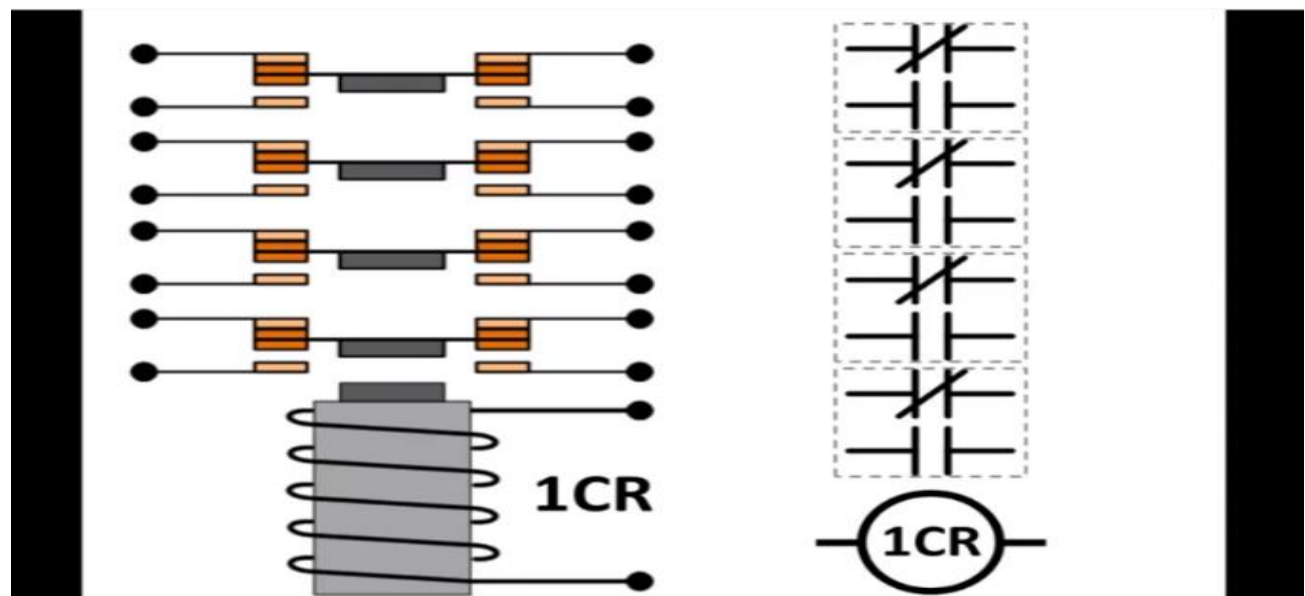
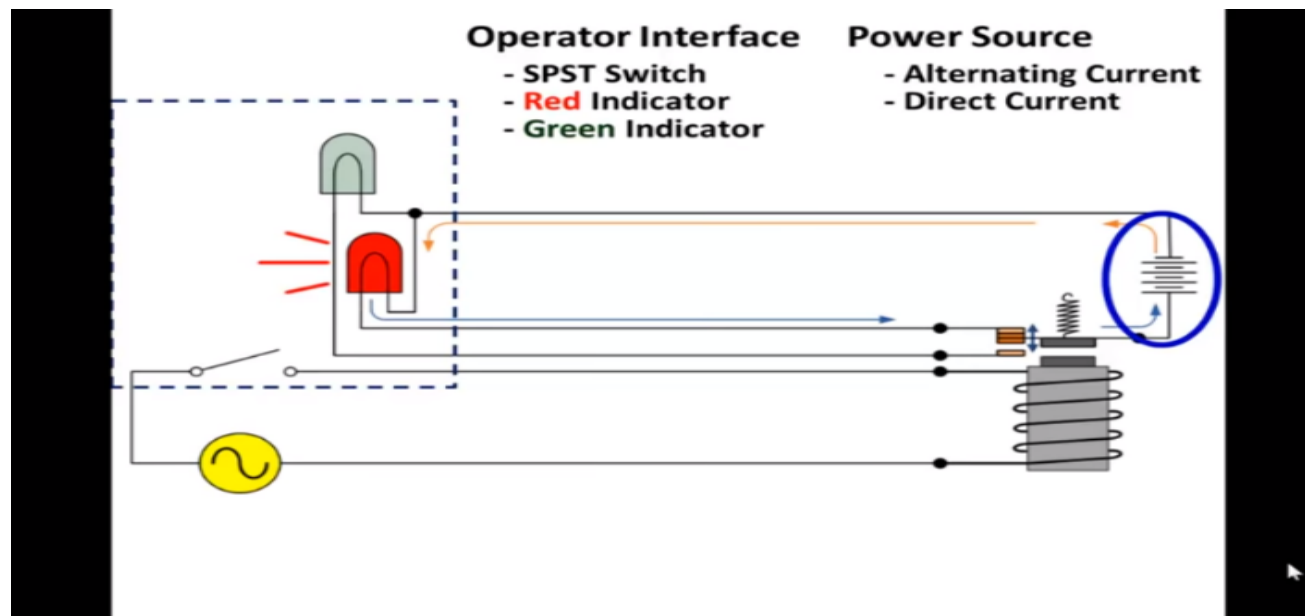


