JÖNKÖPING UNIVERSITY

School of Engineering

FUNCTIONS IN PYTHON

Peter Larsson-Green

Jönköping University

Autumn 2018

COMPUTING MULTIPLE SUMS

```
sum = 0
for i in range(4):
  sum = sum + i

sum2 = 0
for i in range(6):
  sum2 = sum2 + i
```

Duplicated a lot of code ☹

- We can copy-paste code with errors.
  - Need to fix the code at multiple places…
- We may come up with a better algorithm.
  - Need to change the code at multiple places…

Functions to the rescue! ☺

FUNCTIONS

A function = a value that contains code
(a sequence of statements).

- Is usually stored in a variable.

Syntax:

```
def variable():
    Statement 1
    Statement 2
    Statement ...
```

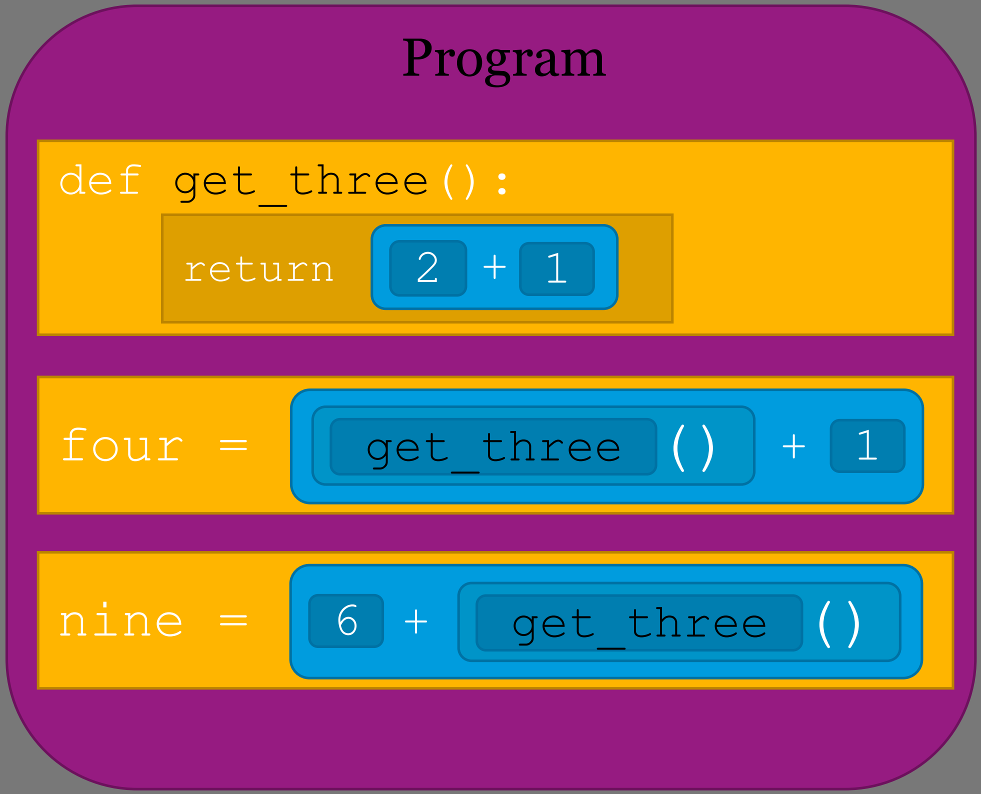
- Statements in the function are executed when the function is called.
  - The call expression: `<expr>()`
- Can return a value.
  - The return statement: `return <expr>`

FUNCTION EXAMPLE

```
def get_three():
    return 2 + 1

four = get_three() + 1

nine = 6 + get_three()
```



