Proposal

Team members:

- Justin Stephenson - 216008419
- Chloe Dimpas - 214151021
- Neves Soares - 215782832

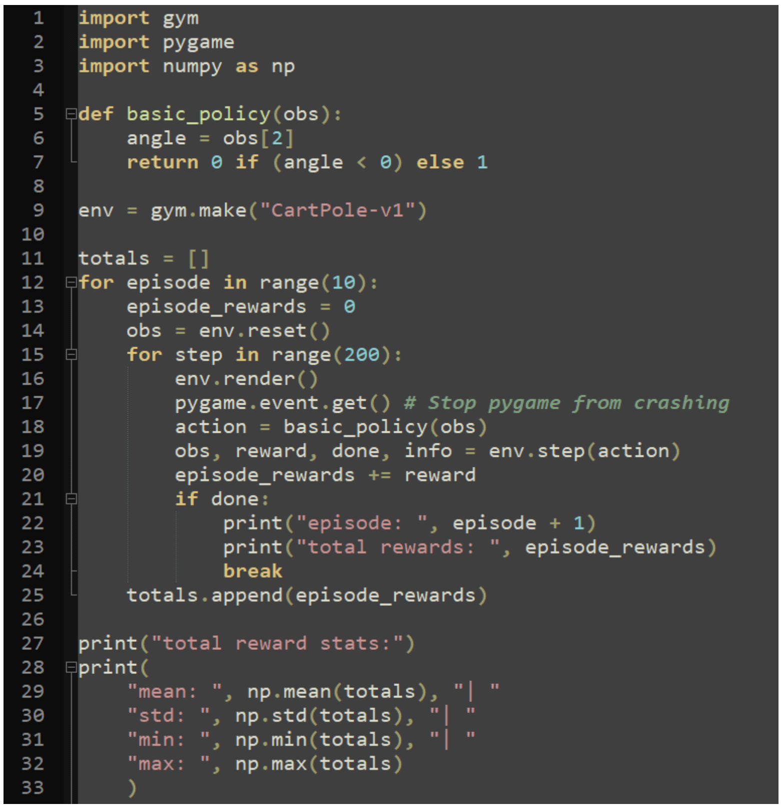
Topic Outline:
The topic our group is interested in is the implementation of reinforcement algorithms on various different problems from OpenAI Gym. We will compare and contrast different algorithms, as well as compare and contrast how these algorithms perform when increasing the complexity of the problem space (environment).

Project Objective:
The objective of this project is to get a better understanding of how reinforcement algorithms work and the pros and cons that pertain to each of them. We will compare and contrast each algorithm and figure out what algorithms are better for certain problems and why. We will be exploring reinforcement algorithms such as Q-learning, and SARSA.

Task Timetable:

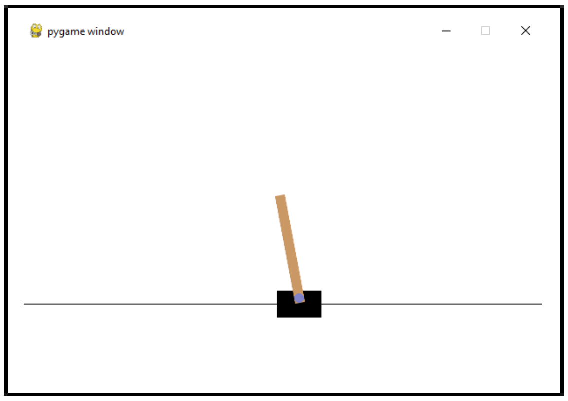
Task	**Time Frame**
Get a grasp of all the different reinforcement algorithms out there, and choose the ones we want to implement.	March 12 - March 18
Implement the various algorithms using problems from OpenAI Gym.	March 19 - March 25
Tinker with the algorithms and different problem spaces to get a better understanding of their benefits and faults.	March 26 - April 1
Create presentation for our project	April 1 - April 4
Rehearse and present project presentation	April 5 - April 7
Create report for our project	April 8 - April 11

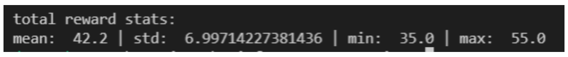
Preliminary Design of the System:
The goal of this project is to explore reinforcement algorithms. Specifically we want to explore different ‘policies’. A policy is the action an agent will take at any given state. A basic design with a crude policy might look something like this:



```python
import gym
import pygame
import numpy as np

def basic_policy(obs):
    angle = obs[2]
    return 0 if (angle < 0) else 1

env = gym.make("CartPole-v1")

totals = []
for episode in range(10):
    episode_rewards = 0
    obs = env.reset()
    for step in range(200):
        env.render()
        pygame.event.get() # Stop pygame from crashing
        action = basic_policy(obs)
        obs, reward, done, info = env.step(action)
        episode_rewards += reward
        if done:
            print("episode: ", episode + 1)
            print("total rewards: ", episode_rewards)
            break
    totals.append(episode_rewards)

print("total reward stats:")
print(
    "mean: ", np.mean(totals), "| "
    "std: ", np.std(totals), "| "
    "min: ", np.min(totals), "| "
    "max: ", np.max(totals)
    )
```

Cart Pole problem from OpenAI





```
total reward stats:
mean:  42.2 | std:  6.99714227381436 | min:  35.0 | max:  55.0
```

This policy just detects if the pole is tilting to the left or the right and tries to move the cart in the opposite direction. While this is a poor policy, this is a quick mock up of an example that could be used to show off reinforcement learning algorithms. We might for instance replace the policy with a Q-learning algorithm that will make better decisions based on the state of the agent. Of course we will output more useful stats that we can use to compare and contrast; statistics on total rewards is a good start.

References

Géron, A., & Demarest, R. (2019). *Hands-on machine learning with scikit-learn and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. O'Reilly.