humans.py

import: Agent from mesa.agent, college_likelihood from fnnr_config_file, random, math, decimal

create lists: single_male_list, married_male_list, human_birth_list, human_death_list, human_avoidance_list (contains latest coordinates of human movement for monkeys to avoid), birth_flag_list, marriage_flag_list, human_demographic_structure_list, hh_migration_flag (for head of household), num_labor_list (households are tracked by list indices), human_marriage_list, head_of_household_list, former_hoh_list (for heads of households who have migrated), hh_size_list, total_migration_list (not cumulative; total at a time), total_re_migration_list (cumulative), first_step_income_list (to be used in the first step only)

Define _readCSV, which reads in .csv files

class Human(Agent):
Set up all attributes (see model.py for the full list).
Each step, loop through all individual agents.

First step of the model only (if self.model.time == 1/73):
if not migrated (self.migration_status ==0):
add self.income_local_off_farm to household_income_list[self.hh_id] (the household is the list index)
set first_step_income_list[self.hh_id] flag to 1 so duplicate incomes aren't added for each household member
set head of household for each household
calculate total number of laborers (num_labor_list[self.hh_id] += 1 for each agent whose self.work_status == 1)
calculate single males and married males in lists (single_male_list takes both self.unique_id and self.hh_id as arguments because married women migrate to their husband's household, so the husband's self.hh_id needs to be stored explicitly)
elif migrated (self.migration_status == 1):
calculate total number of migrants (set total_migration_list[self.hh_id] +=1 for each agent)

define step() function (for each step of the model):
for non-migrants:
check education/college migration or retirement
check migration (1/73 chance per 5-day time-step, or on average, once a year)
check head of household
move humans on the grid
for migrants:
check re-migration (1/73 chance per 5-day time-step, or on average, once a year)
check marriage (1/73 chance per 5-day time-step, or on average, once a year)
check death (1/73 chance per 5-day time-step, or on average, once a year)

```
        age the human agent
        check age_category
        add 1/73 (in model years) to the last_birth_time
        check birth possibility for married females of birth-giving age
        check and shuffle single_male_list

define movement() function:
        clear human_avoidance_list every step after it reaches a certain length (e.g.
        everyone has moved)
        masterdict = self.model.saveLoad(masterdict archive from model.py)
        if self.current_position not in masterdict where it is out of monkey range (such as
        low elevation):
                human_avoidance_list.append(self.current_position)
                also, human_avoidance_list.append(all 8-cell neighbors of
                self.current_position)
                if self.resource_check == 0 (the human has not gathered the resource yet):
                self.move_to_point(self.resource_position)
                if self.current_position == self.resource_position (if agent is at resource):
                set self.resource_check to 1
                elif self.resource_check == 1 (the human has gathered the resource):
                self.move_to_point(self.home_position)
                Reset resource from random choice in list of resources (only one may be
                gathered at a time); from model import resource_dict
                resource = random.choice(resource_dict[self.hh_id])
                set new resource and repeat gathering process

define age_check() function (working status, education status, death rates,
college/university migration):
        if 15 < self.age < 59:
                if self.work_status == 0:
                        set self.work_status to 1
                        set num_labor_list[self.hh_id] += 1
        else:
                set self.work_status to 0

        if 7 < self.age < 19 and random.random() < 0.9:
        self.education +=1
        This means on average, FNNR residents will have about 10-11+ years of
        education; somewill have more, and some less.
        Death rates per are are also calculated.
            if self.age <= 6:
               self.death_rate = 0.00745 * 0.5
            elif 6 < self.age <= 13:
               self.death_rate = 0.0009 * 0.5
            elif 13 < self.age <= 16:
               self.death_rate = 0.00131 * 0.5
```

elif 16 < self.age <= 21:
self.death_rate = 0.00196 * 0.5
elif 21 < self.age <= 60:
self.death_rate = 0.001291 * 0.5
elif 60 < self.age:
self.death_rate = 0.05354 * 0.5

0.5 is a changeable multiplier that depends on the frequency of death checks (in this case, yearly); what matters is that death rates are proportional by age category.

if 16 < self.age < 20 and random.random() < (0.0192 * college_likelihood) and self.migration_status == 0 and random.random() < ((1/73) / 4):
Division by 4 in the odds represents the chance triggering once in four years during college-going age; college_likelihood is set in fnnr_config_file.py
If these chances succeed, this human agent goes to college and is removed from the household
set self.migration_status to 2
set self.education += 4 years
Out-migration/death-like attribute changes:
set hh_size_list[self.hh_id] -= 1
set self.hh_id to 'Migrated' and take care of head_of_household reassignment (if HoH)
set num_labor_list[self.hh_id] -= 1 if applicable
human_demographic_structure_list[self.age_category] -= 1
remove human agent from schedule and grid

define check_age_category() function:
set age_category to 1-9 if male and to 10-19 if female (checks as the person ages)

define hoh_check() function:
If the head of the household has migrated, designates the next capable person as a substitute head of household; this slows down resource gathering temporarily

