

"""

=====
Title: Module 6.2 Forest Fire Simulation: Program and Revised Flowchart

Original Author: Al Sweigart

Modified By:

Group C (Carmelle Nguessan, Daniel Vance, Scott Johnson, Wade Eckert, Zak Nizam)

Date Modified: 24 January 2026

Description:

A simulation of wildfires spreading in a forest. Press Ctrl-C to stop.
This version adds a lake near the center that never changes (firebreak).

=====
Forest Fire Sim, modified by Sue Sampson, based on a program by Al Sweigart

A simulation of wildfires spreading in a forest. Press Ctrl-C to stop.

Inspired by Nicky Case's Emoji Sim <http://ncase.me/simulating/model/>

** use spaces, not indentation to modify **

Tags: short, bext, simulation

"""

```
import random, sys, time

# bext is used to color output in the terminal.
# If it's missing, we print install instructions and exit cleanly.
try:
    import bext
except ImportError:
    print('This program requires the bext module, which you')
    print('can install by following the instructions at')
    print('https://pypi.org/project/Bext/')
    sys.exit()

# Set up the constants:
WIDTH = 79
HEIGHT = 22

TREE = 'A'
FIRE = '@'
WATER = '~' # Water tile (blue). Acts as a firebreak.
EMPTY = ' '

# (!) Try changing these settings to anything between 0.0 and 1.0:
INITIAL_TREE_DENSITY = 0.20 # Percentage of forest that starts with trees.
GROW_CHANCE = 0.01           # Chance a blank space turns into a tree.
FIRE_CHANCE = 0.01           # Chance a tree is hit by lightning & burns.

# (!) Try setting the pause length to 1.0 or 0.0:
PAUSE_LENGTH = 0.5

def main():
    forest = createNewForest()
    bext.clear()

    while True: # Main program loop.
        displayForest(forest)

        # Run a single simulation step:
        nextForest = {'width': forest['width'],
                      'height': forest['height']}

        for x in range(forest['width']):
            for y in range(forest['height']):
                if (x, y) in nextForest:
                    # If we've already set nextForest[(x, y)] on a
                    # previous iteration, just do nothing here:
                    continue

                    # Water never changes and blocks fire completely.
                    if forest[(x, y)] == WATER:
                        nextForest[(x, y)] = WATER
                        continue # Skip all other logic for water tiles

                    if ((forest[(x, y)] == EMPTY)
                        and (random.random() <= GROW_CHANCE)):
                        # Grow a tree in this empty space.
                        nextForest[(x, y)] = TREE
                    elif ((forest[(x, y)] == TREE)
                        and (random.random() <= FIRE_CHANCE)):
                        # Lightning sets this tree on fire.
                        nextForest[(x, y)] = FIRE
                    elif forest[(x, y)] == FIRE:
                        # This tree is currently burning.
                        # Loop through all the neighboring spaces:
                        for ix in range(-1, 2):
                            for iy in range(-1, 2):
```

```

        # Fire spreads only to neighboring trees.
        # This naturally prevents fire from crossing water.
        if forest.get((x + ix, y + iy)) == TREE:
            nextForest[(x + ix, y + iy)] = FIRE
        # The tree has burned down now, so erase it:
        nextForest[(x, y)] = EMPTY
    else:
        # Just copy the existing object:
        nextForest[(x, y)] = forest[(x, y)]
forest = nextForest

time.sleep(PAUSE_LENGTH)

def createNewForest():
    """Returns a dictionary for a new forest data structure."""
    forest = {'width': WIDTH, 'height': HEIGHT}
    for x in range(WIDTH):
        for y in range(HEIGHT):
            # ORIGINAL (kept for reference):
            # if (random.random() * 100) <= INITIAL_TREE_DENSITY:
                # UPDATED:
                # random.random() returns 0.0-1.0, so compare directly to density
                if random.random() <= INITIAL_TREE_DENSITY:
                    forest[(x, y)] = TREE # Start as a tree.
                else:
                    forest[(x, y)] = EMPTY # Start as an empty space.

    # Add a lake near the center of the forest (firebreak).
    addLake(forest)

    return forest

def addLake(forest):
    """Adds a rectangular lake near the center of the forest grid."""
    # Keep it "roughly center" but not so large that it dominates the map.
    lake_width = forest['width'] // 4
    lake_height = forest['height'] // 4

    start_x = (forest['width'] - lake_width) // 2
    start_y = (forest['height'] - lake_height) // 2

    for x in range(start_x, start_x + lake_width):
        for y in range(start_y, start_y + lake_height):
            forest[(x, y)] = WATER

def displayForest(forest):
    """Display the forest data structure on the screen."""
    # Move the cursor to the top-left so each frame redraws in place.
    bext.goto(0, 0)

    for y in range(forest['height']):
        for x in range(forest['width']):
            if forest[(x, y)] == TREE:
                bext.fg('green')
                print(TREE, end='')
            elif forest[(x, y)] == FIRE:
                bext.fg('red')
                print(FIRE, end='')
            elif forest[(x, y)] == WATER:
                bext.fg('blue')
                print(WATER, end='')
            elif forest[(x, y)] == EMPTY:
                # EMPTY stays default color
                print(EMPTY, end='')

    print()

    # Reset color so the status text prints normally.
    bext.fg('reset') # Use the default font color.
    print('Grow chance: {}%'.format(GROW_CHANCE * 100), end='')
    print('Lightning chance: {}%'.format(FIRE_CHANCE * 100), end='')
    print('Press Ctrl-C to quit.')

# If this program was run (instead of imported), run the game:
if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        sys.exit() # When Ctrl-C is pressed, end the program.

```