TØ 3. Extra

Table of contents

This note contains additional exercises regarding functions and variables.

1 Functions

Functions in Python are like recipes in a cookbook. Writing them down does not "run" them, just like how writing down a recipe for a delicious brownie does not cause a delicous brownie to appear.

To "run" a function we need to *call* it, for example

```
def my_func(x):
    result = 2 * x
    return result

output = my_func(2)
```

Line 1

Function definition, does not run the function.

Line 5

Function call, runs the functions for the input `x=2`.

Additionally, if we want the function to influence our program we need to assign its output to a variable[1]. Above we assigned the output of the function to the variable `output`, so we now have access to that variable

```
print("The output was: ", output)
```

```
The output was:  4
```

Then there is the concept of scope, the function we defined above defines a variable `result`, lets see what happens if we try to access it

```
print(result)
```

```
NameError: name 'result' is not defined
[31m---------------------------------------------------------------------------[39m
```

[1]This is a little bit imprecise. Functions can be written to change variables, or more generally the program state *in-place* or have other side-effects. However, the functions we will consider in this course are not of this type.

```
□[31mNameError□[39m                                 Traceback (most recent call last)
□[36mCell□[39m□[36m □[39m□[32mIn[3]□[39m□[32m, line 1□[39m
□[32m----> □[39m□[32m1□[39m print(result)

□[31mNameError□[39m: name 'result' is not defined
```

Python throws an error, telling us that `result` is not defined. This is because the `result`-variable is only available in the *scope* of the function. That means `result` only exists inside the function.

1.1 Exercise: Functions, calling, assignment and scopes

```
# A function that calculates something but doesn't return it
def function_without_return(A, B):
  result = A * B
  print("Inside function_without_return: ", result)

# A function that calculates something and returns it.
def function_with_return(A, B):
  result = A * B
  print("Inside function_with_return: ", result)
  return result
```

The code block above defines two functions.

Below there are three code blocks that use these functions and an accompanying question. Use the interactive codeblock below to answer these questions.

You are not expected to give technically precise answers, just describe it in your own words. For each question you can play around with the interactive block or the definition of the functions above to help you answer.

1.1.a Without assignment

What happens when the functions are called but the output not assigned?

```
# Call the functions but don't assign to a variable
print('Calling functions without assigning to a variable')
function_without_return(6, 7)
function_with_return(6, 7)

print('End of cell')
```

1.1.b With assignment

What happens when the functions are called and the output is assigned?

```
print('Calling functions and assigning')
without_return = function_without_return(6, 7)
with_return = function_with_return(6, 7)
```

```
print('without_return: ', without_return)
print('with_return: ', with_return)
print('End of cell')
```

1.1.c Without calling

What happens when you assign the function itself to a variable without calling it?

```
print('Assigning to a variable but not actually calling')
without_return_no_call = function_without_return
with_return_no_call = function_with_return

print('without_return_no_call: ', without_return_no_call)
print('with_return_no_call: ', with_return_no_call)
print('End of cell')
```