Curtin University – Department of Computing

# Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

| | | | |
|---|---|---|---|
| Last name: | Mowatt-Gok | Student ID: | 20568818 |
| Other name(s): | Jackson | | |
| Unit name: | Fundamentals of Programming | Unit ID: | COMP1005/5005 |
| Lecturer / unit coordinator: | Dr Valerie Maxwell | Tutor: | Dr Valerie Maxwell |
| Date of submission: | 11/10/2021 | Which assignment? | (Leave blank if the unit has only one assignment.) |

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: Jackson Date of signature: 10/10/2021

*(By submitting this form, you indicate that you agree with all the above text.)*

# Overview (of your programs purpose and features)

The purpose of this assignment is to simulate different cats in an environment. These cats have independent behaviour, interactions and needs that they can vary themselves through a combination of reasoning and randomness. Each cat has varying ages, hunger, thirst, states and desires that are each affected by one another. Each cat's behaviour varies based on their hunger, thirst and type of cat (either good/passive or evil/aggressive) and can be changed by eating food, drinking water and interacting with mysterious boxes. Each cat's movement has a combination of reason and randomness. If a good cat is hungry and near food, it will eat. If an evil cat is near a good cat, the evil cat will eat the good cat. This also comes under interactions between varying types. As aforementioned, there are special zones including water, food, and mysterious boxes. Water increases thirst levels, food increases hunger levels, mysterious boxes transform good cats into new evil cats. This simulation also has the ability to track and display the log of events within the simulation happening including each cat's eating, drinking, dying, transforming and interacting. These stats also only appear for the current cat's that are alive. Visualisation is also implemented to more easily view age increase over time in the simulation. Each cat slowly fades on the graph as it ages overtime

# User Guide (on how to use your simulation)

The entire simulation runs off of two python files: cats.py and catsSimtest.py. 'cats.py' establishes the classes of both the good and evil cats. This includes creating their attributes on initialisation and their methods in terms of how they react in terms of food/hunger, water/thirst and aging. If you are wanting to access something fundamental about the cats, like how much hunger or thirst they have or how old they can age until, this file is the location to do so. 'catsSimtest.py' handles all the rest of the simulation and is the file needed to be run to start the simulation. This file includes basic things like establishing the number of steps, initial population, food and boxes via input, to more complex things like: establishing functions that allow you to scatter animate cats and inanimate features and live update them on a plot, establishing separate movement functions for good and evil cats that allows them to chase or avoid certain things depending on their environment and current statistics, establishing a randomly divided initial population and assigning the cats their initialised attributes (name, position, tag). If you are wanting to change something fundamental about the initialisation of the simulation, i.e. how it scatters the objects, how the cats are moved or how the population is created then the start of this code up until 'for t in range(int(steps)):' is the location to do so. Finally, within the 'for t in range(int(steps)):' loop in 'catsSimtest.py' is the step updates, establishing every live update of the simulation within the range of the steps input by the user. These step updates include any type of feature that could change, be tracked or affect the simulation at varying times. Things like printing the statistics of each cat in that step, printing the events that occurred in each step,

how the cats age, fade, interact with one another and with features, what time of ‘day’ the cats are moving, how their movement is live updated and finally, how the data is plotted in every step. If you are wanting to change anything to do with how each step changes, records and updates the simulation, this is the location to do so.

: printing the statistics of that step (the number of alive good cats, evil cats and total cats. The name, hunger, thirst and age of every alive cat), printing the events of every step (every time a cat eats, drinks, transforms or dies), aging within every step, fading as

# Traceability Matrix

| **Feature** | **Code** | **Test** |
|---|---|---|
| Cat object has name | Class GoodCat: self.name in __init__ function.<br>globals()['GoodCats%s' % i] = GoodCat(np.zeros((RMAX,CMAX), dtype="int16"), 'GoodCat%s' % i, 'N%s' % i) in ‘for i in range(int(initpop)):’ in def main():<br>for obj in listofgoodcats:<br>if (sum(obj.pos) > 0).any():<br>print("Name: ", obj.name, "\t Hunger: ", obj.hunger, "\t Thirst: ", obj.thirst, "\t Age: ", obj.age)<br><br>Class EvilCat: self.name in __init__ function.<br>globals()['EvilCats%s' % i] = EvilCat(np.zeros((RMAX,CMAX), dtype="int16"), 'EvilCat%s' % i, 'N%s' % i) in ‘for i in range(int(initpop)):’ in def main():<br>for x in listofevilcats:<br>if (sum(x.pos) > 0).any():<br>print("Name: ", x.name, "\t Hunger: ", | [PASSED] |

| | | |
|---|---|---|
| | x.hunger, "\t Thirst: ", x.thirst, "\t Age: ", x.age)<br><br>Self.name died in multiple methods and functions | |
| Cat object has position | Class GoodCat: self.pos in __init__ function<br>Self.pos[row,col] -= 1 in multiple methods and functions<br>Current.pos in move_em_good<br>Obj.pos[row,col] = 0<br>Etc, etc<br><br>Class EvilCat: self.pos in __init__ function<br>Self.pos[row,col] -= 1 in multiple methods and functions<br>x.pos[row,col] = 0<br>Etc, etc | [PASSED] |
| Cat object has age | Class GoodCat: self.age = random.randint(0,7) in __init__ function<br>Class GoodCat: self.age += 1. If self.age < etc etc in stepChange function<br><br>Class EvilCat: self.age = random.randint(0,7) in __init__ function<br>Class EvilCat: self.age += 1. If self.age < etc etc in stepChange function | [PASSED] |
| Cat object has state | If self.state == self.states[2] etc etc | [PASSED] |
| Cat object has thirst | Class GoodCat:<br>Self.thirst = initialgoodthirst<br>def drink: self.thirst -= 1<br>If self.thirst == 0:<br><br>Class EvilCat:<br>Self.thirst = initialevilthirst<br>def drink: self.thirst -= 1<br>If self.thirst == 0: | [PASSED] |