define birth_check() function:
Add children to the reserve; this is triggered for married females of age only
if self.children < self.birth_plan and if self.last_birth_time >= random.uniform(1, 4): (random.uniform means random float number)
determine attributes for children (gender is randomly generated with a 50/50 chance; see model.py for normal human generation rules) and add the new human agent to the schedule
ind = Human(for attribute listing, see model.py)
self.model.schedule.add(ind)
set self.model.number_of_humans += 1
set self.model.human_id_count += 1
hh_size_list[self.hh_id] += 1
human_birth_list.append(last + 1)

```
                    if ind.gender == 1, aka female:
                        human_demographic_structure_list[0] += 1
                    elif ind.gender == 2, aka male:
                        human_demographic_structure_list[10] += 1

  define death_check() function:
        Small chance of dying every step; chance increases the older one gets, see
        age_check()
      if random.random() < self.death_rate:
        remove self as head of household (if applicable) or former head of household if
        migrated, remove self from single_male_list or single_married_list (if applicable),
        subtract 1 from num_labor_list[self.hh_id] if working, subtract 1 from
        hh_size_list
        append self to human_death_list
        set self.model.number_of_humans -= 1
        set human_demographic_structure_list[self.age_category] -= 1

        self.model.schedule.remove(self)
                if self in self.model.grid:
                        self.model.grid.remove_agent(self)

define marriage_check(self):
This function occurs approximately once a year (1/73) chance and thus uses a yearly
marriage rate.
The below probability triggers for everyone:
        if random.random() < 0.00767:
                marriage_flag_list.append(1)
The below only happens from the marrying female's point of view:
        if random.uniform(20, 30) < int(self.age) < 45 and int(self.gender) == 2 and
        int(self.marriage) !=  marriage_flag_list != [] and self.migration_status ==
        0:  (marriage occurs)
                set self.marriage = 1
                marriage_flag_list.remove(1)
                modify household size list as female moves to male's household
                add husband (randomly chosen from single male list) to married male list
                human_marriage_list[self.hh_id] += 1

define migration_check() function:
        from land import non_gtgp_part_list, gtgp_part_list, non_gtgp_area_list
        self.non_gtgp_area = non_gtgp_area_list[self.hh_id]
        calculate non_gtgp_land_per_labor as self.non_gtgp_area /
        num_labor_list[self.hh_id]
        determine if human is part of a GTGP household
        Shuang sets remittances of the individual to the normal variate of (1200,400^2),
        but I think this is wrong because it can generate negative numbers; I chose to
```

```
        import mig_remittances from his original data and have descendants inherit
        parents' attributes
        The below is Shuang's migration formula, which is a lifetime probability (the
        yearly chance is divided by 45, for ages 15-59)
            prob = math.exp(2.07 - 0.00015 * float(self.income_local_off_farm) + 0.67 *
        float(num_labor_list[self.hh_id])
                    + 4.36 * float(self.migration_network) - 0.58 *
        float(non_gtgp_land_per_labor)
                    + 0.27 * float(self.gtgp_part) - 0.13 * float(self.age) + 0.07 *
        float(self.gender)
                    + 0.17 * float(self.education) + 0.88 * float(self.marriage) +
                    1.39 * float(self.work_status) + 0.001 * float(self.mig_remittances))  #
        Shuang's formula
            mig_prob = (prob / (prob + 1) / 45)
        Finally, out-migration can only occur if there is at least one other person in the
        household.
            if random.random() < mig_prob and hh_size_list[self.hh_id] >= 2:
                modify household size list, head of household status, self.hh_id,
                self.migration_status, num_labor_list if working, total migration list,
                self.work_status, self.hh_id
                add migrant remittances to household income
                if self in self.model.grid:
                    self.model.grid.remove_agent(self)

def re_migration_check(self):
    if self.migration_status == 1:
        prob = math.exp(-1.2 + 0.06 * float(self.age) - 0.08 * self.mig_years)
        re_mig_prob = (prob / (prob + 1) / 45) # 45 for ages 15-59; migration is a
        lifetime, not yearly, probability
        self.mig_years += 1
        if random.random() < re_mig_prob: they re-migrate
                adjust household size list, num_labor_list if working, migration status,
                migration remittances on household income, household ID, number of
                years migrated, head of household if one was previously HoH
                add self.unique_id to total_re_migration_list and remove from
                total_migration_list

define move_to_point()/move_to() functions:
        Moves human agent to assigned point according to resource-gathering frequency
        (higher frequency = 'exposed' to monkeys more in the wild while gathering
        resources)
        For example, if a human agent gathers a resource 5 times a month and their
        resource is 21 grid coordinates away, they move approximately 4 grid coordinates
        towards the pixel each step.
        This is not accurate in real time (humans move much faster than depicted), but
        makes it so that resources are gathered proportionally to each other.
```

The destination for a human at any given time is always either a resource or their home position (household coordinate).

class Resource(Agent):

set up position, freuency, hh_id, type

see model.py; these are imported from 'hh added to the grid