Romael Simmonds

Homework 4

Problem 1

```
'''''Romael Simmonds
    Z23204457'''

import random
import time
import pylab

class Island (object):
    """Island
       n X n grid where zero value indicates not occupied."""
    def __init__(self, n, prey_count=0, predator_count=0, human_count=0):
        '''''Initialize grid to all 0's, then fill with animals
        '''
        print("Grid Size:",n,"x",n,"Prey:",prey_count,"Predators:",predator_count,"Humans:",human_count)
        self.grid_size = n
        self.grid = []
        for i in range(n):
            row = [0]*n    # row is a list of n zeros
            self.grid.append(row)
        self.init_animals(prey_count,predator_count, human_count)

    def init_animals(self,prey_count, predator_count, human_count):
        ''''' Put some initial animals on the island
        '''
        count = 0
        # while loop continues until prey_count unoccupied positions are found
        while count < prey_count:
            x = random.randint(0,self.grid_size-1)
            y = random.randint(0,self.grid_size-1)
            if not self.animal(x,y):
                new_prey=Prey(island=self,x=x,y=y)
                count += 1
                self.register(new_prey)
        count = 0
        # same while loop but for predator_count
        while count < predator_count:
            x = random.randint(0,self.grid_size-1)
            y = random.randint(0,self.grid_size-1)
            if not self.animal(x,y):
                new_predator=Predator(island=self,x=x,y=y)
                count += 1
                self.register(new_predator)

        count = 0
        # same while loop but for predator_count
        while count < human_count:
            x = random.randint(0,self.grid_size-1)
            y = random.randint(0,self.grid_size-1)
            if not self.animal(x,y):
                new_human=Human(island=self,x=x,y=y)
```

```
                count += 1
                self.register(new_human)

    def clear_all_moved_flags(self):
        ''''' Animals have a moved flag to indicated they moved this turn.
        Clear that so we can do the next turn
        '''
        for x in range(self.grid_size):
            for y in range(self.grid_size):
                if self.grid[x][y]:
                    self.grid[x][y].clear_moved_flag()

    def size(self):
        '''''Return size of the island: one dimension.
        '''
        return self.grid_size

    def register(self,animal):
        '''''Register animal with island, i.e. put it at the
        animal's coordinates
        '''
        x = animal.x
        y = animal.y
        self.grid[x][y] = animal

    def remove(self,animal):
        '''''Remove animal from island.'''
        x = animal.x
        y = animal.y
        self.grid[x][y] = 0

    def animal(self,x,y):
        '''''Return animal at location (x,y)'''
        if 0 <= x < self.grid_size and 0 <= y < self.grid_size:
            return self.grid[x][y]
        else:
            return -1  # outside island boundary

    def __str__(self):
        '''''String representation for printing.
           (0,0) will be in the lower left corner.
        '''
        s = ""
        for j in range(self.grid_size-1,-1,-1):  # print row size-1 first
            for i in range(self.grid_size):      # each row starts at 0
                if not self.grid[i][j]:
                    # print a '.' for an empty space
                    s+= "{:<2s}".format('.' + "  ")
                else:
                            s+= "{:<2s}".format((str(self.grid[i][j])) + "  ")
                   s+="\n"
               return s

           def count_prey(self):
               ''''' count all the prey on the island'''
               count = 0
               for x in range(self.grid_size):
                   for y in range(self.grid_size):
                       animal = self.animal(x,y)
                       if animal:
                           if isinstance(animal,Prey):
```

```
                                count+=1
                return count

            def count_predators(self):
                ''''' count all the predators on the island'''
                count = 0
                for x in range(self.grid_size):
                    for y in range(self.grid_size):
                        animal = self.animal(x,y)
                        if animal:
                            if isinstance(animal,Predator):
                                count+=1
                return count

            def count_humans(self):
                ''''' count all the humans on the island'''
                count = 0
                for x in range(self.grid_size):
                    for y in range(self.grid_size):
                        animal = self.animal(x,y)
                        if animal:
                            if isinstance(animal, Human):
                                count+=1
                return count

        class Animal(object):
            def __init__(self, island, x=0, y=0, s="A"):
                '''''Initialize the animal's and their positions
                '''
                self.island = island
                self.name = s
                self.x = x
                self.y = y
                self.moved=False

            def position(self):
                '''''Return coordinates of current position.
