**Autograder User Manual**

1. **Autograder Configuration file**

This has two fields

[Autograder Setup]

grading\_root = \users\home\manujinda\grading

grading\_master = assignments

grading\_root

This is the path to the root directory where all the grading related stuff are stored within. In the directory directed to by this path, autograder looks for pre-specified directories and files to carry on its work.

grading\_master

This is the directory where all the assignment / project related stuff are stored. Each assignment / project has a sub directory within this directory. Such a sub directory contains the solutions, test input and their expected outputs etc. for that assignment / project.

Based on the user’s preferences it is suggested using either “assignments” or “projects” as the name of the grading\_master directory.

1. **Autograder initial setup**

Run the command:

$ python Autograder.py setup <path the autograder.cfg configuration file>

This creates the directory structure required by the autograder at the grading\_root provided in the autograder.cfg file. If the directory pointed to by grading\_root already exists, this notifies about that and takes no further action. For this to work, the directory pointed to by grading\_root should not exist and in that case this command creates that directory and the necessary autograder directory structure under the grading\_root. Then this copies the autograder.cfg and a cascading style sheet that is used in student grading log files into the grading\_root directory.

This creates a blank example assignment / project as a starting point within the directory specified by grading\_master which can be taken as an example for creating future projects.

The autograder directory structure is as follows:



grading\_root



grading



students



assignments

autograder.cfg

students.csv

+\_1\_assignment\_1.cfg

+\_2\_assignment\_1\_problems.cfg



assignment\_1

students directory – Each student repository is cloned in here. Each student has a unique directory in this.

students.csv – This stores all the student details. This file is stored in students directory.

grading directory – A copy of each student submission is created in this directory. So each student has a unique directory in this. All the provided files are also copied from respective grading master directory into student submission directories. Compiling of student submissions is done with this copied set of files.

+\_1\_assignment\_1.cfg – Each assignment / project has a configuration file that describes the assignment / project. Each assignment / project is comprised of a set of problems.

+\_2\_assignment\_1\_problems.cfg – The configuration file for all the problems that are part of a particular assignment / project.

1. **Student details**

Populate the students.csv file with the student details. Student repository URI’s should be properly recorded in this file for the system to clone / pull them and proceed with the grading.

1. **Setting up an assignment / project**
   1. Create a directory for the assignment / project within the assignments / projects directory.

(e.g. .../grading\_root/assignments/assignment\_3)

* 1. Create the assignment / project configuration file and populate it to describe the assignment / project. This file must have the file name +\_1\_<assignment / project sub directory name>.cfg

(e.g. .../grading\_root/assignments/assignment\_3/+\_1\_assignment\_3.cfg)

[assignment\_3]

asnmt\_no = 3

name = hello world

duedate = 6/28/2016

problem\_ids = 1:prog 2:code 3:ans 4:mcq

asnmt\_no – Assignment / project number. Just to identify the assignment / project.

name – Name of the assignment / project. Again just to make the assignment / project better identifiable

dudate – Due date of the assignment / project. The format should be mm/dd/yyyy. (Do we need to add the due time as well?)

problem\_ids – IDs of problems that are part of this assignment / project. A problem type for each problem identifier should be specified separated by a colon (‘:’) for each problem. Valid problem types are as follows:

prog – A programming problem. The student is supposed to submit complete code. The code is compiled along with provided code and then tested against the test input. Verification is done by comparing the output provided by student submission with the desired output.

code – Student does not submit a complete program but some program segment. Student submission will not compile as it is. Grading is done by comparing student submitted code segments wit

h the desired code segments provided to the autograder. As an extension, student code segments could be plugged into some code stubs, then compiled and tested just like a ‘prog’ type submission.

ans – Student submits a short written answer. Grading is done by comparing the answer with predefined answer keywords.

mcq – Multipole choice type question. Student submits the correct answer choice. To have more freedom, student can either submit the correct answer selected among the choices or just the correct answer choice number / letter. Grading is done by one-to-one mapping between the student answer and the correct answer choices.

* 1. Run the command

$ python Autograder.py genprob <path to assignment / project configuration file>

This creates a skeleton problem configuration file.

(e.g. .../grading\_root/assignments/assignment\_3/+\_1\_assignment\_3\_problems.cfg)

* 1. Populate the problem configuration file to describe the problems that are part of the assignment / project and their respective grading criteria.
  2. Run the command

$ python Autograder.py genfiles <path to problem configuration file>

This generates all the files described in all the problems that come under the particular assignment / project. All these files are stored in the directory created for the assignment / project.