Welcome to:

# UNIT 5 -Python in IoT

# Unit objectives



**After completing this unit, you should be able to know:**

- How to connect hardware devices and sensors to the internet using Python libraries. (also called as IOT)
- Write basic Client Publish and Subscribe calls using the Paho-MQTT library and how to implement them.
- Numpy which is a Numerical Scientific module allowing faster computations and ability to do mathematical calculations like Linear Algebra, Matrix manipulations etc.
- MySQL -Python Modules which help in interfacing to the database, and how to create, insert and other Python MySQL module operations.
- OpenCV - Open Computer Vision library calls which helps in interpreting images such as photos, maps, videos.
- Matplotlib - A Python Module which allows developers to plot charts, lines, graphs.
- Pandas - A data manipulation module - and how to manipulate data, Create a 1 D, 2D and 3D data. This is an introduction to the module

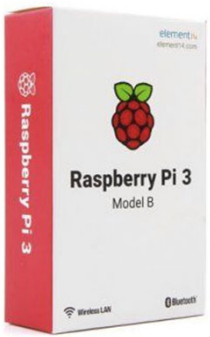
# Python in IoT

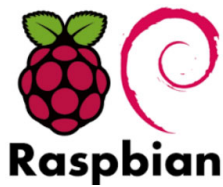


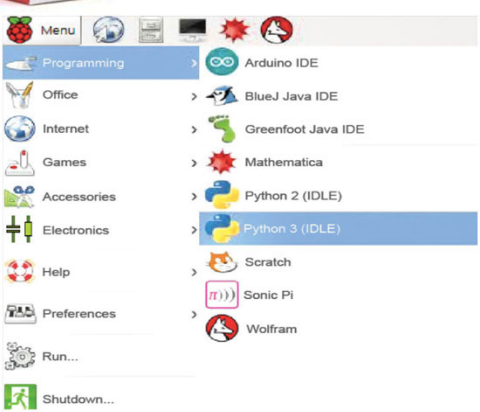
- Python helps in rapid development and deployment of applications that can connect to the sensors at one end
- Python connects to the cloud at the other end
- Most of the popular microcontrollers using MicroPython
- MicroPython is a lean and efficient implementation of the Python 3

# Python on Rasbian image






- Raspbian is a Debian-based computer operating system for Raspberry Pi

- Python is installed on the base Raspbian image
- The most common Python module is the RPI.GPIO module

Python on Raspberry can be used as

- a web server,
- developing utilities,
- connecting it to various sensors at one end
- connecting it to the Cloud at the other end

# RPI.GPIO module



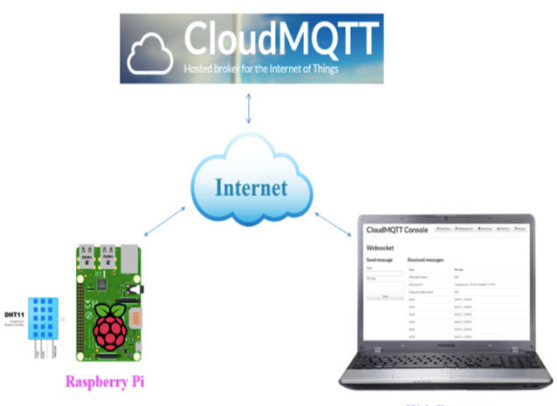
A Simple Python program to turn on and off a LED. Delay the LED on/off for a second.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)  # board mode is set to  Broadcom
GPIO.setup(18, GPIO.OUT)  # set up pin 18

while True:
    GPIO.output(18, HIGH)  # turn on pin 18
    time.sleep (5 ) # Sleep for 5 seconds
    GPIO.output(18,LOW)  #turn off pin
```

# Paho -MQTT



- MQTT (Message Queuing Telemetry Transport) is an ISO standard (ISO/IEC PRF 20922) publish-subscribe-based messaging protocol.
- The MQTT Broker is the centralised broker which can receive and send messages to /from the devices, acting as a broker




- **pip3 install paho-mqtt**
- **Raspberry Pi reads a temperature from a sensor**
- **Python publishes to the MQTT Broker in the cloud**

# Python code using MQTT libraries



- Python code using MQTT libraries and can be used to publish a simulate room temperature to cloudmqtt.com MQTT broker
- Python Client code using the MQTT library, ( runs on the Raspberry Pi)
- Raspberry Pi is connected to the Internet.

# Client Code: Publish and Subscribe