                '''
                return self.x, self.y

            def __str__(self):
                return self.name

            def check_grid(self,type_looking_for=int):
                ''''' Look in the 8 directions from the animal's location
                and return the first location that presently has an object
                of the specified type. Return 0 if no such location exists
                '''
                # neighbor offsets
                offset = [(-1,1),(0,1),(1,1),(-1,0),(1,0),(-1,-1),(0,-1),(1,-1)]
                result = 0
                for i in range(len(offset)):
                    x = self.x + offset[i][0]  # neighboring coordinates
                    y = self.y + offset[i][1]
                    if not 0 <= x < self.island.size() or \
                       not 0 <= y < self.island.size():
                        continue
                    if type(self.island.animal(x,y))==type_looking_for:
                        result=(x,y)
                        break
                return result
```

```
173.
174.             def move(self):
175.                 '''''Move to an open, neighboring position '''
176.                 if not self.moved:
177.                     location = self.check_grid(int)
178.                     if location:
179.                         # print('Move, {}, from {},{} to {},{}'.format( \
180.                         #       type(self),self.x,self.y,location[0],location[1]))
181.                         self.island.remove(self)   # remove from current spot
182.                         self.x = location[0]       # new coordinates
183.                         self.y = location[1]
184.                         self.island.register(self) # register new coordinates
185.                         self.moved=True
186.             def breed(self):
187.                 ''''' Breed a new Animal.If there is room in one of the 8 locations
188.                 place the new Prey there. Otherwise you have to wait.
189.                 '''
190.                 if self.breed_clock <= 0:
191.                     location = self.check_grid(int)
192.                     if location:
193.                         self.breed_clock = self.breed_time
194.                         # print('Breeding Prey {},{}'.format(self.x,self.y))
195.                         the_class = self.__class__
196.                         new_animal = the_class(self.island,x=location[0],y=location[1])

197.                         self.island.register(new_animal)
198.
199.             def clear_moved_flag(self):
200.                 self.moved=False
201.
202.         class Prey(Animal):
203.             def __init__(self, island, x=0,y=0,s="O"):
204.                 Animal.__init__(self,island,x,y,s)
205.                 self.breed_clock = self.breed_time
206.                 # print('Init Prey {},{}, breed:{}'.format(self.x, self.y,self.breed_clo
    ck))
207.
208.             def clock_tick(self):
209.                 '''''Prey only updates its local breed clock
210.                 '''
211.                 self.breed_clock -= 1
212.                 # print('Tick Prey {},{}, breed:{}'.format(self.x,self.y,self.breed_cloc
    k))
213.
214.         class Predator(Animal):
215.             def __init__(self, island, x=0,y=0,s="X"):
216.                 Animal.__init__(self,island,x,y,s)
217.                 self.starve_clock = self.starve_time
218.                 self.breed_clock = self.breed_time
219.                 # print('Init Predator {},{}, starve:{}, breed:{}'.format( \
220.                 #       self.x,self.y,self.starve_clock,self.breed_clock))
221.
222.             def clock_tick(self):
223.                 ''''' Predator updates both breeding and starving
224.                 '''
225.                 self.breed_clock -= 1
226.                 self.starve_clock -= 1
227.                 # print('Tick, Predator at {},{} starve:{}, breed:{}'.format( \
228.                 #       self.x,self.y,self.starve_clock,self.breed_clock))
229.                 if self.starve_clock <= 0:
230.                     # print('Death, Predator at {},{}'.format(self.x,self.y))
```

```
231.                 self.island.remove(self)
232.
233.         def eat(self):
234.             ''''' Predator looks for one of the 8 locations with Prey. If found
235.             moves to that location, updates the starve clock, removes the Prey
236.             '''
237.             if not self.moved:
238.                 location = self.check_grid(Prey)
239.                 if location:
240.                     # print('Eating: pred at {},{}, prey at {},{}'.format( \
241.                     #       self.x,self.y,location[0],location[1]))
242.                     self.island.remove(self.island.animal(location[0],location[1]))
243.                     self.island.remove(self)
244.                     self.x=location[0]
245.                     self.y=location[1]
246.                     self.island.register(self)
247.                     self.starve_clock=self.starve_time
248.                     self.moved=True
249.
250.     class Human(Animal):
251.         def __init__(self, island, x=0, y=0, s="H"):
252.             Animal.__init__(self, island, x, y, s)
253.             self.starve_clock = self.starve_time
254.             self.breed_clock = self.breed_time
255.             self.hunt_clock=self.hunt_time
256.
257.         def clock_tick(self):
258.             self.breed_clock -= 1
259.             self.starve_clock -= 1
260.             self.hunt_clock -= 1
261.
