Spike: 1
Title: Goal oriented behaviour and Simple goal insistence

Author: Ryan Chessum, 102564760

Goals / deliverables:
Create a simple goal insistence (SGI) model simulation of goal-oriented behaviour (GOB) that demonstrates the both the effectiveness and the limitations of the technique.

You will need to deliver (show your tutor) the following items:

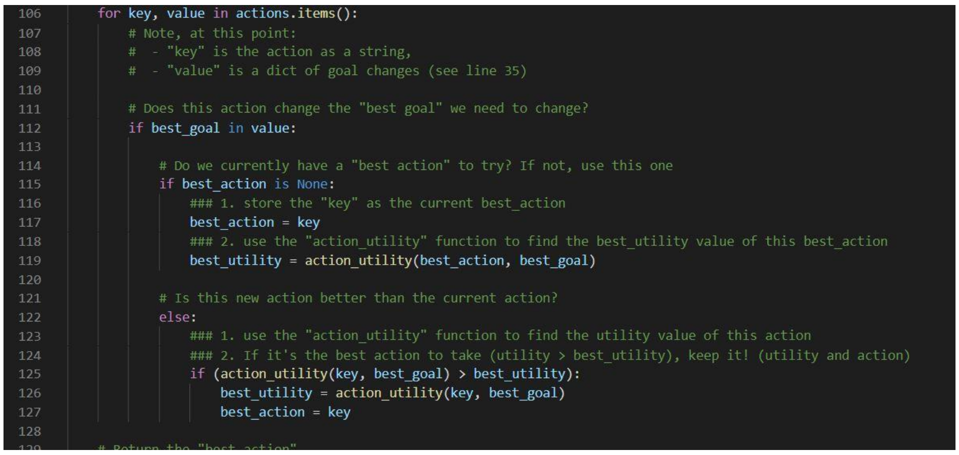
• Working code that simulates and displays GOB using SGI.
• You must demonstrate a situation where SGI works appropriately
• You must demonstrate a situation where SGI does not work well.

Technologies, Tools, and Resources used:
List of information needed by someone trying to reproduce this work

- Visual Studio Code
- Python 3.8.0

Tasks undertaken:

- Download Visual Studio Code
- Install Python
- Download the GOB sample code from Canvas
- Add code into the loop in the choose_action() function to compare each action and determine which one should be used to lower the number of the most urgent goal using the action_utility() function
- Add new effects to each action
- Add code into action utility that disincentivises using an action that would bring the goal to below 0
- Add code that takes into account the negative effects of actions

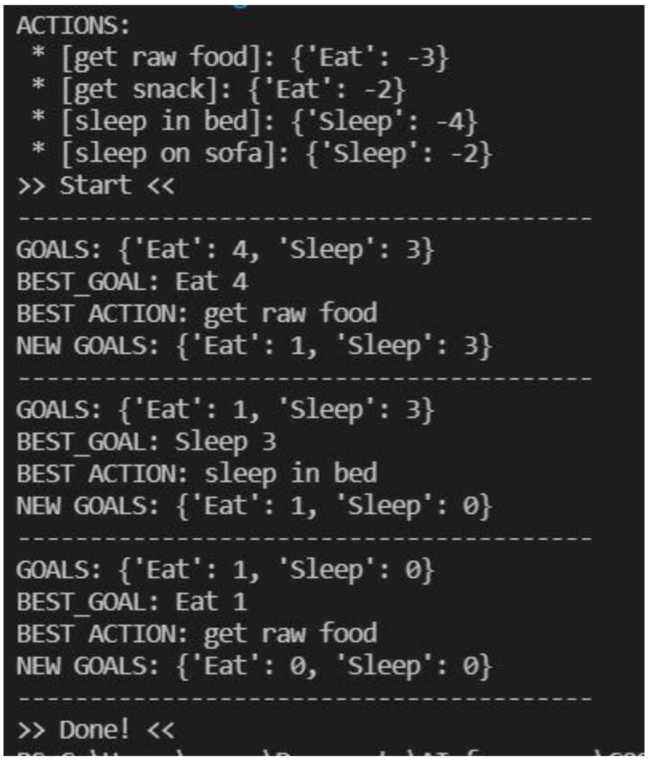


```python
    for key, value in actions.items():
        # Note, at this point:
        #  - "key" is the action as a string,
        #  - "value" is a dict of goal changes (see line 35)

        # Does this action change the "best goal" we need to change?
        if best_goal in value:

            # Do we currently have a "best action" to try? If not, use this one
            if best_action is None:
                ### 1. store the "key" as the current best_action
                best_action = key
                ### 2. use the "action_utility" function to find the best_utility value of this best_action
                best_utility = action_utility(best_action, best_goal)

            # Is this new action better than the current action?
            else:
                ### 1. use the "action_utility" function to find the utility value of this action
                ### 2. If it's the best action to take (utility > best_utility), keep it! (utility and action)
                if (action_utility(key, best_goal) > best_utility):
                    best_utility = action_utility(key, best_goal)
                    best_action = key

    # Return the "best action"
```