| | | |
|---|---|---|
| Cat object has hunger | Class GoodCat:<br>Self.hunger = initialgoodhunger<br>def eat: self.hunger -= 1<br>If self.hunger == 0:<br><br>Class EvilCat:<br>Self.hunger = initialevilhunger<br>def eat: self.hunger -= 1<br>If self.hunger == 0: | [PASSED] |
| Cat object has self identifiable tag/name | Class GoodCat: self.number = number<br>'N%s' % i<br><br>Class EvilCat: self.number = number<br>'N%s' % i | [PASSED] |
| Cat object can drink | Class GoodCat: def drink [obj.drink(obj.pos,water) for obj in listofgoodcats]<br><br>Class EvilCat: def drink [x.drink(x.pos,water) for x in listofevilcats] | [PASSED] |
| Cat object can eat | Class GoodCat: def eat [obj.eat(obj.pos,food) for obj in listofgoodcats]<br><br>Class EvilCat: def eat [x.eat(x.pos,food) for x in listofevilcats] | [PASSED] |
| Cat object can age | Class GoodCat: def stepChange<br>Class EvilCat: def stepChange | [PASSED] |
| Cat object can die | If self.state == self.states[2]:<br>Self.pos[row,col] -= 1 | [PASSED] |
| Interactive choice of number of steps, initial population, food and boxes | Steps = input(...)<br>Initpop = input(...)<br>Foodnum = input(...)<br>Boxnum = input(...) | [PASSED] |
| Scatter function for inanimate features | def make_feature_scatter | [PASSED] |

| | | |
|---|---|---|
| Scatter function for animate cats | def make_cat_scatter | [PASSED] |
| Function for controlling good cats random movement, avoiding evilcats and eating food if hungry and nearby | def move_em_good | [PARTIAL PASS]<br>Everything works apart from the good cats moving to food when hungry |
| Function for controlling evil cats random movement, and eating good cats if nearby | def move_em_evil | [PASSED] |
| Random division of initial population between good and evil cats | for i in range(int(initpop)):<br>toss = random choice([0,1])<br>If toss == 0: create a good cat<br>If toss == 1: create an evil cat | [PASSED] |
| Alpha of cats decreases as they age, essentially fading out as they die | for x in listofevilcats:<br>  alphacalc = 1/(x.maxage - x.initialage)<br>  if (x.alpha - alphacalc) > 0:<br>  x.alpha -= alphacalc<br><br>for obj in listofgoodcats:<br>  alphacalc = 1/(obj.maxage - obj.initialage)<br>  if (obj.alpha - alphacalc) > 0:<br>  obj.alpha -= alphacalc | [PASSED] |
| Evil cats eat good cats (chases them down as previously mentioned in move_em_evil) | for x in listofevilcats:<br>  for obj in listofgoodcats:<br>  for row in range(RMAX):<br>  for col in range(CMAX):<br>  if x.pos[row,col] and obj.pos[row,col] > 0:<br>  obj.pos[row,col] = 0<br>  x.hunger = initialevilhunger<br>  print(x.name, "ate", obj.name) | [PASSED] |
| Cats only move during certain times of the day | If 5 < t < 24 or 29 < t < 48:<br>Execute move_em functions | [PASSED] |
| Prints statistics of each cat alive. Stats include: name, hunger, thirst, age | For obj in listofgoodcats:<br>  If (sum(obj.pos) > 0).any():<br>Print stats<br><br>For x in listofevilcats:<br>  If (sum(x.pos) > 0).any(): | [PASSED] |

| | | |
|---|---|---|
| | Print stats | |
| Plot each cat as its identifying tag instead of a coloured dot | For obj in listofgoodcats:<br>  make_cat_scatter(....., r” {} “.format(obj.number)<br><br>For x in listofevilcats:<br>  make_cat_scatter(....., r” {} “.format(x.number) | [PASSED] |
| Terrain | Not implemented | N/A |
| Scent | Not implemented | N/A |

# Showcase:

## Introduction:

For the introduction, I will be splitting this into two sections. One for the class creation file ‘cats.py’, and one for the simulation creation file ‘catsSimtest.py’.

We will begin with the class creation file ‘cats.py’.
Within cats.py, there are two defined classes, ‘GoodCat’ and ‘EvilCat’. These represent a passive/friendly cat and an aggressive cat. Both are very similar in how they are defined except for a few defining traits, some of which are also addressed in ‘catsSimtest.py’. We will begin with GoodCat. GoodCat starts with the creation of its varying states: [“child”, “adult”, “dead”] which are addressed later on to represent how GoodCat ages and how it dies. Within def __init__, we have attributes given to a cat upon initialization of an object with this class. These include:

- self.name which is used to identify each cat within the log updates of every simulation step (will explain later in the report).
- Self.pos which is each cats individual position on the graph/simulation, in this assignment I decided to plot everything based off arrays and so each cats position is essentially an empty array of size (RMAX,CMAX) with one point plotted in the array, meant to represent the one cats position on the total graph/simulation space. These

individual points on the arrays are then each plotted, creating a total accumulation of cats position in the total simulation (these will be covered more later).