```
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, flags, rc):
    print("Connected with result code
"+str(rc))

client.subscribe("temperature/#")

def on_message(client, userdata, msg):
    print (" Received a message")
 print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
```

```
client.username_pw_set("ox
efqvkn", "uTM7RdarxTPA")
client.connect("m12.cloudm
qtt.com", 19757, 60)
print ("Publishing")

client.publish("temperature/
room", 28)

print ("waiting for incoming
connects from broker ")

client.loop_forever()
```

# Some of the APIs explained in larger scope:



- client_id is an optional argument.

If there is no client id, then the mqtt code generates one automatically.

Some examples E.g:

- client = mqtt.client ()
- client = mqtt.client( " ConnectToCloud")
- clean session is a boolean flag, Can be set to True or False. If set to TRUE, then any information about this client will be lost when the client disconnects.
- If set to False, the client information will still be retained after the Client.disconnect() call.
- This can be restored during reinitialise() call.

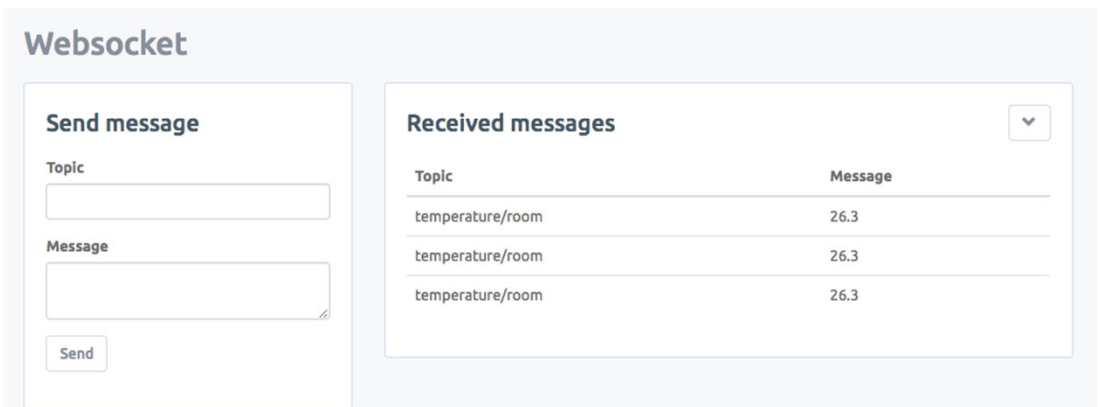
Userdata user defined data of any type that is passed as the userdata parameter to callbacks.

- Protocol is the version of the MQTT that will be used for this client.t.
- Valid flags are MQTTv31 or MQTTv311

# Output



Snapshot showing the messages being received at the CloudMQTT console for a different temperature in Centigrade



# Numpy



- The package is called Numpy ( short for Numerical Python)
- powerful and helps developers in analysis of data, has a lot of capabilities for numerical calculations, Fourier transforms.
- Numpy has  a N-Dimensional array object
- Numpy, helps in working with matrices, and maintains an efficient data structure.
- To install on a Linux/Ubuntu platform:
- sudo apt-get install python-numpy python-scipy python-matplotlib

# Working with Numpy



To start with this Numpy module, user has to import the module as shown into the Python code.

**sample_numpy.py**

```
import numpy as xy
x = "hello world"
print ( x )
```

**Some examples of how Numpy treats the arrays.**

Let's start with a 1 dimensional array and check that the array concept is still untouched.
This is a Python List ( or single dimensional array )

**simple_list.py**

