#Classes Challenge 36: Pythonagachi Simulator App

import random

#Define the Creature class

class Creature():

"""Create a simple Tomogachi clone."""

def \_\_init\_\_(self, name):

"""Initialize attributes"""

self.name = name.title()

#Attributes to track playing the game (0-10)

self.hunger = 0

self.boredom = 0

self.tiredness = 0

self.dirtiness = 0

self.food = 2 #Represents food inventory

self.is\_sleeping = False #Bool to track if creature is sleeping

self.is\_alive = True #Bool to track if creature is alive

def eat(self):

"""Simulate eating. Each time you eat, take one food away from the inventory and randomly take a value away from hunger."""

#First, make sure there is food available

if self.food > 0:

self.food -= 1

self.hunger -= random.randint(1,4)

print("Yum! " + self.name + " ate a great meal!")

else:

print(self.name + " doesn't have any food! Better forage for some.")

#If the hunger is less than zero, set it to zero

if self.hunger < 0:

self.hunger = 0

def play(self):

"""Play a guessing game to lower the creatures boredom.

If you win the game, lower the boredom even more."""

#Simple guessing game

value = random.randint(0,2)

print("\n" + self.name + " is thinking of a number 0, 1 or 2.")

guess = int(input("What is your guess: "))

#Lower the boredom attribute based on the users guess

if guess == value:

print("That is correct!")

self.boredom -= 3

else:

print("WRONG! " + self.name + " was thinking of " + str(value) + ".")

self.boredom -= 1

#If the boredom is less than zero, set it to zero

if self.boredom < 0:

self.boredom = 0

def sleep(self):

"""Simulate sleeping. The only thing a player can do when the creature is sleeping

is try to wake up. However, tiredness and boredom should decrease each round when sleeping"""

self.is\_sleeping = True

self.tiredness -= 3

self.boredom -= 2

print("Zzzzzzz...Zzzzzzz...Zzzzzzz...")

#If tiredness or boredom is less than zero, set it to zero

if self.tiredness < 0:

self.tiredness = 0

if self.boredom < 0:

self.boredom = 0

def awake(self):

"""Simulate randomly waking a creature up."""

#Creature has a 1/3 chance to randomly wake up

value = random.randint(0,2)

#If creature wakes up, set tiredness to zero!

if value == 0:

print(self.name + " just woke up!")

self.is\_sleeping = False

else:

print(self.name + " won't wake up...")

self.sleep()

def clean(self):

"""Simulate taking a bath to completely clean the creature"""

self.dirtiness = 0

print(self.name + " has taken a bath. All clean!")

def forage(self):

"""Simulate foraging for food. This will increase the creatures food attribute

however, it will also increase their dirtiness"""

#Randomly find food from 0 to 4 pieces

food\_found = random.randint(0,4)

self.food += food\_found

#Creature gets dirty from foraging

self.dirtiness += 2

print(self.name + " found " + str(food\_found) + " pieces of food!")

def show\_values(self):

"""Show the current information about the creature"""

#Show creature attributes

print("\nCreature Name: " + self.name)

print("Hunger (0-10): " + str(self.hunger))

print("Boredom (0-10): " + str(self.boredom))

print("Tiredness (0-10): " + str(self.tiredness))

print("Dirtiness (0-10): " + str(self.dirtiness))

print("\nFood Inventory: " + str(self.food) + " pieces")

#Show current sleeping status

if self.is\_sleeping:

print("Current Status: Sleeping")

else:

print("Current Status: Awake")

def increment\_values(self, diff):

"""user must set an arbitrary difficulty. This will control how much "damage" you take

each round. Update the current values of the creature based on this difficulty."""

#Increase the hunger and dirtiness regardless if the creature is awake or sleeping

self.hunger += random.randint(0, diff)

self.dirtiness += random.randint(0, diff)

#If the creature is awake, he should be growing tired and growing bored.

if self.is\_sleeping == False:

self.boredom += random.randint(0, diff)

self.tiredness += random.randint(0, diff)

def kill(self):

"""Check for all conditions to kill or sleep the creature."""

#First two checks, will kill the creature

if self.hunger >= 10:

print(self.name + " has starved to death...")

self.is\_alive = False

elif self.dirtiness >= 10:

print(self.name + " has suffered an infection and died...")

#Next two checks, will put the creature to sleep

elif self.boredom >= 10:

self.boredom = 10

print(self.name + " is bored. Falling asleep...")

self.is\_sleeping = True

elif self.tiredness >= 10:

self.tiredness = 10

print(self.name + " is sleepy. Falling asleep...")

self.is\_sleeping = True

#Helper functions outside of the creature class

def show\_menu(creature):

"""Show the menu options for the player. If the creature is sleeping, the player

can ONLY try to wake the creature up by default."""

#If the creature is sleeping, only allow the user to wake the creature.

#Hard code the value for sneaky users.

if creature.is\_sleeping:

choice = input("\nEnter (6) to try and wake up: ")

choice = "6"

#Creature is awake, give full fuctionality to user

else:

print("\nEnter (1) to eat.")

print("Enter (2) to play.")

print("Enter (3) to sleep.")

print("Enter (4) to take a bath.")

print("Enter (5) to forage for food.")

choice = input("What is your choice: ")

return choice

def call\_action(creature, choice):

"""Given the players choice, call the appropriate class method."""

#Call the appropriate creature method

if choice == "1":

creature.eat()

elif choice == "2":

creature.play()

elif choice == "3":

creature.sleep()

elif choice == "4":

creature.clean()

elif choice == "5":

creature.forage()

elif choice == "6":

creature.awake()

#User entered in invalid input. Do not call any methods.

else:

print("Sorry, that is not a valid move.")

#The main code

print("Welcome to the Pythonagachi Simulator App")

#Set the difficulty level

difficulty = int(input("Please choose a difficulty level (1-5): "))

if difficulty > 5:

difficulty = 5

elif difficulty < 1:

difficulty = 1

#The overall main game loop

running = True

while running:

#Get user input for creature name and make a creature

name = input("What name would you like to give your pet Pythonagachi: ")

player = Creature(name)

rounds = 1

#The game loop that sumulates an individual round

#This loop should run as long as the creature is alive

while player.is\_alive:

print("\n--------------------------------------------------------------------------------")

print("Round #" + str(rounds))

#An individual round should show values, get a players move, call the appropriate method

player.show\_values()

round\_move = show\_menu(player)

call\_action(player, round\_move)

print("\nRound #" + str(rounds) + " Summary: ")

#Summarize the effects of the current round

player.show\_values()

input("\nPress (enter) to continue...")

#Increment values and check for death

player.increment\_values(difficulty)

player.kill()

#Round is over

rounds += 1

#The creature has died. Game over.

print("R.I.P.")

print(player.name + " survived a total of " + str(rounds-1) + " rounds.")

#Ask the user to play again.

choice = input("Would you like to play again (y/n): ").lower()

if choice != "y":

running = False

print("Thank you for playing Pythonagachi!")