become smarter and do more

1.1.1 What is IoT?

IoT contains things which have unique characters and being connected with the utilization of Internet. While many existing devices, for instance, PCs or 4G engaged mobile phones, are distinguished by their uniqueness and are connected to the Internet, IoT emphasize more on setup, control and frameworks organization by methods for the Internet of "Things" that are generally not related with the Internet. These incorporate devices, for example, Indoor Regulators, Utility Meters, a Bluetooth associated Headset, Water System Pumps and Sensors, or Control Circuits for an Electric autos motor. IoT is another change in the limits of the endpoints that are related with the Internet, and is being driven by the types of progress in capacities in sensor frameworks, PDAs, WSN's and cloud developments. Pros assume that by the year 2020 there will be a total of 50 billion devices/things which might be connected and communicated with the Internet. Hence, the Real business players are energized by the outline of new markets for their items. The items incorporate equipment and programming segments for IoT endpoints, centers, or control focuses of the IoT universe.

The degree of the IoT is not confined to just interfacing things (gadgets, contraptions, machines) to the Internet. IoT grants these things to connect and exchange information (control and information, that could fuse data related with customers) while executing huge applications towards an ordinary customer or machine objective. Information itself does not have an importance until the point when it is unified prepared into valuable data. Applications on IoT systems remove and make data from bringing down level information by separating, handling, ordering, gathering and contextualizing the information. This data acquired is then composed and organized to surmise information about the framework and additionally its clients, its condition, and its operations and advance towards its goals, permitting a more brilliant execution. **Figure 1-1** demonstrates the process of deriving knowledge and information from the data.

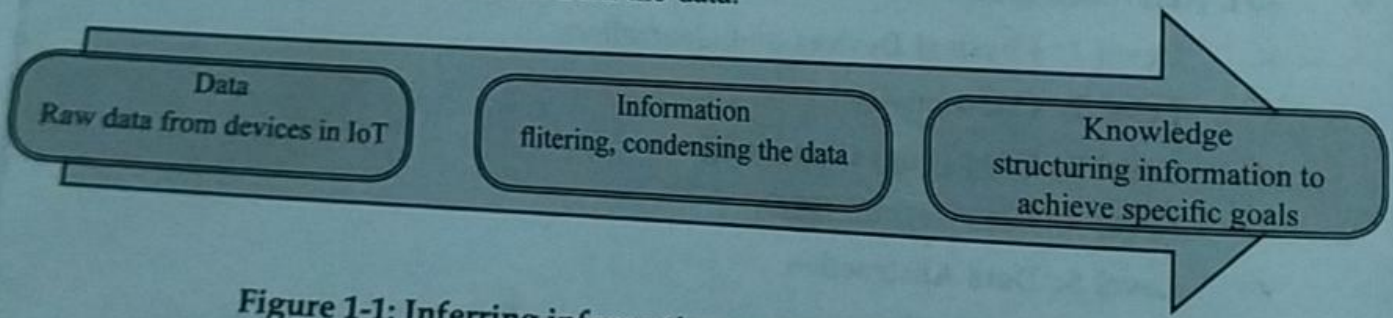


Figure 1-1: Inferring information and knowledge

1.1.3 Disambiguation of IoT vs IoE vs M2M vs others

"Internet of Things" is a prominent term in depicting this era of interconnected world. How do these terms identify with each other? What's more, how would they vary from the Internet of Things definition? How would the concepts do get differ in different coined terms from each other.

M2M

The term "Machine to Machine- (M2M)" being used for over 10 years, and is notable in the Telecom area. M2M correspondence had at first been a balanced association, connecting one machine to another. In any case, today's blast of portable availability implies that information can now be more effectively transmitted, through an orderly arrangement of IP systems, to a level where much more extensive scope of devices exist.

Industrial Internet of Things

This term modern web is unequivocally pushed by GE which goes past M2M since it mainly concentrates on associations among machines as well as incorporates human interventions.

Internet of Things (IoT)

IoT has yet a more extensive reach as it likewise incorporates associations past the modern setting for example, wearable gadgets on individuals.

Internet (as we know it)

the web is a genuinely a little box. In its center it associates just individuals.

Web of Things

The Web of Things is much smaller in extension as alternate ideas as it exclusively concentrates on programming design.

Internet of Everything (IoE)

It's Still a somewhat unclear idea, where IoE plans to incorporate a wide range of associations that one can imagine. The idea has consequently the most astounding scope.

Industry 4.0

Industry 4.0 that has been firmly pushed by the governing bodies from German to be restricted as the mechanical web in reach as it just spotlights on assembling situations. In any case, it has the biggest extent of the considerable number of ideas. Industry 4.0 portrays an arrangement of ideas to drive the following modern insurgency, that incorporates a wide range of availability ideas in the mechanical setting. In any case, it goes advance and incorporates genuine, adapting to the changes occurring in the physical world around us, for example, 3D-printing advances or the presentation of new enlarged reality equipment.

1.2.2.4 Communications

As for sending and receiving information, wired and wireless models are enhanced to such an exten that almost every kind of electronic hardware can give information since they are connected to th Internet. This has permitted the steadily contracting sensors implanted in smart items to send and receive information over the cloud for gathering, stockpiling and analysis in case of some events.

The conventions for enabling IoT sensors to hand-off information incorporate wireless innovations, for example, RFID, NFC, Wi-Fi, Bluetooth, Bluetooth Low Energy (BLE), XBee, ZigBee, Z-Wave, Wireless M-Bus, SIGFOX and NuelNET, and additionally satellite associations and versatile systems utilizing GSM, GPRS, 3G, LTE, or WiMAX. Wired conventions, useable by stationary keen articles, incorporate Ethernet, HomePlug, HomePNA, HomeGrid/G.hn and LonWorks, and ordinary phone lines.

