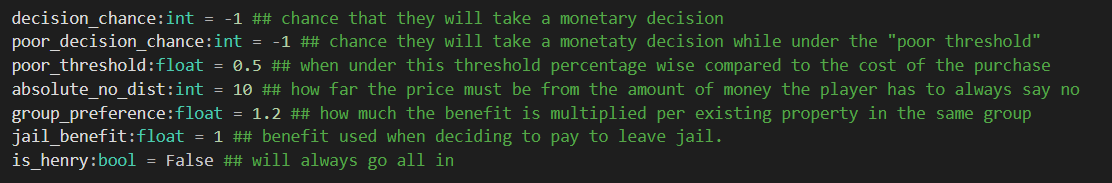
The player agent – Callum Hemsley

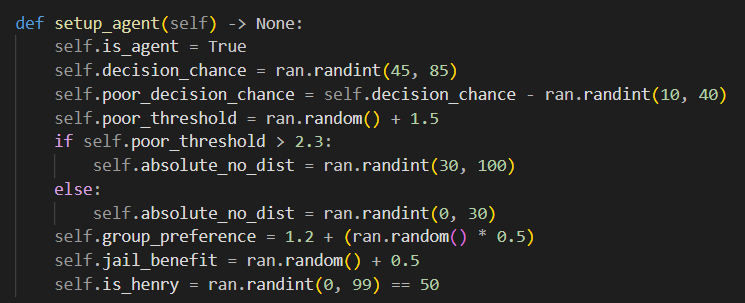
The player class itself is simple – it handles most of the behaviour for the player that is not explicitly part of the loop and should be handled there. These features handled in the class include moving, going to spaces, paying players, purchasing properties, and some parts of jail.

The Player agent is enabled by setting the player’s “is\_agent”. When active, the gameplay loop will handle decisions differently. When inactive, the respective windows for the user to interact with the game will show up, otherwise it uses built in methods as part of the player class.



An agent has multiple variables for it to work. Each of these variables will be a randomly generated value which determines its behaviour. The decision chance is simple – how likely it is to buy something. The poor decision chance is the same but is always a lower value/less likely and the poor threshold decides when to use this value.

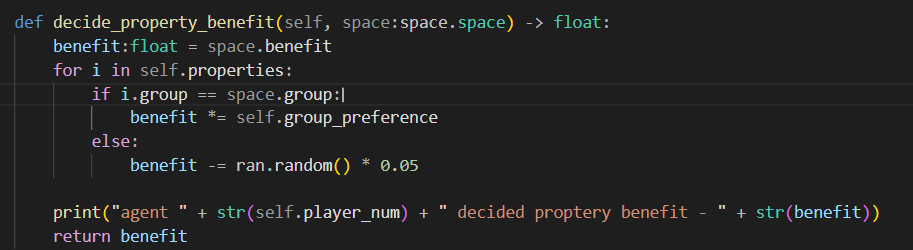
The “absolute no distance” will cause the agent to always say no when buying something. Group preference and jail benefit are both used when deciding a purchase’s benefit used for decisions later. “Is Henry” is a fun/joke value that causes the agent to always buy a property no matter what.



When starting a player agent, the method “setup agent” should be run to initialize all the variables shown above. First it sets the agent value to true, and then the chances are randomized. Decision chance is set to a value between 45 and 85 to make sure that the agent is never too likely or too unlikely to buy something.

To keep consistency throughout the agent, the poor decision chance is decided based off the decision chance, so that if the decision chance is high, the poor decision chance is likely to be a higher value too, and vice versa. The poor threshold is set to a value between 1.5 and 2.5, and this is also used to decide the absolutely no distance to again keep consistency. This means that if the poor threshold is higher, the more likely it is for the agent to decide that the purchase is too expensive and vice versa. These consistencies create agents that will hold a personality, like being likely to spend a lot of money but only when having enough money and being likely to spend.

Finally, the preferences are set to values between 1.2 to 1.7 or 0.5 to 1.5 respectively. Lastly, the is henry value is set to true 1 in 99.



The decide property benefit value is passed a space as an argument and returns a float value used for decisions later. It starts by taking the benefit saved in the space. This value is associated with the return expected from the space. For example, the cheaper spaces have a lower benefit value, while the later spaces are much more beneficial. It then will loop through each of the properties owned by the agent. If they own a space within the same group as the passed space, the benefit is increased by multiplying itself by the group preference defined previously. If its not in the same group, the benefit is randomly decreased by a minor amount. This decrease is to minorly deficit the agents to make sure agents that wish to buy every property have less of a chance to (ironically) get a monopoly too easily.

A screen shot of a computer program

AI-generated content may be incorrect.

Finally, the agent decision method is passed a benefit value and a cost and returns a Boolean value that when true means the agent has decided to purchase whatever this may be. The benefit value is defined in 3 ways. The first is using the decide property benefit method when asking if they wish to purchase a property. The second is using the jail benefit value when asking if they wish use 50 money to leave jail, and finally a constant value for any other cost that may be outside of this.

To begin with, if the cost is more expensive than the money minus the absolutely no distance, the agent will always return false no matter what. For example, if the AND (absolute no distance) is 20, the agent has 100 money, but the cost is above 80, although the agent can purchase this, it will always decide not to since they won’t have much money remaining. Next, is henry is checked and if true will return true.

After this, the true decision chance is calculated. First, it is set to the base decision chance, but if the cost multiplied by the poor threshold is above the amount of money the agent has, the poor decision chance is used instead. For example, say the poor threshold is 2, and the cost is 200, the the agent has over 400 money they will decide to use the base decision chance, otherwise the poor decision chance will be used. This then decreases the likelihood of buying something as the agent runs out of money.

Finally, the decision chance is multiplied by the benefit, so purchases with higher benefit are more likely to be bought and vice versa. This is then checked against a value between 0 and 100, and if the decision chance is greater than this value, true is returned.

Ideally, there are still some changes that could be used to make the agent more “human”, such as linearly decreasing the decision chance, making agents react to the highest rent value on the board, or add more awareness of what’s on the board in general, but this could overcomplicate the agent. As it stands, the agent has a set “personality”, so the depth it has works for a game like monopoly. However, if this was to be implemented, maybe adding a difficulty decided by the player could add more awareness.