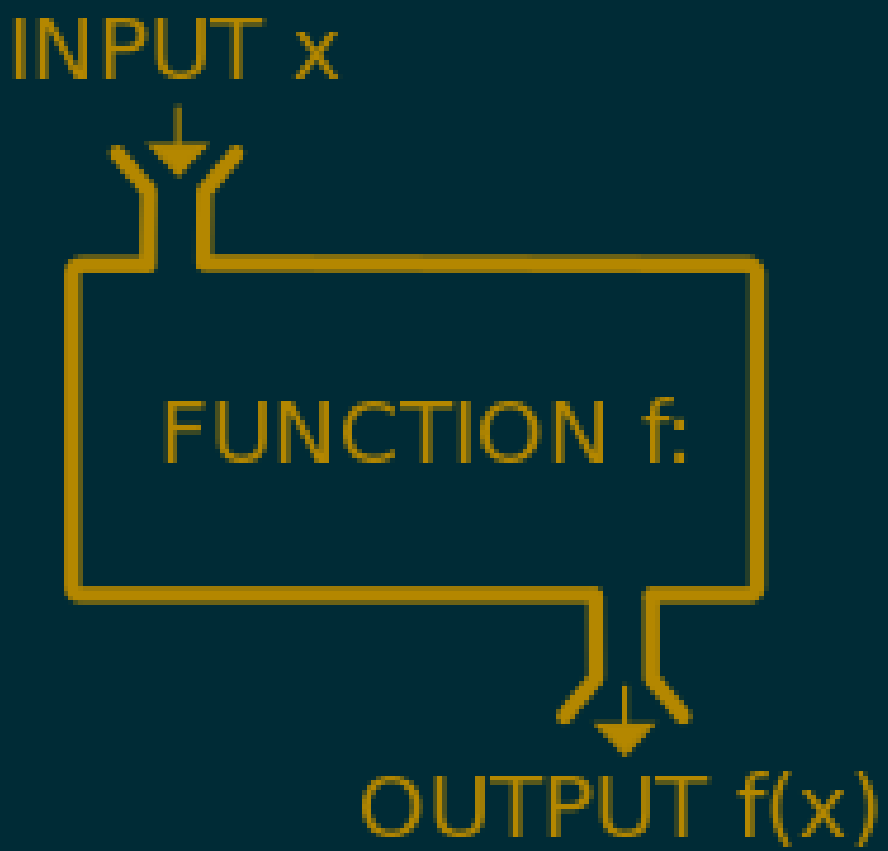
LEARN CODING

ale66

github.com/ale66/learn-coding

FUNCTIONS



Functions are a key abstraction to model nature and processes

a regular input/output or cause/effect behaviour is identified and *given a name*

```
1  The higher the temperature the quicker the veggies cook.
2
3  Cooking time is a function of the temperature in the oven.
```

FUNCTIONS IN CODING

A function is a block of code that

- has a clear input/output definition and
- executes in a separated environment

```
1  def marks2pc(marks: int):
2    ''' Convert Italian exam marks into percentages '''
3
4    pc = int((marks / 30) * 100)
5
6    return pc
```

`marks` is a *parameter* of the f.

`pc` is the *return value* of the f.

OBSERVATIONS

Functions only run when they are called (‘invoked’) within a code in execution

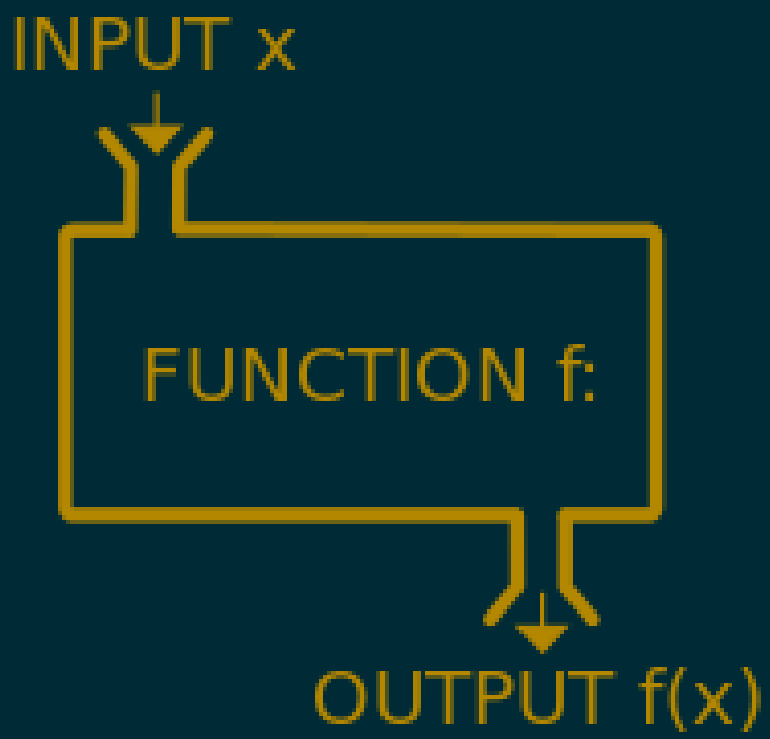
```
1  for m in my_italian_exam_marks:
2    uk_marks = marks2pc(m)
3    print(uk_marks)
```