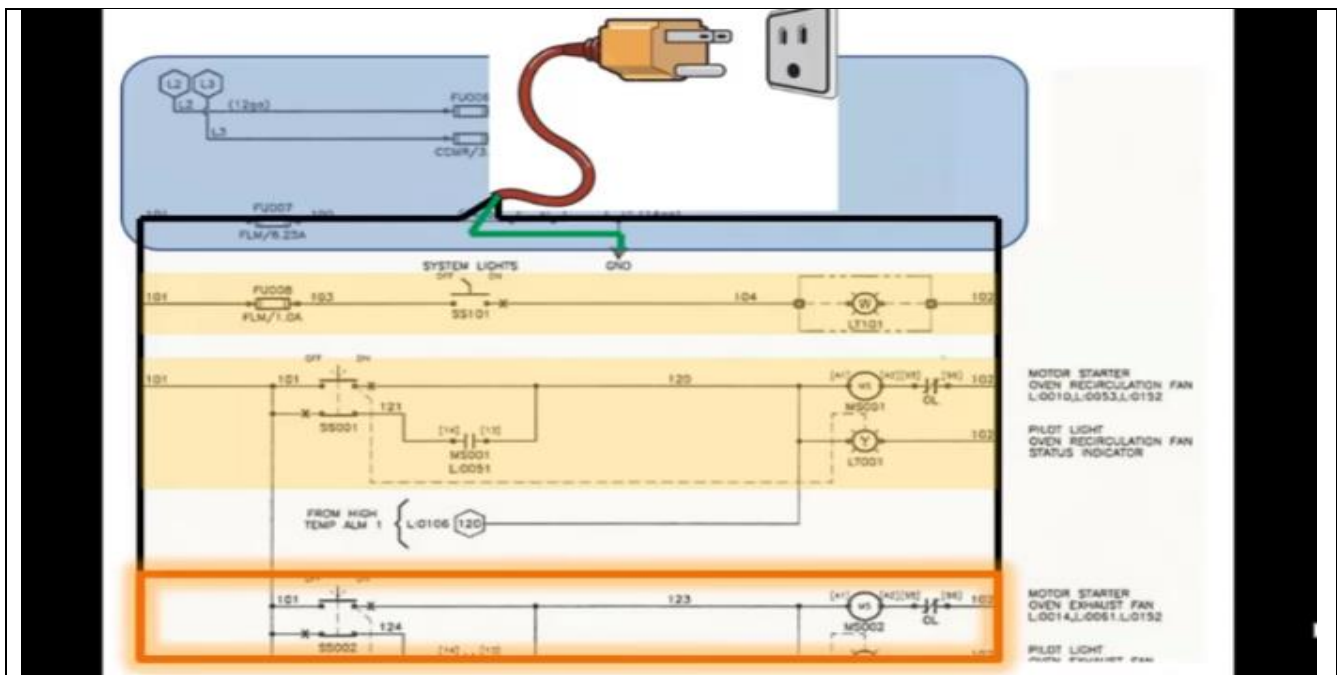
## DAILY ASSESSMENT FORMAT

Date:	30/5/2020	Name:	Shilpa S
Course:	Logic design	USN:	4AL14EC078
Topic:	Application of programmable logic controllers	Semester & Section:	8 <sup>th</sup> sem A sec
Github Repository:	Shilpa_online		

### FORENOON SESSION DETAILS

Image of session





**Report –**

## **PLC Analog I/O and Network I/O:**

In the early days of programmable logic controllers, processor speed and memory were too limited to support anything but discrete (on/off) control functions. Consequently, the only I/O capability found on early PLCs were discrete in nature. Modern PLC technology, though, is powerful enough to support the measurement, processing, and output of analog (continuously variable) signals.

