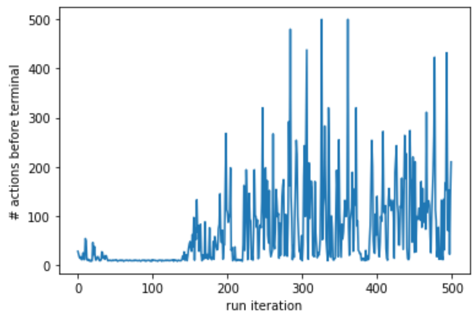
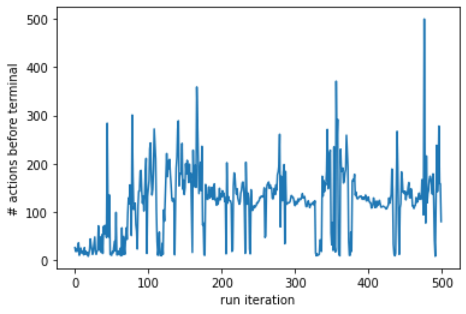
I tried to keep my model as simple and small as possible as reinforced learning requires a lot of computing resources. I started with a model which was 5 layers deep and this yielded me a base line result of 125 as the mean of the last 15 runs. My second run just took way to long to compute >48h so I had to stop that one and change the parameters in such a way to consume less computing resources. The three most impactful factors for computing time are the number of runs, the batch size and the size of the model. When trying bigger vs smaller models I could conclude that bigger models don't give much of an advantage and especially not at the increased computing time that they require, therefore I choose to only add 2 hidden layers. For the number of runs I kept it at 500, as models might still improve after 500 runs, but after 500 runs it is possible to properly compare different models or parameters as they should have learned significantly by then. And finally the batch size, from my testing batch size seemed to be the most influential parameter for increasing the quality, however it also drastically increased the computing time (2nd attempt had a batch size of 50 and took way too long), this is due to the fact that after each run the model has to be trained again by taking the batch size number of previous decisions and results.

Results:

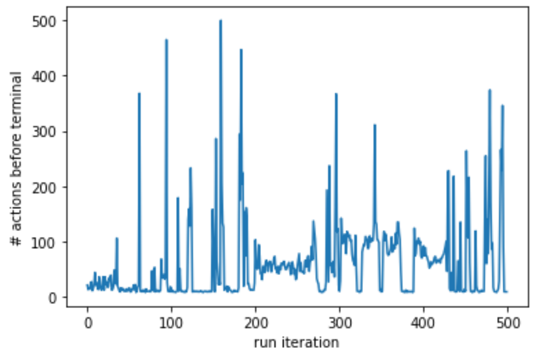
Attempt 1



attempt 3



Attempt 4



Attempt 1 is very volatile after 150 iterations, this is probably due to the relatively low batch size. In attempt 3 I tried a higher batch size, this resulted in longer computing times but also a more stable result. In attempt 4 I tried using more layers and nodes, but this didn't really help.

At the time of writing this, I'm still running attempt 5 and 6 which use a higher batch size than attempt 3 but lower than attempt 2 (which I had to stop). 5 and 6 each use a different amount of nodes and slightly different amount of batch size.

For this exercise the computing times proved to be exceptionally difficult, I choose to run the code locally as this is faster than google colab and secondly it doesn't stop after a couple of hours. Over the last week I've been running code almost continuously 24/7 and still only managed to get 3 attempts done (and attempt 2 which had to be stopped early). And while I was planning on doing the pac man game as well, I even installed all packages and got it too run, but the pacman game is way more complicated (more inputs and outputs) than the cartpole game and thus it would require even longer to run the code and obtain results. Therefore, I decided to focus my computing time on the cartpole game to ensure I got somewhat decent results.

Of the attempts that are finished, nr 3 is clearly the best, it has a high average while not being as volatile as nr 1 or 4. However attempt 5 and 6 seem to be doing better as they are busy with iteration 180-ish and seem to be having a high average (scores between 120-300 most common) and seem to be not too volatile. Unfortunately due to the high computing demands of reinforced learning these attempts will not be finished before the deadline.