FUNCTION EXAMPLE

```
# Let's be a computer and execute the statements!
def get_three():
   return 2 + 1


four = get_three() + 1


nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three(): # I create a function...
   return 2 + 1  # ...consisting of this statement...
                 # and store it in the variable get_three.
four = get_three() + 1

nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three():
   return 2 + 1

four = get_three() + 1 #To compute the value I should store in four
                       #I need to call the function get_three.
nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three(): # So I call this function,
   return 2 + 1  # and start to execute the statements in it.

four = get_three() + 1

nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three():
    return 2 + 1 # I return: 2+1 → 3.

four = get_three() + 1

nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three():
   return 2 + 1

four = get_three() + 1 # I store get_three()+1 → 3+1 → 4 in four.

nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three():
   return 2 + 1

four = get_three() + 1

nine = 6 + get_three()#To compute the value I should store in nine,
                      #I need to call the function get_three.
```

FUNCTION EXAMPLE

```
def get_three(): # So I call this function,
   return 2 + 1  # and start to execute the statements in it.

four = get_three() + 1

nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three():
    return 2 + 1 # I return: 2+1 → 3.

four = get_three() + 1

nine = 6 + get_three()
```

FUNCTION EXAMPLE

```
def get_three():
    return 2 + 1

four = get_three() + 1

nine = 6 + get_three() # I store 6+get_three() → 6+3 → 9 in nine.
```

FUNCTION EXAMPLE

```
def get_three():
   return 2 + 1

four = get_three() + 1

nine = 6 + get_three()
# And I'm done!
```

PRACTICAL EXAMPLE

GIVING FUNCTIONS INPUT

```
def sum_of_ints():
    sum = 0
    for i in range(6):
        sum = sum + i
    return sum


fifteen = sum_of_ints()
```

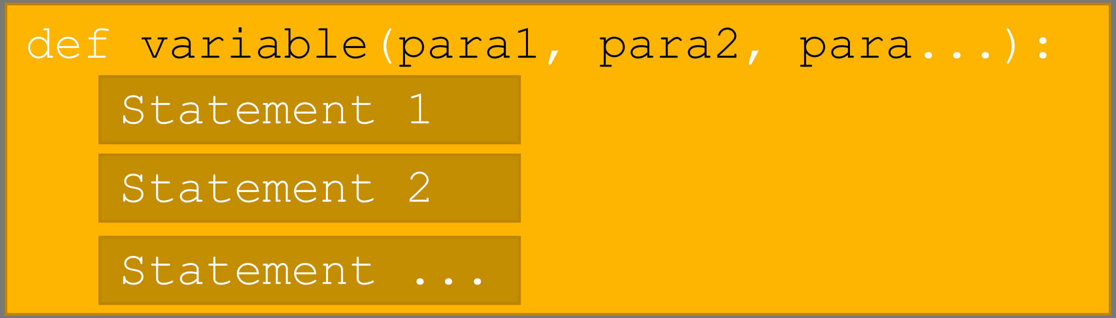
How can we change this 6 each call?

```
last_int = 0
def sum_of_ints():
    sum = 0
    for i in range(last_int):
        sum = sum + i
    return sum
last_int = 6
fifteen = sum_of_ints()
```

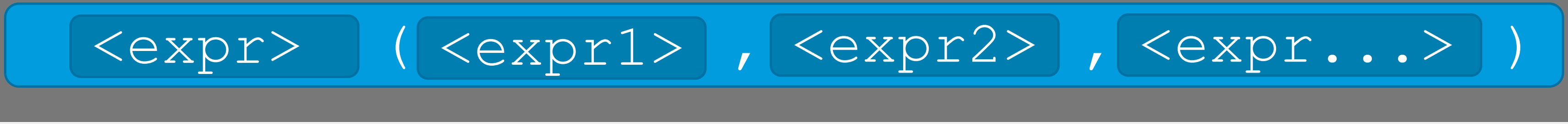
PARAMETERS AND ARGUMENTS

```
def add(number_a, number_b):
   return number_a + number_b


three = add(1, 2)
five = add(4, 1)
```



```
def variable(para1, para2, para...):
    Statement 1
    Statement 2
    Statement ...
```



```
<expr> (<expr1> , <expr2> , <expr...> )
```

THE FINAL SOLUTION

```
def sum_of_ints(last_int):
    sum = 0
    for i in range(last_int):
        sum = sum + i
    return sum

fifteen = sum_of_ints(6)
```

Poorly named variable (it is not added to the returned sum!).

PRACTICAL EXAMPLE