CAL POLY

MEMORANDUM

January 25, 2026

TO: Charlie Refvem
FROM: Jack Toyama

SUBJECT: Homework 0x01 – Finite State Machines and Python Basics

---

My solution to creating the Mastermind game in the console required an FSM, a couple helper functions, and a simple "main" that constantly updates the FSM state. The FSM uses 15 states; an initialization for resetting the board between plays, 12 game states, and a win and a lose. These states update based on user input. Between game states, the FSM changes state to the 'win' state if the correct guess is input, if the guess is valid but incorrect, the FSM moves on to the next game state. Invalid guesses keep the program in the waiting for input loop until a valid input is given. My implementation allows for somewhat non blocking code since the input() function holds up the program, but other functions can run in the main loop at the same time. My implementation also utilizes a display board function that updates based on what the guesses have been in the round. This means that the board is just printed every time a state is entered and no calculations on what to display must be done. Since the update happens after the state is entered, I left out the action from the diagram.

The game can be replayed infinite times, and the win and loss count are both displayed. My solution to determining the correctness of each guess is to turn both the correct and input answer as a list. I then compare indices and remove any matching guesses. I then use a for-loop to iterate through comparing each number in the input answer to whatever is left of the correct answer, add those as guesses that are correct but in the wrong position, and remove it from the answer list. This is all done locally so it does not effect the rest of the game.