1.2.2.5 Wireless Sensor Networks

A WSN contains distributed devices connected with sensors which are utilized to observe and monitor the ecological and physical conditions. A WSN comprise of various end-hubs and switches and a coordinator. End hubs have a few sensors appended to them. End hubs can be also served as switches. Switches are in charge of routing the information bundles from end-hubs to the coordinator. The coordinator gathers the information from every one of the hubs. Coordinator can also act as a facilitator/gateway that connects WSN to the web. A few cases of WSNs utilized as a part of IoT frameworks are portrayed as follows:

- Smart networks utilize WSNs for checking the matrix at different focuses.
- Indoor air quality checking frameworks utilize WSNs to gather information on the indoor air quality and convergence of different gasses.
- Soil dampness observing frameworks utilize WSNs to screen at different areas.
- Structural wellbeing observing frameworks utilize WSNs to screen the strength of (structures, spans) by gathering vibration information from sensor hubs conveyed at different focuses in the structure.
- Weather checking frameworks are WSNs in which the hubs gather temperature, stickiness and other information, which is accumulated and broke down.
- Surveillance frameworks utilize WSNs for gathering observation information, for example, movement recognition information)

1.2.2.6 Cloud computing

As "cloud" is frequently utilized as an analogy for the Internet, "Cloud computing" alludes to having the capacity to get access to processing/computing resources by means of the Internet, as opposed to traditional frameworks where figuring equipment is physically situated on the premises of the client and any product applications are introduced on such nearby equipment. All the more formally, "distributed computing" is characterized as:

" A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services)

that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Cloud computing – and three administration models of Software as a Service (SaaS) of it, Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) – are essential to the IoT in light of the fact that it permits any client with a program and the Internet to transform any of the smart object/device into an intelligent model performing relevant actions. The cloud computing provides *"the virtual infrastructure for utility of computing integrating applications, monitoring devices, storage devices, analytics tools, visualization platforms, and client delivery to enable businesses and users to access IoT-enabled applications on demand anytime, anyplace and anywhere."*

1.2.2.7 Digital Twin

Another result of the developing and advancing IoT is the idea of a "Digital twin," presented in 2003 by John Vickers, chief of NASA's National Center for Advanced Manufacturing. The idea alludes to a computerized duplicate of a physical resource (i.e., a savvy protest inside the IoT), that lives and advances in a virtual situation over the physical resource's lifetime. That is, as the sensors inside the question gather continuous information, an arrangement of models framing the advanced twin is refreshed with the greater part of a similar data. In this manner, an examination of the computerized twin would uncover an indistinguishable data from a physical review of the keen question itself – but remotely. The computerized twin of the shrewd protest would then be able to be concentrated to not just enhance operations of the keen question through decreased support expenses and downtime, however to enhance the up and coming era of its plan.

1.2.2.8 Big Data Analytics

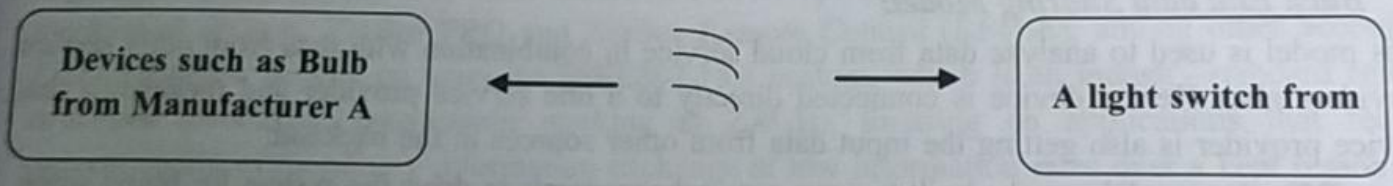
As more things (or "savvy objects") are associated with the IoT, more information is gathered from them keeping in mind the end goal to perform investigation to decide patterns and affiliations that prompt bits of knowledge. For instance, an oil very much furnished with 20-30 sensors can produce 500,000 information focuses each 15 seconds, a jetliner with 6,000 sensors creates 2.5 terabytes of information for every day, and the more than 46 million keen utility meters introduced in the U.S. produce more than 1 billion information focuses every day. In this way, the term "Big data" prompts to these extensive informational collections that should be gathered, questioned, stored and analyzed and by and large overseen keeping in mind the end goal to convey on the guarantee of the IoT — understanding!

Additionally exacerbating the specialized difficulties of enormous information is the way that IoT frameworks must manage the information gathered from smart objetcs, as well as subordinate information that is expected to appropriately perform such analytics (e.g., open and private informational collections identified with climate, GIS, money related, seismic, delineate, wrongdoing, and so on.). In this way, as more smart objects come on the web, no less than three measurements ("the three V's") are regularly utilized by IoT administrators to depict the enormous information they handle: Volume (i.e., the measure of information they gather from their IoT sensors measured in gigabytes, terabytes and petabytes); velocity (i.e., the speed at which information is gathered from the sensors); and assortment (i.e., the distinctive sorts of organized and unstructured

1.3.2.1 Communication model in IoT

1. Device to Device Communications:

In this model shown in **Figure 1-8** demonstrates the communication between two or more end devices through an intermediate server via different networks like internet and establish communication using protocols like Bluetooth, Zigbee etc. This model can be used to transfer data packets within devices which operate at small data rate.



Wireless network Such as Blutooth, Zigbee etc.