262.             if self.starve_clock <= 0:
263.                 self.island.remove(self)
264.
265.         def eat(self):
266.             ''''' Human looks for one of the 8 locations with Prey. If found
267.             moves to that location, updates the starve clock, removes the Prey
268.             '''
269.             if not self.moved:
270.                 location = self.check_grid(Prey)
271.                 if location:
272.                     # print('Eating: human at {},{}, prey at {},{}'.format( \
273.                     #       self.x,self.y,location[0],location[1]))
274.                     self.island.remove(self.island.animal(location[0],location[1]))
275.                     self.island.remove(self)
276.                     self.x=location[0]
277.                     self.y=location[1]
278.                     self.island.register(self)
279.                     self.starve_clock=self.starve_time
280.                     self.moved=True
281.
282.
283.         def hunt(self):
284.             ''''' Human looks for one of the 8 locations with Predators. If found
285.             moves to that location, updates the hunt clock, removes the Prey
286.             '''
287.             if not self.moved:
288.                 location = self.check_grid(Predator)
289.                 if location and self.hunt_clock == 0:
```

```
290.                         # print('Hunting: Human at {},{}, Predator at {},{}'.format( \
291.                         #       self.x,self.y,location[0],location[1]))
292.                         self.island.remove(self.island.animal(location[0],location[1]))
293.                         self.island.remove(self)
294.                         self.x=location[0]
295.                         self.y=location[1]
296.                         self.island.register(self)
297.                         self.hunt_clock=self.hunt_time
298.                         self.moved=True
299.
300.
301.         ########################################
302.         def main(predator_breed_time=6, predator_starve_time=3, human_breed_time=7, huma
    n_starve_time=6,human_hunt_time=8, initial_humans=5, initial_predators=10, prey_breed_t
    ime=4, initial_prey=50, \
303.                  size=10, ticks=300):
304.             ''''' main simulation. Sets defaults, runs event loop, plots at the end
305.             '''
306.             # initialization values
307.             Predator.breed_time = predator_breed_time
308.             Predator.starve_time = predator_starve_time
309.             Human.breed_time = human_breed_time
310.             Human.starve_time = human_starve_time
311.             Human.hunt_time = human_hunt_time
312.             Prey.breed_time = prey_breed_time
313.
314.             # for graphing
315.             predator_list=[]
316.             prey_list=[]
317.             human_list=[]
318.
319.             # make an island
320.             isle = Island(size,initial_prey, initial_predators, initial_humans)
321.             print(isle)
322.
323.             # event loop.
324.             # For all the ticks, for every x,y location.
325.             # If there is an animal there, try eat, move, breed and clock_tick
326.             print("Prey|Predators|Humans")
327.             for i in range(ticks):
328.                 # important to clear all the moved flags!
329.                 isle.clear_all_moved_flags()
330.                 for x in range(size):
331.                     for y in range(size):
332.                         animal = isle.animal(x,y)
333.                         if animal:
334.                             if isinstance(animal,Predator):
335.                                 animal.eat()
336.                             if isinstance(animal,Human):
337.                                 animal.hunt()
338.                                 animal.eat()
339.
340.                             animal.move()
341.                             animal.breed()
342.                             animal.clock_tick()
343.
344.                 # record info for display, plotting
345.                 prey_count = isle.count_prey()
346.                 predator_count = isle.count_predators()
347.                 human_count = isle.count_humans()
```

```
348.
349.                 if prey_count == 0:
350.                     print('Lost the Prey population. Quiting.')
351.                     break
352.                 if predator_count == 0:
353.                     print('Lost the Predator population.')
354.                     #break
355.                 if human_count == 0:
356.                     #break
357.                     print("Lost the Human population.")
358.
359.                 prey_list.append(prey_count)
360.                 predator_list.append(predator_count)
361.                 human_list.append(human_count)
362.
363.                 # print out every 10th cycle, see what's going on
364.                 if not i%10:
365.
366.                     print(prey_count,"     ",predator_count,"     ",human_count)
367.                 # print the island, hold at the end of each cycle to get a look
368.         #         print('*'*20)
369.         #         print(isle)
370.         #         ans = input("Return to continue")
371.
372.             print("\n",isle)
373.
374.             pylab.plot(range(i), predator_list, label="Predators")
375.             pylab.plot(range(i), prey_list, label="Prey")
376.             pylab.plot(range(i), human_list, label="Humans")
377.             pylab.legend(loc="best", shadow=True)
378.             pylab.show()
379.
