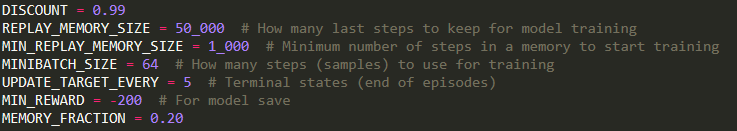
Handover Documentation

# Environment Setup

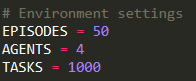
* Download python 3.6.8 from: <https://www.python.org/downloads/windows/>
* At command prompt enter the following to install the required packages:
  + pip install numpy
  + pip install Keras == 2.3.1
  + pip install tensorflow == 1.15
  + pip install tqdm
  + pip install Pillow
  + pip install opencv-python
  + pip install xlwt

# Program Parameters

* Parameters of the **system** can be modified in the header of the script



* Parameters of the **environment** can be modified in the header of the script



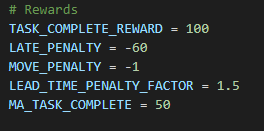
* **Look-ahead** and **individual** **queue** parameters can be modified in the header of the script



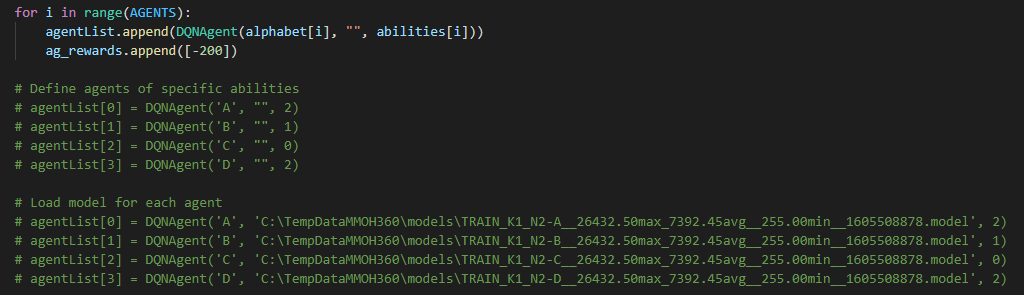
* Abilities of agents can be manually specified in the header of the script



* **Reward** values can be modified in the *Env* class



* To **load model** or define agent of **specific abilities** manually, uncomment the respective part in the main loop
* Adjust path name to load model as required



# Running Program

* To run from terminal
  + python speed\_test.py
* If using supercomputer
  + python3.6 speed\_test.py
  + if installing packages on supercomputer use python3.6 -m pip install
* Recommended start with 4 agents, 20 tasks, 50 episodes, k = 1 and N = 0 for testing functionality
* To display results
  + tensorboard --logdir=logs in directory where program is stored
  + follow address that is output e.g. [http://EN402316:6006/](http://en402316:6006/)
  + .csv files can be downloaded from tensorboard

# Using the Supercomputer

* Username: IAI-lab
* Password: yuqianlu
* Follow instructions in ‘Running Program‘

# Classes

***Env***

* Defines the environment
* Used to get environment observations and rewards based on actions taken by the robot

***ModifiedTensorBoard***

* Used for logging results and viewing results on TensorBoard

***DQNAgent***

* Creates the DQN model
* Contains function for training models

***Robot***

* Handles robot location, status, actions and order handling

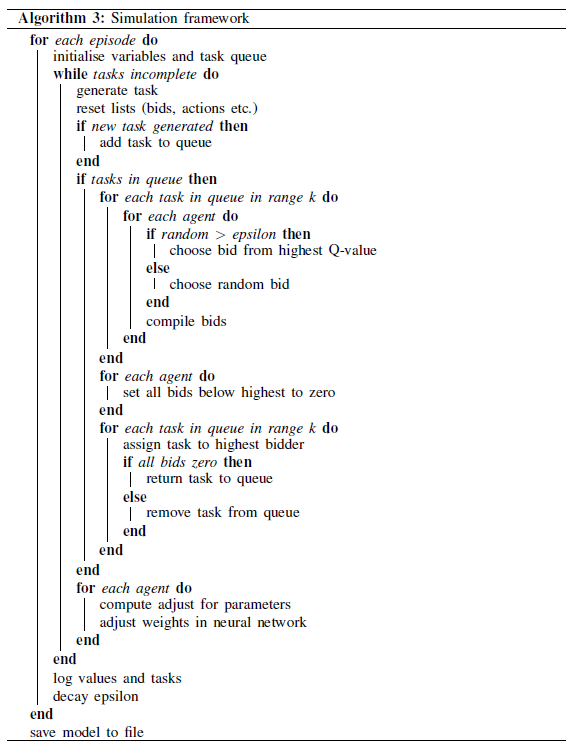
***Order***

* Defines an order and all the information attached to the order

**Helper** **functions**

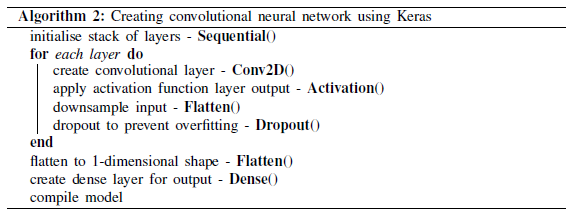
* generateParam(): used to generate random parameters for tasks
* argMax(): used to find the index of the highest bidder
* checkBids(): check if all bids are zero

**Main** **loop** (pg 24 of Thesis)



# Other Information

* DQNAgent and ModifiedTensorBoard classes inspired from <https://pythonprogramming.net/deep-q-learning-dqn-reinforcement-learning-python-tutorial/>
  + Useful tutorial to start off with deep q networks and to further understand these classes
  + Creation of neural network model (pg 19 of thesis)



* + Overall training process:
    - Checks if there is sufficient memory to train
    - Takes minibatch - a sample of the memory available
    - Get the **current states** from minibatch
    - Get the q-values based on the current states taken from minibatch - these values are predicted using the **model**
    - Get the **new states** from minibatch
    - Get the new q-values based on the new states taken from minibatch - these are predicted using the **target model**
    - For each set of data in the minibatch
      * The q-values are updated using the Bellman equation
      * The current state and new q is appended to the list of states and q values
    - Fit the model based on the “current states” and the q-values
    - If the desired number of iterations has been reached, then the target model is updated
  + Refer to page 18-19 of thesis for further information on TensorFlow and Keras
* State/Observation is taken as an RGB array that consists of
  + the end positions of the task the robot currently has - which is its current position if no task has been assigned
  + the start and end position of the task being offered
* Results.zip – contains Matlab scripts that can be adapted to plot own results; also contains figures of results obtained for different parameters – useful for comparisons.