```
list=[1,2,3,4,5]
print list
```

```
$ python3 first.py
[1, 2, 3, 4, 5]
```

# Working with Numpy Contd



**simple_numpylist.py**

```
import numpy as nptest
numpylist = nptest.array ([1,2,3,4,5])
print ( numpylist)
```

```
$ python3 simplenumpylist.py
[1 2 3 4 5]
```

So far, so good. but what about the shape of the array. It is a 1-dimensional array of 5 elements.

The “Shape” is a Numpy attribute that tells what array type this is.

**printshape.py**

```
import numpy as nptest
numpylist = nptest.array ([1,2,3,4,5])
print ( numpylist)
print ( numpylist.shape). # Notice we are printing shape.
```

**Resulting Output:**

```
[1 2 3 4 5]
(5,)
```

# Working with Numpy Contd



Now, for some more examples, where we have a single line but really representing a 2 d-array.

```
dlist=nptest.array( [1,2,3,4] )
print ( dlist.reshape(2,2))
```

**Resulting Output:** Note it printed rows X columns

```
[[1 2]
 [3 4]]
```

Another example of a 3d array, and rendered using the reshape.

```
dlist=nptest.array( [9,8,7,6,5,4,3,2,1] )
print ( dlist.reshape(3,3))
```

**Resulting Output:**

```
[[9 8 7]
 [6 5 4]
 [3 2 1]]
```

# Constructing Numpy arrays



The Numpy array constructor is created as follows :
**numpy.array(objectname, datatype = None, copy = True, order = None, subok = False, ndmin = 0)**

The first argument object is the array itself,
The datatype argument is (optional ) It is the data type of the array objects.
The data types are

- bool ( data is stored as a Byte - true or false )
- int_ ( data is stored as long )
- float
- complex

There are more variants to this.

copy = (optional ) if true then the object will be copied
order = C, or F. If "C" then column order.
subok = bool, subclasses will be passed thru
ndmin = specifies the minimum number of dimensions.
e.g: if '2' then it means the array is going to be a 2d array.

Reference :
https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.array.html#numpy.array
https://www.tutorialspoint.com/numpy/numpy_data_types.htm

# Constructing Numpy arrays Contd



**Examples of ndmin:**

If ndmin = 2, then a 2 X 2 matrix will be created. [ [ ...] ]
If ndmin = 3, then a 3 X 3 matrix will be created. [ [ [ ... ] ] ]
If admin = 4, then a 4 X 4 matrix will be created [ [ [ [ .... ] ] ] ]

**Examples of type**

If dtype = complex - the array will be treated as a complex elements.

```
dlist = nptest.array ( [ 4,5,6,7 ], dtype=complex )
print ( dlist )
```

**Resulting Output :**

```
[1.+0.j 2.+0.j 3.+0.j 4.+0.j]
```

If dtype = float - the array will be treated as a float elements.

```
dlist = nptest.array ( [ 1,2,3,4 ], dtype=float )
print ( dlist )
```

**Resulting Output :**

```
[1. 2. 3. 4.]
```

# Constructing Numpy arrays Contd



If dtype = bool - the array will be treated as a Boolean elements.

```
dlist = nptest.array ( [ 4,5,6,7 ], dtype=bool  )
print ( dlist )
```

**Resulting Output :**

```
[True  True  True  True]
```

Another example : ( Watch the last element, it is 0)

```
dlist = nptest.array ( [ 4,5,6,0 ], dtype=bool  )
print ( dlist )
```

**Resulting Output :**

```
[True  True  True  False]
```

Set up dtype to be a structure, and create an array of structures with different data types.