Figure 1-8: Device to Device Communications

2. Device to cloud Communications:

In this model shown in **Figure 1-9** an edge device directly connects to an internet service through wired network or wireless connections to exchange the data to establish a connection between an edge device and an IP network which connects to service provider namely a cloud.

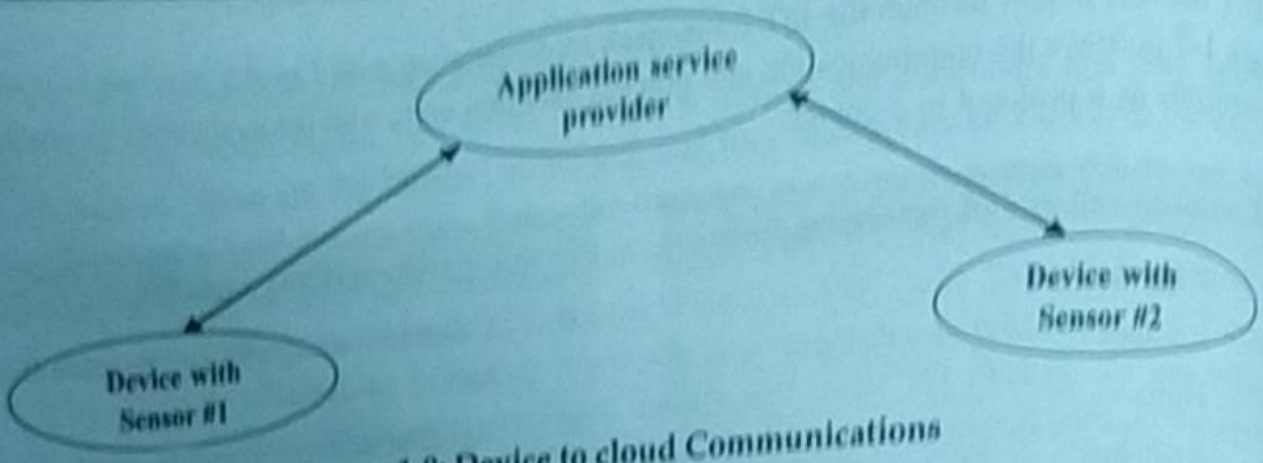


Figure 1-9: Device to cloud Communications

3. Device to Gateway model:

In this model shown in **Figure 1-10**, the edge device is connected to an intermediate server (for example an application in a smart phone to communicate with device and exchange data with device to cloud service.

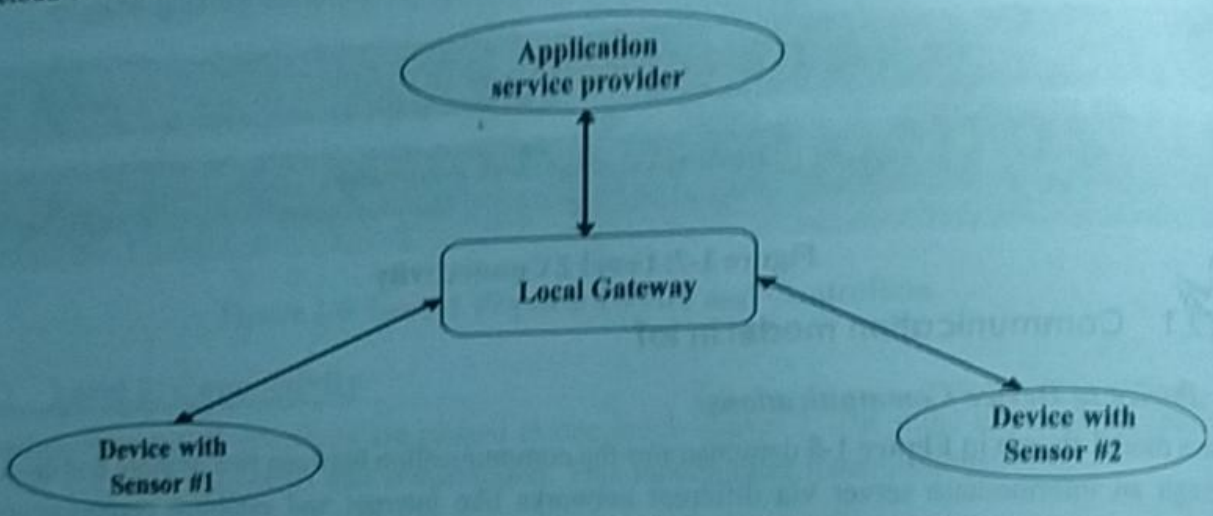


Figure 1-10: Devices connected to a Gateway model

4. Back End data Sharing Model:

This model is used to analyze data from cloud service in combination with data from other service providers also. Here a device is connected directly to a one service provider and meanwhile that service provider is also getting the input data from other sources in the backend.

Effective models can be built to open new enhancements to drive the system for future scope. To build a small application like switching on/off the lights, we can use A simple model network device to device communication model and in rest three forms the connectivity is mainly focused on cloud services provided to the end user.

WebSocket-based Communication APIs

WebSocket APIs allow bi-directional, full duplex communication between clients and servers. WebSocket APIs follow the exclusive pair communication model. Unlike request-response APIs such as REST, the Websocket APIs allow full duplex communication and do not require a new connection to be setup for each message to be sent as shown in **Figure 1-12**.