All PLCs are digital devices at heart. Thus, in order to interface with an analog sensor or control device, some “translation” is necessary between the analog and digital worlds. Inside every analog input module is an ADC, or Analog-to-Digital Converter, circuit designed to convert an analog electrical signal into a multi-bit binary word. Conversely, every analog output module contains a DAC, or Digital-to-Analog Converter, circuit to convert the PLC’s digital command words into analog electrical quantities.

Analog I/O is commonly available for modular PLCs for many different analog signal types, including:

- Voltage (0 to 10 volt, 0 to 5 volt)
- Current (0 to 20 mA, 4 to 20 mA)
- Thermocouple (millivoltage)
- RTD
- Strain gauge

### **PLC Analog I/O**

The following photographs show two analog I/O cards for an Allen-Bradley SLC 500 modular PLC system, an analog input card and an analog output card.

### **PLC Network I/O**

Many different digital network standards exist for PLCs to communicate with, from PLC to PLC and between PLCs and field devices. One of the earliest digital protocols developed for PLC communication was Modbus, originally for the Modicon brand of PLC.

Modbus was adopted by other PLC and industrial device manufacturers as a de facto standard, and remains perhaps the most universal digital protocol available for industrial digital devices today. Another digital network standard developed by a particular manufacturer and later adopted as a de facto standard is Profibus, originally developed by Siemens.

A Programmable Logic Controller, also called a PLC or programmable controller, is a computer-type device used to control equipment in an industrial facility.

The kinds of equipment that PLCs can control are as varied as industrial facilities themselves. Utility Plants, Batch Control Application, Chemical Processing, Conveyor

systems, food processing machinery, auto assembly lines etc...you name it and there's probably a PLC out there controlling it.

## **PLC Advantages**

In addition to the programming flexibility we just mentioned, PLCs offer other advantages over traditional control systems.

These advantages include:

- high reliability
- small space requirements
- computing capabilities
- reduced costs
- ability to withstand harsh environments
- expandability

A PLC basically consists of two elements:

1. CENTRAL PROCESSING UNIT
2. INPUT/OUTPUT SYSTEM

### **The Central Processing Unit**

The central processing unit (CPU) is the part of a programmable controller that retrieves, decodes, stores, and processes information. It also executes the control program stored in the PLC's memory. In essence, the CPU is the "brains" of a programmable controller.

It functions much the same way the CPU of a regular computer does, except that it uses special instructions and coding to perform its functions.

The CPU has three parts:

- the processor
- the memory system
- the power supply

The processor is the section of the CPU that codes, decodes, and computes data.

Memory system is the section of the CPU that stores both the control program and data from the equipment connected to the PLC. Power supply is the section that provides the PLC with the voltage and current it needs to operate.

### **The Input/Output System**

The input/output (I/O) system is the section of a PLC to which all of the field devices are connected. If the CPU can be thought of as the brains of a PLC, then the I/O system can be thought of as the arms and legs. The I/O system is what actually physically carries out the control commands from the program stored in the PLC's memory.

The I/O system consists of two main parts:

- the Rack
- I/O modules

The rack is an enclosure with slots in it that is connected to the CPU. I/O modules are devices with connection terminals to which the field devices are wired. Together, the rack and the I/O modules form the interface between the field devices and the PLC.

