

Computer Games Development CW208

Software Functional Specification

Year IV

Joshua Dunne

C00241588

|  |  |
| --- | --- |
|  |  |
| 12/12/2021 | |

[Declaration form to be attached]

Contents

[**Acknowledgements**](#_heading=h.gjdgxs) **2**

[**Functional Specification**](#_heading=h.45lvye5eaz43) **2**

[**References**](#_heading=h.f1m76clq9u8g) **6**

# Acknowledgements

I would like to thank the following people who assisted in completing this project including;

My project supervisor, Oisin Cawley, for helping to keep me on track with my progress, and giving me advice on what way to go with the project overall.

Nicholas Renotte for creating an incredibly helpful video that helps understand the basics of Reinforcement Learning and how to apply it for OpenAI Gym. [1]

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave and Anssi Kanervisto for creating and maintaining Stable Baselines 3, which helped greatly with policy algorithms to help make the process of creating models and using them on environments a lot easier. [2]

The many people behind creating Gym, which is part of Open AI, making it easier to try out environments other people made to help further my understanding of how to make one myself. [3]

# Functional Specification

**Specify the major functionalities of the software system (use screen shots if appropriate).**

For my FYP, I wish to look into how randomness affects Reinforcement Learning. By using pre-existing environments, alongside my own custom made environment, I hope to compare two versions of agents per environment, one that works off the non-random environment, and another that is exposed to a non-deterministic element during training.

In order to achieve this, the software system to be implemented will require use of OpenAI Gym, specifically using two environments, called CartPole and CarRacing.

The software system will run on Jupyter Notebook, which will be supplied by Anaconda. These notebooks will be able to run off a fresh install, with one or two packages required for the environments to run properly (such as Pyglet or Swig).

At the end of this project, there will be a few notebooks which will house the majority of the code that I’ve implemented. These will be set up in a way such that they are easy to run without needing any external code.

This will all be set up such that all notebooks can be run to generate logs for Tensorflow, in order to compare final model data against each other to compare and contrast differences between models trained in predetermined environments, and environments full of entropy (chaos).

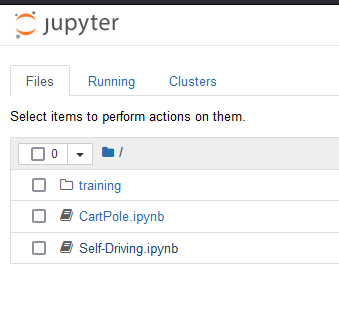
**Design and describe how the application will be used: e.g. Navigation sequence in Web based project.**

**This may necessitate some high level design in order to convey how the application may be used to access its functionality.**

**User interfaces should be specified loosely here (possibly screenshots of prototype user interfaces - you will not be held to using these UIs; they are just to assist in explanation of your project functionality).**

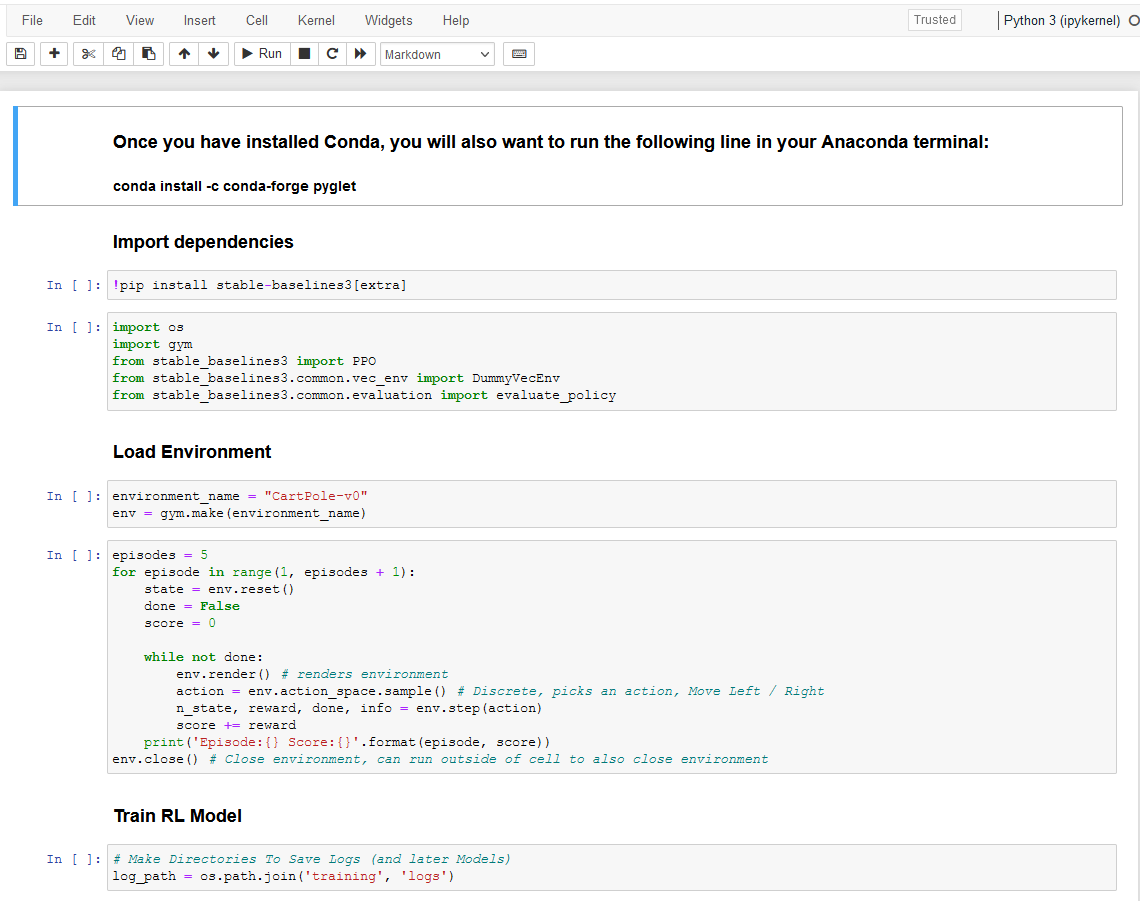
Before the User can access the notebooks, they will be prompted to install Anaconda, in order to access Jupyter Notebook which will be used to run said notebooks.