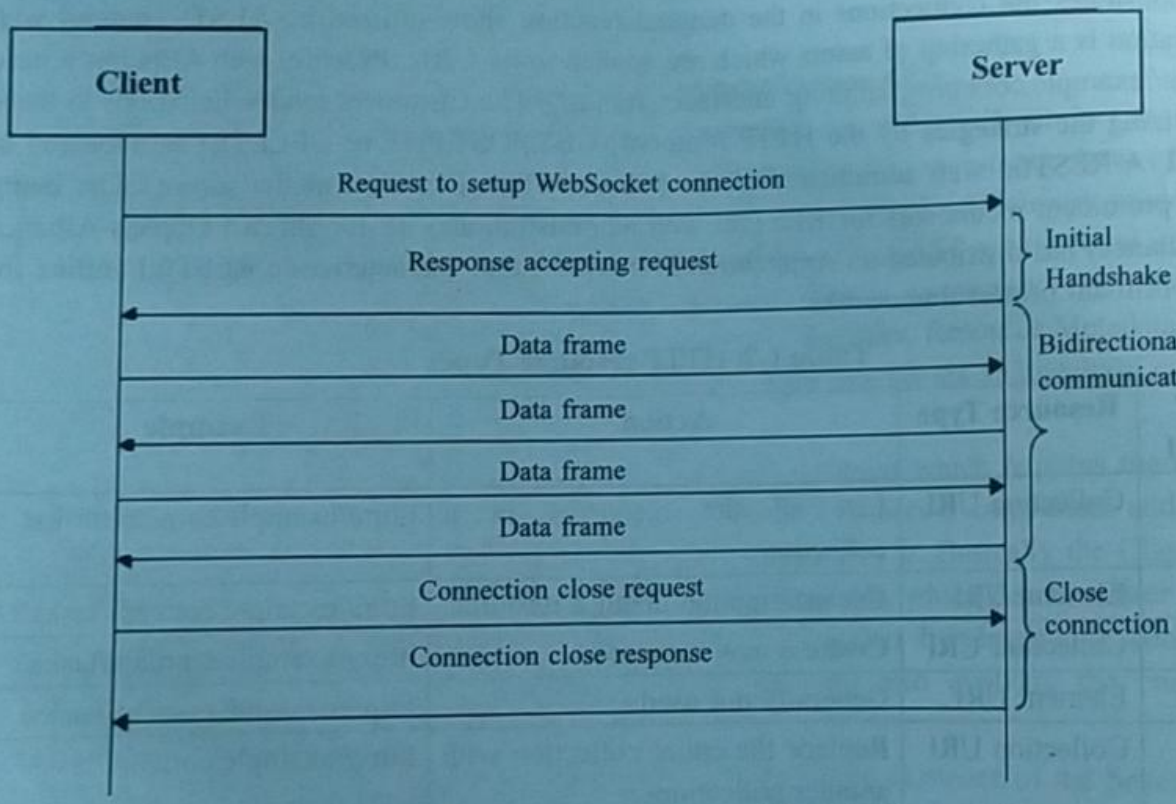


Figure 1-12: Exclusive pair model used by WebSocket APIs

WebSocket communication begins with a connection setup request sent by the client to the Server. This request (called a WebSocket handshake) is sent over HTTP and the server interprets it as an upgrade request. If the server supports WebSocket protocol, the server responds to the WebSocket handshake response. After the connection is setup, the client and server can send data/messages to each other in full-duplex mode. WebSocket APIS reduce the network traffic and latency as there is no overhead for connection setup that have low latency or high throughput requirements.

3.7.2 REST Constraints

There are six directing requirements that characterize a RESTful framework.

1. **Client-Server:** The primary requirements added to our cross breed style are those of the customer server building style. Division of concerns is the standard behind the customer server limitations. By isolating the UI worries from the information stockpiling concerns, we enhance the convenience of the UI over different stages and enhance versatility by streamlining the server parts.

2. **Stateless:** The client–server correspondence is obliged by no customer setting being put away on the server between solicitations. Each ask for from any customer contains all the data important to benefit the demand, and session state is held in the customer. The session state can be exchanged by the server to another administration, for example, a database to keep up an industrious state for a period and permit confirmation. The customer starts sending demands when it is prepared to make the move to another state. While at least one solicitations are extraordinary, the customer is thought to be on the move. The portrayal of every application state contains joins that might be utilized whenever the customer starts another state-move.

3. **Cacheable:** As on the World Wide Web, customers and mediators can store reactions. Reactions should subsequently, verifiably or expressly, characterize themselves as cacheable, or not, to keep customers from reusing stale or wrong information in light of further demands. All around oversaw storing halfway or totally dispenses with some client–server collaborations, additionally enhancing adaptability and execution.

4. **Layered framework:** A customer can't usually advise whether it is associated straightforwardly to the end server, or to a middle person en route. Delegate servers may enhance framework

adaptability by empowering load adjusting and by giving shared reserves. They may likewise uphold security approaches.

5. **Code on request (discretionary):** Servers can incidentally broaden or modify the usefulness of a customer by the exchange of executable code. Cases of this may incorporate assembled parts, for example, Java applets and customer side scripts, for example, JavaScript.

6. **Uniform interface:** The uniform interface imperative is principal to the plan of any REST benefit. The uniform interface streamlines and decouples the design, which empowers each part to advance autonomously. The four requirements for this uniform interface are Identification of assets, Manipulation of assets through portrayals, Self-enlightening messages, Hypermedia as the motor of utilization state (HATEOAS)

3.7.3 REST Architecture

The basic design shown in **Figure 3-15** is characterized beneath which utilized HTTP convention for sending messages.. A portion of the elements imperative to a data system are the adaptability of segment collaborations, all inclusive statement of interfaces and free organization. It is additionally critical that any delegate segments, for example, firewalls for security and reserves to lessen arrange inertness can be worked amongst servers and customer without affecting on accessibility of data.