- Self.age which is assigned a random value from 0 to 7 for every single cat created via this class, this enables randomness and varied situations in every simulation. The age is then used by each cat later to eventually fade and die.
- Self.initialage is created to record each cats age on initialisation, this becomes useful later on within 'catsSimtest.py' to enable fading while aging.
- Self.state makes each cat a child on creation.
- Self.thirst and self.hunger both equal to initialgoodthirst and initialgoodhunger respectively, which are predefined outside of the class, this enables the values to be easily altered. Thirst and hunger are used later to incentive chasing food and water and to eventually kill of cats if they do not find food or water.
- Self.alpha is used to fade the cats as they age.
- Self.maxage represents the age at which the cats will die.
- Finally self.number represents each cats identifying tag which is used in plotting each cat on the graph. Each cat is plotted as a marker of their identifying tag instead of a circle, which allows each cat to be more easily identified when cross referencing with the log updates of every step.

def __str__ is a function/method inspired by the 'shrimp.py' file given to us in the assignment handout. This function allows the state and position of each cat to be more easily available.
def __drink__ is a function/method that controls the GoodCats entire thirst and interaction with water. Each iteration of the function reduces the cats thirst by 1 (self.thirst -= 1) and if its thirst equals to zero, it will remove the cat from the graph/array, essentially killing it via the line self.pos[row,col] -= 1, this will also print a statement saying that that specific cat has died. It also will increase the cats thirst back to its max (initialgoodthirst) if the cat is in water and if it has less than its max thirst, this will also print a statement saying that that specific cat has drank.
def __eat__ is identical to def __drink__ but for food/hunger instead, therefore also printing a statement that says that that specific cat has eaten when it eats food.
def __stepChange__ ages the cat with each iteration via self.age += 1 and if its age becomes greater than its maxage, the cat is "killed" in the same way it would be via loss of hunger or thirst (self.pos[row,col] -= 1). It also prints a statement saying that that specific cat has died.
The code for the EvilCat class in 'cats.py' is essentially identical to the GoodCat class apart from the changing of the initial thirst and hunger (initialevilthirst and initialevilhunger). The changes between the two cats becomes more apparent within 'catsSimtest.py' and the two classes are used to create two separate types of entities that can interact with one another and behave differently.

'catsSimtest.py' is significantly larger than 'cats.py' and holds all the code for running the simulation. It begins with importing all necessary files and establishes the RMAX and CMAX for the graph. 4 lines of input are then created to give the user easier interaction and variation within every simulation. The user can choose how many steps of the simulation, how large the

initial population is, how many food drops and how many mysterious box drops there are. The food and box drops are then randomly assigned positions across the simulation space.

def make_feature_scatter is a function created to be used when plotting inanimate features in the simulation, i.e. food, water and boxes. This function streamlines the plt.scatter process and reduces the amount of overall code. Every feature is also assigned a square marker within this function to more easily distinguish them from the objects. This function was inspired from the gremlin simulation code within prac test 3.

def make_cat_scatter is the function created to plot the animated objects (cats). It ensures to plot only the cats within the graph space and creates a streamlined scatter process for the cats. This function was inspired by the gremlin simulation code within prac test 3, however it has a few key differences. The size of the scattered plots is increased from 20 to 100. This size increase makes the identifying tags of each cat on the graph more visible to the viewer. Therefore, more code has to be added to the scatter feature: ‘marker=symbol’ where symbol is also a new defining parameter of make_cat_scatter, allowing each plot to be represented by its tag instead of a circle. Alpha was also added to this function to allow for each cat to fade over time as they age.

def move_em_good is the function that controls the entirety of the GoodCat’s movement. Some of the base code is inspired from the gremlin simulation code within prac test 3, however it has a lot built on top. move_em_good has a combination of reasoning and randomness built into every move. Moore neighbourhood is used to identify if food is near a good cat, and that if the cats hunger is less than half its initial hunger, it will move to the food. Unfortunately, whilst I was able to get Moore Neighbourhood working for the move_em_evil function, it does not appear to be working for the move_em_good function and I have run out of time to correct this mistake. If the good cat is not hungry and near food, then it will choose a random movement based on however many spaces it is allowed to move in accordance with the moves parameter. However , if that random choice is on the position of an evil cat, it will stay put and not move there, in order to not be eaten. However, with how I’ve ordered the code, an evil cat will actually move after a good cat and the evil cat is coded to chase the good cat if it's near so essentially the good cat can only survive in two scenarios if it is right next to an evil cat. Either the evil cat has two options to pick from, and it happens to be the lucky one or the good cat picks a random spot to move that isn't the evil cat's position and is also far away enough to be out of reach of the evil cat.

def move_em_evil is very similar to def move_em_good with a few differences. For one, the chase of the evil cat is always active, essentially meaning that if a good cat is close enough to an evil cat at any point (using Moore Neighbourhood), it will be eaten. move_em_evil also has no avoid list within the code.

def main() contains all the rest of the code necessary for the simulation. Moves and moves2 is created to assign allowed movement spaces for the good and evil cats (good cats can move

max one space at a time, evil cats can move two). Listofgoodcats, listofgoodcatspos, listofevilcats and listofevilcatspos are all created to store every type of cat and their individual positions, these will all be utilised later.
‘For i in range(int(initpop)):’ randomly divides the initial population into good cats and evil cats. Global variables are used here to name the cat variables. Whilst there is a lot of stigma against using global variable names, I believe here it is useful for my code as it better allows me to debug and test my code whilst going through the process of coding. Assigning each cat variable with a specific numbered name allowed the identification process of my debugging to be much smoother. I have seen the suggestion of using dictionaries instead of globals multiple times, however I believed that in this specific instance, global variable names were the better option for me and my code. Given the time constraints of this assignment, this is definitely something I could’ve misinterpreted and overlooked and would be happy to revisit at a later time, however it has worked well for me and for this assignment I will continue to use it. Within this loop, each cat is created with a global variable, a name and a tag that all relate to the number loop it was created in, e.g. (‘N%s’ % i) creates a tag/number that changes depending on what number loop it was created. This loop name creation was chosen to individually identify each cat within the simulation when cross referencing the log events/statistics with the graph. Each cat and its position is then added to the lists aforementioned (listofgoodcats, listofgoodcatspos, etc). listofgoodcatspostotal and listofevilcatspostotal are used to record all the cats positions at once in one array, this will be utilised later on.

