Spike: Spike Week 8
Title: Goal-Oriented Action Planning (GOAP)

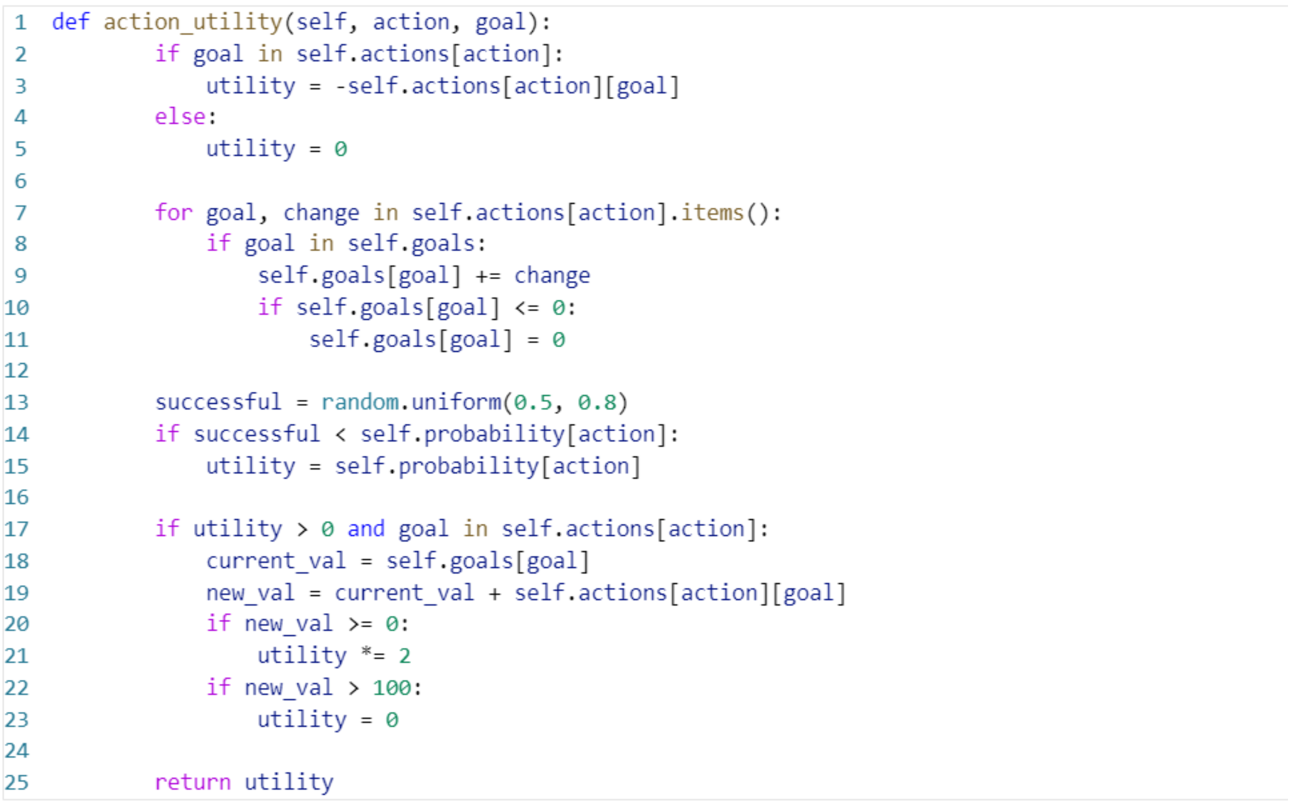
Author: Nguyen Khanh Toan - 104180605

Goals / deliverables:
Develop a GOAP simulation that showing the effectiveness of the approach in factoring in the long-term consequences of actions, such as side-effects and time delays, while also exhibiting intelligent planning and execution.

Technologies, Tools, and Resources used:
- Visual Studio Code
- Python 3.12

Tasks undertaken:
- Install Python 3+
- Install and setup compatible IDE for the language, e.g.: Visual Studio Code
- Pay attention to the comment of how the code work and functionality. Can use debug tool to observe the program more clearly.
- Run the code and observing the output.

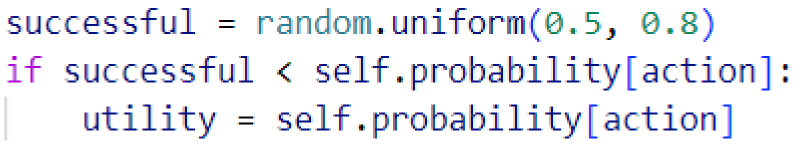
What we found out:
- Action with the consequences are incorporated within a set and can be able to modify.
- Until all of the goals downed to 0, the program repeatedly choosing best action to achieve goal.
- However, the goal oriented of the agent might not be “smart” because it would not comparing which goal with action to priority, so in the “gob_simple_SGI_fail.py” file, the program stuck in the infinite loop because while solving the goal, it chooses the action (faster solve the current goal) that raise “unhappiness” value instead of the action solve the goal slower but not raising other undesirable goal.
- This spike is an enhancement from SGI task 7 by leverage the action_ultility function. In addition, this task also include several possibility of factors that could contribute to the successful of actions.



```
1  def action_utility(self, action, goal):
2          if goal in self.actions[action]:
3              utility = -self.actions[action][goal]
4          else:
5              utility = 0
6
7          for goal, change in self.actions[action].items():
8              if goal in self.goals:
9                  self.goals[goal] += change
10                 if self.goals[goal] <= 0:
11                     self.goals[goal] = 0
12
13         successful = random.uniform(0.5, 0.8)
14         if successful < self.probability[action]:
15             utility = self.probability[action]
16
17         if utility > 0 and goal in self.actions[action]:
18             current_val = self.goals[goal]
19             new_val = current_val + self.actions[action][goal]
20             if new_val >= 0:
21                 utility *= 2
22             if new_val > 100:
23                 utility = 0
24
25         return utility
```