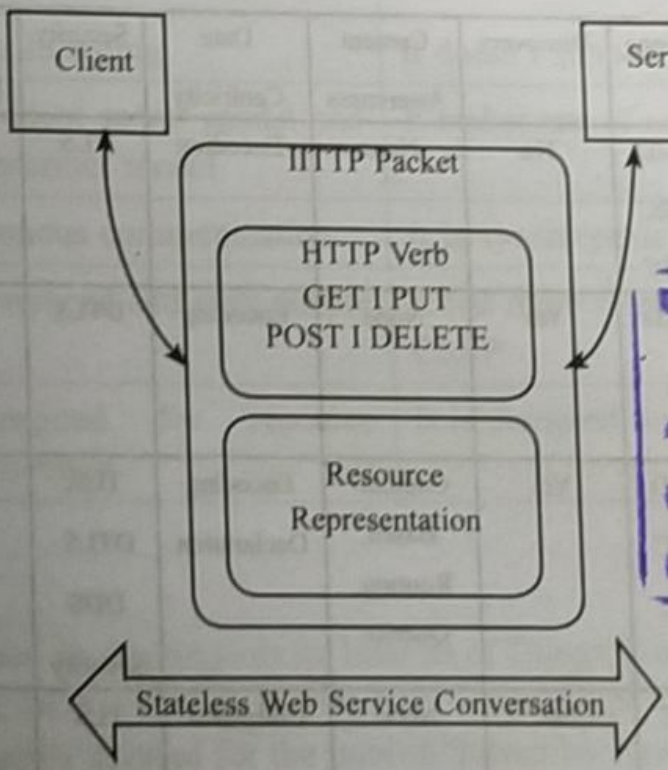


Figure 3-15: REST Architecture

guided through detailed implementations

4.2 HOME AUTOMATION

Smart Lighting- Lighting for homes is made smart by automatically dimming or switching on/ off the lights depending on the ambience of the surroundings. This feature helps in saving energy in domestic usage. Technologies that aid for smart lighting include solid state lighting (such as LED lights) and IP enabled lights. For solid state lighting solutions both spectral and temporal characteristics can be configured to adapt illumination to various needs. Domestic lighting solutions are made smart to sense the human movements and the environmental factors like cloudy or sunny weather and control the lights accordingly. Thus, smart home lighting achieve huge energy savings. Mobile devices or web based IoT applications can be used to remotely control Wireless enabled and Internet connected lights. Smart lights with sensors for occupancy, temperature, etc.., can be configured to adapt the lighting based on the ambient conditions sensed in order to provide a good ambience.

Smart Appliances- There are a number of appliances such as TVs, refrigerators, music systems etc that are used in modern homes these days. Smart thermostats allow to control the temperature remotely and can learn the user preferences. Smart refrigerators can take a stock of items stored and communicate their updates to the users whenever the stock is low. Smart TVs allows users to search and stream videos and movies from the Internet on a local storage drive, search TV channel schedules and fetch news, weather updates and other content from the Internet.

Intrusion detection- Home Intrusion detection system uses security cameras and sensors such as PIR sensors and door sensors to detect intrusions and raise alerts. Owners can be alerted about the intrusion through an SMS or an email. Alerts could be made more detailed by sending image grab or a short video clip as email attachment. Location aware services are used by intrusion detection system that is cloud based to get the geo location of each node of a home automation system and store it in the cloud.

Smoke detectors- When fire bursts in homes and buildings, lot of smoke is generated. Smoke detectors are installed in such places to detect this fire breaks. Smoke detectors use optical detection, ionization or sampling techniques to detect smoke. A fire alarm system can be alerted by these smoke detectors. Harmful gases such as carbon monoxide (CO), liquid petroleum can be detected using gas detectors. These systems can immediately send an SMS or email alerting the user or the local fire safety department about the status of the same.

4.8 Health and Lifestyle

Health and Fitness Monitoring- Wearable IoT devices that allow non-invasive and continuous monitoring of physiological parameters can help in continuous health and fitness monitoring. These wearable devices may can be in various forms such as belts and wrist bands. The wearable devices form a type of wireless sensor networks called body area networks in which the measurements from a number of wearable devices are continuous sent to a master node such as smart phone which then sends the data to a server or a cloud based back end for analysis and archiving. Health care providers can analyze the collected health care data to determine any health conditions or anomalies. Commonly uses body sensors include body temperature, heart rate, pulse oximeter oxygen saturation (SPo2), blood pressure, electrocardiogram (ECG).

Wearable Electronics- Wearable electronics such as wearable gadgets smart watches, smart glasses, wristbands etc and fashion electronics with electronics integrated in clothing and accessories provide various functions and features to assist us in daily activities and making us lead healthy lifestyles. Smart watches that run mobile operating systems provide enhanced functionality beyond just timekeeping. With smart watches, the users can search the internet, play audio/video files, make calls and use various kinds of mobile applications. Smart glasses allows users to take photos and record videos, get map directions, check flight status and search Internet using Voice commands.

Figure 4-3 shows one of the examples how IoT helps in Smart Healthcare systems.

Figure 4-3: IoT in Health care Monitoring

- ✓ During manual mode, the controller service retrieves the current state from the database and switches the light on/off.

5.5 Step 4: Information Model specification

Defining the Information model is the fourth step in the IoT design methodology. Information model defines the structure of the information in the IoT system, for example, attributes of Virtual Entities, relations etc but does not describe the specifics of how information is being represented or stored.

