displaying LED pattern

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
led_pins = [11, 13, 15]

for pin in led_pins:
    GPIO.setup(pin, GPIO.OUT)

patterns = [
    [1, 0, 0],
    [0, 1, 0],
    [0, 0, 1],
    [1, 1, 0],
    [0, 1, 1],
    [1, 0, 1],
    [1, 1, 1],
    [0, 0, 0]
]

try:
    while True:
        for pattern in patterns:
            for i, pin in enumerate(led_pins):
                GPIO.output(pin, pattern[i])
            time.sleep(1)
except KeyboardInterrupt:
    GPIO.cleanup()
```

Display time over 4 digit 7 segment display

```
from datetime import datetime
import TM1637
import time

# Initialize the TM1637 display
tm = TM1637.TM1637(clk=23, dio=24)  # Replace these pin numbers with your actual connections

# Set brightness (optional)
tm.brightness(1)  # You can adjust brightness from 0 to 7

try:
    while True:
        now = datetime.now()
        current_time = now.strftime("%H%M")  # Get hours and minutes in HHMM format

        # Display time on the 4-digit 7-segment display
        tm.write([int(current_time[0]), int(current_time[1]), int(current_time[2]),
int(current_time[3])])
```

```
        time.sleep(1)  # Update every second
except KeyboardInterrupt:
    tm.cleanup()
```

---

capturing image with raspberry pi and pi camera

```
import picamera
import time

# Initialize the camera
camera = picamera.PiCamera()

try:
    # Start preview (optional)
    camera.start_preview()
    time.sleep(2)  # Allow the camera to adjust to light levels

    # Capture an image
    camera.capture('image.jpg')  # Save the captured image as 'image.jpg'

    # Stop preview (optional)
    camera.stop_preview()
finally:
    # Close the camera to release resources
    camera.close()
```

Oscilloscope

```
sudo pip3 install matplotlib
sudo pip3 install Adafruit-ADS1x15

import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import Adafruit_ADS1x15

# Create an ADS1115 ADC (16-bit) instance.
adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
val = []

# Start continuous ADC conversions on channel 0 using the previous gain value.
adc.start_adc(0, gain=GAIN)
print('Reading ADS1x15 channel 0')

fig, ax = plt.subplots()
ax.set_ylim(-5000, 5000)
ax.set_title('Oscilloscope')
ax.grid(True)
ax.set_ylabel('ADC outputs')
line, = ax.plot([], 'ro-', label='Channel 0')
```

```
ax.legend(loc='lower right')

def update(cnt):
    # Read the last ADC conversion value and print it out.
    value = adc.get_last_result()
    print('Channel 0: {0}'.format(value))

    # Set new data to line
    line.set_data(list(range(len(val))), val)
    ax.relim()
    ax.autoscale_view()

    # Store values for later
    val.append(int(value))
    if cnt > 50:
        val.pop(0)

ani = FuncAnimation(fig, update, interval=500)
plt.show()
```