```
student = nptest.dtype([('student_name','S20'), ('student_age', 'i1'), ('student_marks', 'f4')])
s = nptest.array([('sam', 18, 80),('joe', 19, 75)], dtype = student)
print (s)
```

**Resulting Output :**

```
[(b'sam', 18, 80.) (b'joe', 19, 75.)]
```

# Printing arrays:



One-dimensional arrays are then printed as rows,
Bi dimensional as matrices in the form of rows X columns.
and Tridimensionals are printed as lists of matrices.

Let's create a small 1d array.
Small program to create 4 elements in an array

```
x = nptest.arange(4)
print ( x)
```

**Resulting Output :**

```
[0 1 2 3]
```

**Printing a 2d array, Note it prints a 4X3 matrix**

```
x = nptest.arange(12)
ret= x.reshape(4,3)
print ( ret )
```

**Resulting Output :**

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

# Program



## Printing 4 arrays of 3 rows and 2 columns

```
x = nptest.arange(24)
ret= x.reshape(4,3,2)
print ( ret )
```

**Resulting Output :**

```
[[[ 0  1]
  [ 2  3]
  [ 4  5]]

 [[ 6  7]
  [ 8  9]
  [10 11]]

 [[12 13]
  [14 15]
  [16 17]]

 [[18 19]
  [20 21]
  [22 23]]]
```

# Arithmetic Operations on matrix's



**Addition :**

```
a = nptest.array( [8,6,4,2] )
b = nptest.arange( 4 )
print ("a= ", a)
print ("b= ", b)
c = a + b
print ( "c = a + b ", c)
```

**Resulting Output :**

```
a=  [8 6 4 2]
b=  [0 1 2 3]
c = a + b  [8 7 6 5]
```

Note :Subtraction, Multiplication and Square roots are calculated similarly

# numpy.zeros( )



To create an array of 5 elements and fill it with zeroes

numpy.zeros(shape, datatype=float, order='C')

Arguments are :
**shape** is the array,
**datatype** is the array ( it is optional )
**order** indicates it is row first or column first. ( Optional )
Returns the array, filled with zeros.

**Sample Code :**
nptest.zeros ( 6 )

**Resulting Output :**
[0. 0. 0. 0. 0. 0.]

# numpy.zeros( )



**Getting count of elements in the array**

```
narr = ([ 1, 2, 3,4 , 5)]
print (narr)
print ( len ( narr))
```

**Convert arrays to lists**

```
narray = np.array( [1,2] )
print ( narray)

nlist =  narray.tolist()
print ( "Printing the count", len(nlist) )

for item in nlist:
    print (item)
```

**Initialise the array with a certain value, here it is set to 99.**

```
import numpy as np

narray = np.array([9,8])
nlist =  narray.tolist()

i=0
while ( i < len(nlist )) :
    nlist[i] = 99
    i=i+1
```

# Slicing the arrays



It follows the array[ start: stop: increment ] format.
The start and stop are where the element should start, and the stop is just one less than the STOP.
Step is the increment (the element is “including” the step count.

If no step is given , it defaults to every element from START to STOP.

x = np.array [ 1,2,3,4,5,6,7,8,9,0 ]

print ( x[1:9:2 ]) = start from Start, exclude it though, and end at 9 and step ( or jump ) 2 steps.

So, the answer is : 2,4,6,8

# Random number generation



np.random.rand(m,n)

Creates a matrix of m rows X n columns matrix with random numbers.

```
[[0.93321914 0.85861314 0.73888415]
 [0.77566318 0.42627905 0.32774669]]
```

**Generate random number.**

numpy.random.randint(*low*, *high=None*, *size=None*, *dtype='l'*)

Return random numbers as integers. ( A single number is generated if size is not specified, or a matrix of numbers is generated if size is set )
Return random numbers ( integers) from low to high ( both are inclusive )
If high argument set to None, then results can be from 0, to low.

# Load data from a text file



This call loads the external file and returns an object.

numpy.loadtxt (fname, dtype=<type
'float'>, comments='#', delimiter=None, converters=None, skiprows=0, usecols=None, unp
ack=False, ndmin=0, encoding='bytes')

Sample program:

```
# Load CSV
import numpy
filename = 'tesdb.csv'
raw_data = open(filename, 'rt')
data = numpy.loadtxt(raw_data, delimiter=",")
print(data.shape)
```

Reference:

https://docs.scipy.org/doc/numpy/reference/index.html

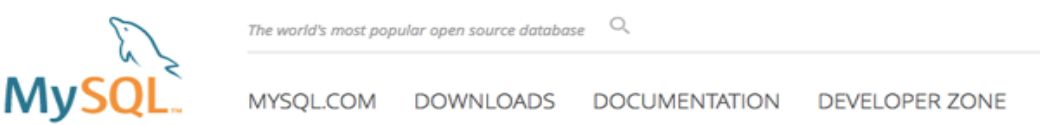
# Mysqldb



- Pymysql is the Python Module interface for connecting the Python code to the Database
- Installation of MySQL:
- Login to https://www.mysql.com/downloads/ and go to “Trial downloads”.

**MySQL Server 5.7.21**

Download this and install it on your computer.
Follow the installation instructions on
https://dev.mysql.com/doc/refman/5.7/en/installing.html



# Mysqldb Contd



After installation, start the MySQL server as described in the documentation.


Check that the default databases are there before proceeding.

# Installation of Python Interface module



Many databases are supported by Python, however a separate module to support that module has to be installed.
PyMySQL is a third party module, that has to be installed separately.
To install it:

**pip3 install PyMySQL**

To include that in to the Python code.

**import pymsql**

Save and run this above program once. If it fails that means the module is not properly installed.
This has to be resolved before proceeding.
Install the appropriate package for your platform, validate and then proceed to below steps.

# Creating a database



```
mysql> create database testdb;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
| testdb             |
+--------------------+
6 rows in set (0.00 sec)
```

# Tested Database



## Create a table called “employee” in the tested database

Instructions to create are listed in this MySQL reference manual
https://dev.mysql.com/doc/refman/5.7/en/creating-tables.html

Get into the employee database space,

use employee,
Create a table as shown:

```
CREATE TABLE employee (firstname VARCHAR(20), lastname VARCHAR(20), age
INTEGER, sex CHAR(1), birth DATE);
```

Check that the table is created right.

```
mysql> describe employee;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| firstname | varchar(20) | YES  |     | NULL    |       |
| lastname  | varchar(20) | YES  |     | NULL    |       |
| age       | int(11)     | YES  |     | NULL    |       |
| sex       | char(1)     | YES  |     | NULL    |       |
| birth     | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

# Python : PyMySQL module



- creating a table using Python PyMySQL modules for managing “cars” in the “testdb” database in the Python code.

Code to create the table.

```
# Create table as per requirement
CREATE TABLE car (
        name CHAR(20) NOT NULL,
        model CHAR(20),
        introduced_in DATE,
        sales INTEGER )
```

Execute it thru the cursor pointer.

Now, login back to the MySQL command line interface, go to the “testdb”, and run the “describe car” command.

```
mysql> describe car;
+---------------+----------+------+-----+---------+-------+
| Field         | Type     | Null | Key | Default | Extra |
+---------------+----------+------+-----+---------+-------+
| name          | char(20) | NO   |     | NULL    |       |
| model         | char(20) | YES  |     | NULL    |       |
| introduced_in | date     | YES  |     | NULL    |       |
| sales         | int(11)  | YES  |     | NULL    |       |
+---------------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

# Python : PyMySQL module Contd



**To print results of the mysql execute commands on the screen:**

Use the fetchall () command.

```
a = cursor.fetchall()
print ( a )
```

**Example:**

```
import pymysql

# Open database connection
db = pymysql.connect("localhost","r
# disconnect from server
db.close()
db" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

sql = """show databases """

cursor.execute(sql)