- To define the information model, virtual entities are listed to define in the domain model.
- Information model adds more details to the virtual entities by defining their attributes and relations.

- In the Home Automation example, there are two virtual entities, one is for the light appliance and another is for the room.
- **Figure 5-3** shows the information model for the home automation system example.

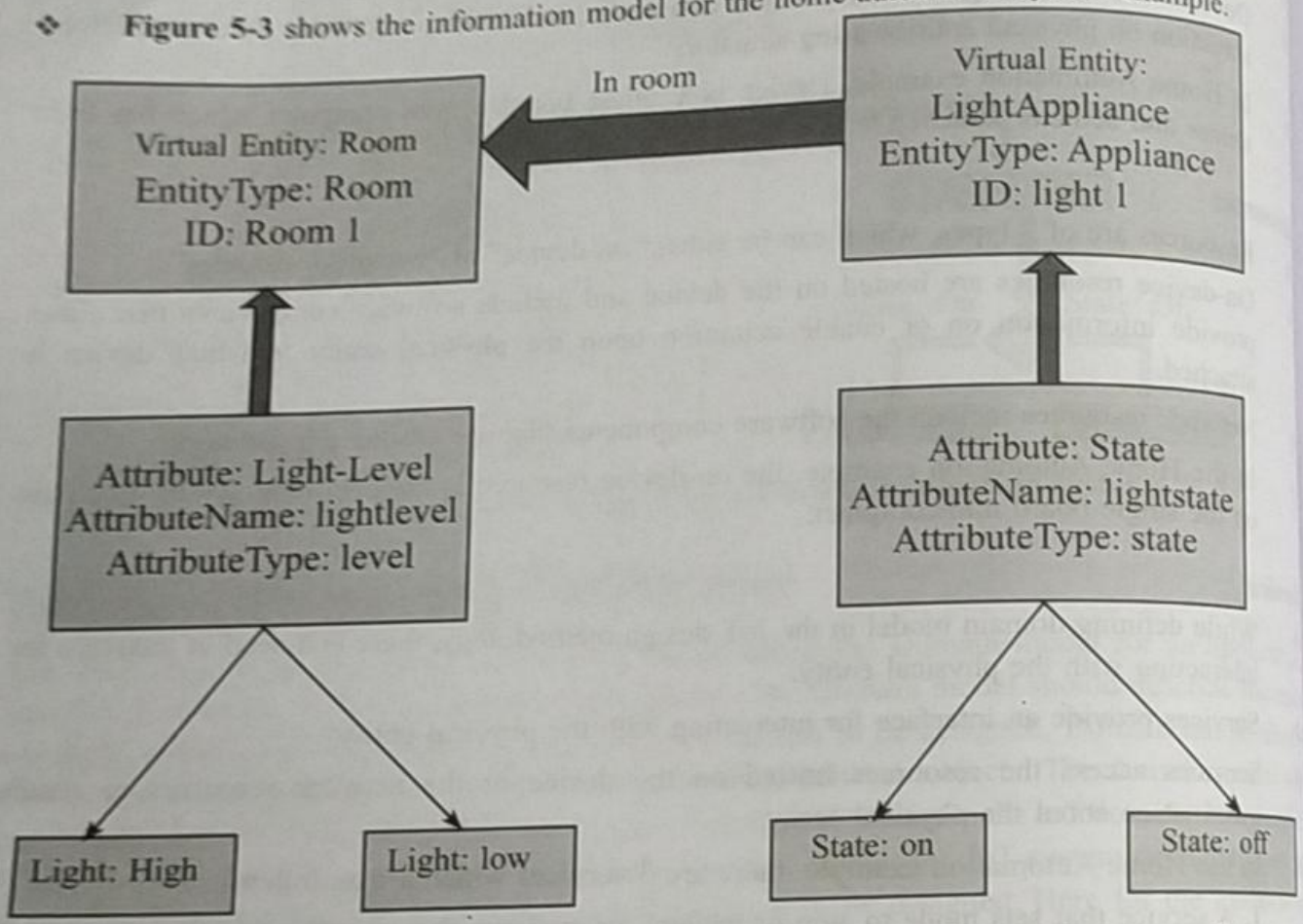


Figure 5-3: The Information model for the Home automation system example

Example use of the Python shell is demonstrated in **Figure 6-1.**

6.4 Python Data Types & Data Structures

6.4.1 Numbers

Number is a data type which stores numeric values and termed as an immutable type, creat
object when value of a number is changed. Table 6.-1 shows examples of numbers.

Table 6-1: Working with Numbers in Python

In this section different data types supported by Python are illustrated namely integer, complex, floating etc and the class they belong to.			
#Integer	#Complex	#Subtraction	#Division
>>>a=10	>>>x=3+8j	>>>c=a-b	>>>div=a/b
>>>type (a)	>>>type(x)	>>>c	>>>div
<type 'int'>	<type 'complex'>	4.4	1.7857
#Float	#Addition	#Multiplication	#Power
>>b=5.6	>>>add=a+b	>>>mul=a*b	>>>g=a**2
>>>type(b)	>>>add	>>>mul	>>>g
<type 'float'>	15.6	56.0	100

6.4.2 Strings

Strings in python are sequence of characters which has no limits on the number of characters, empty string is one which has no characters. Table 6.-2 shows examples of Strings.