‘for t in range(int(steps)):’ then controls everything that happens within each step of the simulation.
At the start of every step, the timestep number, number of alive good cats, number of alive evil cats and number of alive cats total is printed in the log updates. The name, hunger, thirst and age of every alive good and evil cat is then printed into the log statistics to keep track of each cat. This is using the ‘for _ in _list:’ format which is very helpful within my code.
The class method stepChange is then applied to every good and evil cat within the lists.
Using each objects initialised max age and initial age, an alphacalc is then created to see how much each specific cat’s alpha should be reduced each step to make them fade away until death, each cats alpha is then reduced by this calc each step, ensuring each cat fades away at the right pace as it ages because each cat has a different age.
The next chunks of code then determines the next movement of the cats and records their positions to be used later. The movement of the cats is only changed when the timestep is in accordance within certain hours of the ‘day’. The movement follows a simple formula: the next position equals the current position put into a move function, the current position is then updated to equal the next move, the current position is then plotted later on. For the good cats specifically, the position of each good cat is also then added to goodnext, which is summed into listofgoodcatspostotal, which is then used in the evil cats movement function after to enable the evil cats to move based on what the good cats most recent movement was, allowing them to be chased. The good cats chase food and avoid evil cats. The evil cats chase the good cats.

The next chunk of code determines what happens when a good cat and a box interact with one another. The interaction check is coded in the same way as the food and water interaction

checks. When they interact, a new evil cat is created with a name identifying it was created in this way, it is then given the position of the good cat, and the good cat is destroyed, effectively replacing the cat and giving the illusion of ‘transforming’. A statement is printed saying that that specific cat has turned evil.

The next chunk of code makes it so that if an evil and a good cat are on top of each other, the evil cat will ‘eat’ the good cat, effectively killing the good cat and raising the evil cat's hunger back to max. This is done by removing the good cat's position (obj.pos[row,col] = 0) and x.hunger = initialevilhunger. This works in tandem with the movement function of the evil cats as aforementioned where the evil cats will chase down and move on top of any surrounding good cats if possible. A statement is then printed identifying which evil cat has eaten which good cat

The next 4 lines of code execute the eat and drink function for both good and evil cats.

The next chunk of code creates the listofgoodcatspostotal and listofevilcatspostotal lists that record each class of cats positions to then be used at the top of the next loop in this sequence. (print(“### Number of good cats: “, (listofgoodcatspostotal).sum(), “###”)). As you may have noticed, this code is duplicated from earlier in the code where it was used to record the good cats positions right after movement for the evil cats to utilise. This code is intentionally used twice in the same loop. It is used originally for the good cats to record instantly what the good cats’ next move is so that the evil cats can immediately use it to chase/hunt them down on their next move. It is then used again later to track the new positions of every cat after they have: lost hunger/thirst, transformed, been eaten or anything that may affect their position so that on the start of the next step, an accurate counter can be given of the number of cats.

Finally, the last chunk of code simply plots everything so far. It scatters the water, food, boxes, each individual good and evil cat, the title, the labels and the pauses and clf needed to live update the graph in this specific way.

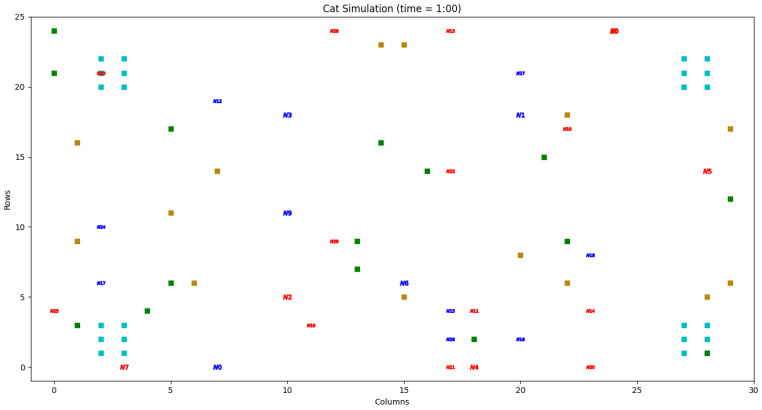
## Methodology

For my results I will be keeping initialgoodthirst, initialgoodhunger, initialevilthirst, initialevilhunger and both maxage the same throughout each simulation for the sake of not having too many varying values. These values can be seen within cats.py and have unchanged. For these results I will provide 3 examples. The difference between each simulation will be the difference in input values I choose for each simulation. The first simulation will have steps = 30, initpop = 30, food drops = 15, boxes = 15. The second simulation will have steps = 30, initpop = 60, food drops = 15, boxes = 15. The third simulation will have steps = 30, initpop = 30, food drops = 30, boxes = 30. These can all be input when running ‘catsSimtest.py’, there is no need to actually enter the code files or anything. That is all that is necessary. Of course, your output will be slightly different as the plotting of features and cats is randomized as well as the division of the initial population. I will be providing 6 images for each simulation, a screenshot near the start, middle and end of each simulation as well as a corresponding log update for each.