380.         if (__name__ == "__main__"):
381.             main()
```

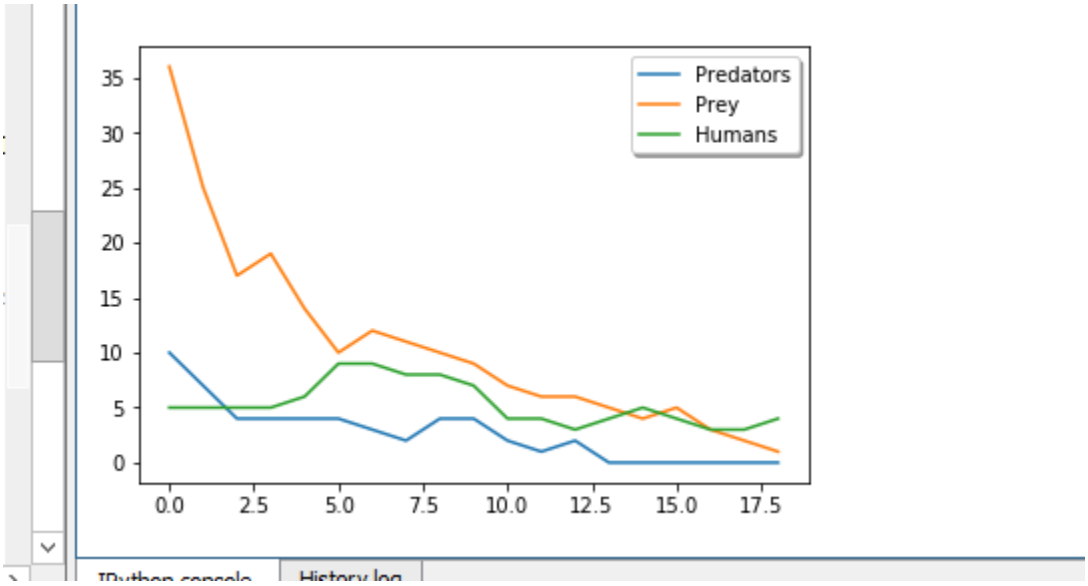
```
305                 self.hunt_clock=self.hunt_time
306                 self.moved=True
307
308
309 ########################################
310 def main(predator_breed_time=6, predator_starve_time=3, human_b
311          size=10, ticks=300):
312     ''' main simulation. Sets defaults, runs event loop, plots
313     '''
314     # initialization values
315     Predator.breed_time = predator_breed_time
316     Predator.starve_time = predator_starve_time
317     Human.breed_time = human_breed_time
318     Human.starve_time = human_starve_time
319     Human.hunt_time = human_hunt_time
320     Prey.breed_time = prey_breed_time
321
322     # for graphing
323     predator_list=[]
324     prey_list=[]
325     human_list=[]
326
327     # make an island
328     isle = Island(size,initial_prey, initial_predators, initial
329     print(isle)
330
331     # event loop.
332     # For all the ticks, for every x,y location.
333     # If there is an animal there, try eat, move, breed and cle
334     print("Prey|Predators|Humans")
335     for i in range(ticks):
336         # important to clear all the moved flags!
337         isle.clear_all_moved_flags()
338         for x in range(size):
339             for y in range(size):
340                 animal = isle.animal(x,y)
341                 if animal:
```

```
In [1]: runfile('C:/Users/rsimmo19/Desktop/Summer 2018/Py
Python Programming - COP 4045/Module 5')
Grid Size: 10 x 10 Prey: 50 Predators: 10 Humans: 5
O  O  .  O  O  H  O  .  H  O
.  O  O  H  .  O  O  .  .  O
O  O  .  .  .  O  O  O  X  O
O  X  O  H  O  O  .  O  .  O
O  .  O  O  X  O  .  O  O  X
O  O  O  O  .  O  .  X  X  O
.  O  .  O  O  O  .  .  .  O
.  .  .  H  O  .  X  .  .  X
.  O  .  O  .  .  O  O  O  .
O  O  .  X  O  .  O  .  .  X

Prey|Predators|Humans
36       10       5
7        2        4
Lost the Predator population.
Lost the Predator population.
Lost the Predator population.
Lost the Predator population.
Lost the Predator population.
Lost the Predator population.
Lost the Prey population. Quiting.

 H  .  H  .  .  .  .  .  .  .
H  .  .  .  .  .  .  .  .  .
.  H  .  .  .  .  .  .  .  .
H  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .  .  .
```

IPython console | History log



IPython console | History log