Greet the world!

Practical Assignment 0

Informatics 1 for Biomedical Engineers

Graz University of Technology

Abstract

Ensuring a working python 3.5.2 environment is vital in order to do the practical assignments for this lecture. 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. Finally, to complete the assignment the file is put in an archive and uploaded the the Palme.

---

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. We strongly suggest using the anaconda framework[1] as it comes with all the needed libraries (packages) pre-installed.

In order 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 (paths can be different):

```
C:\Anaconda3>python
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC ...
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Once 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

As is tradition the first program anyone has to write when learning a programming language is called "Hello World!". The student's 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 3 lines of code you are doing it wrong!)

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

2. 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 feel free to 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.
###############################################################################
```

3. 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!

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 focus especially on the following three aspects.

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

We will test your submissions automatically. The test machines will have the latest anaconda version on the 5$^{\text{th}}$ of October 2016. Your file will be executed with the following command:

```
>python assignment_0.py
```

We will monitor the standard output. For this task please make sure you do not have any other output and you use proper capitalisation (as defined by the task).
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.

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

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

18^{th} of October 2016 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 automatically be pushed back for 24 hours.
If for whatever reason you are unable to submit your submission on time please contact your tutor ASAP.

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. This is important to us as immediate feedback so we can adjust the tutor sessions. Please note that all submissions will be anonymised prior to being read. Hence please feel free to be honest (but do stay polite).
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 very file
- Coding standard upheld

6.3. Your submission file

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

Appendix A. The Anaconda framework

One problem with interpreter based programming languages is that you have to trust the interpreter on the executing machine to do the right things. Whilst this is manageable for the interpreter alone it becomes an order of magnitude more difficult when you use a plethora of 3rd party packages. (Such as numpy or mathplotlib) The Anaconda platform attempts to tackle this very problem.
Built specifically for data science the anaconda platform is a single installer available for all common operating systems that installs python environments with all common packages. Hence a virgin installation is reliable to a point that I can say when code runs on my machine it will also run on yours.
Additionally it installs a few other very useful tools. Both *pip* and *easy install* allow the installation of various packages. Additionally a public repository for pre-compiled binaries is available. If you want to have multiple separate python installations Anaconda can set up multiple environments with different interpreter/package configurations for each. Using these features maintaining a clean python setup is a manageable with limited effort.
For this course the most important feature is the *Jupyter Notebook*. An Web-based IDE which serving as ideal playbook. This is the software we use during presentations. For more information on this refer to Appendix B. Please note that this is just a short excerpt of the Anaconda features for the scope of this course and the platform can do a lot more.

Appendix B. Jupyter Notebooks

Jupyter Notebooks (previously known as *IPython Notebooks*) is an open source web application for live code of the most common data science programming languages. It comes pre-packaged with the Anaconda platform so you don't need to worry about setting up the kernels needed for language support. It can be started by either typing *jupyter notebook* into your command line of choice or you can launch it via the Anaconda Navigator if you have it installed. (You will need write permissions to the notebook-folder for the web user on unix operating systems)
Per default it starts on port 8888. So open `http://localhost:8888` on your machine after you started the notebook server and you will see the Jupyter interface. To create a new notebook click new on the right-hand-side. Depending on your installation the Python 3 environment will either be called "Python 3" or "Python [Root]". To test your environment create a new notebook and type the following code into the cell visible:

```
import sys
sys.version_info
```

To execute this cell either click the ⏭ - button or hit *ctrl + enter*. The output should be the following (which translates to python version 3.5.2):

```
sys.version_info(major=3, minor=5, micro=2, releaselevel='final', serial=0)
```

Ipython is persistent meaning as long as you do not restart the kernel (via the menu or by hitting ↻) the variables will stay set. You can execute the individual cells in any order you want. This can be a little confusing at start but is a very nice ability once you got used to them.
All notebooks used by us will be available for download in the teach center. You can upload and use them using the Jupyter start page. This way you can play around with the data and see what happens when you change individual cells or variables. Please note we will not be accepting notebook files (.ipynb) for assignment submissions!
We will be using notebooks a lot during the tutor sessions. So if you have any questions regarding them please feel free to ask your respective tutor!