Internet of Things (Advance) 2000605 C

UNIT-1 Python Basics

Python:

- Python is a widely used high-level programming language.
- Python was created by "Guido Van Rossum" in "1991".
- Python is a powerful, flexible and easy to use language.

Features of Python;

1. Readable : Python is easily readable language.
2. Easy to Learn : Coding in python focusses more on the solution of problem and less on syntax.
3. Cross platform and portable : Python works on and with OS (operating system) like MAC, Windows, Linux etc.
4. Open Source : Python is open source programming lang.
5. Free : Python is free to download use and distribute.
6. Rich library support : Python has rich in built library support and do not requires external library support.

Advantages of Python;

1. Easy to read, learn and write - It has english like syntax.
2. Improved Productivity - In python more things are done using less no. of codes.
3. Interpreted language - Python is an interpreter language which runs the codes line-by-line, so debugging is easy.
4. Free and open source - Python program and source codes are free to download, use and distribute.
5. Rich library support - It has in built libraries and we don't need external libraries.
6. Portability - Python codes can be run on any platform.

Disadvantages :

1. Slow speed - Python is slow because it is an interpreter and dynamic language.
2. Poor Memory Efficiency - Python uses and need large amount of memory.
3. Weak in mobile computing - It is more suitable for server computers and not for clint, computers or cell phones.
4. Poor database access - Python is not suitable for large database.
5. Runtime errors - Python is dynamic language so data type of a variable can change anytime. It gives runtime errors.

Installation of Python

Step-1; In any internet browser's address bar type -

https://www.python.org/download/

Step-2; Click on python for Windows. It will display different versions of python.

Step-3; Select the latest version of python. Eg. Python 3.9.1 and click on download.

Step-4; After clicking on download, various installers are displayed. Choose the installer according to system OS. Eg. Windows installer (64 bits).

Step-5; The download will complete. Now we have download-python 3.9.1 windows (64-bits) installer

Step-6; Run and install. It will open installer window. Click on "install now" button and installation process starts. Installation is finished in few minutes.

Variable

- A variable is nothing but a reserved memory location to store a value.

- Python variable –

(i) Must start with a letter or underscore character.

(ii) Cannot start with a number.

(iii) Can only contain alphanumeric characters (A-Z, a-z. 0-9) and underscore.

(iv) Variable names are case sensitive, eg; BOOK & book are different variables.

(v) A variable name cannot be any of the python keywords.

Types of variables

According to data types, variables can be following types.

1. Numbers Type :–

Number type variables are used for numeric data which can be integer, long integer, floating-point or complex.

Examples;–

x = 25 ; integer , x = 519243 61L ; Long integer

x = 5.6 ; floating-point, x = 2+3y ; complex.

2. String type : –

- It consist of characters in quotation marks.
- Single quotes and double quotes both are valid.

Example - x = "Raj" , x = 'Raj'.

3. List type :–

It consist of a number of numeric and/or string type data seperated by comma (,).

Example , x = ["Ray", 23,"abcd", 4.36].

4. Tuple type :-

Tuple type is similar to list type except –

(i) It cannot be changed or edited.

(ii) It is enclosed in parentheses ( ).

According to case type, python variables are of three types –

1. Camel case – a middle character is capital. Eg.

nameOfstudent

2. Pascal case – First and middle characters are capital.

Eg. Name Of stdent

3. Snake case – Characters are seperated by underscore.

Eg. name_of_student.

Print () Function

- Print function prints a message on the screen or other output devices.
- The content inside ( ) is called argument which can be a message or a variable.
- The syntax for using print () are given below ;–

1. To print a message directly

Here the ~~arrang~~ argument is message itself inside single or double quotes. Example, >> Print ("I am Raj.")

output ;- I am Raj.

Note : If quotes are not used print ( ) assumes it a variable.

2. To print a variable

Write the variable inside ( ).

```
Examples, i. >> x = 5                 ii. >> x = "I am Raj"
             >> Print (x)                 >> Print (x)
     Output : 5                       Output : I am Raj"
```

3. To print multiple arguments

Seperate the arguments by comma (,).

```
Example ;                                   Output
   >> Print("Hello","Hi!")                  Hello Hi!
   >> full-name = "Raj"                     ______
   >> Print ( "Hi!",fullname)               Hi! Raj
```

Escape Character

- There are illegal characters in python that are never printed. For example; single quote (' '), double quote (" "), backslash (\) etc.
- Escape character is used to print illegal characters in a string. syntax is backslash before illegal character. i.e.

