Grassroots - Blog 1 – Design of Online Evaluation of Python Assignments for Scoring and Direct Feedback
Thomas Mensink and Spencer Cappallo - Sun 26th October.

Next week our course 'Visual Search Engines' will start. Roughly 50 students from the MSc Information Studies program have enrolled, and we're excited to start using our **automatic online evaluation server for the python** lab assignments. The aim of the lab is to support the understanding of the theories discussed during the lectures, but learning-to-program is not one of the class (key) objectives.

The goal of online evaluation of the students’ code is to enable students to study on their own pace, by receiving immediate feedback about the correctness of their code and being able to track their own progress. This is inspired -in part- by the people behind the first MOOC initiatives, nicely illustrated by Salman Khan (founder of Khan Academy):

> When you learn to ride a bicycle, and you fail to learn to ride a bicycle, you don’t stop learning to ride the bicycle, give the person a D, and then move on to a unicycle. You keep training them as long as it takes. And then they can ride a bicycle.

During the past weeks, we have been discussing the requirements, possible setups, and existing solutions for automatic evaluation of code. There are many online platforms, which focus either on learning-to-program, or collaboration on (python) programming. Most notable is CodeCademy (http://www.codecademy.com), which allows the creation of new courses, and which has a ‘pupil-tracking’ module to track progress of students. Unfortunately, those two cannot yet be combined; although this is planned for the (near) future.

Therefore we have designed our own evaluation protocol for students’ code. This required some serious consideration. Evaluating honest attempts of the lab assignment is rather trivial, however how to prevent malicious code to be run on the evaluation server? Restricting a general programming language (such as Python) is hardly possible, without restricting so much that the program is not of any use anymore.

**We came up with the following design:**

Instead of executing student code on the server, we sandbox the code to the student’s computer. The students are evaluated on a series of pre-specified functions, the guts of which have been left for the students to fill in. In this way, we insure that we know exactly what form of data each function expects to receive and is expected to return.

As we already know the function's expected behaviour, we are able to provide the function with input data to be fed through the function locally, on the student’s machine. The output of this evaluation is then returned to the server, where it checks it against a pre-computed expected output, and returns whether the student was successful or not. This allows to unit-test most of the required functions, although this might not be suited
for all kinds of programming exercises. The final lab evaluation will be based on a lab-report and the submitted code.

The server logs the attempt by the student, and its success, so that both the course administrators as well as the student are able to track the student’s progress through a web interface. For the hosting of the server software, we use a service called PythonAnywhere (http://pythonanywhere.com) that allows users to easily create web applications using a full python installation. The python backbone of the service allows administrators to easily add pre-computed test inputs and outputs by simply implementing gold-standard versions of the functions the students are expected to complete.

A diagram giving an overview of the communication between a student’s computer and the server in the use case of a student submitting a function for review is provided below.

In the next blog, we will discuss our in-class experience!
Spencer and Thomas

![Diagram of communication between student's computer and server](attachment:communication-diagram.png)