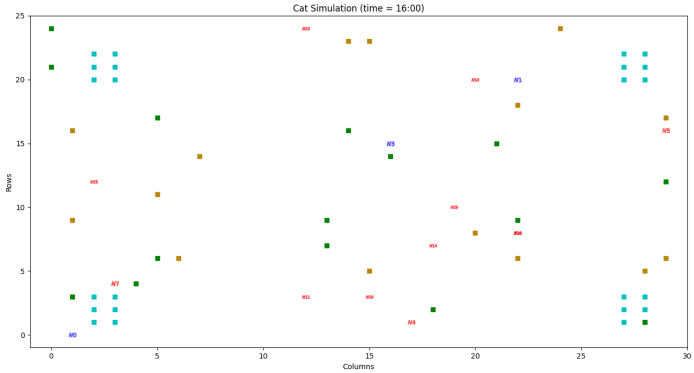
# Results

You will have to zoom in to see the finer details (e.g. individual tags).The size is reduced to reduce page number on report

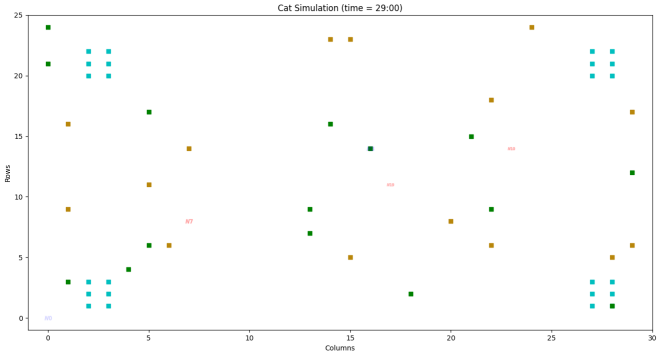
Sim 1:




```
### Timestep  1 ###
### Number of good cats:  13 ###
### Number of evil cats:  17 ###
### Number of total cats:  30 ###
Name:  GoodCat0          Hunger:  34     Thirst:  49     Age:  8
Name:  GoodCat1          Hunger:  34     Thirst:  49     Age:  7
Name:  GoodCat3          Hunger:  34     Thirst:  49     Age:  5
Name:  GoodCat6          Hunger:  34     Thirst:  49     Age:  1
Name:  GoodCat9          Hunger:  34     Thirst:  49     Age:  2
Name:  GoodCat12         Hunger:  34     Thirst:  49     Age:  7
Name:  GoodCat15         Hunger:  34     Thirst:  49     Age:  7
Name:  GoodCat16         Hunger:  34     Thirst:  49     Age:  1
Name:  GoodCat17         Hunger:  34     Thirst:  49     Age:  1
Name:  GoodCat18         Hunger:  34     Thirst:  49     Age:  7
Name:  GoodCat24         Hunger:  34     Thirst:  49     Age:  7
Name:  GoodCat26         Hunger:  34     Thirst:  49     Age:  3
Name:  GoodCat27         Hunger:  34     Thirst:  49     Age:  6
Name:  EvilCat2          Hunger:  14     Thirst:  39     Age:  5
Name:  EvilCat4          Hunger:  14     Thirst:  39     Age:  8
Name:  EvilCat5          Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat7          Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat10         Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat11         Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat13         Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat14         Hunger:  14     Thirst:  39     Age:  4
Name:  EvilCat19         Hunger:  14     Thirst:  39     Age:  4
Name:  EvilCat20         Hunger:  14     Thirst:  39     Age:  8
Name:  EvilCat21         Hunger:  14     Thirst:  39     Age:  7
Name:  EvilCat22         Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat23         Hunger:  14     Thirst:  40     Age:  4
Name:  EvilCat25         Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat28         Hunger:  14     Thirst:  39     Age:  8
Name:  EvilCat29         Hunger:  14     Thirst:  39     Age:  7
Name:  NewEvilCat born at timestep: 0    Hunger:  14     Thirst:  39     Age:  5
EvilCat23 drank!
```

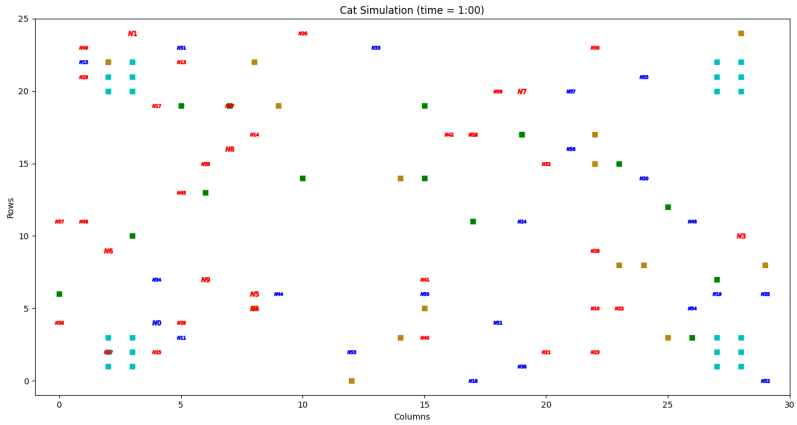


```
### Timestep  16 ###
### Number of good cats:  3 ###
### Number of evil cats:  11 ###
### Number of total cats:  14 ###
Name:  GoodCat0          Hunger:  19     Thirst:  50     Age:  23
Name:  GoodCat1          Hunger:  19     Thirst:  34     Age:  22
Name:  GoodCat3          Hunger:  19     Thirst:  34     Age:  20
Name:  EvilCat4          Hunger:  13     Thirst:  24     Age:  23
Name:  EvilCat5          Hunger:  14     Thirst:  24     Age:  21
Name:  EvilCat7          Hunger:  13     Thirst:  34     Age:  17
Name:  EvilCat10         Hunger:  9      Thirst:  24     Age:  18
Name:  EvilCat11         Hunger:  6      Thirst:  24     Age:  18
Name:  EvilCat14         Hunger:  9      Thirst:  24     Age:  19
Name:  EvilCat19         Hunger:  8      Thirst:  24     Age:  19
Name:  EvilCat23         Hunger:  10     Thirst:  31     Age:  19
Name:  EvilCat25         Hunger:  10     Thirst:  24     Age:  21
Name:  EvilCat29         Hunger:  11     Thirst:  24     Age:  22
Name:  NewEvilCat born at timestep: 10   Hunger:  9      Thirst:  34     Age:  12
```

