Using techniques learned in class, you will be expected to create a formal design for the program described in Assignment 2. Specifically, the assignment calls on you to use BoUML in creating a UML design of the program. Using features of BoUML, you will then need to automatically generate code that implements the design. This document describes how to create models in BoUML and then generate code.

1. **Getting Started**: BoUML is installed on the lab machines in Trottier, so accessing the software is not an issue. If you wish to install BoUML on your own PC, it is available as an open source project and can be found online at `http://bouml.free.fr/`. On the website, documentation is provided. There are also some very nice tutorial videos available on the documentation page that you may want to watch.

   (a) Upon opening BoUML for the first time, an environment dialog appears. You must enter an "Own Identifier" to continue.

   (b) A pop-up will appear telling you to select your language. Go to *Languages* > *Python* to choose Python as your working language.

2. **Modelling**: BoUML is now ready for you to begin modelling your program.

   (a) Create a new *project* to begin working. You should see your project appear in the browser view on the left. Right-clicking on your project in the browser view, create a new *package*. Right-clicking on your new package, create a new *class view*. Finally, right-click on your class view to create a new *class diagram*. The class diagram is your UML model.

   (b) Double click your class diagram in the browser to bring up the GUI modelling interface. You will need to decide how to draw your UML diagram based upon what you have learned in this class. You will need to edit the classes you create to give them appropriate settings for your design.

   (c) To fill in classes with attributes and operations, right click on the class and add them as appropriate. Editing attributes allows you to set initial values, type, etc., while editing operations allows you to define input and output parameters. Additionally, clicking on the Python tab of an operation shows the Python code. On that

same type, in the bottom right there is a button 'Edit Body'. This will allow you to enter your Python code to complete the body of the operation.

3. **Code Generation**: Once the UML diagram is complete, the final step is to generate code through BoUML.

   (a) Right-click your project in the browser view and select 'Edit Generation Settings'. Select the directory tab and set your Python directory.

   (b) Right-click your package again and add a *deployment view*. This will hold your generated code artifacts. You must explicitly associate this view to generate code. Right-click your class view and select 'Edit'. Then, on the deployment view dropdown, select the deployment view you just created.

   (c) To generate code for a class, right-click the class and select 'Create Associated Artifact'. Once you have created artifacts for each class, go to your deployment view, right-click, and select *Generate > Python*. This will generate your code for all classes.

Make sure to test your code! If it doesn't work properly, go back to your design and edit as necessary!

Good luck!