What we found out:

Simple Goal Insistence (SGI) is a basic way of making an AI with Goal-Oriented Behaviour (GOB). It compares each goal to see which is more urgent or insistent and then compares each action to see which one would be best to use to satisfy that goal.

The example code provided had 2 goals, food and sleep. It also had 4 actions which affected each goal by different amounts.



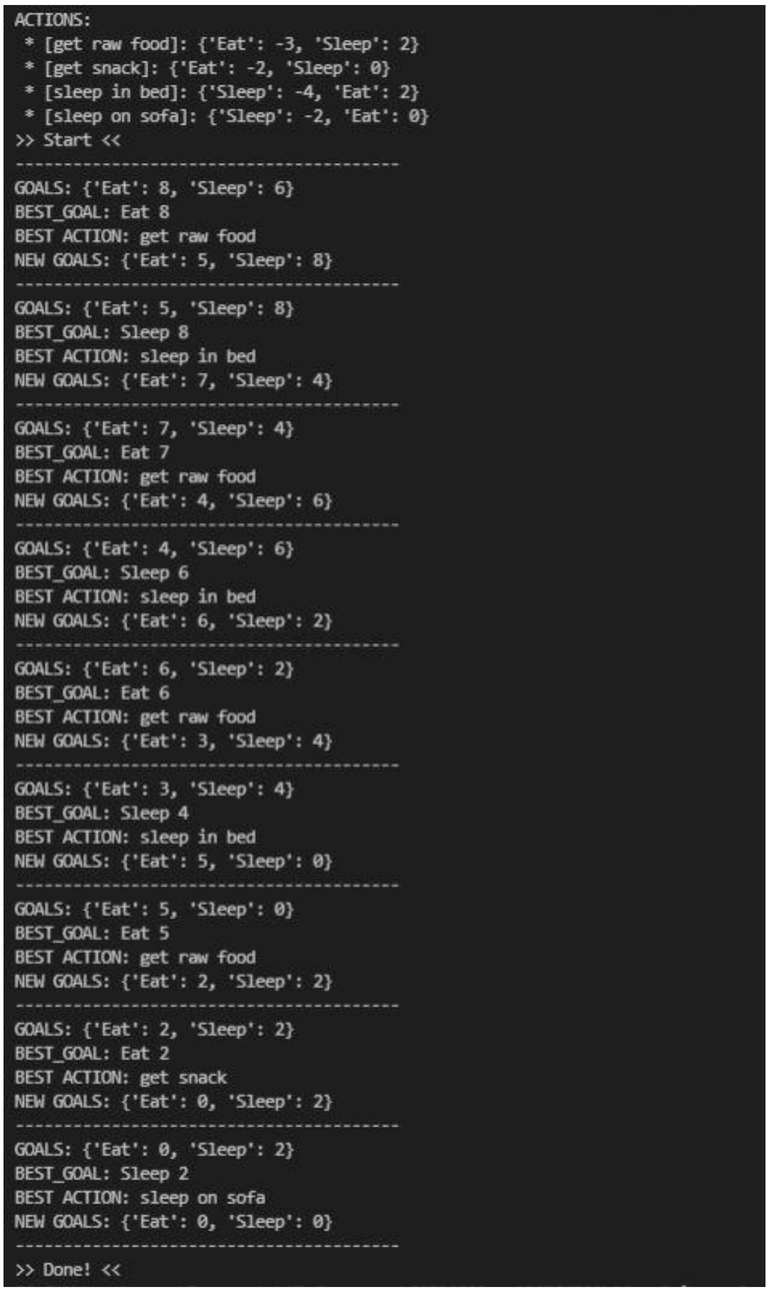
```
ACTIONS:
 * [get raw food]: {'Eat': -3}
 * [get snack]: {'Eat': -2}
 * [sleep in bed]: {'Sleep': -4}
 * [sleep on sofa]: {'Sleep': -2}
>> Start <<
----------------------------------------
GOALS: {'Eat': 4, 'Sleep': 3}
BEST_GOAL: Eat 4
BEST_ACTION: get raw food
NEW GOALS: {'Eat': 1, 'Sleep': 3}
----------------------------------------
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! <<
```