Figure 1: action_utility function

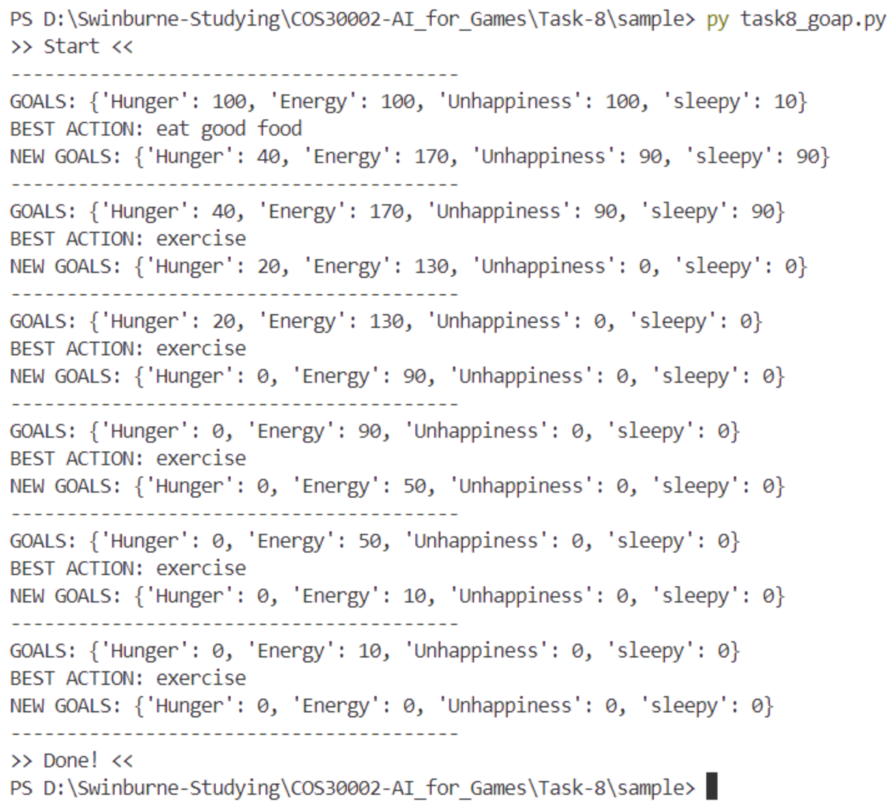
- For example, the previous task, action_ultility only calculate based on the current goal, but in this task extension, it add some extra functionality to calculate ultility:
    - It check for successful of the action.



```
successful = random.uniform(0.5, 0.8)
if successful < self.probability[action]:
    utility = self.probability[action]
```

    - It considers the positive utility actions and multiplies the utility by 2 if the new goal value is greater than or equal to 0. If the new goal value is greater than 100, it sets the utility to 0.
    - If the utility action is positive and it already execute in line 3, it mean that if action consequences lead to a bad result (new_val >100), then it would considered as low utility.

-



```
PS D:\Swinburne-Studying\COS30002-AI_for_Games\Task-8\sample> py task8_goap.py
>> Start <<
----------------------------------------
GOALS: {'Hunger': 100, 'Energy': 100, 'Unhappiness': 100, 'sleepy': 10}
BEST ACTION: eat good food
NEW GOALS: {'Hunger': 40, 'Energy': 170, 'Unhappiness': 90, 'sleepy': 90}
----------------------------------------
GOALS: {'Hunger': 40, 'Energy': 170, 'Unhappiness': 90, 'sleepy': 90}
BEST ACTION: exercise
NEW GOALS: {'Hunger': 20, 'Energy': 130, 'Unhappiness': 0, 'sleepy': 0}
----------------------------------------
GOALS: {'Hunger': 20, 'Energy': 130, 'Unhappiness': 0, 'sleepy': 0}
BEST ACTION: exercise
NEW GOALS: {'Hunger': 0, 'Energy': 90, 'Unhappiness': 0, 'sleepy': 0}
----------------------------------------
GOALS: {'Hunger': 0, 'Energy': 90, 'Unhappiness': 0, 'sleepy': 0}
BEST ACTION: exercise
NEW GOALS: {'Hunger': 0, 'Energy': 50, 'Unhappiness': 0, 'sleepy': 0}
----------------------------------------
GOALS: {'Hunger': 0, 'Energy': 50, 'Unhappiness': 0, 'sleepy': 0}
BEST ACTION: exercise
NEW GOALS: {'Hunger': 0, 'Energy': 10, 'Unhappiness': 0, 'sleepy': 0}
----------------------------------------
GOALS: {'Hunger': 0, 'Energy': 10, 'Unhappiness': 0, 'sleepy': 0}
BEST ACTION: exercise
NEW GOALS: {'Hunger': 0, 'Energy': 0, 'Unhappiness': 0, 'sleepy': 0}
----------------------------------------
>> Done! <<
PS D:\Swinburne-Studying\COS30002-AI_for_Games\Task-8\sample>
```

Figure 2: Output