INFPROG2 P01 – Objects

1. Recap Python Knowledge

Reactivate your programming skills. This involves properly structured scripts, appropriate code quality, robustness, and of course achieving a functional solution.

1.1 “Draw a match” game

The game (simple version)
The game starts with a stack of matches (**random** number: no less than 10, no more than 20). The players are playing one after another. A player can either draw **one**, **two** or **three** matches from the stack per turn. The player who draws the last match loses.

Your Task
Work out a concept for a Python program that can be used to let a human play against the computer. You need to define the rules and a strategy for the computer player. On each turn the actual stack should be displayed. This could look like following (abbreviated):

```
========O
========O
========O
========O
   ....
========O
========O
========O
|The stack has 12 matches.|
```

Use the `print()` and `input()` functions to interact with the human player. The game starts with a random number of matches. The player who starts (human or computer) is also random. Note: The use of object-oriented programming is still optional here.

Procedure
1. Think about how the workflow of the game looks like and how the computer needs to react on different situations. Write down a list of tasks that the program needs to solve.
2. Think about what functions it needs for all the tasks and how they interact.
3. Think about what data are needed when.
4. Set some rules for the computer player in order to let it draw the matches intelligently without losing every time. Can you come up with a mathematical formula?

2. Object Orientation

2.1 Classroom objects

Represent contents of a typical classroom (table, blackboard, student, teacher, document) as Python class structures, each with class name, constructor method and attributes. Some attributes may cross-reference objects of other classes. Instantiate the classes as multiple objects, using range iterations if adequate, to represent the following scenario: The teacher has 20 copies of a document. There are 10 tables, and each table is occupied by two students. Each document belongs to one of the students. On the blackboard, the name of the teacher is written.

2.2 Person

In software, a person is typically represented by an object. Write a class `Person` and add some data (hair color, age, etc.) and behaviors (speak (i.e. write to terminal), getting older, etc.) to it. How can data be set? Use proper object orientation, i.e. attributes for data and methods for behaviors.

2.3 Bank account

Develop a class `BankAccount` in an object-oriented way with following specification:

A bank account is used to manage money and is uniquely determined by an identifier (digits and/or letters, eg. IBAN). The account can be opened or closed. Money can be deposit or withdrawn. The actual bank balance can be retrieved at any moment. The amount is expressed by an account-specific currency, by default Swiss francs (Fr, rp). The bank balance can not be below zero nor above 100'000.- in its currency.

- **Use attributes for data**: Think what data such a bank account needs and declare some class variables (attributes) for it.
- **Use a constructor to initialize the object**: The bank account starts with 0.- Fr and the mandatory identifier.
- **Use methods for actions**: Think about what behavior the bank account has (what action can be taken). Methods are similar to functions except that they are declared in a class scope.

Test your class with a non-interactive test script that initialises the bank account object, pays in some money, shows the bank balance, etc. The main script should be guarded (`if __name__ == “__main__”`) to prepare the script for module use. Submit the file as `bankaccount.py`.

Note: This class will later be extended. Make sure to avoid ‘spaghetti code’.