Table 6-2: Working with Strings in Python

Consider two strings with **string1="Hello"** and **string2="world"**. In this section string operations such as string concatenation, usage of slicing operation, in and not in operator are illustrated respectively.	
#Create two Strings with **a="Hello"** and **b="world"**	
>>>a="Hello"	
>>>b="World"	
String operations	**Example and its output**
String concatenation[+]	#Concat two strings string1 and string2 using '+' operator >>>c=string1+string2 >>>c HelloWorld
Slice operator[:]	#use slicing operator to find the character at position 1 with a string variable a. >>>string1[1] e
	#use slicing operator to find the characters between position 1 and 4 with a string variable a. >>>string1[1:4] ell

in and not in operator	#usage of in and not in operators which returns 1 if evaluation results to true #else returns 0. >>>H in string1
	>>>M not in string1 1
Formatting output	#use formatting specifiers and print a string by name and age of a person >>>print("My name is %s and my age is %d"('Alisha',21)) My name is Alisha and my age is 21

Consider a string **str= "Monty Python circus"** and In this section different predefined string methods available in python are expressed with example..	
String methods	**Example and its output**
count()	>>> str.count('on') #counts how many substrings? 2
find()	>>> str.find('Pyt') #give starting location of substring, if any 7

isdigit()	>>> four = '4'
	>>> four.isdigit() # to find digit characters in string?
	True

upper()	>>> string_upper = str.upper(# convert to upper case
	>>> string_upper
	'MONTY PYTHON CIRCUS'

just()	>>> str.rjust(30) # right justify string by adding blanks ' Monty Python circus '
strip()	>>> "newlines\n\n\n".strip() # a string literal also has methods! 'newlines'
replace()	>>> str.replace('circus', 'comic') # replace substring (all occurrences) 'Monty Python comic'
replace()	>>> s.replace('M', 'P', 1) # replace only once 'Ponty Python circus'

split()	>>> "Hello string methods!".split() # split a string when a blank space exists
	['Hello', 'string', 'methods!']
	>>> "32,24,25,57".split(",") #split a string on specified character

6.4.3 Lists

A List is a rundown information structure that holds a requested gathering of things i.e. you can store a succession of things in a rundown. The rundown of things ought to be encased in square sections. List is a mutable (variable) information.. Table 6.-3 shows examples of Lists.

Table 6-3: Working with Lists in Python

In this section we learn how to Create a list and demonstrate list operations such as add, delete, update, access elements of a List etc.	
List operation	**Example and its output**
Accessing values in the list	# create three Lists subjects, number and alphabets with the following #elements in to it. #subjects = ['python', 'java', 1999, 2001] #number= [11, 22, 33, 44, 55] #alphabets= ["ab", "bc", "cd", "de"] >>>subjects[0] python >>>number[3] 44 >>>subjects[-1] 2001 >>>alphabets[-3] bc
Finding type of a List	>>>type(subjects) <type 'List'>
updating list	#Create a list of three numbers and perform update operations >>>a=[1,2,3] >>>a [1,2,3]

	[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

6.4.4 Tuples

In python a tuple is termed as grouping of immutable(permanent) Python objects and are sequences, much the same as Lists. The main contrast is that tuples are unchanging. Tuples utilize enclosures and records utilize square sections.. Table 6-4 shows examples of Tuples.

Table 6-4: Working with Tuples in Python

Here we learn how to Create a tuple and demonstrate how to find length, combining tuple and to access elements of a tuple.	
tuple operation	**Example and its output**
Accessing values in the tuple	# create three Lists subjects, number and alphabets with the following #elements in to it. #subjects = ['python', 'java', 1999, 2001] #number= [11, 22, 33, 44, 55] #alphabets= ["ab", "bc", "cd", "de"] >>>subjects[0] python >>>number[3] 44
Finding type of a tuple	>>>type(subjects) <type 'tuple'>
Length of a tuple	#Find length of a tuple number >>>len(number) 5

6.4.5 Dictionaries

In Python Dictionary reference is mutable(changeable) data type and is another container type that can store any number of Python items, including other types. Dictionaries references comprise of sets (called things) of keys and their comparing values. Python lexicons are otherwise called acquainted exhibits or hash tables. Keys are one of a kind inside a dictionary while values may not be. The values of a dictionary can be of any type, yet the keys must be of a immutable(permanent) information type, for example, strings, numbers, or tuples. Table 6-5 indicates cases of Dictionaries.

Table 6-5: Working with Dictionaries in Python

Here we learn how to Create a dictionary and demonstrate how to find length, value of a key, accessing items, accessing all values and deleting dictionary elements.	
Dictionary operation	**Example and its output**
create dictionary	# create dictionary of a person with name, id and stream as key entries >>>student={'name':'Bob','id':'1234','stream':'CS'} >>>student {'id':'1234','stream':'CS',' name':'Bob'}
Finding type of a dictionary	>>>type(student) <type 'dict'>
length of a dictionary	>>>len(student) 3
value of a key in dictionary	>>>student('name') 'Bob'
Get all items in a dictionary	>>>student.items() {('stream':'CS'),(' name':'Bob'), ('id':'1234')}
Get all keys in a dictionary	>>>student.keys() ['id','stream','name']
Get all values in a dictionary	>>>student.values() ['1234','CS',' ,'Bob']

convert to list	>>>s="Hello" >>>list(s) ['H','e','l','l','o']

6.5 CONTROL FLOW

Python programming language provides following types of decision making statements.

- **if statements**

An if statement comprises of a Boolean expression took after by at least one or more statements.

- **if...else statements**

An if statement can be trailed by a discretionary else statement, which executes when the Boolean expression is false.

- **nested if statements**

You can use one if or else if statement inside another if or else if statement(s).