a = cursor.fetchall()
print (a)
```

# Exception handling in pymysql



**Some of the exceptions that can arise are :**

**DatabaseError -**Occurs where there are database errors. Must subclass Error.

**DataError -** Occurs where the is errors in data. Must Subclass DatabaseError

**Error-**This is the Base class for all pymsql errors. Must subclass StandardError.

**IntegrityError-**This happens where there is an issue with integrity of data, Subclass of Database Error

**InterfaceError-**This happens when the Python mysql module has some errors! Must subclass Error.

**InternalError-**Occurs when there is an instance of cursor not being active and related errors. Must subclass DatabaseError.

**NotSupportedError-**Happens when we are attempting some unsupported calls. Must subclass DatabaseError

**OperationalError-**This can happen where the connection is lost to the database. Subclass of DatabaseError.

**ProgrammingError-**Happens when there is a bad table name given during the sql statement execution. Subclass of DatabaseError.

**Warning-**Used for non-fatal issues. Must subclass StandardError.

# Example



**Example of Error Exception Handling in pymysql**

```
import pymysql
try :
      # Open database connection
      db = pymysql.connect("localhost","root","root124","testdb"  )

except pymysql.Error as err :
      print ( err )
      quit()
```

**Resulting Output :**

```
$ python3 createdb.py
(1045, "Access denied for user 'root'@'localhost' (using password: YES)")
```

# OpenCV



- Open CV is a “Open Source Computer Vision” project which works on computer vision
- Library is used for with real-time applications, and can identify pictures and validate the authenticity.
- Module is written in C/C++

# Open CV - Installation



To install on other platforms : ( this can be used on Mac OS as well )

pip3 install opencv-python

```
$ pip3 install opencv-python
Collecting opencv-python
  Downloading opencv_python-3.4.0.12-cp36-cp36m-
macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x86_64.macosx_10_10_intel.maco
sx_10_10_x86_64.whl (41.2MB)
    100% |                                        | 41.2MB 31kB/s
Requirement already satisfied: numpy>=1.11.1 in
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages (from
opencv-python)
Installing collected packages: opencv-python
Successfully installed opencv-python-3.4.0.12
```

# Sample code – Open CV



Sample code :

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load an color image in grayscale
img = cv2.imread('watch.png',cv2.IMREAD_GRAYSCALE)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

**Read Function:**

cv2.imread()

The cv2.imread ()function will read the input file ( jpg or png) and the second argument lists how the image has to be read in.

The various modes are :

cv2.IMREAD_COLOR : Loads a colour image. Transparency is ignored. This is the
cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode
cv2.IMREAD_UNCHANGED : Loads image as such including alpha channel

# Matplotlib



- This is a Python library to plot graphs and bar charts, histograms, scatterplots, and various other types of graphical representations
- The code is simple, easy to develop and can run on multiple platforms. Users can embed images as well.
- Pyplot -library which allows users to program bars and charts
- Additional reference :
- https://matplotlib.org/api/pyplot_summary.html

# Matplotlib-Installation & Sample code

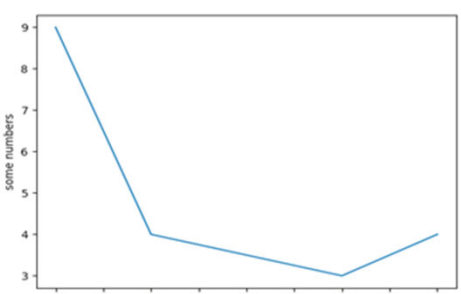


**Simple example code of charting a single line**

```
import matplotlib.pyplot as plt
plt.plot([1,2,4,5], [9,4,3,4], )
plt.ylabel('some numbers')
plt.show()
```

```
pip3 install matplotlib
```

Include this module into the Python code.

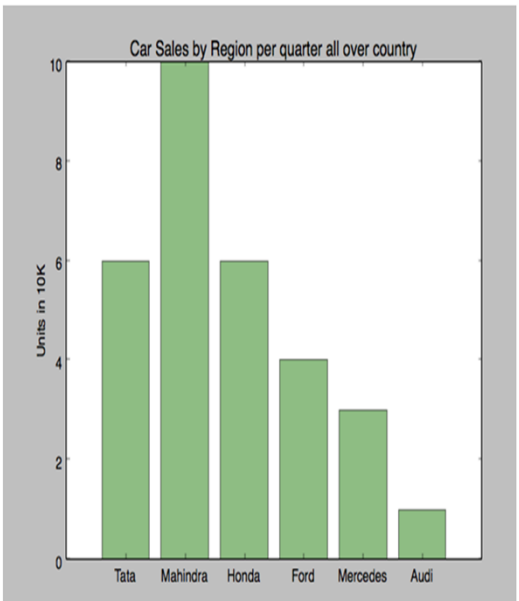


# Sample code – Bar Chart