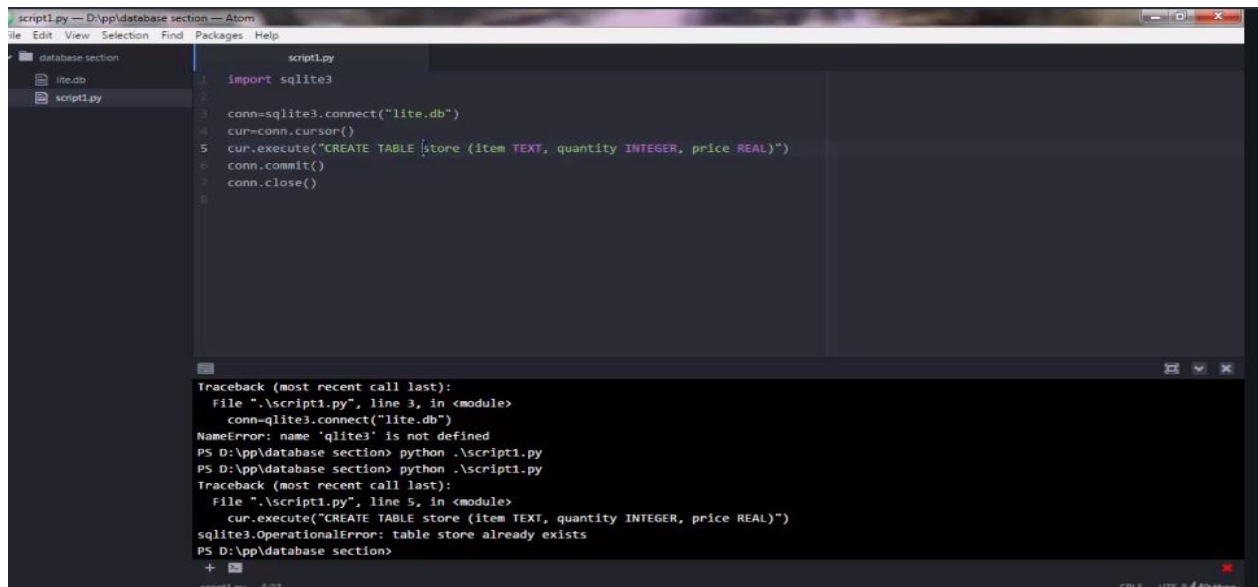
When set up properly, each I/O module is both securely wired to its corresponding field devices and securely installed in a slot in the rack. This creates the physical connection between the field equipment and the PLC. In some small PLCs, the rack and the I/O modules come prepackaged as one unit.

Date:30/5/2020  
Course: Python  
Topic: Python for image and video  
processing with OpenCV

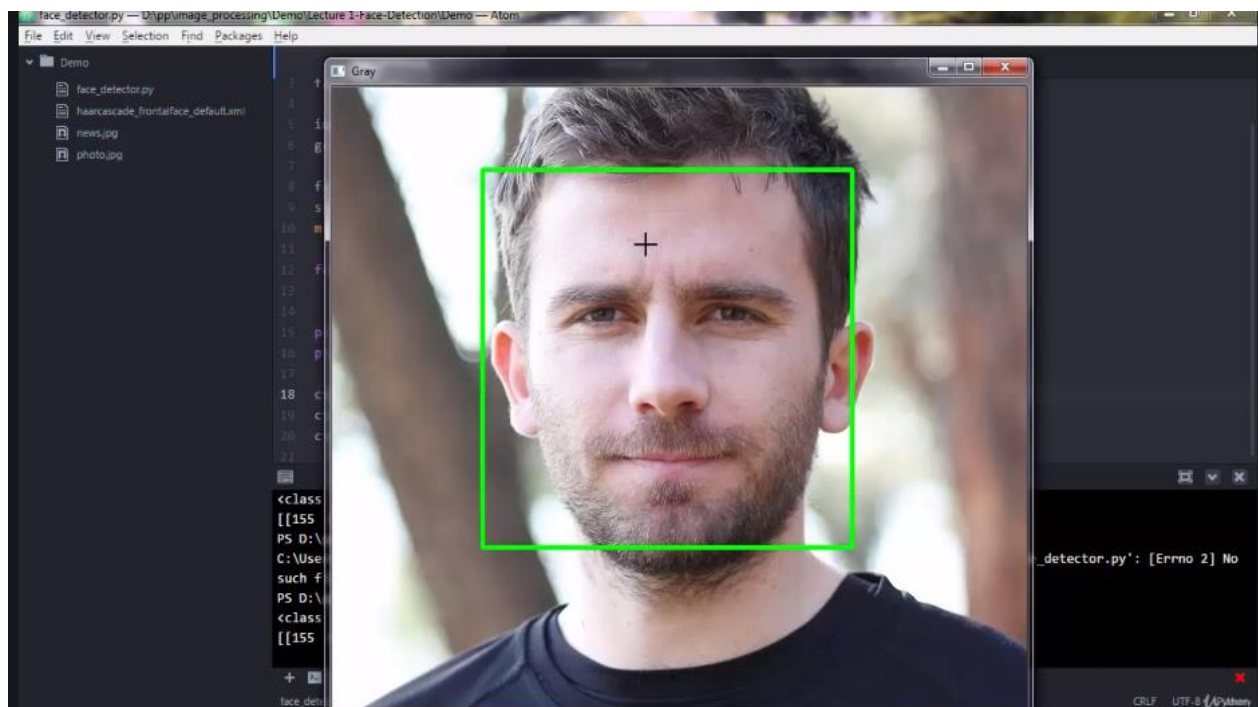
Name: Shilpa S  
USN:4AL14EC078  
Sem :8<sup>th</sup> sem  
Section: A sec

