“Hello, World!”
Practical Assignment 0

Informatics 1 for Biomedical Engineers

Graz University of Technology

Abstract

In the practical assignments for this course, you will be working with the Python programming language (3.5 or above). Your initial task will be to set up your local development environment.

The most famous programming task for learners of any programming language is “Hello World!”. Thus it only feels suitable for our needs. The student’s task is to prepare the software on their local machine using the anaconda framework. To test this setup they have to write a single short Python script which outputs a short “Hello World!” to the command-line.

1. Tasks

1.1. Set up Python

Before starting to program you will have to install a Python environment. The reference for this course will be Python 3.5 and 3.6. We strongly suggest using the anaconda framework[1] as it comes with all the needed libraries (packages) pre-installed.

To verify which version of Python is running on your system start Python using the command line. What you see should be along the lines of the following:

```
>python
Python 3.6.2 |Anaconda, Inc.| (default, Sep 19 2017, 08:03:39) [MSC v.1900 64 ...
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Again, we do not strictly enforce the use of the anaconda framework, but we will not openly support other installations. Hence, if you run into problems, you might be on your own!

1.2. Your First Program

Traditionally, the first program anyone writes when learning a programming language is called "Hello World!". The student task is to write a program with the following output:

```
Hello, World!
```

The objective of this task is to ensure that the Python 3 environment is set up properly and to get used to the basic Python functionality. Think of the shortest way to cleanly do this. (**Hint**: if you have more than 2 or 3 lines of code, excluding the comment header, you might be doing it wrong!)

[1] `https://www.continuum.io/downloads`

2. Restrictions

- Do not add any bloat to your submission
- Use built-in functions where possible
- Do NOT load extra packages (no imports)
- Do NOT use any command line arguments!

3. File Headers

All Python source files have to contain a comment header with the following information:

- Author: – Your name
- MatNr: – Your matriculate number
- Description: – Purpose of the file
- Comments: – Any comments as to why if you deviate from the task or restrictions

Please copy and paste the example and change its contents to your data. (Depending on your PDF-viewer you may have to fix the whitespaces)!

Example:

```
######################################################################
# Author:      Patrick Kasper
# MatNr:       0730294
# Description: The main file. Assignment only has 1 file...
# Comments:    This is the example comment. I just made it a bit
#              longer so it spans across multiple lines.
######################################################################
```

4. Coding Standard

For this lecture and the practicals we use PEP 8 [2] as a coding standard guideline. Please note that whilst we do not strictly enforce all these rules we will not tolerate messy or unreadable code. Please concentrate in particular on the following aspects.

[2] https://www.python.org/dev/peps/pep-0008/

Language Programming is done in English. This is to ensure someone else at the other end of the world can read your code. Please make sure all sources you submit are thus written in English. This covers both variable names and comments!

Spaces, not tabs. Indent your files with 4 spaces instead of tabs. PEP 8 forces this in Python 3 (whilst allowing more freedom in Python 2). Most editors have an option to indent with 4 spaces when you press the tab button!

Descriptive names. Use descriptive names for variables where possible! Whilst a simple *i* can be enough for a simple single loop code can become very messy really fast. If a variable has no purpose at all, use a single underscore (_) for its name.

Max 72 characters. Do not have lines longer than 72 characters. Whilst PEP 8 allows 79 in some cases we ask you to use the lower cap of 72 characters per line. Please note that this includes indentation.

5. Automated Tests

Your file will be executed with the following command:

```
>python assignment_0.py
```

Please make sure your submission follows all the restrictions defined in this document. Your program will have a limited runtime of 2 minutes. If your submission exceeds this threshold, then it will be considered non-executable. Please note that this is a very generous amount of time for any of the tasks in this course.
If your submission fails any of the automated tests, we will look into your code to ensure the reason was not on our end and to evaluate which parts of your code are correct and awards points accordingly.

6. Submission

6.1. Deadline

17$^{\text{th}}$ of October 2017 at 23:59:59.

Any submission handed in too late will be ignored with the obvious exception of emergencies. In case the submission system is globally unreachable at the deadline it will be pushed back for 24 hours. If for whatever reason you are unable to submit your work, contact your tutor *PRIOR* to the deadline.

6.2. Uploading your submission

Assignment submissions in this course will always be archives. You are allowed to use .zip or .tar.gz formats.
In addition to your Python source files, please also pack a *readme.txt.* The file is mandatory but its contents are optional. In this file please write down how much time you spent on the assignment and where you ran into issues. Your feedback allows for immediate adjustments to the lecture and tutor sessions.
Upload your work to the Palme website. Before you do please double-check the following aspects:

- File names and folder structure (see below)
- Comment headers in every source code file
- Coding standard upheld

6.3. Your submission file

```
assignment_0.zip (or assignment_0.tar.gz)
    assignment_0.py
    readme.txt
```