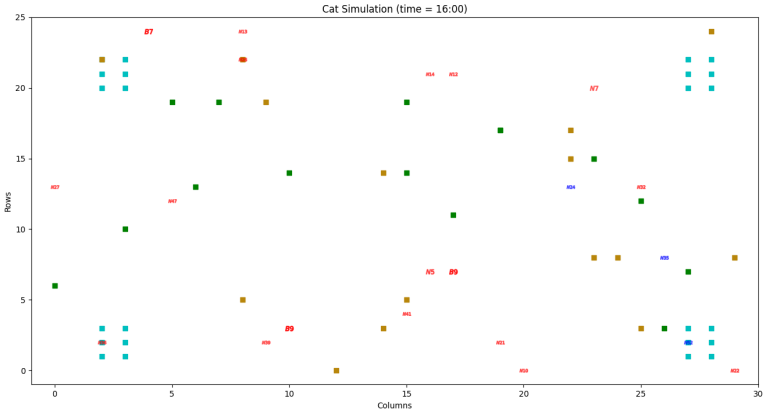



```
### Timestep  29 ###
### Number of good cats:  2 ###
### Number of evil cats:  4 ###
### Number of total cats:  6 ###
Name:  GoodCat0          Hunger:  6      Thirst:  37     Age:  36
Name:  GoodCat3          Hunger:  35     Thirst:  21     Age:  33
Name:  EvilCat5          Hunger:  1      Thirst:  11     Age:  34
Name:  EvilCat7          Hunger:  9      Thirst:  21     Age:  30
Name:  EvilCat10         Hunger:  4      Thirst:  11     Age:  31
Name:  EvilCat19         Hunger:  7      Thirst:  11     Age:  32
GoodCat3 ate!
EvilCat5 died!
```

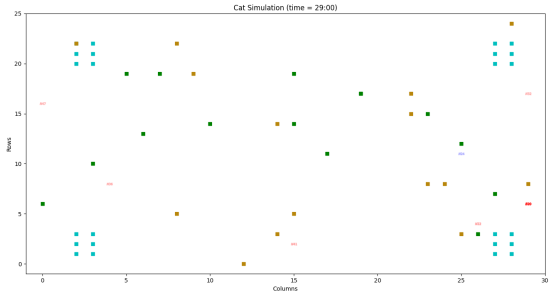
Sim 2:



```
### Timestep  1 ###
### Number of good cats:  22 ###
### Number of evil cats:  36 ###
### Number of total cats:  58 ###
Name:  GoodCat0          Hunger:  34     Thirst:  40     Age:  1
Name:  GoodCat11         Hunger:  34     Thirst:  40     Age:  7
Name:  GoodCat15         Hunger:  34     Thirst:  40     Age:  8
Name:  GoodCat18         Hunger:  34     Thirst:  40     Age:  5
Name:  GoodCat19         Hunger:  34     Thirst:  40     Age:  3
Name:  GoodCat20         Hunger:  34     Thirst:  40     Age:  3
Name:  GoodCat24         Hunger:  34     Thirst:  40     Age:  3
Name:  GoodCat31         Hunger:  34     Thirst:  40     Age:  2
Name:  GoodCat33         Hunger:  34     Thirst:  40     Age:  4
Name:  GoodCat34         Hunger:  34     Thirst:  40     Age:  4
Name:  GoodCat35         Hunger:  34     Thirst:  40     Age:  1
Name:  GoodCat38         Hunger:  34     Thirst:  40     Age:  2
Name:  GoodCat44         Hunger:  34     Thirst:  40     Age:  4
Name:  GoodCat46         Hunger:  34     Thirst:  40     Age:  1
Name:  GoodCat50         Hunger:  34     Thirst:  40     Age:  4
Name:  GoodCat51         Hunger:  34     Thirst:  40     Age:  6
Name:  GoodCat52         Hunger:  34     Thirst:  40     Age:  5
Name:  GoodCat53         Hunger:  34     Thirst:  40     Age:  4
Name:  GoodCat54         Hunger:  34     Thirst:  40     Age:  4
Name:  GoodCat55         Hunger:  34     Thirst:  40     Age:  7
Name:  GoodCat56         Hunger:  34     Thirst:  40     Age:  4
Name:  GoodCat57         Hunger:  34     Thirst:  40     Age:  3
Name:  EvilCat1          Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat3          Hunger:  14     Thirst:  39     Age:  1
Name:  EvilCat4          Hunger:  14     Thirst:  39     Age:  1
Name:  EvilCat5          Hunger:  14     Thirst:  39     Age:  1
Name:  EvilCat6          Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat7          Hunger:  14     Thirst:  39     Age:  8
Name:  EvilCat8          Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat9          Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat10         Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat12         Hunger:  14     Thirst:  39     Age:  4
Name:  EvilCat13         Hunger:  14     Thirst:  39     Age:  4
Name:  EvilCat14         Hunger:  14     Thirst:  39     Age:  4
Name:  EvilCat16         Hunger:  14     Thirst:  39     Age:  7
Name:  EvilCat17         Hunger:  14     Thirst:  39     Age:  7
Name:  EvilCat21         Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat22         Hunger:  14     Thirst:  39     Age:  1
Name:  EvilCat23         Hunger:  14     Thirst:  39     Age:  1
Name:  EvilCat25         Hunger:  14     Thirst:  39     Age:  1
Name:  EvilCat26         Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat27         Hunger:  15     Thirst:  39     Age:  4
Name:  EvilCat28         Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat29         Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat30         Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat32         Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat36         Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat37         Hunger:  14     Thirst:  39     Age:  8
Name:  EvilCat39         Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat40         Hunger:  14     Thirst:  39     Age:  5
Name:  EvilCat41         Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat42         Hunger:  14     Thirst:  39     Age:  5
Name:  EvilCat45         Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat47         Hunger:  14     Thirst:  40     Age:  3
Name:  EvilCat48         Hunger:  14     Thirst:  39     Age:  7
Name:  EvilCat49         Hunger:  14     Thirst:  39     Age:  4
Name:  EvilCat58         Hunger:  14     Thirst:  39     Age:  7
Name:  EvilCat59         Hunger:  14     Thirst:  39     Age:  1
EvilCat47 drank!
EvllCat27 ate!
```