  \ illegal character

Example ;

```
1. x = "We are "polytechnic" students."
2. Print (x)
```

Here, there are illegal characters (" ") in line 1, so the above code will not run and python interpreter will indicate error. This problem can be solved by escape character. ie

```
x = "We are \" Polytechnic \" students."
```

Escape Character Type	Example	Output
1. Single quote \'	x = "It\'s alright."	It's alright.
2. Back slash \\	x = "D\\A"	D\A
3. New line \n	x = "Hello\n world!"	Hello World!
4. Carriage Return (Enter) \r	x = "Hello\r world!"	Hello World!
5. Tab \t	x = "Hello \t World!"	Hello World!
6. Backspace \b	x = "Hello \b World!"	Hello· World!
7. Form feed		
8. Octal value		
9. Hexadecimal value		

Run Python Program

- Python is interpreter type programming software.
- Python program files are called 'scripts'.
- Python program can be run in two modes –

i) Interactive mode

ii) Script mode

Interactive Mode

Steps to run python programs in this mode are –

Step 1. Open command line and write python 3. It will display prompt as >>

Step 2. Write the codes after the prompt and press enter. It will run the code. For example;

```
>> 3+4 ↵           >> Print ("Hello") ↵
>> 7               >> Hello
```

Step 3. To exit interactive mode write exit () or quit () and press enter(↵).

Note: The main demerit of interactive mode is that program is lost after exit. It is used only for testing of codes.

Script Mode

Steps to run python codes are given below –

Step-1 Write the code in a text editor and save the file wit extension .py. for example if file name is xyz then file name will be xyz.py.

Step 2 Open command line.

Step 3 On command prompt write python 3 and then python script file name. for example python 3 xyz.py.

Step-4 Press enter and it will run the python Script.

Python Touble

- In python multiple items can be stored in a single variable.
- This set of multiple items is called collection of data.
- According to collection data types (Properties), variables in Python are of four types are List, Touple, Set and Directory.
- Their properties are given below;–

Collection Data type	Written in	Properties	Duplicate Items
1. List	Square Brackets [ ]	ordered, changable indexed	Yes
2. Tuple	Rounded brackets ( ) or no brackets	Ordered, unchangeable indexed	Yes
3. Set	Curley brackets { }	Unordered, unchangeable unindexed	No
4. Dictionary	– Same –	ordered, changeable indexed	No

Touple

– A tuple is a collection of multiple data. It is used to store multiple items in a single variable.

– Tuple is one of the four types used to store multiple items. The other three are list, set, dictionary.

– **Syntax** – Tuple is a comma seperated list of values written inside round brackets ( ) or no brackets.

Example; x = ('a','b','c','d','e'). y = (1,2,3,4,5).

– Properties

1. Tuple items are indexed starting from [0].
   [0] = a , [1] = b , [2] = c , [3] = d , [4] = e
2. Unchangeable – We cannot change, add or remove items.
3. Ordered – Items in a touple have fixed order and cannot be changed.

4. Duplicates - Tuples allow duplicate values.

Ex ; x = ('a','b','c','d','e','a').

Example	Display output
1. Empty tuple x=() Print(x)	( )

2. Tuple having integers
x = (1,2,3)
Print(x)
	(1,2,3)

3. Tuple with mixed data x = (1,"Raj"3.4) Print (x)	(1,"Raj",3.4)
4. Nested tuple x = ("Raj",[1,2,3], (1,2,3),3.4] Print (x)	("Raj",[1,2,3] (1,2,3),3.4)

Example	Display Output
5. Tuple with one element x=('a',) Print (x) x = ('a',) is not touble.	('a',)
6. x = ('a','b','c','d','e') Print (len(x)) Print (max(x)) Print (min(x))	Print(·) '5' 'e' 'a'
7. Indexing x = (1,2,3) Print (x[0]) Print (x[1]	1 2

Note; Tuple items can be any data type. ie integer, float, complex etc.

Python Dictionary

Dictionary is a collection of items which is ordered and changeable and do not allows duplicate.

Syntax : 1. Dictionary items are written in key : value pair
2. Dictionary is written in curley brackets { }
3. Items are saperated by comma,

x = { "Name":"Raj", "College":" Patna7,"DOB": 2006)
Print (x).

