Spike: 08
Title: Goal-Oriented Action Planning (GOAP)

Author: Ben Holmes, 103024841

Goals / deliverables:
1. Create a GOAP simulation that demonstrates the effectiveness of the technique in considering long-term outcomes of actions (related to side-effects and/or time delays) and can plan and act intelligently.

Technologies, Tools, and Resources used:
- Visual Studio Code
- Python 3.12.2

Tasks undertaken:

- Copied gop from task 7
- Adjusted to use 3 values instead of 2 and renamed them to energy, hunger and fitness (fitness still wants to go to 0, its lack of fitness shortened to fitness for ease of typing)
- Created new recursive function for path planning called action_paths, it loops through each possible action and applies it to a temp instance of the goals, in the loop this temp instance of the goals is then passed to another action_paths which loops through the next moves (move number 2). This eventually generates a path (moves done and final goals of moves done) in a Path object that is then added to a list of path objects called paths which is returned. Action paths also has a temp_paths which is looped through and appended to paths if another action_paths is called

```
def action_paths(path_goals, max_move, counter, path_moves):
    counter += 1
    result_paths = []
    for key, value in actions.items():
        path_goals_temp = path_goals.copy()
        path_moves_temp = path_moves[:]
        path_moves_temp.append(key)
        path_apply_action(key,path_goals_temp)
        if counter < max_move and any(value != 0 for goal, value in path_goals_temp.items()):
            temp_paths = action_paths(path_goals_temp,max_move,counter,path_moves_temp)
            for path in temp_paths:
                result_paths.append(path)
        else:
            path = Path(path_moves_temp,path_goals_temp)
            result_paths.append(path)

    return result_paths
```

- Path_apply_action, is the same as the old apply action just to a provided set of goals (path_goals

```
def path_apply_action(action,path_goals):
    '''Change all provided goal values using this action. An action can change multiple
    goals (positive and negative side effects).
    Negative changes are limited to a minimum goal value of 0.
    '''
    for goal, change in actions[action].items():
        path_goals[goal] = max(path_goals[goal] + change, 0)
```

- Action_paths is called in the chose_action_path function, which if a path has not been generated before, generates a paths list and then loops through it checking against the first paths goals to see if the path produces better results using the path_check function. (it also compares length of path and choses the shortest path if one is shorter as there is a check in action_paths that shortens the path if 0 0 0 is hit) The best path is stored in the global variable, the path moves are then iterated through till the move_counter (number of path moves) hits the move_amount (max number of path moves), once move_amount is hit, the counter is reset and next time chose action_paths is called, then the paths are regenerated.

```
def path_check(moves_number_check, goal_check,path):
    global move_depth
    best_goal, best_goal_value = max(goals.items(), key=lambda item: item[1])
    if len(path.moves) < move_depth:
        return True
    elif len(path.moves) > moves_number_check:
        return False

    if goal_check.get('Energy') >= path.goals.get('Energy') and goal_check.get('Hunger') >= path.goals.get('Hunger') and goal_check.get('Fitness') >= path.goals.get('Fitness
        return True
    if goal_check.get(best_goal) > path.goals.get(best_goal):
        return True

    return False
```

```
def choose_action_path():
    '''chose an action path and return current action on that path'''

    assert len(goals) > 0, 'Need at least one goal'
    assert len(actions) > 0, 'Need at least one action'
    global move_counter
    global path_chosen
    global move_amount

    if move_counter == 0:
        paths = action_paths(goals,move_depth,0,[])
        path_number = 0
        goals_check = paths[0].goals.copy()
        moves_number_check = len(paths[0].moves)
        i = 0
        for path in paths:
            if path_check(moves_number_check,goals_check,path):
                path_number = i
                moves_number_check = len(path.moves)
                goals_check = path.goals.copy()
            i += 1

        path_chosen = paths[path_number]

    move_amount = len(path_chosen.moves) - 1
    apply_action(path_chosen.moves[move_counter])
    return path_chosen.moves[move_counter]
```

Results:

Path move number shows how many moves through each path the choice is through.
Moved number shows the overall move count
Path moves is the list of moves the path generates (shows the same for all 3 or more path moves done)
best action is action done on that move
goals is current overall goals
path goals is final goals after path is run

Depth 3 last 2 paths chosen:

```
----------------------------------------
MOVE NUMBER: 55
GOALS: {'Energy': 2, 'Hunger': 1, 'Fitness': 0}
PATH MOVES: ['cook and eat food', 'have a nap', 'go to sleep']
PATH GOALS: {'Energy': 0, 'Hunger': 1, 'Fitness': 0}
PATH MOVE NUMBER: 1
BEST ACTION: cook and eat food
NEW GOALS: {'Energy': 4, 'Hunger': 0, 'Fitness': 0}
----------------------------------------
MOVE NUMBER: 56
GOALS: {'Energy': 4, 'Hunger': 0, 'Fitness': 0}
PATH MOVES: ['cook and eat food', 'have a nap', 'go to sleep']
PATH GOALS: {'Energy': 0, 'Hunger': 1, 'Fitness': 0}
PATH MOVE NUMBER: 2
BEST ACTION: have a nap
NEW GOALS: {'Energy': 3, 'Hunger': 0, 'Fitness': 0}
----------------------------------------
MOVE NUMBER: 57
GOALS: {'Energy': 3, 'Hunger': 0, 'Fitness': 0}
PATH MOVES: ['cook and eat food', 'have a nap', 'go to sleep']
PATH GOALS: {'Energy': 0, 'Hunger': 1, 'Fitness': 0}
PATH MOVE NUMBER: 3
BEST ACTION: go to sleep
NEW GOALS: {'Energy': 0, 'Hunger': 1, 'Fitness': 0}
----------------------------------------
MOVE NUMBER: 58
GOALS: {'Energy': 0, 'Hunger': 1, 'Fitness': 0}
PATH MOVES: ['cook and eat food', 'have a nap', 'have a nap']
PATH GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
PATH MOVE NUMBER: 1
BEST ACTION: cook and eat food
NEW GOALS: {'Energy': 2, 'Hunger': 0, 'Fitness': 0}
----------------------------------------
MOVE NUMBER: 59
GOALS: {'Energy': 2, 'Hunger': 0, 'Fitness': 0}
PATH MOVES: ['cook and eat food', 'have a nap', 'have a nap']
PATH GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
PATH MOVE NUMBER: 2
BEST ACTION: have a nap
NEW GOALS: {'Energy': 1, 'Hunger': 0, 'Fitness': 0}
----------------------------------------
MOVE NUMBER: 60
GOALS: {'Energy': 1, 'Hunger': 0, 'Fitness': 0}
PATH MOVES: ['cook and eat food', 'have a nap', 'have a nap']
PATH GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
PATH MOVE NUMBER: 3
BEST ACTION: have a nap
NEW GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
----------------------------------------
>> Done! <<
```