6.5.1 if statements

Syntax of if statement: ***if expression: statement(s)***

Single Statement Suites: If the suite of an if comprises only one statement, it might go on an same line line from the header defined statement. Case of a one-line if statement appeared in Table 6.7:

Table 6-7: working with if statements in Python

Here we demonstrate the working of if statement, In the program given below code evaluates the value of p in if statement to validate at the desired output.	
Example	**output**
>>>p=10 >>>if(p==10) : print("value of p is :10") print("Bye")	value of p is :10 Bye

If-else statement

Syntax of If-else statement ***if expression:***

The *elif* Statement

The elif explanation permits you to check various expressions for truth esteem and execute a piece of code when one of the conditions assesses to genuine. Python does not as of now bolster switch or case explanations as in different dialects.

Syntax of elif Statement

```
if expression1:
        statement(s)
elif expression2:
        statement(s)
elif expression3:
        statement(s)
else: statement(s)
```

Table 6-9: working with elif statement in Python

Here we demonstrate the working of elif statement, In the program given below code evaluates the value of p in nested conditional statements to validate at the desired output.

Example	output

>>>num_x=1000
>>>if num_x < 2000:
print("value is less than 2000")
if num_x == 1200:
print("It is 1200")
elif num_x == 1400 :
print("It is 1400")
elif num_x < 1000 :
print("Value less than 1000")
else:
print("Could not find true expression"
print"Bye")
	value is less than 2000
	It is 1200
	Bye

6.5.2 for

The for loop in Python can emphasize over the things of any grouping, for example, a rundown ,string, tuple and other inherent iterables.

Syntax:

```
for iterating_var in sequence:
        Statement(s)
```

6.5.3 while

While loop repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

Table 6-11: working with while statement in Python

Here we demonstrate the working of while loop to which evaluates over executing the loop body	
while loop usage	**Example and its output**
print odd numbers upto 100	>>>i=0 >>>while I <=100:

[20,40,60,80,100]

6.5.4 break/continue

break statement terminates the loop statement and transfers execution to the statement immediately following the loop.

continue statement causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

Table 6-12: working with break/continue statement in Python

Here we demonstrate the working of break and continue statements evaluates over executing the loop body	
Break statement	# when character evaluate to a value of 'a' control break out of a loop >>>for character in 'Break': if character == 'a': break print(character) B r e

k

6.5.5 Pass

The pass statement in Python is utilized when an statement is required grammatically however don't need any summon or code to execute.

Table 6-13: working with break/continue statement in Python

Here we demonstrate the working of pass statements evaluates over executing the loop body	

pass statement	>>> for num in [1,2,3,-2,5]:
	if num <0:
	pass
	print("This is pass block")
	else:
	print(num)
	1
	2
	3
	This is pass block

6.6 FUNCTIONS

Functions are characterized to have reusable code and arrange and simplify code. Functions characterized utilizing keyword def followed with function name and brackets (()). The prim statement of a function can be a optional statement - the documentation string of the functio docstring. The code hinder inside each function begins with a colon (:) and is indented.

The statement return [expression] exits a function, alternatively going back an expressio the caller.

Table 6-14: Example of a function in Python

Here we define function to find minimum of two numbers	
Example	**Output**

#Defining a function min having two parameters n1 and n2
>>>def min(n1,n2):
if n1 < n2:
res=n1
else:
res=n2
return res
# Function call 	
>>>result=min(10,20)
>>>print("The minimum of two numbers is" result)
	The minimum of two numbers is 10

Function with default arguments

For a few Functions, a few parameters can be made as optional and utilize default values if the client does not have any desire to give values to such parameters. This is finished with the assistance of default contention values. Take note of that the default contention esteem ought to be changeless. Parameters which are toward the end of the parameter rundown can be given as default contention values.

Table 6-15: Example of a Function with default arguments in Python

Here we define function Hello with default arguments passed as an argument list	
Example	**output**
#define a function Hello with times as default parameter set to 1 >>>def Hello(s,times=1): print(s*times) #call a function Hello >>>Hello('python) Python >>>Hello('default',5)	defaultdefaultdefaultdefaultdefault

Function with keyword arguments

Here Arguments are identified by name of the parameter are keyword arguments. Name of a argument is used instead of the position.

Table 6-16: Example of a Function with keyword arguments in Python.

Here we define function functionX with keyword arguments passed as an argument list	
Example	**output**
#define a function with keyword arguments b set to 6 and c set to 12 >>>def funcionX(x,y=6,m=12): print('x is',x,'y is' ,y, 'm is', m) >>>functionX(13,17)15 >>>functionX(12,c=22) >>>functionX(m=5,x=10)	x is 13 y is 6 m is 12 x is 12 y is 17 m is 12 x is 10 y is 6 m is 5

Function with Required arguments

Required arguments are the contentions go to a function in right positional request. Number contentions in the function call ought to coordinate precisely with the function definition.

Table 6-17: Example of a Function with Required arguments in Python.

Here we define function show with required arguments passed as an argument list	
Example	**output**
#define a function with required arguments b set to str variable >>>def show(str): print(str) return >>>printme() >>>printme("Hello")	#output of function call #printme() Error #output of function ca #printme("Hello") Hello

Function with Variable length arguments

Function called with a greater number of arguments than indicated while characterizing the functi called as variable length arguments. A mark (*) is prefixed before the variable name which hold t estimations of all non keyword variable arguments. This tuple stays purge if no extra arguments a indicated amid the function call.

Table 6-18: Example of a Function with Variable length arguments in Python

Here we define function info with variable length arguments passed as an argument list	
Example	**output**

#define a function info where last parameter should be defined with #* as #variable length argument
>>>def info(a1,*vartuple):
	#output of function ca
	info(12)
	12