Exercise Sheet 1

Send your solutions to ics-ex@goesec.de

Deadline: Wednesday, November 4th, 11:59 a.m. CET

Student Details

1. (1 point) Design a cover page that clearly states your...
   - surname and forename
   - enrollment number
   - university and study programme

Python Basics

In this part of the exercise you will implement simple tasks covering basic concepts of the Python programming language. Each task corresponds to one unit test in `ex1.py`. Test input and output are defined in `ex1_testdata.py`. For evaluating the exercises we will use a similar but different configuration that contains alternative input.

2. (1 point) Modules and packages. Create a Python package called `goesec` that in turn contains a module `exercises`. This module needs to contain a class `Exercise1` that is used to implement the following tasks. Unit test: `ex1.py Ex1.test_00_packages`

3. (1 point) Static fields. Implement a static field `STUDENT_NAME` for class `Exercise1`. This field should contain a string static your name. Unit test: `ex1.py Ex1.test_01_static_field`
4. (1 point) *Static methods.* Implement a static method `deadline` for class `Exercise1`. This method should return the exercise’s deadline (November 4th, 11:59) as specified by the Python format string passed in as argument. *Hint: Make use of standard library functionality.* Unit test: `ex1.py Ex1.test_02_static_method`

```python
Exercise1.deadline("%H:%M %d.%m.%Y") should return "11:59 04.11.2015"
```

5. *Properties.* Unit test: `ex1.py Ex1.test_03_property`
   (a) (1 point) Implement a readable but not writable property for class `Exercise1` with the name `txt`.
   (b) (1 point) The property should return the first 17 characters of the string passed to the constructor of `Exercise1` plus "...".

```python
ex = Exercise("abcdefghijklmnopqrstuvwxyz")
ex.txt returns "abcdefghijklmnopqrstuvwxyz...
```

6. (points) *Format strings.* Implement a function `format` for class `Exercise1` that returns different Python 3 format strings according to two different mode specifiers: "order" and "dict".
Unit test: `ex1.py Ex1.test_04_format_strings`

   (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.

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

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

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

7. *Generators.* Unit test: `ex1.py Ex1.test_05_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.
8. (1 point) **Function parameters.** Unit test: ex1.py Ex1.test_06_functionparams

(a) (1 point) Make your Exercise1 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.

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

9. (8 points) **Base64.** Implement the base64 encoding without the help of any packages or modules such as Python's base64 module. When an Exercise1 object is converted to a string `s = str(ex1)`, the result should be the base64 encoding of the string initially passed to the constructor.

Unit test: ex1.py Ex1.test_07_base64

10. **Argparse.** Finally, implement program arguments for your script goesec.exercises using Python's argparse module.

Unit test: ex1.py Ex1.test_08_argparse

(a) (3 points) The following arguments need to be present:

- A positional argument.
- `-h`, `--help` Printing usage information.
- `-b`, `--bool` A boolean flag.
- `-f`, `--float` A parameter of type float.
- `-i`, `--int` A parameter of type int.

(b) (1 point) Your script needs to end with exit code 42

(c) (1 point) Print the input arguments.