Depth 4 last 2 paths chosen:

```
----------------------------------------
MOVE NUMBER: 49
GOALS: {'Energy': 0, 'Hunger': 2, 'Fitness': 8}
PATH MOVES: ['go to the gym', 'cook and eat food', 'go to the gym', 'go to sleep']
PATH GOALS: {'Energy': 3, 'Hunger': 2, 'Fitness': 0}
PATH MOVE NUMBER: 1
BEST ACTION: go to the gym
NEW GOALS: {'Energy': 2, 'Hunger': 3, 'Fitness': 4}
----------------------------------------
MOVE NUMBER: 50
GOALS: {'Energy': 2, 'Hunger': 3, 'Fitness': 4}
PATH MOVES: ['go to the gym', 'cook and eat food', 'go to the gym', 'go to sleep']
PATH GOALS: {'Energy': 3, 'Hunger': 2, 'Fitness': 0}
PATH MOVE NUMBER: 2
BEST ACTION: cook and eat food
NEW GOALS: {'Energy': 4, 'Hunger': 0, 'Fitness': 4}
----------------------------------------
MOVE NUMBER: 51
GOALS: {'Energy': 4, 'Hunger': 0, 'Fitness': 4}
PATH MOVES: ['go to the gym', 'cook and eat food', 'go to the gym', 'go to sleep']
PATH GOALS: {'Energy': 3, 'Hunger': 2, 'Fitness': 0}
PATH MOVE NUMBER: 3
BEST ACTION: go to the gym
NEW GOALS: {'Energy': 6, 'Hunger': 1, 'Fitness': 0}
----------------------------------------
MOVE NUMBER: 52
GOALS: {'Energy': 6, 'Hunger': 1, 'Fitness': 0}
PATH MOVES: ['go to the gym', 'cook and eat food', 'go to the gym', 'go to sleep']
PATH GOALS: {'Energy': 3, 'Hunger': 2, 'Fitness': 0}
PATH MOVE NUMBER: 4
BEST ACTION: go to sleep
NEW GOALS: {'Energy': 3, 'Hunger': 2, 'Fitness': 0}
```

```
--------------------------------------
MOVE NUMBER: 53
GOALS: {'Energy': 3, 'Hunger': 2, 'Fitness': 0}
PATH MOVES: ['go to sleep', 'cook and eat food', 'have a nap', 'have a nap']
PATH GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
PATH MOVE NUMBER: 1
BEST ACTION: go to sleep
NEW GOALS: {'Energy': 0, 'Hunger': 3, 'Fitness': 0}
--------------------------------------
MOVE NUMBER: 54
GOALS: {'Energy': 0, 'Hunger': 3, 'Fitness': 0}
PATH MOVES: ['go to sleep', 'cook and eat food', 'have a nap', 'have a nap']
PATH GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
PATH MOVE NUMBER: 2
BEST ACTION: cook and eat food
NEW GOALS: {'Energy': 2, 'Hunger': 0, 'Fitness': 0}
--------------------------------------
MOVE NUMBER: 55
GOALS: {'Energy': 2, 'Hunger': 0, 'Fitness': 0}
PATH MOVES: ['go to sleep', 'cook and eat food', 'have a nap', 'have a nap']
PATH GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
PATH MOVE NUMBER: 3
BEST ACTION: have a nap
NEW GOALS: {'Energy': 1, 'Hunger': 0, 'Fitness': 0}
--------------------------------------
MOVE NUMBER: 56
GOALS: {'Energy': 1, 'Hunger': 0, 'Fitness': 0}
PATH MOVES: ['go to sleep', 'cook and eat food', 'have a nap', 'have a nap']
PATH GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
PATH MOVE NUMBER: 4
BEST ACTION: have a nap
NEW GOALS: {'Energy': 0, 'Hunger': 0, 'Fitness': 0}
--------------------------------------
>> Done! <<
```

What we found out:

For the deliverable, the simulation was created to have 6 actions with different increases and decreases and stability of the 3 goals so that the planning function had to account for side effects of its actions. The intelligent actions are determined in 2 ways, first is checking to see if the move count is less than the move depth (or current best path) then it choses that path. The next intelligent check is when it checks for the decrease in the largest goal after checking for all 3 values decreasing. This is to make each path more efficient. The second check improves efficiency at depth 3-4 but depth 5 its nearly identical.

If you wish to test the depth side of things, simply change the move depth number, although be warned at depth 6 and above it becomes much slower each increment.

If you wish to test the efficiency of the largest goal check please comment out these lines

```
90      if goal_check.get(best_goal) > path.goals.get(best_goal):
91          return True
```