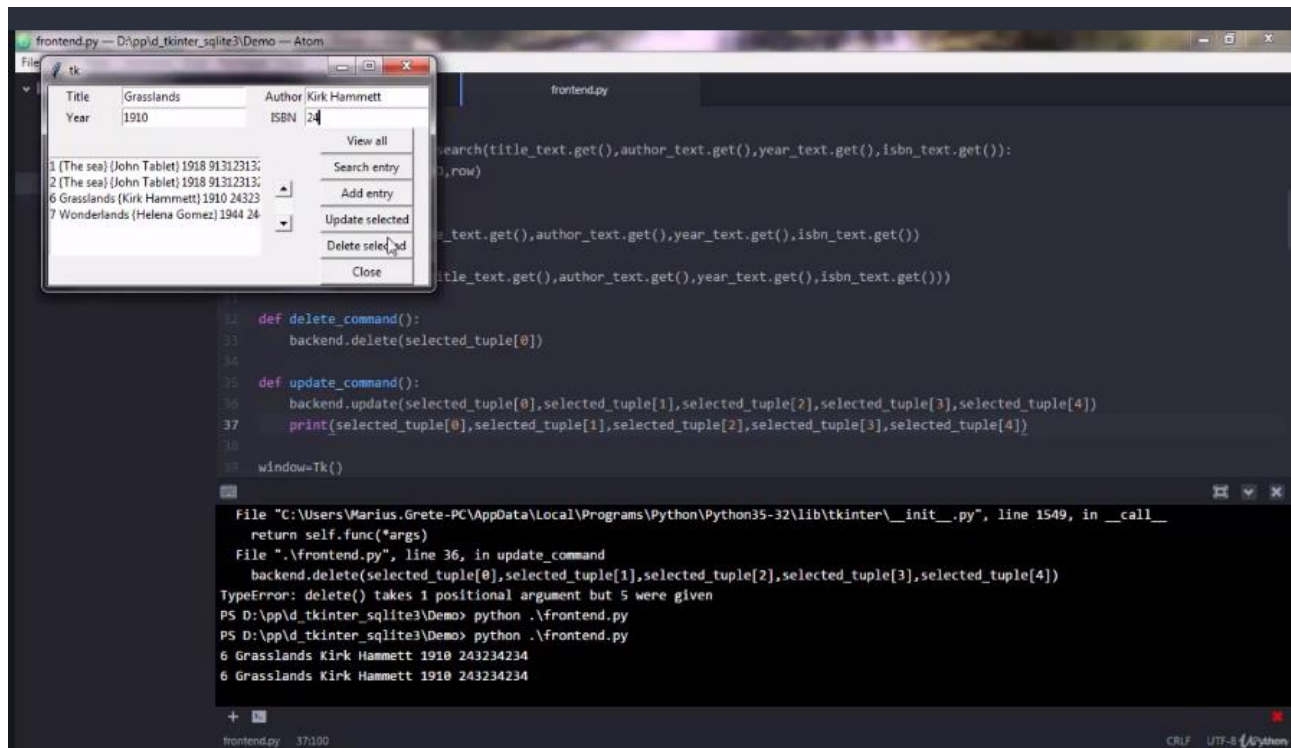
### AFTERNOON SESSION DETAILS

#### SESSION IMAGE :



```
script1.py — D:\pp\database section — Atom
File Edit View Selection Find Packages Help
database section
lite.db
script1.py
script1.py
1 import sqlite3
2
3 conn=sqlite3.connect("lite.db")
4 cur=conn.cursor()
5 cur.execute("CREATE TABLE store (item TEXT, quantity INTEGER, price REAL)")
6 conn.commit()
7 conn.close()
8
Traceback (most recent call last):
  File ".\script1.py", line 3, in <module>
    conn=qlite3.connect("lite.db")
NameError: name 'qlite3' is not defined
PS D:\pp\database section> python .\script1.py
PS D:\pp\database section> python .\script1.py
Traceback (most recent call last):
  File ".\script1.py", line 5, in <module>
    cur.execute("CREATE TABLE store (item TEXT, quantity INTEGER, price REAL)")
sqlite3.OperationalError: table store already exists
PS D:\pp\database section>
```





## REPORT:

### Installing the Library

For installing OpenCV, please do the following the instructions below. If you don't know if you have OpenCV, please open Python and type `import cv2`. If you don't get an error, it means OpenCV is installed.

### To install:

1. Open the command line and type:

```
pip install opencv-python
```

2. Then open a Python session and try:

```
import cv2
```

3. If you get no errors, that means you installed OpenCV successfully. If you get an error please see the FAQs below:

## **FAQs**

### **1. My opencv installation didn't go well on Windows**

Solution:

1. Uninstall opencv with:

```
pip uninstall opencv-python
```

2. Download a wheel (.whl) file from this link and install it with pip. Make sure you download the correct file for your Windows version and your Python version. For example, for Python 3.6 on Windows 64-bit you would do this:

```
pip install opencv_python-3.2.0-cp36-cp36m-win_amd64.whl
```

3. Then try to import cv2 in Python again. If there's still an error, then please type the following again in the command line:

```
pip install opencv-python
```

4. Now you should successfully import cv2 in Python.

