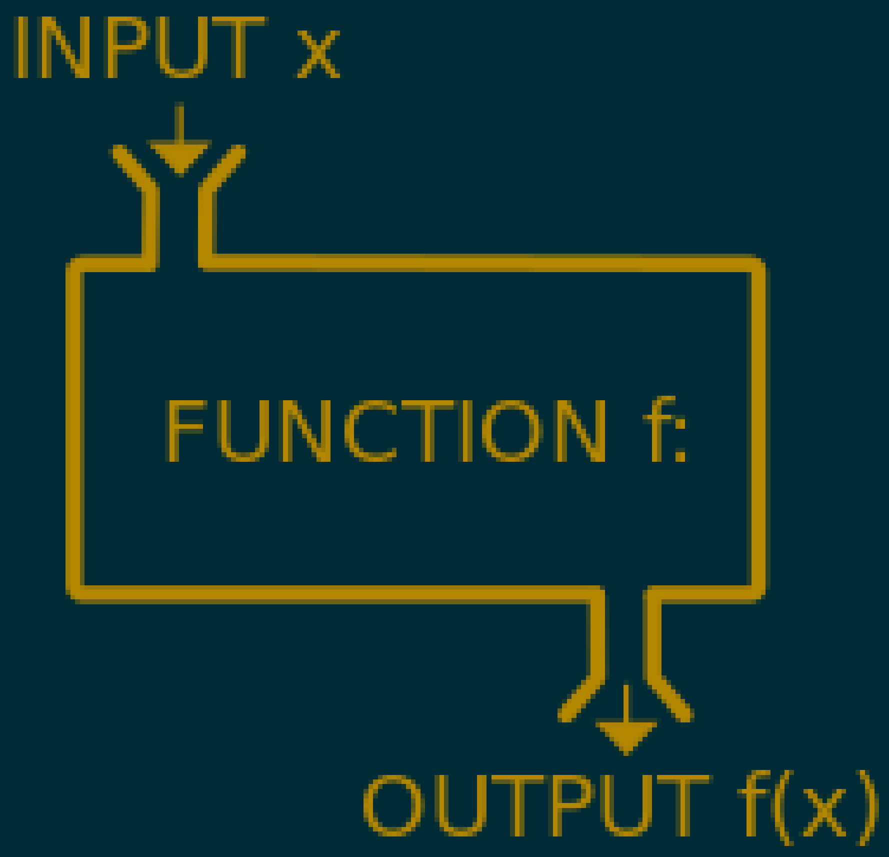
LEARN CODING

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*

```
The higher the temperature the quicker pizza cooks.

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

```python
def marks2pc(marks: int):
  ''' Convert Italian exam marks into percentages '''

  pc = int((marks / 30) * 100)

  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

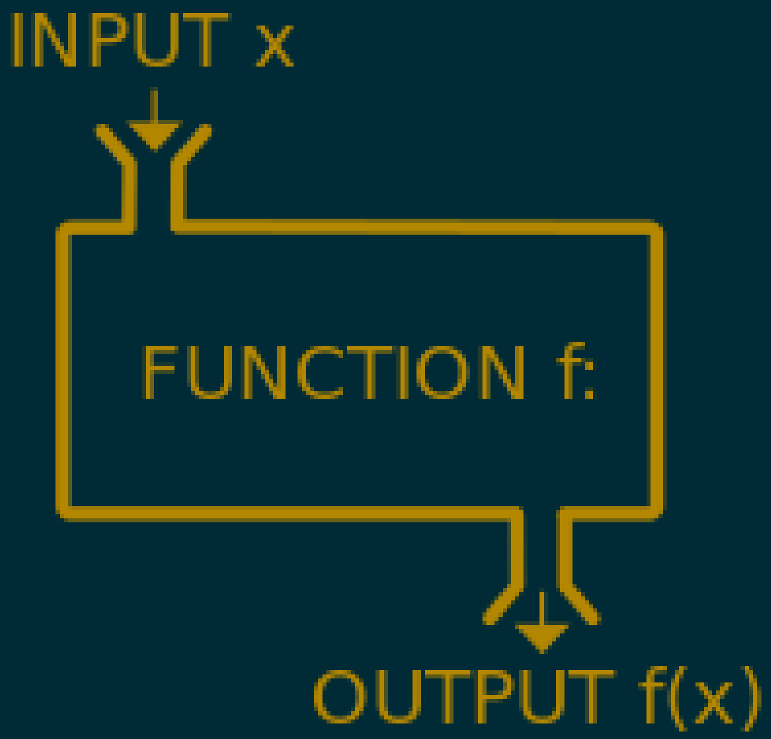
```python
for m in my_italian_exam_marks:
  uk_marks = marks2pc(m)
  print(uk_marks)
```

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

- improve readability
- improve maintainance

SYNTAX

```
def <name>(<parameter(s)>):
  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

MULTIPLE INPUTS

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

```python
def char_counter(text, c):
  '''returns the number of times we found c in text'''

  # this cannot be empty
  pass
```

On to live coding session

A VERSION WITH TYPES

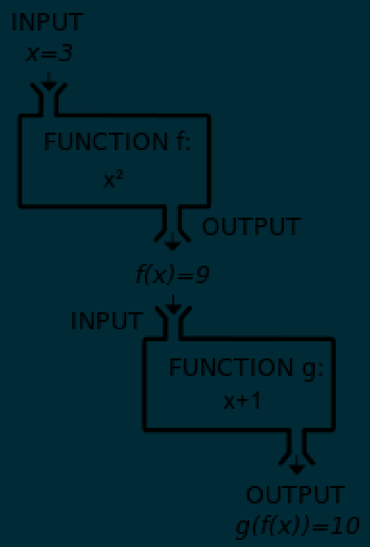
Expliciting data types helps in catching coding errors as soon as possible

```python
def char_counter(text: str, c: chr) -> int:
    '''returns the number of times we found c in text'''

    # this cannot be empty
    pass
```

NESTING

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



```python
def square(x):
  return x**2

def average_of_three(x, y, z):
  return (x+y+z)/3
```

void FUNCTIONS

Functions may *not* return a value

```python
def greet(lang):

  if lang == 'es':
    print('Hola')

  elif lang == 'fr':
    print('Bonjour')

  else:
    print('Hello')

greet('es')
```

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