for Item in ListArg:
        Sum = Sum + Item
    return(Sum)
```



```
MySum = SumItems([1,2,3,4,5])
```

```
print(MySum)
```

```
6
```

```
print(MySum * 2)
```

```
12
```

Return Statements

If you return more values, Python converts into a tuple.

```
import math
def MinMax(ListArg):
    Min = math.inf
    Max = -1*math.inf
    for Item in ListArg:
        if Item < Min:
            Min = Item
        if Item > Max:
            Max = Item
    return(Min,Max)
```

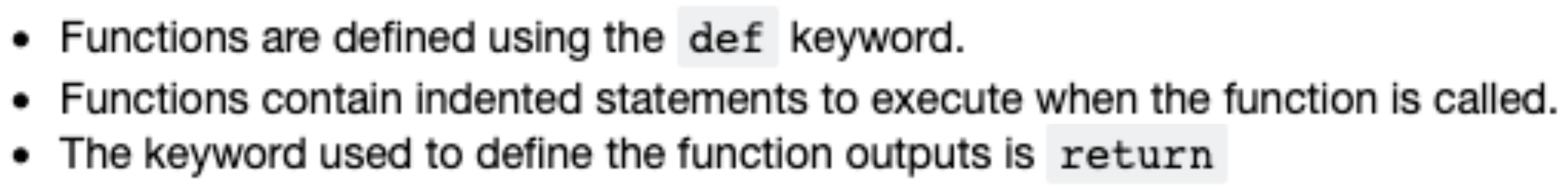
Python returns here a tuple with 2 element

This is variable is a tuple

```
MyMinMax = MinMax([math.pi,1.934,-3,45,9])
print("Min: " + str(MyMinMax[0]))
print("Max: " + str(MyMinMax[1]))
```

individual values can be accessed

Summary



- Functions are defined using the `def` keyword.
- Functions contain indented statements to execute when the function is called.
- The keyword used to define the function outputs is `return`