Properties

1. Items are referred by key name. For example;- Print(x["Name"])

   output ;- "Raj"

2. Ordered - Items are ordered and order cannot be changed after dictionary is created.
3. Changeable - We can change, add or remove items.
4. No duplicates - Duplicate items are invalid.
5. Dictionary items can be of any data type.

Example	Display Output
1. Empty dictionary x = { } Print (x)	{ }
2. x = {'a':1, 'b':2} Print (x)	{'a':1, 'b':2}
3. Single Element x = {'a':1} Print (x)	{'a':1}
4. x = {'a':1, 'b':2, 'c':3} Print (len(x)) Print (max(x)) Print (min(x))	3 c a

Operators

- Operators are used to perform operations on values.
- Python operators are Arithmetic, Assignment, comparison, logical, Identity, membership and bitwise.

1. Arithmetic Operators

Operator	Name	Example	Output for x=5, y=3
+	Addition	x+y	Print (x+y) 8
–	Substraction	x-y	2 15
*	Multiplication	x*y	1.66666
/	Division	x/y	2
%	Modulus (remainder)	x%y	125
**	Exponentiation	x**y	1
//	Floor division	x//y	

Precedence : **, *, /, //, %, +, – or Hierarchy.

2. Assignment Operators

Operator	Example	Same to
=	x = 5	x = 5
+ =	x + = 5	x = x + 5
- =	x - = 5	x = x - 5
* =	x* = 5	x = x*5
/ =	x/ = 5	x = x/5
% =	x % = 5	x = x%5
// =	x // = 5	x = x//5
** =	x** = 5	x = x**5
& =	x & = 5	x = x & 5
| =	x | = 5	x = x | 5
^ =	x ^ = 5	x = x^5
>> =	x >>=5	x = x >>5
<< =	x <<= 5	x = x <<5

Arithmetic operation Assignments

3. Comparison Operator

Operator	Name	Example
==	equal	x == y
!=	not equal	x != y
>	greater than	x > y
<	less than	x < y
>=	greater than or equal	x >= y
<=	less than or equal .	x <= y

4. Logical Operators

Operator	Description	Example
AND	Returns true if both statements are true.	x = 5 Print (x>3 AND x<8)
OR	Returns ~~t~~true if any of the statements is true	print (x>3 OR x>8) True
NOT	Returns false if the result is true and vice-versa	NOT(x>3 and x<8) False

5. Identity Operators

Operator	Description	Example
is	Returns true if both variables have same object	x is y
is not	Returns falso is above condition is false	x is not y

6. Membership Operator

operator	Description	Example
in	Returns true if one value is in other	x in y
not in	Returns false if above condition is false	x not in y

7. Bitwise Operator

Operator	Description	Example	$x=3$ $y=5$	
&	AND logic on binary	011 101 001 $\Big\vert_2$	1_{10}	x & y
|	OR logic	011 101 111	7_{10}	x | y
^	XOR logic	011 101 110	6_{10}	x ^ y
~	NOT logic	011 100	4_{10}	~ x
<<	Zero fill left shift	011 0110 01100	12_{10}	x << 2
>>	Zero fill right shift	100 - 4 010 001 - 1		x >> 2

Python Array

Python array is a special variable to store multiple values.

Example -

(i) x = [1,2,3,4]

(ii) y = ["Physics", "Chemistry", "Maths"]

Important Terms

1. Element - It is any value in an array. eg 1,2,3 , physics are elements.

2. Index - It is the sequence number of an element.

Array	Element	Index
x	1 2 3 4	[0] [1] [2]
y	physics chemistry maths	[0] [1] [2]

Array is useful and it is similar to list but the basic difference is -

(i) Elements in array are same data types

(ii) Array is useful for large number of elements

Comparision between Array and List

Array	List
1. Homogeneous set of elements.	1. Homogeneous
2. Same data type elements.	2. Same or mixed data types
3. Indexing is fast because elements are strored in contiguous memory locations.	3. Indexing is slow elements are stored in distant locations
4. Supports arithematic operation.	4. Do Not supports
5. Useful for longer sequence of elements.	5. Useful for shorter Sequence.

Advantages of array

1. Can handle very large data sets.
2. Memory efficient
3. Fasten calculation and analysis
4. Diverse functionality ie supports different functions.

