<u>Assignment 2:</u> Controlling the Arm Using Code with Keyboard

Step 0: Create the python file

Enter the the enviroment and Cd all the way into your robotic arm folder

Create a file and name it: **assignment2.py**

Step 1: Import Libraries and Connect to the xArm

```
import xarm
import time
```

Step 2: Connect to the xArm

```
arm = xarm.Controller('USB')
```

Step 3: Set Initial Positions for All Servos

For this part, I will provide you all with the initial starting position of the servos.

```
# Initial angles for servos
a = 800     # Servo 1 (base)
b = -90.25  # Servo 2 (fixed in this activity)
c = 51.49   # Servo 3
d = -81.75  # Servo 4
e = 8.25    # Servo 5

# Move to starting pose
arm.setPosition([[1, a], [2, b], [3, c], [4, d], [5, e]], wait=True)
print("Set to Original Position")
```

Step 4: Start an Input Loop to Read User Commands

We will start by creating an infinite loop where your program waits for keyboard input.

```
while True:
    cmd = input("Command (u/d/q): ").strip().lower()
```

Step 5: Simulate "Smile Detected" with ‘u’ Key

When the user presses `'u'`, this simulates a smile detection and adjusts servos to a "smiling pose" using your original logic.

Even if you didn't assign the a and c variables to numbers, this code would still run. So you can take a look at step 6 and after pasting the code from step 6 you can come back to change the numbers that would best represent a movement of a flower.

```
    if cmd == 'u':  # Simulate "smile"
        if a > 200 and c > -2.25 and d < 2.25 and e > 8.25:
            a -= #replace this with numbers
            c -= #replace this with numbers
            d += 4.08*2
            e -= 0.98*2
            arm.setPosition([[1, a], [2, b], [3, c], [4, d], [5, e]],
wait=False)
            print("Smiling pose")
        else:
            arm.setPosition([[1, 200], [2, b], [3, -2.75], [4, 2.25], [5,
8.25]], wait=False)
```

Step 6: Simulate "Not Smiling" with ‘d’ Key

Pressing `'d'` simulates a neutral expression — servos move back to the default position.
Hint: the replace number for variable a and c should be the same as step 5

```
elif cmd == 'd':  # Simulate "not smiling"
        if a < 800 and c < 51.49 and d > -81.75 and e < 25.75:
            a += #replace this with numbers
            c += #replace this with numbers
            d -= 4.08*2
            e += 0.98*2
            arm.setPosition([[1, a], [2, b], [3, c], [4, d], [5, e]],
wait=False)
            print("Neutral pose")
        else:
            arm.setPosition([[1, 800], [2, b], [3, 51.49], [4, -81.75], [5,
25.7499]], wait=False)
```

Step 6: Quit with the ‘q’ Key

Pressing `'q'` ends the loop and the program.

```
    elif cmd == 'q':
        print("Quitting program.")
        break
```

Step 7: Handle Invalid Keys

This gives helpful feedback if the user presses a wrong key.

```
    else:
        print("Invalid input. Use 'u', 'd', or 'q'.")
    time.sleep(0.15)
```

Step 8: Discussion Question

1. Which servos changed when the arm was "smiling"?

   Servo 5

   Servo 1

2. Why are there conditions like if a > 200 before updating positions?

   So it stays in a certain range for better movement

3. What would happen if we removed those checks?

   The flower arm could break