

**Spike:** Spike\_No\_1

**Title:** Spike\_GOB

**Author:** Hasaan Akhtar, 102400615

**Goals / deliverables:**

- I updated the code so that it can exhibit goal Oriented behaviour
- The best action being opted based on the given goals

```
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:
            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
            utility = action_utility(best_action, best_goal)
            ### 2. If it's the best action to take (utility > best_utility), keep it! (utility and action)
            if utility > best_utility:
                utility = action_utility(best_action, best_goal)
                best_action = key

    # Return the "best action"
    return best_action
```

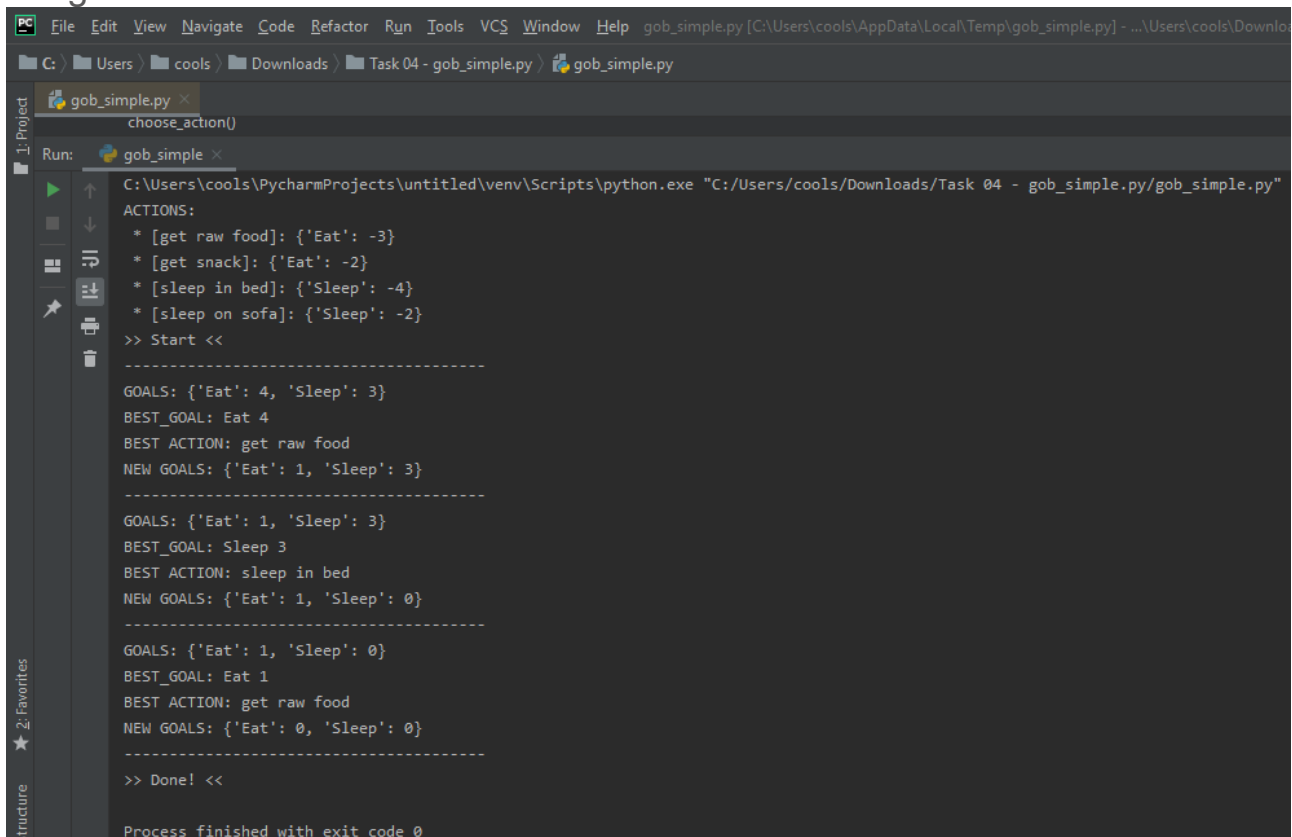
**Technologies, Tools, and Resources used:**

- Python IDE (PyCharm) used along with python interpreter

**Tasks undertaken:**

- Download and install PyCharm IDE
- Download and install Python interpreter
- Comments and scenario put up in the code wherever possible so that it can make it easier for someone new to understand what the function/code is doing (making it easier to understand the logic).
- Go through the sample code before Compiling it.
- One of the mistakes that I did was Compiling the code before actually figuring out the logic behind the code and how it should work.
- Speaking from experience, even though this is a bad habit of mine I strongly suggest people specially newbies to not remove comments even after you are finished with the program since it is a good practice to have it in the code plus it makes it easier for someone new to understand your code's functionality or a single module's functionality.

Based on the output of the program (shown below), while also going through the comments and scenarios described in the game, it was pretty much obvious what the expected output should be while also grasping the logic behind the output; for instance in the below output it was made clear that the program was going through each of the available actions while comparing with the resources utilization factor to opt the best available action to achieve the goal.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help gob_simple.py [C:\Users\cools\AppData\Local\Temp\gob_simple.py] - ...Users\cools\Downlo
C:\Users\cools\Downloads\Task 04 - gob_simple.py > gob_simple.py
gob_simple.py x
choose_action()
Run: gob_simple x
C:\Users\cools\PycharmProjects\untitled\venv\Scripts\python.exe "C:\Users\cools\Downloads\Task 04 - gob_simple.py/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! <<
Process finished with exit code 0
```