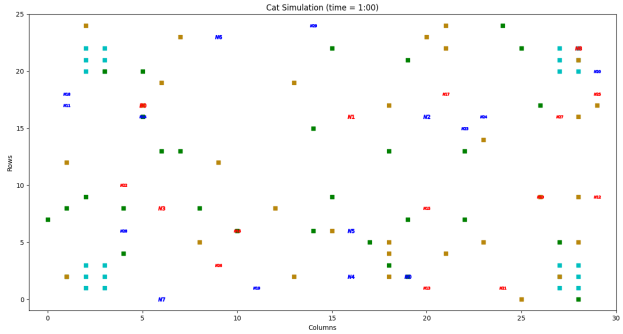



```
### Timestep  16 ###
### Number of good cats:  4 ###
### Number of evil cats:  18 ###
### Number of total cats:  22 ###
Name:  GoodCat24         Hunger:  19     Thirst:  34     Age:  18
Name:  GoodCat35         Hunger:  35     Thirst:  34     Age:  16
Name:  GoodCat38         Hunger:  19     Thirst:  34     Age:  17
Name:  GoodCat52         Hunger:  19     Thirst:  50     Age:  20
Name:  EvilCat5          Hunger:  5      Thirst:  24     Age:  16
Name:  EvilCat7          Hunger:  10     Thirst:  24     Age:  23
Name:  EvilCat10         Hunger:  6      Thirst:  24     Age:  17
Name:  EvilCat12         Hunger:  10     Thirst:  24     Age:  19
Name:  EvilCat13         Hunger:  5      Thirst:  24     Age:  19
Name:  EvilCat14         Hunger:  7      Thirst:  24     Age:  19
Name:  EvilCat21         Hunger:  8      Thirst:  24     Age:  17
Name:  EvilCat22         Hunger:  13     Thirst:  24     Age:  16
Name:  EvilCat26         Hunger:  7      Thirst:  24     Age:  17
Name:  EvilCat27         Hunger:  8      Thirst:  24     Age:  19
Name:  EvilCat32         Hunger:  15     Thirst:  24     Age:  18
Name:  EvilCat36         Hunger:  11     Thirst:  37     Age:  18
Name:  EvilCat39         Hunger:  5      Thirst:  24     Age:  18
Name:  EvilCat41         Hunger:  5      Thirst:  24     Age:  17
Name:  EvilCat47         Hunger:  15     Thirst:  30     Age:  18
Name:  NewEvilCat born at timestep: 7    Hunger:  6      Thirst:  32     Age:  12
Name:  NewEvilCat born at timestep: 9    Hunger:  8      Thirst:  33     Age:  9
Name:  NewEvilCat born at timestep: 9    Hunger:  8      Thirst:  33     Age:  12
EvilCat41 ate GoodCat38
GoodCat52 drank!
EvllCat36 drank!
```

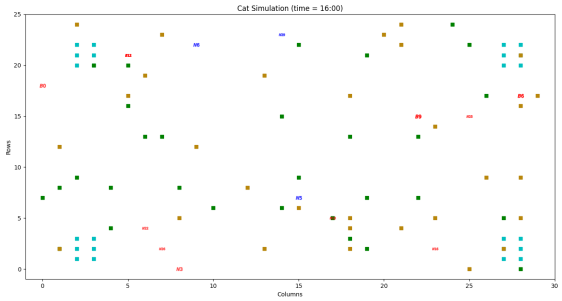



```
### Timestep  29 ###
### Number of good cats:  1 ###
### Number of evil cats:  6 ###
### Number of total cats:  7 ###
Name:  GoodCat24          Hunger:  29     Thirst:  21     Age:  31
Name:  EvilCat22          Hunger:  9      Thirst:  11     Age:  29
Name:  EvilCat32          Hunger:  2      Thirst:  11     Age:  31
Name:  EvilCat36          Hunger:  8      Thirst:  28     Age:  31
Name:  EvilCat41          Hunger:  2      Thirst:  11     Age:  30
Name:  EvilCat47          Hunger:  4      Thirst:  17     Age:  31
Name:  NewEvilCat born at timestep: 20    Hunger:  6      Thirst:  31     Age:  12
```