When a user has Anaconda installed, they will be prompted to run Jupyter Notebook in the same folder that features the notebooks. The user will then select a notebook of their choice in order to view / run the environment and model.

  
*Jupyter Notebook on the Home Screen, where a Notebook can be accessed.*

Once the user has access to the Notebooks, and can open them, they will be instructed to follow each step, and will be told how to run each cell, and in what order, so that they can choose whether to train a model, or to view a pre-existing model running in the environment.

The Notebook will also tell the user if there are any other libraries that Anaconda requires in order to run the environment, and will specify where/how to install it on Anaconda.

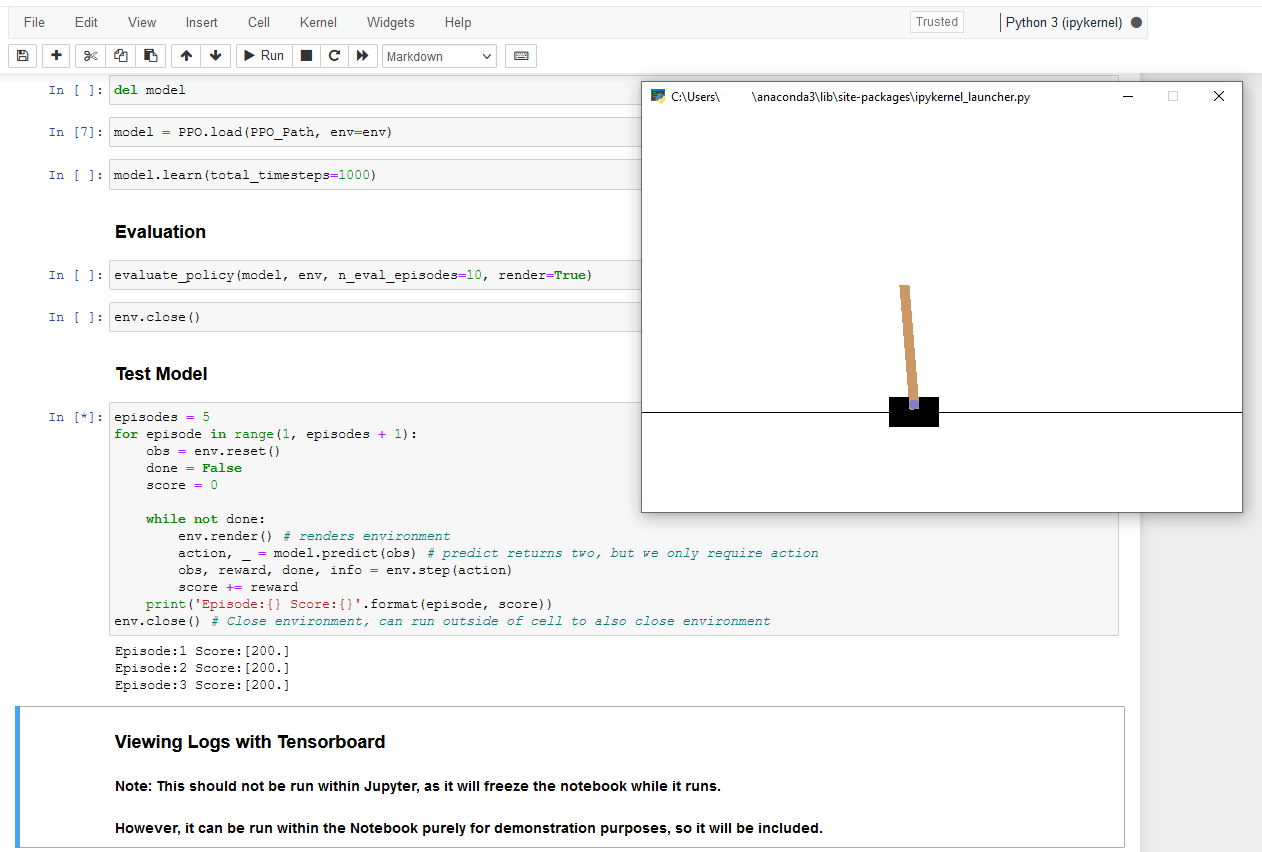
  
*An example of a Notebook opened, with instructions on what is needed to download, and what each section is for*

The user will be able to run each cell. The notebook will set up the environment it was created for, and create a new model or load an existing model, depending on what the user chooses.

When the user is ready, they can run a cell to either put the model through reinforcement learning to better improve how they solve the environment, or they can run the model and watch it play the environment to judge for themselves if the model requires more training.

While the model is playing the environment, the user will see a window pop up. This window will render the environment to the user, so they may observe the model.

This same window will not appear during reinforcement learning, as we want to save resources during training.



*What the user will see when they run the cell to view the model running in the environment.*

Once the user has viewed the model, they can freely choose to train the model more, open a different notebook, or close Jupyter Notebook, as they are finished with viewing.

# References

*[1] Renotte, Nicholas. “Reinforcement Learning in 3 Hours | Full Course Using Python.” YouTube, 6 June 2021,* [*https://www.youtube.com/watch?v=Mut\_u40Sqz4*](https://www.youtube.com/watch?v=Mut_u40Sqz4)*.*

*[2] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, & Noah Dormann (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. Journal of Machine Learning Research, 22(268), 1-8.*

*[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, & Wojciech Zaremba. (2016). OpenAI Gym.*

*[4] Tewari, U. (2020, April 14). Which reinforcement learning-RL algorithm to use where, when and in what scenario? Medium.Datadriveninvestor.Com. Retrieved January 27, 2022, from* [*https://medium.datadriveninvestor.com/which-reinforcement-learning-rl-algorithm-to-use-where-when-and-in-what-scenario-e3e7617fb0b1*](https://medium.datadriveninvestor.com/which-reinforcement-learning-rl-algorithm-to-use-where-when-and-in-what-scenario-e3e7617fb0b1)

