**TU856/858-2 Object Oriented Programming with Python**
**Semester 1, Lab Week 10, 27.11.2020**
**Dr. Bianca Schoen-Phelan**

**Please join the virtual classroom in brightspace first, before going to individual lab links. I will provide explanations and hints and tricks regarding the lab at the beginning of the lab sessions. These are recorded.**

Please sign your group's lab sign in sheet!
Lab solutions will be discussed in the Monday lecture. Labs are not marked.
Don't forget about your revision item for the week!

Link to the lab groups (click).
Link to python documentation (click).

## Objectives:

- Practice python classes and inheritance
- Use abstract classes via the abc module
- Code encapsulation, provide docstrings

**This lab does not have a GitHub download.You have to program from scratch!**

## Your task:

- You work as a Python software developer in a company focuses on facilitating edutainment games for schools. One of the secondary schools in your county has tasked your company with a maths game. They are not entirely sure yet how many different maths games they want to provide, as currently they only have an idea for one. They want to keep their options open to extend this.
- The first game should just illustrate the principle of Fibonacci numbers. The student should input how many Fibonacci numbers to generate and the programme then shows these numbers.
    - Next, the student should be asked what is the next number in the series. If a correct number has been entered a counter should be increased that keeps track of how many times the right fibonacci number was entered. If an incorrect number has been added, there should be an appropriate number displayed.
    - This should run in an endless loop:
      `while True:` or `while some_boolean_variable:` in your program.
      If 1 is entered, the program should ask again for Fibonacci terms, if 2 is entered, the program should stop and before exit display how many Fibonacci numbers were guessed correctly. See next page for an example of game play.
- The lead architect on your team decides to use an **abstract class** via the abc module in order to facilitate the entire maths game. The abstract class acts as a scaffold of the math game. And then every actual game inherits from this abstract class for every new game that they are tasked to implement.
    - The abstract class implements the following that must be implemented by your derived Fibonacci playing class:

        - Abstract methods: `__init__`, `play_game`
        - Abstract property: `user_input_property`
    - Validate user inputs: `try...except` is great here, also see [link] for more examples. Revise `try...except` in our lecture notes if needed.
    - Use appropriate commenting throughout
    - Use docstrings
    - Use encapsulation
    - Could any of the methods you write be declared `@staticmethod`?

Observe how the game can be played:
[link]

**Hint**: Start off very simple with the most basic version of the program. Then add functionality gradually. This is a good place to use **git** to commit every new feature.

**Advanced:**

- Think of other maths games to implement using your framework. Have a look at recommendations, such as: https://home.oxfordowl.co.uk/kids-activities/fun-maths-games-and-activities/
    - This is another lab that you could choose to take further and add to your git portfolio. **Seize those opportunities!**