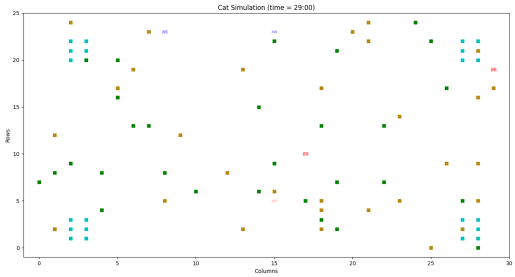
## Sim 3:




```
### Timestep  1 ###
### Number of good cats:  15 ###
### Number of evil cats:  15 ###
### Number of total cats:  30 ###
Name:  GoodCat0           Hunger:  35     Thirst:  49     Age:  4
Name:  GoodCat2           Hunger:  34     Thirst:  49     Age:  4
Name:  GoodCat4           Hunger:  34     Thirst:  49     Age:  3
Name:  GoodCat5           Hunger:  34     Thirst:  49     Age:  1
Name:  GoodCat6           Hunger:  34     Thirst:  49     Age:  1
Name:  GoodCat7           Hunger:  34     Thirst:  49     Age:  2
Name:  GoodCat11          Hunger:  34     Thirst:  49     Age:  6
Name:  GoodCat14          Hunger:  35     Thirst:  49     Age:  7
Name:  GoodCat18          Hunger:  34     Thirst:  49     Age:  8
Name:  GoodCat19          Hunger:  34     Thirst:  49     Age:  5
Name:  GoodCat20          Hunger:  34     Thirst:  49     Age:  6
Name:  GoodCat23          Hunger:  34     Thirst:  49     Age:  6
Name:  GoodCat24          Hunger:  34     Thirst:  49     Age:  4
Name:  GoodCat28          Hunger:  34     Thirst:  49     Age:  7
Name:  GoodCat29          Hunger:  34     Thirst:  49     Age:  2
Name:  EvilCat1           Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat3           Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat8           Hunger:  14     Thirst:  40     Age:  3
Name:  EvilCat10          Hunger:  14     Thirst:  39     Age:  1
Name:  EvilCat12          Hunger:  14     Thirst:  39     Age:  2
Name:  EvilCat13          Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat15          Hunger:  14     Thirst:  39     Age:  8
Name:  EvilCat16          Hunger:  15     Thirst:  39     Age:  5
Name:  EvilCat17          Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat21          Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat22          Hunger:  14     Thirst:  39     Age:  6
Name:  EvilCat25          Hunger:  14     Thirst:  39     Age:  3
Name:  EvilCat26          Hunger:  14     Thirst:  39     Age:  7
Name:  EvilCat27          Hunger:  14     Thirst:  39     Age:  1
Name:  NewEvilCat born at timestep: 0     Hunger:  14     Thirst:  39     Age:  5
EvilCat8 drank!
GoodCat0 ate!
GoodCat14 ate!
EvilCat16 ate!
```




```
### Timestep  16 ###
### Number of good cats:  3 ###
### Number of evil cats:  10 ###
### Number of total cats:  13 ###
Name:  GoodCat5          Hunger:  27     Thirst:  34     Age:  16
Name:  GoodCat6          Hunger:  19     Thirst:  34     Age:  16
Name:  GoodCat29         Hunger:  19     Thirst:  34     Age:  17
Name:  EvilCat3          Hunger:  6      Thirst:  37     Age:  21
Name:  EvilCat13         Hunger:  13     Thirst:  24     Age:  21
Name:  EvilCat16         Hunger:  9      Thirst:  24     Age:  20
Name:  EvilCat22         Hunger:  9      Thirst:  24     Age:  21
Name:  EvilCat25         Hunger:  11     Thirst:  33     Age:  18
Name:  EvilCat26         Hunger:  9      Thirst:  24     Age:  22
Name:  NewEvilCat born at timestep: 0    Hunger:  13     Thirst:  38     Age:  20
Name:  NewEvilCat born at timestep: 6    Hunger:  14     Thirst:  31     Age:  9
Name:  NewEvilCat born at timestep: 9    Hunger:  12     Thirst:  33     Age:  13
Name:  NewEvilCat born at timestep: 12   Hunger:  11     Thirst:  36     Age:  3
EvilCat13 ate!
```




```
### Timestep  29 ###
### Number of good cats:  2 ###
### Number of evil cats:  3 ###
### Number of total cats:  5 ###
Name:  GoodCat6          Hunger:  6      Thirst:  21     Age:  29
Name:  GoodCat29         Hunger:  6      Thirst:  21     Age:  30
Name:  EvilCat13         Hunger:  8      Thirst:  11     Age:  34
Name:  NewEvilCat born at timestep: 6    Hunger:  7      Thirst:  18     Age:  22
Name:  NewEvilCat born at timestep: 9    Hunger:  8      Thirst:  20     Age:  26
```

## Conclusion and Future Work

In conclusion, whilst there is much more I would like to add/improve on my code, I am happy with the progress so far. Through this assignment I've been able to learn a lot through trial and error, finding my own solutions to problems and using a lot of stackoverflow forums. Further investigations I would've liked to add would be terrain, scent, fixing of the move em good function and more user interaction with the simulation

References:

https://stackoverflow.com/questions/6181935/how-do-you-create-different-variable-names-while-in-a-loop
Second answer shows global variable use

https://stackoverflow.com/questions/2682012/how-to-call-same-method-for-a-list-of-objects
Second answer shows how to apply method and function to a list of objects

https://stackoverflow.com/questions/26830697/moore-neighbourhood-in-python
Moore neighbourhood

https://stackoverflow.com/questions/22175489/numpy-valueerror-the-truth-value-of-an-array-with-more-than-one-element-is-ambi
https://stackoverflow.com/questions/54191677/numpy-error-the-truth-value-of-an-array-with-more-than-one-element-is-ambiguous
.any()

https://www.quora.com/What-does-t-do-in-Python
Print functions

https://matplotlib.org/stable/api/markers_api.html
https://stackoverflow.com/questions/37004356/matplotlib-using-text-instead-of-marker-but-cant-be-the-best-way-to-do-it
Rendering string using text

Gremlin sim from prac test 3 used for scattering functions and movement function
The four python scripts given in the assignment