4 - Functions in Python

Introduction to Python
efl Data Science Courses

Timo Schäfer

What is a function?

- Mathematically speaking, a function isa mapping *f* from an input *x* to an output *y*

$$f(x) = y$$

- In a programming language, a pseudo code looks as follows:

```
f(argument1, argument2, ...) = result
```

```
def functionName(argument1,argument2,…):
    do something
    return result
```

Introduction to functions in Python

- A function is an **executable statement**
- Very useful when, e.g., a **set of operations is applied repeatedly** → key concept **don't repeat yourself (DRY)**
    - Example: verify whether an integer is odd/even for several integers
- A function can have several **arguments** that are evaluated in the **body** of a function
- Results obtained in body can be **returned** that could be used for further calculations
- A function has to be **called** to be executed, i.e., a definition of a function itself does not execute a function's body
- Let's be more specific on the next slides!

Defines and initializes a function

```
def fDoubleMe(int1):
    result = 2*int1
    return result
```

Name of the function

```
def fDoubleMe(int1):
    result = 2*int1
    return result
```

Arguments

Can also be set of a **default parameter**, e.g., int1=12

```
def fDoubleMe(int1):
    result = 2*int1
    return result
```

Functions **without** any input **arguments** are possible as well, e.g., returning a predefined string:

```
def fNoArgs():
    result = "this is an example"
    return result
```

Body of the function

The variable `result` belongs to the local namespace of the function `doubleMe` and thus, cannot be accessed outside the function's body.

```
def fDoubleMe(int1):
    result = 2*int1
    return result
```

For instance, having defined `doubleMe` and executing following line

```
print(result)
```

throws an error because it is defined in a different namespace!

BUT: `doubleMe` has a reference to the current global namespace since the global namespace is used when `doubleMe` is called (see next slide what *calling* means).

Return result computed in function's body

Return result to a variable that **calls** the function `doubleMe`

```
def fDoubleMe(int1):
    result = 2*int1
    return result
```

For instance,

```
resultFromDoubleMe = fDoubleMe(12)
print(resultFromDoubleMe)
```

yields 24. **Any questions?**

Example: a function using conditional statements

```
def fIsPositive(number):
    if number>=0:
        return True
    elif number<0:
        return False
```

Control for the type of the input argument(s)

```
def fDoubleMe(int1):
    if type(int1) is int:
        result = 2*int1
        return result
    else:
        print("int1 must be of type int.")
        return
```

Some exercises for writing functions in Python

Write a function

- `fSumElems` that sums up all elements in a list and returns a scalar,
- `fSquareElems` that squares all elements in a list and returns a list.

Assume that all elements in the list are of type numeric and, for the sake of simplicity, we do not control for other types at the moment.

Afterwards, verify that these functions work properly by testing them on the example lists provided in the code skeleton.

Solution

```
def fSumElems(list1):
    ##use a loop
    sume = 0
    for i in list1:
        sume += i
    return sume

def fSquareElems(list1):
    sq = [i*i for i in list1]
    return sq
```

- Does everyone understand the definition of `sq`?

Cont'd: compute mean and variance

- Write a function `fMean` that computes the average of a list. This function shall use the function `fSumElems` that you programmed previously:

$$\text{mean}((x_1, x_2, \ldots, x_n)') = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- Write a function `fVariance` that computes the variance of a list. This function shall use the function `fMean` and `fSquareElems` that you programmed previously:

$$\text{var}((x_1, x_2, \ldots, x_n)') = \frac{1}{n} \sum_{i} x_i^2 - \text{mean}((x_1, x_2, \ldots, x_n)')^2$$

Solution

```
def fMean(list1):
    me = fSumElems(list1) / len(list1)
    return me

def fVariance(list1):
    var = fMean(fSquareElems(list1)) - fMean(list1)**2
    return var
```

Outlook: tip of the iceberg

- `lambda` operator for **anonymous functions**, i.e., a function that is not called by its name
  - Example:

```
squareMe = lambda x: x*x
```

- `map` operator: a “faster” alternative to loops
  - Instead of writing

```
for i in range(10):
    print(i*i)
```

  - You can write:

```
map(lambda x: x*x, range(10))
```

- Any many more (that are not part of this course)!

References

- https://docs.python.org/2.0/ref/function.html
- https://docs.python.org/3/library/functions.html#map
- https://docs.python.org/3/tutorial/controlflow.html