import random

import time

from dataclasses import dataclass

from typing import List, Optional

# Town Planning Configuration

MEMORY\_TOWN\_SIZE = 50 # How many lots are in our memory neighborhood

SIMULATION\_DAYS = 15 # How many days to run our simulation

@dataclass

class MemoryHouse:

"""A house built by a process in our memory neighborhood"""

address: int # Where the house starts on Memory Street

lots\_occupied: int # How many lots the house covers

owner: str # Which process family lives here (e.g., "Baker")

move\_in\_day: int # When the family moved in

move\_out\_day: Optional[int] = None # When they left (None if still living here)

class MemoryTown:

"""Manages our memory neighborhood and its houses"""

def \_\_init\_\_(self, total\_lots: int):

self.total\_lots = total\_lots

self.all\_houses = [] # Records all houses ever built

self.current\_day = 0 # Tracks time in our simulation

def build\_house(self, needed\_lots: int, family\_name: str) -> bool:

"""A family tries to build a house on available lots"""

if needed\_lots > self.total\_lots:

print(f"Family {family\_name} can't build - wants {needed\_lots} lots but town only has {self.total\_lots}!")

return False

# Sort houses by address to find empty spaces between them

self.all\_houses.sort(key=lambda house: house.address)

next\_empty\_lot = 0 # Start looking from the beginning of town

for house in self.all\_houses:

# Skip houses where families have moved out

if house.move\_out\_day is not None:

continue

# Check for space before this house

if next\_empty\_lot + needed\_lots <= house.address:

self.\_welcome\_new\_family(next\_empty\_lot, needed\_lots, family\_name)

return True

next\_empty\_lot = house.address + house.lots\_occupied

# Check if there's space at the end of town

if next\_empty\_lot + needed\_lots <= self.total\_lots:

self.\_welcome\_new\_family(next\_empty\_lot, needed\_lots, family\_name)

return True

print(f"Family {family\_name} couldn't find {needed\_lots} adjacent lots to build")

return False

def \_welcome\_new\_family(self, address: int, lots: int, family: str) -> None:

"""Helper to create and record a new house"""

new\_house = MemoryHouse(

address=address,

lots\_occupied=lots,

owner=family,

move\_in\_day=self.current\_day

)

self.all\_houses.append(new\_house)

print(f"Family {family} built a {lots}-lot house at address {address}")

def family\_moves\_out(self, family\_name: str) -> bool:

"""A process family leaves town, freeing their house"""

found\_family = False

for house in self.all\_houses:

if house.owner == family\_name and house.move\_out\_day is None:

house.move\_out\_day = self.current\_day

found\_family = True

print(f"Family {family\_name} moved out from {house.lots\_occupied}-lot house at {house.address}")

if not found\_family:

print(f"Family {family\_name} doesn't have any active houses in town")

return found\_family

def draw\_town\_map(self) -> None:

"""Create a visual map of current memory neighborhood"""

# Start with all empty lots

town\_map = ['\_'] \* self.total\_lots

# Mark occupied houses

for house in self.all\_houses:

if house.move\_out\_day is None or house.move\_out\_day > self.current\_day:

for lot in range(house.address, house.address + house.lots\_occupied):

town\_map[lot] = house.owner[0] # Use first letter of family name

# Format the map into blocks of 10 lots

map\_rows = []

for i in range(0, self.total\_lots, 10):

map\_rows.append("".join(town\_map[i:i+10]))

# Print the town status

print(f"\n Day {self.current\_day} - Memory Neighborhood Map:")

print("┌" + "──────────┬" \* (len(map\_rows)-1) + "──────────┐")

for row in map\_rows:

print("│" + "│".join(f" {lot} " for lot in row) + "│")

print("└" + "──────────┴" \* (len(map\_rows)-1) + "──────────┘")

print(f"Key: \_ = Empty lot, First letter = Family (e.g., 'B' for Baker)")

print("═" \* 60) # Decorative separator

self.current\_day += 1

def run\_town\_simulation():

"""Run our memory neighborhood simulation"""

memory\_town = MemoryTown(MEMORY\_TOWN\_SIZE)

family\_names = ["Baker", "Taylor", "Miller", "Weaver", "Fisher"]

print("\n Welcome to Memory Neighborhood Simulator!")

print(f"Watching {MEMORY\_TOWN\_SIZE} memory lots over {SIMULATION\_DAYS} days")

print("Families will build houses and move out randomly\n")

for day in range(SIMULATION\_DAYS):

print(f"\n Morning of Day {day + 1}:")

# Random neighborhood activity

activity = random.choices(

["build", "move\_out", "quiet\_day"],

weights=[0.45, 0.45, 0.1], # Most days have activity

k=1

)[0]

if activity == "build":

family = random.choice(family\_names)

lots\_needed = random.randint(5, 10) # Medium-sized houses

memory\_town.build\_house(lots\_needed, family)

elif activity == "move\_out":

family = random.choice(family\_names)

memory\_town.family\_moves\_out(family)

# Show the town map at end of day

memory\_town.draw\_town\_map()

time.sleep(1.5) # Comfortable reading pace

if \_\_name\_\_ == "\_\_main\_\_":

run\_town\_simulation()

print("\nSimulation complete! Thanks for visiting Memory Neighborhood!")