Exercise Sheet 0

Introduction to IT-Security

Deadline: *Wednesday, November 12th, 09:00 a.m. CET 2025*

Preliminaries

You will use the version control tool `git` to work on your homework and submit your solutions. Go to the `Gitlab` instance on `https://git.rz.tu-bs.de/ias/itsec/ws2526` and use your GITZ SSO credentials to login. You have received a team repository.

- Save the code for the programming exercises in the `src/` directory.
- If there are theoretical exercises create a `Markdown` file with the solutions and save it in the `doc/` directory.
  Hint: It is useful to create a template for this document that you use throughout the semester.
- To check your solutions we use the Gitlab CI. To activate the Master tests, edit the `config.json`.

Python Basics

This exercise sheet covers some basic concepts of the Python programming language. For evaluating your solutions we will use simple unit tests on the `Gitea` server.

1. (1 point) *Modules and packages.* Create a Python package called `sectubs` that in turn contains a module `exercises`. This modules needs to contain a class `Exercise00` that is used to implement the following tasks.

2. (1 point) *Static fields.* Implement a static field `STUDENT_NAME` for class `Exercise00`. This field should contain a string stating your name.

3. (1 point) *Static methods.* Implement a static method `deadline` for class `Exercise00`. This method should return the date (`November 15th, 09:00, 2023`) as specified by the Python format string passed in as argument. *Hint:* Make use of standard library functionality.

   `Exercise00.deadline("%H:%M %d.%m.%Y")` should return `"09:00 15.11.2023"`
   `Exercise00.deadline("%d.%m.%Y %H:%M")` should return `"15.11.2023 09:00"`

4. *Properties.*
   (a) (1 point) Implement a readable but **not** writable property for class `Exercise00` with the name `txt`.
   (b) (1 point) The property should return the first 17 characters of the string passed to the constructor of `Exercise00` plus `"..."`.

   ```
   ex = Exercise00("abcdefghijklmnopqrstuvwxyz")
   ex.txt returns "abcdefghijklmnopq..."
   ```

5. (2 points) *Format strings.* Implement a function `format` for class `Exercise00` that returns different Python 3 format strings according to two different mode specifiers: `"order"` and `"dict"`.
   (a) (1 point) Reorder three input arguments such that the first appears on third place, the second on second and the third on first place.

   ```
   ex = Exercise00()
   ex.format('order').format('third', 'second', 'first')
   returns "first - second - third"
   ```

   (b) (1 point) *Dictionary-based formating.* Format name input parameters such that they are interpreted as float numbers and appear in specifc order. The `x` with 1 digit precision and `y` with 4 digits of precision.

   ```
   ex = Exercise00()
   ex.format('dict').format(x=41.123, y=71.091)
   returns "x, y = (41.1, 71.0910)"
   ```

6. (2 points) *Generators.*
   (a) (1 point) Implement a function `listfiles` as generator.
   (b) (1 point) This generator should list all files of a particular type in a directory and its subdirectories. The type thereby is passed to the function as optional parameter.

7. (3 points) *Loops*
   Consider the sequence of positive integers $x_1, x_2, \dots$ given by

$$x_{n+1} = \begin{cases} 3x_n + 1 & \text{if } x_n \equiv 1 \mod 2 \\ \frac{1}{2}x_n & \text{else.} \end{cases}$$

   The Collatz conjecture states that for each initial value $x_1 \in \mathbb{N}$ this sequence ends in the infinite loop $\{4, 2, 1, 4, 2, 1, \dots\}$. Implement a function `collatz` that takes as input a positive integer x and returns a tuple with the Collatz sequence with $x_1 = x$ as a list and the *total stopping time*, i.e. the smallest index i such that $x_i = 1$.

```
ex = Exercise00()
ex.collatz(12) returns ([12,6,3,10,5,16,8,4,2,1], 10)
ex.collatz(3.1415) returns ([], 0)
```

8. (2 points) *Function parameters.*
   (a) (1 point) Make your `Exercise00` class a function object.
   (b) (1 point) Return a string that lists the provided arguments as key/ value pairs in alphabetically order separated by newline characters.

   ```
   ex = Exercise00()
   ex(c=None, a=1, d=4, b='2')
   returns "'a = 1\nb = 2\nc = None\nd = 4"'
   ```

9. (8 points) **Master:** *Base64.* Implement the base64 encoding **without** the help of any packages or modules such as Python's `base64` module. When an `Exercise00` object is converted to a string `s = str(ex1)`, the result should be the base64 encoding of the string initially passed to the constructor.

10. (3 points) *Argparse.* Make yourself familiar with the `argparse` module and turn your `exercise` module into an `argparse` script such that the call `python exercises.py -h` results in the following output

```
usage:  exercises.py [-h] [-b] [-f FLOAT] [-i INT] FILE

positional arguments:
FILE              The input positional parameter.

optional arguments:
-h, -help          show this help message and exit.
-b                 An optional boolean flag (Default:  False).
-f, FLOAT          An optional parameter of type float (Default:  0.0).
-i, INT            An optional parameter of type int (Default:  0).
```