```
import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt

objects = ('Tata', 'Mahindra', 'Honda', 'Ford', 'Mercedes', 'Audi')
y_pos = np.arange(len(objects))
performance = [6,10,6,4,3,1]

plt.bar(y_pos, performance, align='center', alpha=0.5, color='g')
plt.xticks(y_pos, objects)
plt.ylabel('Units in 10K')
plt.title('Car Sales by Region per quarter all over country')

plt.show()
```

**Resulting Output:**



# Pandas : Data Processing Module



- Pandas is a Python Module tool for data manipulation, processing and analysis
- It can handle and manage complex data structures, and also manage time series data
- Integrated with Numpy and other modules
- Pandas  is an Open Source tool, and binaries can be downloaded

# Pandas-Installation



```
pip3 install pandas
...
Installing collected packages: pandas
Successfully installed pandas-0.22.0
```

**The Home page of Pandas is https://pandas.pydata.org/**

# Pandas-Data structure



- The main data structures that Pandas can handle are Series (1-d), Data Frame ( 2-d ), Panel ( 3D ) 3D labelled
- This is built on top of Numpy and hence data can be easily manipulated.
- Series is 1Dimensional, homogenous array, it can handle Integer, string, float or python objects.
- Some of the advantages of this package is listed in the PyPi project page. Audience is encouraged to visit this page. https://pypi.org/project/pandas/

# Pandas-Data structure Contd



Some of the advantages of this package is listed in the PyPi project page.
Audience is encouraged to visit this page. https://pypi.org/project/pandas/

Data Frames are 2d , tabular structure like SQL databases,

Panel is 3D size - mutable array.

Some examples of Series are :

[ 12, 44, 78, 25,69, 29 ]

Some examples of tabular data are :

| Name | salary | Birthdate |
|---|---|---|
| John | 10,000. | 28 Mar 1950 |
| Joe. | 12,000. | 15. Mar. 1953 |

# Pandas-Series data



The Series data can handle integer, string, float, or python objects.

To construct a Series :

pandas.Series( data, index, dtype, copy)

data can be a ndarray, a list,strings, dict, as described above.

index has to be the length of the data, It defaults to np.arange(n) if nothing is passed.

dType is optional and if not passed, will be determined by the data. (accepted values are int, float, bool )

copy is the last argument, optional again. If passed, then data will be copied.
If nothing is passed, data will be not be copied.

# Pandas – Series data Sample Code



```
import pandas as pd

s = pd.Series([0,3,4])
print (s)
s = pd.Series([0,3,4],  dtype=float)
print (s)
s = pd.Series(["john",  "joe", "sam"] )
print (s)
```

**Resulting Output :**

```
0   0
1   3
2   4
dtype: int64
0   0.0
1   3.0
2   4.0
dtype: float64
0    john
1     joe
2     sam
dtype: object
```

# Example



An example involving dict as data.

```
import pandas as pd
import numpy as np
data = {'Jan' : 31, 'Feb' : 28, 'Mar' : 31}
s = pd.Series(data)
print (s)
```

Resulting Output:

```
Feb    28
Jan    31
Mar    31
dtype: int64
```

# Pandas-Data Frames



- The Data Frame is a 2D data structure as described earlier. Data is in tabular form like in a SQL database.
- Here the column sizes are not fixed, This data can be changed midway in the operation
- The Pandas tool can be used to create such a data from a structure as above. It is called DataFrame
- Pandas is a very intensive tool and there are many applications that can be developed

# Pandas-Data Frames Contd



A 2D can be thought of as follows.

| Name | Age | Address |
|---|---|---|
| John | 30 | Elbonia |

The Pandas tool can be used to create such a data from a structure as above. It is called DataFrame.

The construct is as follows:

pandas.DataFrame( data, index, columns, dtype, copy)

**data**
data can be a ndarray, series, map, lists, dict, constants or another DataFrame

**index** ( this is optional argument )
Default: If no index is passed then np.arange(n)is used.this is the size of the data.



**columns** ( optional )
For column labels, Defaults to np.arrange(n). This is only true if no index is passed.

**dtype** ( optional )
Data type of each column.

**copy** ( optional )
If this argument is set , then copies the data. Defaults to false.

# Pandas Basic functionality



Some of the Pandas Basic function calls are listed below :

**axes** This call returns a data list

**dtype** This call shows the data type of the Pandas Object.

**empty** indicates if the list is empty or not.

**ndim** returns the dimensions of data, Is it 1 dimension , 2 or 3 - default is 1.

**size** Returns the number of elements

**values** converts the series to a Numpy ndarray.

**head()** Returns the first element.

**tail()** Returns the last element. Option is there to show

# Example code of a 1-dimensional list



```
import pandas as pd
import numpy as np

#Create a series with 100 random numbers
s = pd.Series(np.random.randn(3))
print ( s )
print ("axes ", s.axes ).  # print. [RangeIndex(start=0, stop=3, step=1)]
print ("dtype ", s.dtype ).   # prints float64
print ("empty ", s.empty ). # prints false, since there are elements in it.
print ("dimension ", s.ndim ). # prints 1, as it is 1D
print ("size ", s.size ).  # 3 , as there are 3 elements .
print ("values ", s.values ).  # prints the data AS_IS
print ("head ", s.head ).     # prints the first element in the list
print ("tail ", s.tail ).    # prints the last element in the list.
```

# Checkpoint



1. What does the abbreviation GPIO stands for ?

2. How many GPIO pins does a Raspberry Pi have ?

3. What is the most popular OS installed on Raspberry Pi
   - Ubuntu
   - Windows
   - Raspbian.

4. The Pins can be either accessed as GPIO.setmode(GPIO.BCM) or
   - GPIO.BROAD
   - GPIO.BOARD.

# Checkpoint



5. To install GPIO module, what is module should be installed ?
   a) A.RPi.GPIO.
   b) B.GPIO_modules

6. When the Python MQTT module receives a message, What is the call back function “name” used ? Is it
   a) callback.connect_on
   b) Callback.connect()
   c) callback.on_connect.

7. Identify the paho mqtt client module during installation ?
   a) pip3 install pho
   b) pip3 install paho-mqtt
   c) pip3 install pahomqtt

8. The message that gets published to the cloud ? What format is the message that is published to the cloud in ?
   – A.topic/variable
   – B.topic
   – C.variable

# Checkpoint



9. What Paho MQTT library call gets called on incoming messages ?
   a) A.client.connect()
   b) B.client.on_message().
   c) C.client.loop ()

10.What is the default port used to connect ? if no port number is specified in the client. connect() call
   a) A.8080
   b) B.1883.
   c) C.241

# Unit summary



**Having completed this unit, you should be able to know:**

- How to connect hardware devices and sensors to the internet using Python libraries. (also called as IOT)
- Write basic Client Publish and Subscribe calls using the Paho-MQTT library and how to implement them.
- Numpy which is a Numerical Scientific module allowing faster computations and ability to do mathematical calculations like Linear Algebra, Matrix manipulations etc.
- MySQL -Python Modules which help in interfacing to the database, and how to create, insert and other Python MySQL module operations.
- OpenCV - Open Computer Vision library calls which helps in interpreting images such as photos, maps, videos.
- Matplotlib - A Python Module which allows developers to plot charts, lines, graphs.
- Pandas - A data manipulation module - and how to manipulate data, Create a 1 D, 2D and 3D data. This is an introduction to the module