```
from machine import Pin, I2C
import time
import sys
import select

# Setup push button on GPIO 0 with pull-up resistor
button = Pin(0, Pin.IN, Pin.PULL_UP)
last_button_state = 1  # Start unpressed

# --- RADIO CLASS DEFINITION ---
class Radio:
    def __init__(self, NewFrequency, NewVolume, NewMute):
        self.Volume = 2
        self.Frequency = 88
        self.Mute = False

        self.SetVolume(NewVolume)
        self.SetFrequency(NewFrequency)
        self.SetMute(NewMute)

        self.i2c_sda = Pin(26)
        self.i2c_scl = Pin(27)

        self.i2c_device = 1
        self.i2c_device_address = 0x10
        self.Settings = bytearray(8)

        self.radio_i2c = I2C(self.i2c_device, scl=self.i2c_scl, sda=self.i2c_sda, freq=200000)
        self.ProgramRadio()

    def SetVolume(self, NewVolume):
        try:
            NewVolume = int(NewVolume)
        except:
            return False
        if not isinstance(NewVolume, int):
            return False
        if NewVolume < 0 or NewVolume >= 16:
            return False
        self.Volume = NewVolume
        return True

    def SetFrequency(self, NewFrequency):
        try:
            NewFrequency = float(NewFrequency)
        except:
            return False
        if not isinstance(NewFrequency, float):
            return False
        if NewFrequency < 88.0 or NewFrequency > 108.0:
            return False
        self.Frequency = NewFrequency
        return True

    def SetMute(self, NewMute):
        try:
            self.Mute = bool(int(NewMute))
        except:
            return False
        return True

    def ComputeChannelSetting(self, Frequency):
        Frequency = int(Frequency * 10) - 870
        ByteCode = bytearray(2)
```

```
        ByteCode[0] = (Frequency >> 2) & 0xFF
        ByteCode[1] = ((Frequency & 0x03) << 6) & 0xC0
        return ByteCode

    def UpdateSettings(self):
        self.Settings[0] = 0x80 if self.Mute else 0xC0
        self.Settings[1] = 0x09 | 0x04
        self.Settings[2:3] = self.ComputeChannelSetting(self.Frequency)
        self.Settings[3] = self.Settings[3] | 0x10
        self.Settings[4] = 0x04
        self.Settings[5] = 0x00
        self.Settings[6] = 0x84
        self.Settings[7] = 0x80 + self.Volume

    def ProgramRadio(self):
        self.UpdateSettings()
        self.radio_i2c.writeto(self.i2c_device_address, self.Settings)

    def GetSettings(self):
        self.RadioStatus = self.radio_i2c.readfrom(self.i2c_device_address, 256)
        MuteStatus = not ((self.RadioStatus[0xF0] & 0x40) != 0x00)
        VolumeStatus = self.RadioStatus[0xF7] & 0x0F
        FrequencyStatus = ((self.RadioStatus[0x00] & 0x03) << 8) | (self.RadioStatus[0x01] & 0xFF)
        FrequencyStatus = (FrequencyStatus * 0.1) + 87.0
        StereoStatus = (self.RadioStatus[0x00] & 0x04) != 0x00
        return MuteStatus, VolumeStatus, FrequencyStatus, StereoStatus


# --- MAIN PROGRAM STARTS HERE ---
fm_radio = Radio(101.9, 2, False)


def poll_button(fm_radio):
    global last_button_state
    current_state = button.value()
    if last_button_state == 1 and current_state == 0:
        print("Button Pressed: Increasing Volume")
        if fm_radio.Volume < 15:
            fm_radio.SetVolume(fm_radio.Volume + 1)
            fm_radio.ProgramRadio()
            print("Volume increased to", fm_radio.Volume)
        else:
            print("Volume already at max (15)")
        time.sleep(0.5)  # debounce
    last_button_state = current_state


# --- Main interactive loop ---
while True:
    print("\nECE 299 FM Radio Demo Menu")
    print("1 - change radio frequency")
    print("2 - change volume level")
    print("3 - mute audio")
    print("4 - read current settings")
    print("Enter menu number > ", end="")

    user_input = ""
    while True:
        poll_button(fm_radio)

        ready = select.select([sys.stdin], [], [], 0)
        if ready and sys.stdin in ready[0]:
            user_input = sys.stdin.readline().strip()
            break

```

```
130         time.sleep(0.05)
131
132     select_option = user_input
133
134     # --- Handle Menu Selection ---
135     if select_option == "1":
136         Frequency = input("Enter frequency in MHz (e.g., 100.3) > ")
137         if fm_radio.SetFrequency(Frequency):
138             fm_radio.ProgramRadio()
139         else:
140             print("Invalid frequency")
141
142     elif select_option == "2":
143         Volume = input("Enter volume level (0 to 15) > ")
144         if fm_radio.SetVolume(Volume):
145             fm_radio.ProgramRadio()
146         else:
147             print("Invalid volume")
148
149     elif select_option == "3":
150         Mute = input("Enter mute (1 for Mute, 0 for Audio) > ")
151         if fm_radio.SetMute(Mute):
152             fm_radio.ProgramRadio()
153         else:
154             print("Invalid mute setting")
155
156     elif select_option == "4":
157         Settings = fm_radio.GetSettings()
158         print("\nRadio Status")
159         print("Mute:", "enabled" if Settings[0] else "disabled")
160         print("Volume:", Settings[1])
161         print("Frequency: %5.1f MHz" % Settings[2])
162         print("Mode:", "stereo" if Settings[3] else "mono")
163     else:
164         print("Invalid option")
165
166     time.sleep(0.05)
```