Spike: Lab 3
Title: Goal Oriented Behaviour

Author: Thomas Latimer, 9718648

Goals / deliverables:
The primary goal of this spike is to fill in the blanks on a simple A.I. program such as how certain actions are applied and comparing values and implementation. The main goal of this spike is to understand simple goal instance when used to resolve a problem using Goal Oriented Behaviour Extra functions were added such as rendering, update and other human out related functions.

Technologies, Tools, and Resources used:

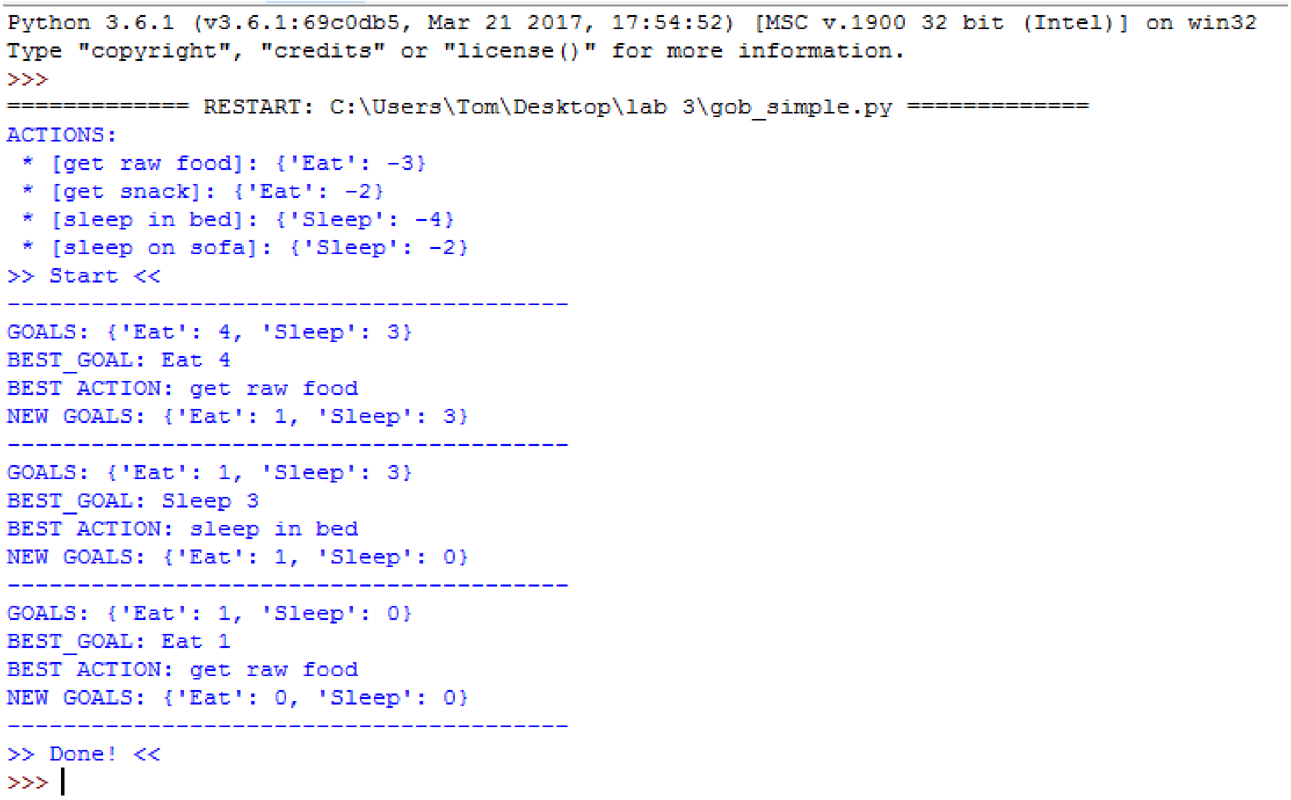
- Pycharm 3.1. for Microsoft computers
- Python 3.3, for separate testing.
- Additional research e.g. the python website.
- Microsoft word for the construction of this document.
- Refer to simple code A.I. in lab 3.
- This spike is known as lab_3_report_9718648.

Tasks undertaken:
This section should resemble a tutorial – the goal is to allow another coder to reproduce your work following these steps.

- Download and install PyCharm.
- Read through the gob-simple.py file and go over the code.
- Find out what the blank spaces are (### - comment type hint) and fill them in.
- Compile sample code.
- Keep testing to make sure the program works and find out how it can possibly be extended.
- Read the source code.
- Download and install Python (only good for testing).
- Run code.
- Write Spike Report.

What we found out:

The outcome was testing the program, Making sure the errors were corrected and what happened once the code was executed. We found out the effectiveness of Simple Goal Instance is that it allows the game agents to perform actions based on the insistence or goal varying on what it is. Referring to the output below, the best goal changes after an action, the higher the goal, the more it will influence the agent into making a certain action.



```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
============= RESTART: C:\Users\Tom\Desktop\lab 3\gob_simple.py =============
ACTIONS:
 * [get raw food]: {'Eat': -3}
 * [get snack]: {'Eat': -2}
 * [sleep in bed]: {'Sleep': -4}
 * [sleep on sofa]: {'Sleep': -2}
>> Start <<
----------------------------------------
GOALS: {'Eat': 4, 'Sleep': 3}
BEST_GOAL: Eat 4
BEST ACTION: get raw food
NEW GOALS: {'Eat': 1, 'Sleep': 3}
----------------------------------------
GOALS: {'Eat': 1, 'Sleep': 3}
BEST_GOAL: Sleep 3
BEST ACTION: sleep in bed
NEW GOALS: {'Eat': 1, 'Sleep': 0}
----------------------------------------
GOALS: {'Eat': 1, 'Sleep': 0}
BEST_GOAL: Eat 1
BEST ACTION: get raw food
NEW GOALS: {'Eat': 0, 'Sleep': 0}
----------------------------------------
>> Done! <<
>>> 
```

The disadvantage of using Simple Goal Instance is that it leaves the agent feeling too content and cannot do anything afterwards (unless the goal was too simple). It doesn't take into consideration the side effects because so far, there were only positive outcomes, to make it more convincing there needs to be negative effects as well, such as certain values to balance things out. Add some additional functions or values to make it more realistic.

Open issues/risks:

- I did have some trouble with the code prior because even though the comments specified what needed to be done with the program such as fill in some of the missing blanks but the issues I had was how to fill in those missing pieces.
- There was also the issue earlier on with what program to run it on whether it would be a browser, python or Pycharm. Getting those programs mixed up is not good for progress.
- I eventually figured out what

Recommendations:

- Make sure you use one program you are comfortable with and stick with it because performance varies on each program you come across when coding in them. My choice was Pycharm rather than python.
- Make sure you do the right kind of research because you need to make sure what those missing spaces are and try to find a good way to fill them without ripping another person off.
- The types of functions I mentioned earlier on were references to the second lab where we had to create a Smart A.I. which behaved according to how the player did. Those functions were supposed to actually get the program working properly not necessarily program an A.I. the errors were listed and acted on accordingly.