As you can see from the output the AI is effective at making choices based on it’s currently most insistent goal. This model works great for ai that need to perform different tasks based on the most important one at the time.

For instance, if the AI had low health in a game, this model could be set up so the lower the health of the ai the higher the insistence value for healing actions would go up.

However, due to the way the example determines which action to take it never uses 2 of the 4 actions, even when the values for the goals are low. In our ai healing example this might also happen, perhaps it might use a healing item that heals an excessive amount instead of one that heals a smaller amount. To fix this issue the AI would have to consider more factors when deciding what action to take.

After editing the example code provided the example now also considers weather the amount the action reduces the goal by will take it below 0. It also has penalties for using some actions and the penalties are taken into account when the action could bring a goal below 0.



```
ACTIONS:
 * [get raw food]: {'Eat': -3, 'Sleep': 2}
 * [get snack]: {'Eat': -2, 'Sleep': 0}
 * [sleep in bed]: {'Sleep': -4, 'Eat': 2}
 * [sleep on sofa]: {'Sleep': -2, 'Eat': 0}
>> Start <<
----------------------------------------
GOALS: {'Eat': 8, 'Sleep': 6}
BEST_GOAL: Eat 8
BEST ACTION: get raw food
NEW GOALS: {'Eat': 5, 'Sleep': 8}
----------------------------------------
GOALS: {'Eat': 5, 'Sleep': 8}
BEST_GOAL: Sleep 8
BEST ACTION: sleep in bed
NEW GOALS: {'Eat': 7, 'Sleep': 4}
----------------------------------------
GOALS: {'Eat': 7, 'Sleep': 4}
BEST_GOAL: Eat 7
BEST ACTION: get raw food
NEW GOALS: {'Eat': 4, 'Sleep': 6}
----------------------------------------
GOALS: {'Eat': 4, 'Sleep': 6}
BEST_GOAL: Sleep 6
BEST ACTION: sleep in bed
NEW GOALS: {'Eat': 6, 'Sleep': 2}
----------------------------------------
GOALS: {'Eat': 6, 'Sleep': 2}
BEST_GOAL: Eat 6
BEST ACTION: get raw food
NEW GOALS: {'Eat': 3, 'Sleep': 4}
----------------------------------------
GOALS: {'Eat': 3, 'Sleep': 4}
BEST_GOAL: Sleep 4
BEST ACTION: sleep in bed
NEW GOALS: {'Eat': 5, 'Sleep': 0}
----------------------------------------
GOALS: {'Eat': 5, 'Sleep': 0}
BEST_GOAL: Eat 5
BEST ACTION: get raw food
NEW GOALS: {'Eat': 2, 'Sleep': 2}
----------------------------------------
GOALS: {'Eat': 2, 'Sleep': 2}
BEST_GOAL: Eat 2
BEST ACTION: get snack
NEW GOALS: {'Eat': 0, 'Sleep': 2}
----------------------------------------
GOALS: {'Eat': 0, 'Sleep': 2}
BEST_GOAL: Sleep 2
BEST ACTION: sleep on sofa
NEW GOALS: {'Eat': 0, 'Sleep': 0}
----------------------------------------
>> Done! <<
```

As you can see in the output the example now also chooses the other actions when making its decisions.

To achieve this requires the AI to make more assessments to determine it's next action. This presents another issue with Goal Oriented Behaviours, as the complexity increases with the amount of goals and actions, so does the cost in computing power. In the example code this is barely an issue however if you were to use this method in a real game with an AI that had many goals and possible actions as well as many instances of that same ai running the calculations at the same time, the amount of processing power can become very high.

So, if you were to make an AI with similar Goal Oriented Behaviours you should keep it simpler in order to keep the processing cost as low as possible. Perhaps it would be better to only run the assessments at certain times, say after every action has been fully completed or only once in a set timeframe.