CA

Michèle Fille

manually set left and right margins to 2cm, and fontsize to 8 for nicer format :)

# Define variables  
generation = 0 # Start at generation 0  
ruleset = [0, 0, 0, 1, 1, 1, 1, 0] # Rule 30  
  
# Initialize cells array  
cells = [0] \* 55 # Array with 55 cells all set to 0 (dead)  
cells[len(cells) // 2] = 1 # Set the center cell to 1 (alive)  
  
  
# Function to determine the state of a cell based on its neighbourhood  
def determine\_cell\_state(left\_cell, middle\_cell, right\_cell):  
 neighbourhood = (str(left\_cell) + str(middle\_cell) +  
 str(right\_cell)) # Convert the states of neighboring cells into a string  
 index = int(neighbourhood, 2) # Convert the binary string into a decimal integer  
 return ruleset[7 - index] # Reverse the index since ruleset is defined in descending order  
  
  
# Function to print the state of cells as black or with square for a given generation  
def print\_cells(cells, generation):  
 # Iterate through each cell in the list of cells and check the state of the cell  
 for cell in cells:  
 if cell == 1: # If the cell is 1 (alive), print a black square (■)  
 print('\u25a0', end="") # to check ASCII code: ascii("■") -> '\u25a0'  
 else: # If the cell is 0 (dead), print a withe square (□)  
 print('\u25a1', end="") # to check ASCII code: ascii("□") -> '\u25a1'  
  
 print(f" Generation {generation}") # Print the generation number  
  
  
# Print initial generation  
print\_cells(cells, generation)  
  
# Compute the states of cells (excluding edge cells) for multiple generations  
# Loop through 31 generations (0 to 20)  
for generation in range(1, 21):  
 nextgen = cells[:] # Create a copy of the current state of cells for the next generation  
 # Loop through each cell, excluding the edge cells  
 for i in range(1, len(cells) - 1):  
 left = cells[i - 1] # Get the state of the left neighbour cell  
 middle = cells[i] # Get the state of the middle (target) cell  
 right = cells[i + 1] # Get the state of the right neighbour cell  
 nextgen[i] = determine\_cell\_state(left, middle, right) # Determine the state of the middle (target) cell  
  
 cells = nextgen # Update the state of cells to the next generation  
 print\_cells(cells, generation) # Print the state of cells for the current generation

□□□□□□□□□□□□□□□□□□□□□□□□□□□■□□□□□□□□□□□□□□□□□□□□□□□□□□□ Generation 0  
□□□□□□□□□□□□□□□□□□□□□□□□□□■■■□□□□□□□□□□□□□□□□□□□□□□□□□□ Generation 1  
□□□□□□□□□□□□□□□□□□□□□□□□□■■□□■□□□□□□□□□□□□□□□□□□□□□□□□□ Generation 2  
□□□□□□□□□□□□□□□□□□□□□□□□■■□■■■■□□□□□□□□□□□□□□□□□□□□□□□□ Generation 3  
□□□□□□□□□□□□□□□□□□□□□□□■■□□■□□□■□□□□□□□□□□□□□□□□□□□□□□□ Generation 4  
□□□□□□□□□□□□□□□□□□□□□□■■□■■■■□■■■□□□□□□□□□□□□□□□□□□□□□□ Generation 5  
□□□□□□□□□□□□□□□□□□□□□■■□□■□□□□■□□■□□□□□□□□□□□□□□□□□□□□□ Generation 6  
□□□□□□□□□□□□□□□□□□□□■■□■■■■□□■■■■■■□□□□□□□□□□□□□□□□□□□□ Generation 7  
□□□□□□□□□□□□□□□□□□□■■□□■□□□■■■□□□□□■□□□□□□□□□□□□□□□□□□□ Generation 8  
□□□□□□□□□□□□□□□□□□■■□■■■■□■■□□■□□□■■■□□□□□□□□□□□□□□□□□□ Generation 9  
□□□□□□□□□□□□□□□□□■■□□■□□□□■□■■■■□■■□□■□□□□□□□□□□□□□□□□□ Generation 10  
□□□□□□□□□□□□□□□□■■□■■■■□□■■□■□□□□■□■■■■□□□□□□□□□□□□□□□□ Generation 11  
□□□□□□□□□□□□□□□■■□□■□□□■■■□□■■□□■■□■□□□■□□□□□□□□□□□□□□□ Generation 12  
□□□□□□□□□□□□□□■■□■■■■□■■□□■■■□■■■□□■■□■■■□□□□□□□□□□□□□□ Generation 13  
□□□□□□□□□□□□□■■□□■□□□□■□■■■□□□■□□■■■□□■□□■□□□□□□□□□□□□□ Generation 14  
□□□□□□□□□□□□■■□■■■■□□■■□■□□■□■■■■■□□■■■■■■■□□□□□□□□□□□□ Generation 15  
□□□□□□□□□□□■■□□■□□□■■■□□■■■■□■□□□□■■■□□□□□□■□□□□□□□□□□□ Generation 16  
□□□□□□□□□□■■□■■■■□■■□□■■■□□□□■■□□■■□□■□□□□■■■□□□□□□□□□□ Generation 17  
□□□□□□□□□■■□□■□□□□■□■■■□□■□□■■□■■■□■■■■□□■■□□■□□□□□□□□□ Generation 18  
□□□□□□□□■■□■■■■□□■■□■□□■■■■■■□□■□□□■□□□■■■□■■■■□□□□□□□□ Generation 19  
□□□□□□□■■□□■□□□■■■□□■■■■□□□□□■■■■□■■■□■■□□□■□□□■□□□□□□□ Generation 20