*[5] Base RL Class — Stable Baselines3 1.5.1a4 documentation. (2020, June 9). Stable-Baselines3.Readthedocs.Io. Retrieved January 27, 2022, from* [*https://stable-baselines3.readthedocs.io/en/master/modules/base.html*](https://stable-baselines3.readthedocs.io/en/master/modules/base.html)

*[6] Moni, R. (2021, December 7). Reinforcement Learning algorithms — an intuitive overview. Medium. Retrieved January 27, 2022, from* [*https://smartlabai.medium.com/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc*](https://smartlabai.medium.com/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc)

*[7] Simonini, T. (2019, February 5). Proximal Policy Optimization (PPO) with Sonic the Hedgehog 2 and 3. Medium. Retrieved January 27, 2022, from* [*https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e*](https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e)

*[8] freeCodeCamp.org. (2018, March 29). An introduction to Deep Q-Learning: let’s play Doom. Retrieved January 27, 2022, from* [*https://www.freecodecamp.org/news/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8/*](https://www.freecodecamp.org/news/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8/)

*[9] Reinforcement Learning Tips and Tricks — Stable Baselines 2.10.2 documentation. (n.d.). Stable-Baselines.Readthedocs.Io. Retrieved January 27, 2022, from* [*https://stable-baselines.readthedocs.io/en/master/guide/rl\_tips.html#which-algorithm-should-i-use*](https://stable-baselines.readthedocs.io/en/master/guide/rl_tips.html#which-algorithm-should-i-use)

*[10] Wikipedia contributors. (2022, April 12). Multilayer perceptron. Wikipedia. Retrieved March 24, 2022, from* [*https://en.wikipedia.org/wiki/Multilayer\_perceptron*](https://en.wikipedia.org/wiki/Multilayer_perceptron)

*[11] An Overview on Multilayer Perceptron (MLP). (2022, February 21). Simplilearn.Com. Retrieved March 24, 2022, from* [*https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron*](https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron)

*[12] Miyoung Han. Reinforcement Learning Approaches in Dynamic Environments. Databases [cs.DB]. Télécom ParisTech, 2018. English. tel-01891805*

*[13] R. S. Sutton and A. G. Barto. Introduction to Reinforcement Learning. MIT Press, 1998.*

*[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, 2017.*

*[15] PPO — Stable Baselines3 1.5.1a4 documentation. (n.d.). Stable Baslines 3. Retrieved January 27, 2022, from https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html*