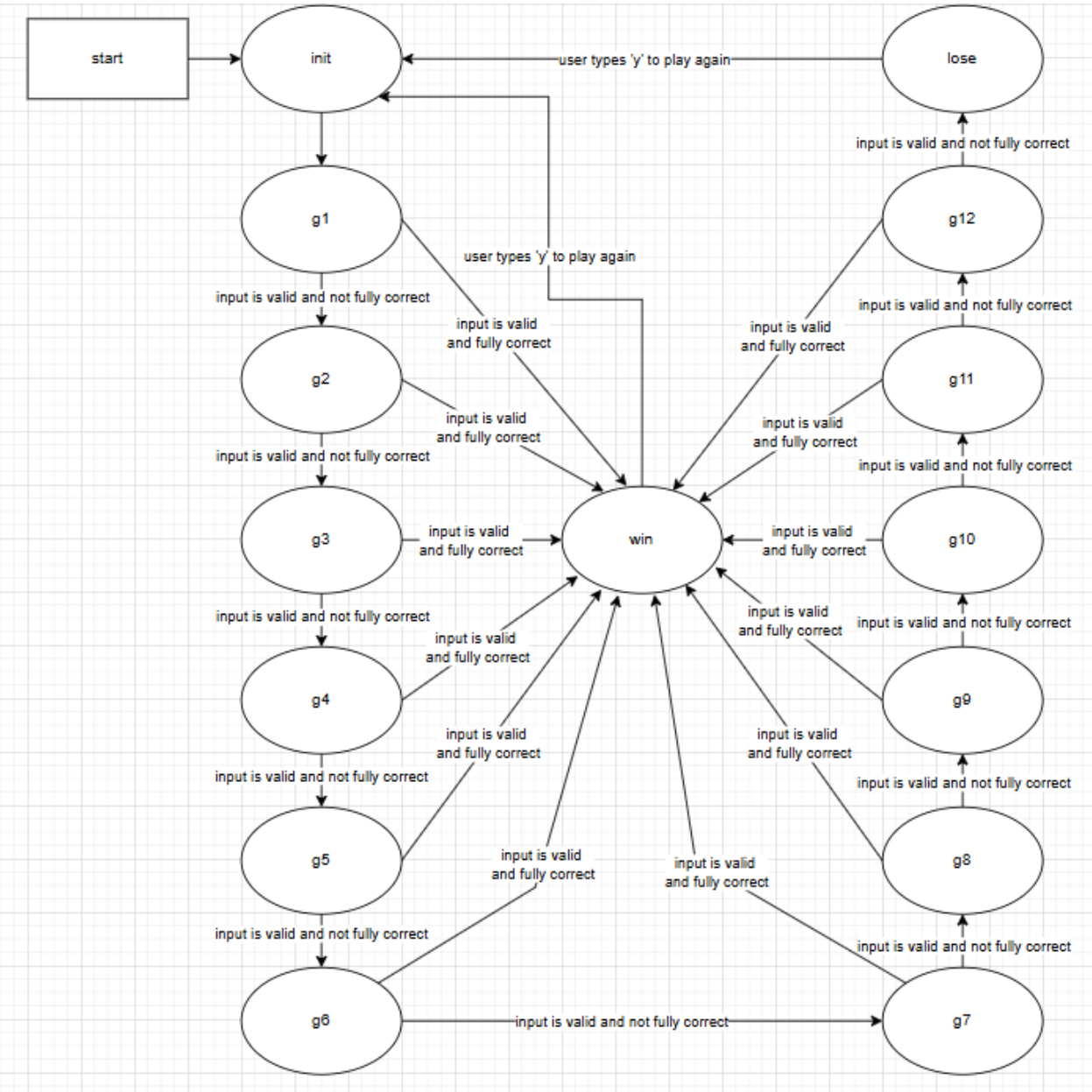

Figure 1. FSM Diagram with states and transition conditionals.

Sources:

W3Schools.com for function definitions and the Random library information

www.geeksforgeeks.org for string and list help, as well as .split() and .replace()

```
import random

state = "init"
wins = 0
losses = 0
valid_values = ["0","1","2","3","4","5"]
answer = ""

def update_FSM():
    global state
    if state == "init":
        # initialize all the game values as global variables, set to blank
        global answer
        global wins
        global losses
        global g1
        global g2
        global g3
        global g4
        global g5
        global g6
        global g7
        global g8
        global g9
        global g10
        global g11
        global g12
        global c1
        global c2
        global c3
        global c4
        global c5
        global c6
        global c7
        global c8
        global c9
        global c10
        global c11
        global c12
        state = "g1"
        reset_board()
    if state == "g1":
        # create the answer code
        answer = random_4_digit_string()
        # display board
```

```
        display_board()
        # get input from user
        g1 = get_next_input()
        # determine correctness
        correct, c1 = calc_correctness(g1)
        # move to win state if fully correct, otherwise next game state
        if correct == 4:
            state = "win"
        else:
            state = "g2"
    elif state == "g2":
        display_board()
        g2 = get_next_input()
        correct, c2 = calc_correctness(g2)
        if correct == 4:
            state = "win"
        else:
            state = "g3"
    elif state == "g3":
        display_board()
        g3 = get_next_input()
        correct, c3 = calc_correctness(g3)
        if correct == 4:
            state = "win"
        else:
            state = "g4"
    elif state == "g4":
        display_board()
        g4 = get_next_input()
        correct, c4 = calc_correctness(g4)
        if correct == 4:
            state = "win"
        else:
            state = "g5"
    elif state == "g5":
        display_board()
        g5 = get_next_input()
        correct, c5 = calc_correctness(g5)
        if correct == 4:
            state = "win"
        else:
            state = "g6"
    elif state == "g6":
        display_board()
        g6 = get_next_input()
        correct, c6 = calc_correctness(g6)
```

```
        if correct == 4:
            state = "win"
        else:
            state = "g7"
    elif state == "g7":
        display_board()
        g7 = get_next_input()
        correct, c7 = calc_correctness(g7)
        if correct == 4:
            state = "win"
        else:
            state = "g8"
    elif state == "g8":
        display_board()
        g8 = get_next_input()
        correct, c8 = calc_correctness(g8)
        if correct == 4:
            state = "win"
        else:
            state = "g9"
    elif state == "g9":
        display_board()
        g9 = get_next_input()
        correct, c9 = calc_correctness(g9)
        if correct == 4:
            state = "win"
        else:
            state = "g10"
    elif state == "g10":
        display_board()
        g10 = get_next_input()
        correct, c10 = calc_correctness(g10)
        if correct == 4:
            state = "win"
        else:
            state = "g11"
    elif state == "g11":
        display_board()
        g11 = get_next_input()
        correct, c11 = calc_correctness(g11)
        if correct == 4:
            state = "win"
        else:
            state = "g12"
    elif state == "g12":
        display_board()
```

```
        g12 = get_next_input()
        correct, c12 = calc_correctness(g12)
        if correct == 4:
            state = "win"
        else:
            state = "lose"
    elif state == "win":
        display_board()
        # increment wins, prompt user to play again, return if not
        wins += 1
        user = input("you win\nplay again: (y) for yes: ")
        if user == "y":
            state = "init"
        else:
            return False
    elif state == "lose":
        display_board()
        losses += 1
        user = input("you lose\nplay again: (y) for yes: ")
        if user == "y":
            state = "init"
        else:
            return False
    return True

def get_next_input():
    # continuously ask user for input until it is valid
    while True:
        guess = input("enter guess:")
        if check_input_validity(guess):
            return guess
        else:
            print("invalid entry")

def reset_board():
    global answer
    global wins
    global losses
    global g1
    global g2
    global g3
    global g4
    global g5
    global g6
    global g7
    global g8
```

```
    global g9
    global g10
    global g11
    global g12
    global c1
    global c2
    global c3
    global c4
    global c5
    global c6
    global c7
    global c8
    global c9
    global c10
    global c11
    global c12
    answer = ""
    g1 = "    "
    g2 = "    "
    g3 = "    "
    g4 = "    "
    g5 = "    "
    g6 = "    "
    g7 = "    "
    g8 = "    "
    g9 = "    "
    g10 = "    "
    g11 = "    "
    g12 = "    "
    c1 = ""
    c2 = ""
    c3 = ""
    c4 = ""
    c5 = ""
    c6 = ""
    c7 = ""
    c8 = ""
    c9 = ""
    c10 = ""
    c11 = ""
    c12 = ""
    return

def random_4_digit_string():
    # creates 4 digit string of random integers from 0-5 inclusive
    return ''.join(str(random.randint(0, 5)) for _ in range(4))
```

```
def display_board():
    # displays board with all global variable information
    print(f"""



Mastermind! Try to break the code.
wins:  {wins}
losses: {losses}
code: {answer}
+---+---+---+---+
| {g12[0]} | {g12[1]} | {g12[2]} | {g12[3]} | {c12}
+---+---+---+---+
| {g11[0]} | {g11[1]} | {g11[2]} | {g11[3]} | {c11}
+---+---+---+---+
| {g10[0]} | {g10[1]} | {g10[2]} | {g10[3]} | {c10}
+---+---+---+---+
| {g9[0]} | {g9[1]} | {g9[2]} | {g9[3]} | {c9}
+---+---+---+---+
| {g8[0]} | {g8[1]} | {g8[2]} | {g8[3]} | {c8}
+---+---+---+---+
| {g7[0]} | {g7[1]} | {g7[2]} | {g7[3]} | {c7}
+---+---+---+---+
| {g6[0]} | {g6[1]} | {g6[2]} | {g6[3]} | {c6}
+---+---+---+---+
| {g5[0]} | {g5[1]} | {g5[2]} | {g5[3]} | {c5}
+---+---+---+---+
| {g4[0]} | {g4[1]} | {g4[2]} | {g4[3]} | {c4}
+---+---+---+---+
| {g3[0]} | {g3[1]} | {g3[2]} | {g3[3]} | {c3}
+---+---+---+---+
| {g2[0]} | {g2[1]} | {g2[2]} | {g2[3]} | {c2}
+---+---+---+---+
| {g1[0]} | {g1[1]} | {g1[2]} | {g1[3]} | {c1}
+---+---+---+---+""")

def check_input_validity(input):
    # check if input is 4 consecutive integers each being between 0 and 5
    if len(input) != 4:
        return False
    for char in input:
        if char not in valid_values:
```

```
            return False
    return True

def calc_correctness(input):
    # calculate correctness, set both half and full correct to 0
    full_correct = 0
    correct_val = 0

    # turn answer and input into local lists
    temp_ans = list(answer)
    temp_in = list(input)

    # exact matches based on indices
    for i in range(4):
        if temp_in[i] == temp_ans[i]:
            full_correct += 1
            temp_in[i] = " "
            temp_ans[i] = " "

    temp_ans = "".join(temp_ans).replace(" ", "")
    temp_in = "".join(temp_in).replace(" ", "")

    # value-only matches have to check every value in the answer list with
every entry in the input list
    for char in temp_in:
        if char in temp_ans:
            correct_val += 1
            temp_ans = temp_ans.replace(char, "", 1)

    return full_correct, "+"*full_correct + "-"*correct_val

# main loop
while True:
    if not update_FSM():
        break
```