Students have to write an elaboration, where they have to transfer the learned concepts into their existing knowledge of OOP. In the written elaboration students have to:

1. Discuss the concepts learned in this course in the context of another programming language of their 🡪 **PYTHON**  
   using code sketches with written explanations and discussions. The following concepts have to be discussed:

**Immutable Data**

The principle of Immutable Data compromises an object whose internal state can not be changed once it has been created. In Python there are built-in types that are immutable, nevertheless there are also mutable types like lists, sets, dictionaries and other user-defined classes.  
The immutable types of Python are Numbers, Strings, Tuples, Frozen Sets and immutable user-defined classes.

When creating a value the space in memory gets allocated and the variable points to that place in memory. If now the same variable gets used for another value the first value still exists in memory, but the binding to it is lost.

A picture containing text, electronics

Description automatically generated

Even though it seems that you can change the value of a variable, Python will not allow it and will make a copy of the variable. This mechanism is called copy-on-modify.

x = 100  
print('Type: ', type(x), '\n',  
 'Memory id: ', id(x))  
#Type: <class 'int'>   
#Memory id: 140715477443440  
  
x = 200  
print('Type: ', type(x), '\n',  
 'Memory id: ', id(x))  
# Type: <class 'int'>  
# Memory id: 140715477446640  
  
x = float(x)  
print('Type: ', type(x), '\n',  
 'Memory id: ', id(x))  
# Type: <class 'float'>   
# Memory id: 1974826931504

In contrast to that lists are mutable in python and are changeable without changing the position in memory.

myList = [1, 2, 3, 4, 5]  
print('Type: ', type(list), '\n',  
 'Memory id: ', id(list))  
# Type: <class 'type'>  
# Memory id: 140715477207088  
  
myList = [12, 33, 11, 42]  
print('Type: ', type(list), '\n',  
 'Memory id: ', id(list))  
# Type: <class 'type'>   
# Memory id: 140715477207088

<https://www.mygreatlearning.com/blog/understanding-mutable-and-immutable-in-python/>

<https://freecontent.manning.com/mutable-and-immutable-objects/>

* + Type Variables

In Python variable type annotations are not enforced, so the variables do not need a declaration to reserve memory. By assigning a value to a variable the declaration happens automatically.

Furthermore, Python allows to change the type of a variable during runtime. It is possible for example to assign an integer value to a variable and change it to another value of type string.

At runtime a type can be checked by using the built in type() function. This can be very useful, as the declaration happens automatically and the type can not be

<https://docs.python.org/3.5/extending/newtypes.html>

<https://python-course.eu/python-tutorial/data-types-and-variables.php>

* + Higher-Order Functions

Functions are called Higher Order Functions if they contain other functions as parameter or return a function as value. Python also supports the concept of Higher Order Functions.

A picture containing text, person

Description automatically generated

There are different ways to define Higher Order Functions in python:

* Passing functions as parameter for another function
* Returning a function from another function
* Using decorators as High Order Function

The most common way of Higher Order Functions is to pass them as parameter to another function. This can be accomplished by assigning a reference of a function to a variable. This variable can then be used to pass it as parameter to other functions.

# pass function to function  
def calculateSum(nums):  
 sum = 0  
 for num in nums:  
 sum += num  
 return sum  
  
  
def calculateProduct(nums):  
 prod = 1  
 for num in nums:  
 prod = prod \* num  
 return prod  
  
  
def calculator(func, nums):  
 solution = func(nums)  
 print(solution)  
 return solution  
  
nums = [1, 2, 3, 4]  
calcSum = calculateSum  
calcProduct = calculateProduct  
calculator(calcSum, nums)  
calculator(calcProduct, nums)

<https://www.geeksforgeeks.org/higher-order-functions-in-python/>

<https://medium.com/analytics-vidhya/higher-order-functions-python-716f508a8f41>

<https://www.javatpoint.com/python-high-order-function>

* + Lambda Expressions

Lambda Expressions are small anonymous functions that take any number of arguments but can only have on expression. Lambda functions are mostly used as anonymous functions inside another function.

def myfunc(n):  
 return lambda a: a \* n  
  
mydoubler = myfunc(2)  
mytrippler = myfunc(3)  
  
print(mydoubler(11))  
print(mytrippler(11))

<https://realpython.com/python-lambda/>

* + Currying

<https://python-course.eu/advanced-python/currying-in-python.php>

* + Function Composition and Streaming

<https://mathieularose.com/function-composition-in-python>

streaming: <https://docs.python.org/3/library/io.html>

* + Algebraic Data Types

<https://www.gidware.com/python-adts/>

* + Pure and Impure Side Effects

<https://stackoverflow.com/questions/20027087/how-to-judge-or-how-to-write-a-python-function-with-no-side-effects>

1. Implement the State Monad in the selected language and demonstrate its use through a simple example such as tree labeling.

<https://gaius.tech/2010/09/06/on-monads/>

<https://medium.com/swlh/monads-in-python-e3c9592285d6>