Assignment 2 - Python Project Introduction to Computer Programming

# Summary of the scope of the task and the rules

* The objective of Assignment 2 is to write **an original Python program** that shows off your Python programming skills and some of your creative flair.
* Note that this project is to assess your ability to use the learned concepts to solve an interesting problem.
* Although you are free to choose any topic, if you are not sure what to do, we will suggest some potential topics for your assignment (see further below). It will be possible to get a solid but unspectacular mark by good delivery of one of these standard topics. But to get a top mark, you will need to either stretch **significantly** the boundaries of one of the standard topics; or
* This assignment is worth 60% of your overall mark for “Introduction to Computer Programming”.

## The deliverables will be:

1. A working Python program in the form of a .py file (or potentially a collection of files which are collected together as a .zip). We will only be accepting projects in Python 3 or above. It has to run on the university build machines. Please test it there before submitting.
2. A short report (in .pdf format) of 1-2 pages in length, with the following format:
   * Describe the goal and motivation behind your project.
   * Cite your sources. Mention any external code used with clear URLs.
   * Describe how you use the program (since it should be clear from the game itself, this should be short).
   * **[very important]** What were the design decisions? Why did you choose this solution (e.g., this data structure, this module, etc.)?
   * **[very important]** Challenges encountered and how you overcame them (give us at least one example).
   * Anything else you would like to highlight and we should know to be able to mark your submission fairly and correctly.

## The deadline is:

* The deadline is **16:00 on Friday 11th January.**
* Multiple submissions (drafts) are possible. We will only mark the latest submission.
* You must upload your assignment including the program (.py or .zip) and report (.pdf) **to Blackboard** (link under *Assessment and Feedback*)
* If for some reason, you cannot submit by the deadline, you need to seek approval from **Helmut before the deadline**. You will need to provide proof!
* Note that a loss of data is **not a valid reason** to get an extension. It’s your obligation to backup regularly!
* For this assignment, it is required that **you submit individual work**. You may discuss the creative scope of your assignments with each other, but you must work individually on the programming and the accompanying report. **We will cross-check projects!**

## What should a good assignment do?

* Note that we want to assess your ability to use the learned concept to solve interesting problems.
* [A] Good programs should:
  + Apply learned concept in a useful and interesting way (i.e., variables, loops, conditions, functions, basic data structures, modules, user interaction, etc.).
  + Include useful comments (describing functions, explains hard to read code, etc.)
  + Use good variable names (self-explainatory names, CamelCase, consistent naming)
  + Should be well-structured and therefore easy to understand
  + Loops, functions and classes should be used **where appropriate** to simplify the structure of the code and make it more taut, avoiding duplicated code
  + Should check for different inputs and provide appropriate feedback to the user
* [B] For higher marks:
  + Have an original idea (i.e. beyond standard ideas we provide)
  + Use a number of modules and combine them in an interesting way.
  + Use external modules like Tkinter, PyGame, matplotlib, numpy, etc.
  + Use data structures that go beyond built-in data structures
  + Use a simple AI in case you make game
  + Save and read relevant data to/from file ((highscores, status of program, , last data points, etc.)
  + Use a Graphical User Interface (GUI), e.g. with Tkinter
* [C] Exceptional submissions may include:
  + Implementing an outstanding and original idea
  + Have complex multiple files, your own classes,
  + Use modules that haven’t been explicitly shown in the lecture, e.g., Pandas, PySpice, BeautifulSoup, etc.
  + Use a more sophisticated Graphical User Interface (GUI)

**Tip:** Start with a simple program (i.e. complexity of [A]), get it working and then add more features and complexity to achieve [B] and [C].

## Lab time

We will set aside time to work on your projects in the labs — and to ask questions as the need arises. Note that we expect a top-end assignment to take more time than what is allocated during lab sessions, but this assignment shouldn’t ruin your Christmas. It should be enough to put in some solid time before you go away. If you wish to work at home on this during the Christmas break — that is fine, but it shouldn’t be necessary. Please make sure you are being effective with the time in the labs, and asking questions while the TAs are available.

## Plagiarism

It is very easy for us to run Python code through plagiarism checking software that analyses similarities between all the submitted assignments and between those assignments and other bits of code that are out there on the internet. The University penalties for copying work are horrible — so don’t be tempted to cheat.

**We will check every submission for plagiarism!!**

It is fine for you to research ideas, or to develop an idea that you find on the internet, but you should not copy code. If you are basing your project on something that you find on the internet, you have to use your report to describe exactly where you found it and how your contribution is different. **Note that we can only mark the part of the code that you have written.** So use of code found online should be highlighted and referenced in a comment, and referenced in your report.

# Possible Python Projects

A simple text interaction game. Note that these are bare minimum and **need to work flawlessly to pass**.

* Noughts and Crosses or Mastermind (but that is too simple - unless perhaps you try to build intelligence into the game for the computer opponent).
* Maze solving (read different text based mazes from a file, display it, and solve it for the shortest way).
* Chess, Connect Four, or Drafts - or other board games - NB need to build in the rules of the pieces etc. (Having a “computer player” is too hard.)
* Card games of various kinds (Poker, Solitary etc.).
* Scrabble - asking the player to find best scoring words from a randomly generated rack.

(NB you can database of words online).

More interesting projects (with higher marks) would a physical simulator (using *odeint* for integration)

* A complex calculator capable of using statistics with a GUI
* Analysis and extracting data from data files and plotting them (with user interaction)
* Drawing application
* A Game implemented with Pygame

**If you are not sure if your idea is valid, please talk to us.   
We can give you feedback on how difficult it is.**

NB you will be generally ok using modules which are built-in to Python 3.0 (the standard library) and the ones that we present, i.e., PyGame, Tkinter, Numpy, matplotlib. But for some more interesting tasks, you might want to experiment with downloading 3rd party module. You can do that within PyCharm, see here for more information.

<https://www.jetbrains.com/help/pycharm/installing-uninstalling-and-upgrading-packages.html>

If you add non-standard packages, please, mentioned that in the report.

# How do we mark your projects

Note that using classes or modules where not necessary will not give you the extra points:

Still working on this part…. We have 5 categories we mark for:

* Originality (5 points): see possible project descriptions above varying in originality from those that are pre-proposed simple text interaction game, to programs that pro- vide a twist on a known game, creative flare, or are entirely original.
* Challenge (10 points): see difficulty levels above varying from a basic text-based pro- grams to ones that uses AI, GUI, default modules vs. external modules.
* Features (5 points): makes appropriate use of data types, functions, and classes and modules.
* Implementation (5 points):the code is well structured, taut and commented, use of CamelCase, is easy to read.
* Usability (5 points): the code is easy to install and use. It’s clear for a new user how to use the program. The user experience is good. The code isn’t buggy, and it is fool-proof (for example by checking user inputs to avoid improper use).
* Report (10 points): The report provides meaningful information, is easy to read, and follows the required structure.

# FAQ

Which modules can I use module? You

Is my idea is interesting enough? Come and talk to us.