### **2. My opencv installation didn't go well on Mac**

Solution:

If pip install opencv-python didn't go well please install OpenCV for Python 2 and use Python 2 to run the programs that contains cv2 code. Its' worth mentioning that Python



2 is installed by default on Mac, so no need to install Python 2. Here are the steps to correctly install OpenCV:

### 1. Install brew:

Open your terminal and execute the following:

```
/usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

2. OpenCV depends on GTK+, so please install that dependency first with brew (always from the terminal):

```
brew install gtk+
```

### 3. Install OpenCV with brew:

```
brew install opencv
```

### 4. Open Python 2 by typing:

```
python
```

### 5. Import cv2 in Python:

```
import cv2
```

If you get no errors, that means you installed OpenCV successfully.

## 3. My opencv installation didn't go well on Linux

1. Please open your terminal and execute the following commands one by one:

```
sudo apt-get install libqt4-dev
```

```
cmake -D WITH_QT=ON ..
```

```
make
```

```
sudo make install
```

2. If that doesn't work, please execute this:

```
sudo apt-get install libopencv-*
```

3. Then install OpenCV with pip:

```
pip install opencv-python
```

4. Import cv2 in Python. If you get no errors, that means you installed OpenCV successfully.

### **Batch Image Resizing**

```
import cv2
```

```
import glob
```

```
images=glob.glob("*.jpg")
```

```
for image in images:
```

```
    img=cv2.imread(image,0)
```

```
    re=cv2.resize(img,(100,100))
```

```
    cv2.imshow("Hey",re)
```

```
    cv2.waitKey(500)
```

```
cv2.destroyAllWindows()
```

```
cv2.imwrite("resized_"+image,re)
```

I first created a list containing the image file paths and then iterated through the aforementioned list.

The loop: reads each image, resizes, and displays the image; waits for the user input key, closes the window once the key is pressed, and writes the resized image. The name of the resized image will be "resized" plus the existing file name of the original image.