Types of Arrays

0 - Dimensional	[1]
1 - Dimensional	[1, 2, 3, 4]
2 - Dimensional	[1, 2, 3, 4] [5, 6, 7, 8]

Creation of Array

Python array can be cracked by two basic methods:-

1. Importing array module or
2. Importing python library eg. numpy.

Creating array using array module

Step - 1 : Import array module

Step - 2 : Create the array by codes

```
x = array. array ("data type", [array elements])
↳ or any nickname for array module eg. arr
```

Example ; Create the array 1,2,3,4 and display

Code

```
Import array as arr
x = arr. array ("i", [1, 2, 3, 4])
for i in range(0,4):
    print (x[i]).
```

Output

```
1
2
3
4
```

Note ; to display horizontally write code

```
print (x[i], end=" ")
```

Creation of array using numPy

Step 1: Import numPy as nm (let)

Step 2: Create the array using code

```
Variable_name = nm. array ([1,2,3,4])
```

Example; Create array 1,2,3,4 and display.

__Codes;__

```
import numpy as nm
x = nm. array ([1,2.3.4])
print (x)
```

↑ Argument

__Display__; [1,2,3,4]

Example; create an array using list x = [1,2,3,4] and display.

__Codes__

```
import numpy
x = [1,2.3.4]
x-array = numpy.array (x)
print (x-array)
```

__Display__ · [1,2,3.4]

Example; Addition of arrays

Code:

```
Import numpy
x = [1,2.3.4]
y = [5.6,7,8]
x-array = numpy.array (x)
y-array = numpy. array (y)
Print (x-array + y-array)
```

Display [6,8,10,12]

Note; Array size must be same.

Reversing Array

1. Using reverse() function

- It is used in array module.
- Syntax: array-name.reverse()

Example;

```
import array
x = array.array('i', [1,2,3,4])
 x. reverse( )
 for i in range (0,4):
 print (x[i], end = " ")
```

Display [4 3 2 1].

2. Using NumPy flip( ) function

- Syntax:

reversed_array.name = numPy.flip(array-name)

Example;

```
 import numPy
x = numpy. array ([1,2,3,4])
y = numpy . flip(x)
 Print (y)
```

Display [4 3 2 1]

3. Using NumPy flipud ( ) function

-Syntax:

reverse. array.name = numpy.flipud (array.name)

Flipud means flip upside-down

Example:

```
import numpy
x= numpy. array ([1,2,3,4])
y = nunpy. flipud (x)
print (y)
```

Display [4321].

4. Using NumPy array slicing

- Syntax

```
reversed_array_name = array-name [:: -1]
```

Example :

```
import numpy
x = numpy.array ([1·2 3 4])

y = x[:: -1]
Print (y)
```

Display [4 3 2 1]

Append Array

- Append means adding new elements in array.
- Append function adds new elements in array after last element.
- Append function methods in NumPy are -

(i) append ( ) function

(ii) Concatenate ( ) function

(iii) Stack ( ) function

(iv) hstack () function

(v) VStack () function

Append ( ) function

Syntax : appended_array-name = numPy.append (array1-name, array2-name)

Example :

```
import numpy
x = numpy.array([1,2,3,4])
y = numpy.array([5,6,7,8])
Z = numpy.append (x,y)
 Print (Z)
```

Display : [1,2,3,4,5,6,7,8].

Concatenate () method

Syntax:

appended-array-name = numpy.concatenate([array1name;array2.name])

Example

```
import numpy
x = Numpy.array([1,2,3,4])
y = numpy.array([5,6,7,8])
z = numpy.concatenate([x,y])
Print (z)
```

Display: [1,2,3,4,5,6,7,8]

Stack ( ) Method

Syntax: appended-array-name = numpy.Stack([array1-name, array2-name])

- Array size shoud be same

Example

```
import numpy
x = numpy.array([1,2,3,4])
y = numpy.array([5,6,7,8])
Z = numpy.stack ([x,y])
Print (z)
```

Display:

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

hstack ( ) method

Syntax: appended-array-name = numpy.hstack([array1-name, array2-name])

```
import numpy
x = numpy.array([1,2,3,4])
y = numpy.array([5,6,7,8])
z = numpy.hstack([x,y])
Print (z)
```

Display: [1 2 3 4 5 6 7 8]

Vstack () method

Syntax :

```
appended-array-name = numpy.vstack([array1-name, array2-name])
```

```
import numpy
x = numpy.array([1,2,34])
y = numpy.array([5,6,7,8])
z = numpy.vstack([x,y])
Print (z)
```