Functions should be defined every time a block of code is required to appear more than once:

- improve readability
- improve maintainance

SYNTAX

```
1  def <name>(<parameter(s)>):
2    return <output(s)>
```



KINDS OF FUNCTIONS:

- **built-in:** come with Python, e.g., `input()`, `print()`, `float()`, `int()` etc.
- **methods:** come with types, e.g., `mylist.len()`, `mylist.append('a')`
- **external** are *imported* into the code upon demand (more later)
- **defined** we define then use them

Names of the built-in functions are reserved: avoid them as variable names

MULTIPLE PARAMETERS AND ARGUMENTS

- functions can take two or more arguments, and even an arbitrary number of them
- mapping arguments to parameters is done by position
- it is also possible to return multiple arguments

```
1  first_out, second_out =  func(first_in, second_in)
2
3  def func(primero, segundo):
4
5    # ...
6
7    return tercero, quarto
```

what are the bindings?

MULTIPLE INPUTS EXAMPLE

Given a text and a character, count the number of occurrences of it

```
1 def char_counter(text, c):
2   '''returns the number of times we found c in text'''
3
4   # this cannot be empty
5   pass
```

FUNCTION NESTING EXAMPLE

```
 1  def find_tendencies(my_temperatures_list):
 2      '''
 3        Nested function calls
 4      '''
 5
 6      l = len(my_temperatures_list)
 7
 8      if l > 0:
 9          min_val = min(my_temperatures_list)
10          max_val = max(my_temperatures_list)
11          avg_val = sum(my_temperatures_list) / l
12
13          # Return multiple values separated by commas (creates a tuple)
14          return min_val, max_val, avg_val, l
15
16      else:
17        print('Empty temperature list, defaulting to None')
18        return None, None, None, 0
```

VISIBILITY RULES ARE ENFORCED

Why is this giving error?

```
1  first_out, second_out =  func(first_in, second_in)
2
3  def func(primero, segundo):
4
5    # print(f'inside here, primero is really first_in: {primero} = {first_is}')
6
7    return tercero, quarto
```

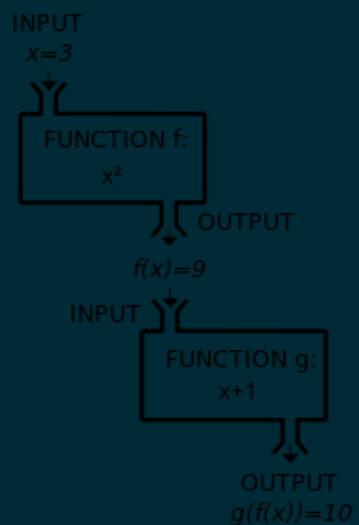
A VERSION WITH TYPES

Expliciting data types helps in catching coding errors as soon as possible

```
1  def char_counter(text: str, c: chr) -> int:
2    '''returns the number of times we found c in text'''
3
4    # this cannot be empty
5    pass
```

NESTING

The **return** value of a f. could be the **input parameter** to another




```
1  def square(x):
2    return x**2
3
4  def average_of_three(x, y, z):
5    return (x+y+z)/3
6
7  print(average_of_three(square(10), square(20), square(30)))
```

`void` FUNCTIONS

Functions may *not* return a value

```
 1  def greet(lang):
 2
 3    if lang == 'es':
 4      print('Hola')
 5
 6    elif lang == 'fr':
 7      print('Bonjour')
 8
 9    else:
10      print('Hello')
11
12  greet('es')
```

they don’t need to be to the right of an assignment symbol

FURTHER OBSERVATIONS

An *argument* is what the caller code sends to the function, e.g., `'es'` in the example

A *parameter* is the local (to the function) given value, e.g., `lang` in the example

Remember: functions are executed in a container and only *see* their parameters, not the variables and values of the calling code

QUIZZES

QUIZ 1/4

What is the default return value for a function that does not return any value explicitly?

- `None`
- `int`
- `null`
- `str`

QUIZ 2/4!

Which of the following items are present in the function header?

- function name
- function name and parameter list
- parameter list
- return value

QUIZ 3/4!

Which of the following keywords marks the beginning of the function block?

- `fun`
- `define`
- `def`
- `function`

QUIZ 4/4!

Which of the following function definition does not return any value?

- print all integers from 1 to 100.
- return a random integer from 1 to 100.
- check whether the current second is an integer from 1 to 100.
- convert an uppercase letter to lowercase.

QUIZ ANSWERS!

What is the default return value ...?

- None

Which ... present in the function header?

- function name and parameter list

Which ... marks the beginning of the function block?

- `def`

Which ... does not return any value?

- print all integers from 1 to 100