Display :
```
[[1 2 3 4]
 [5 6 7 8]]
```

For append module other array functions

1.	Clear ( )	Remove all elements
2.	len ( )	Returns number of element
3.	CoPy ( )	Returns a copy
4.	Count ( )	Returns no. of elements with specified value
5.	Extend ( )	Add elements of a list to the end of current list
6.	Index ( )	Returns index of 1st list
7.	Insert ( )	Add element at Specific position
8.	remove ( )	Removes the first item
9.	Sort ( )	Sorts the list

Syntax : list_name.function_name( )

2-Dimensional Array

- In python, a 2-dimensional array is an array of arrays.
- It is actually a set of multiple list or 1-D arrays.
- So, an array whose elements are 1-D arrays is called a 2D array.

Example

```
       ↙1D      ↙1D    ↙2D
x = [[1,2,3],[4,5,6]]
```

Indexing of 2D array

- Python arrays are zero indexed. ie start from 0.
- Indexing of elements of x is given below:-

```
              Array 0       Array 1
    X  =  [ [1, 2, 3], [4, 5, 6] ]
             ↓  ↓  ↓    ↓  ↓  ↓
             0  1  2    0  1  2
```

- The syntax is

x[0] = [1,2,3]

x[0][0] = 1

x[0][1] = 2

x[0][2] = 3

x[1] = [4,5,6]

x[1][0] = 4

x[1][1] = 5

x[1][2] = 6

Applications of 2D arrays

(i) To represent data in tabular form.

(ii) To store data in matrix form.

(iii) To represent grid of graphic display since screen is a grid of pixels.

Creation of 2D array

Ex-1, Create a 2D array containing two arrays with the values 1,2,3 and 4,5,6.

Code

```
import numpy
x = numpy.array ([1,2,3],[4,5,6])
Print (x)
```

Using array ( ) function of numpy.

Accessing elements of 2D array

Ex-2 Print the elements of 2D array of example 1,

code;

```
Import numpy
x = numpy. array ([1,2,3],[4,5,6])
```

Print (x[0]) = [1,2,3], Print(x[0][0]}= 1 , Print (x[0][1]= 2

print (x[1]) = [4,5,6], Print (x[1] [0] = 4, Print (x[1][1] = 5

Print (x[0] [2]= 3, Print (x[1][2] = 6.

Conditional statements

Conditional statements are used to do an operation if a Condition is satisfied.

for example -

$x = 5 , y = 7$

If x> y then print "x" is larger number."
else print "x is smaller number."

Here,

(i) Greater than (>) is called conditional operator and

(ii) If ... else is called conditional statement.

Conditional operators are given below -

	Operator	Meaning
1.	a = = b	is equal
2.	a ! = b	is not equal
3.	a < b	is less than
4.	a <= b	is less than or equal
5.	a > b	is greater than
6.	a >= b	is greater than or equal
7.	AND	both conditions are true
8.	OR	either condition is true
9.	NOT	Condition is not true.

Conditional statement

1. If
2. Elif
3. If-else
4. Short hand if
5. Short hand if-else

1. If

If condition is true, then, do, else do nothing.

```
Ex-      a=5,   b=7                # .......
            if b>a;
           print ("b is larger number")
```

2. Elif

If the previous condition is not true try next one.

```
Ex-      a=5     b=5
           if b>a
            print ("b is greater than a")
           elif a = = b
             print ("a and b are equal")
```

3. If-else

If condition is true, do operation 1, else do operation 2.

```
Ex-1      a=5    b=7
          if a>b:
          Print ("a is larger number")
          else :
           Print ("b is larger number")
Ex-2      a=5    b=7

           if b>a:
           Print ("b. is larger number")
           elif b = =a
           print ("b and a are equal")
           else:
           print ("b is smaller number")
```

4. Shothand if

Used when there is only one statement

Ex: If a>b: print ("a is greater than b")

5. Short hand if-else

One line statement

Ex: a=5   b=7

print ("A") if a>b else print("B").

Conditional statements with logical operators

1. AND

Ex a = 7, b = 5 , c = 10

```
if a>b and c>a :
print (" both conditions are true")
```

2. OR

Ex a=7 b=5 c=10

```
if a>b or a>c:
print ("at least one condition is true")
```

3. NOT

Ex a =5 b =7

```
if not a>b
print ("a is not greater than b")
```

UNIT-2

Python Do and While Loop

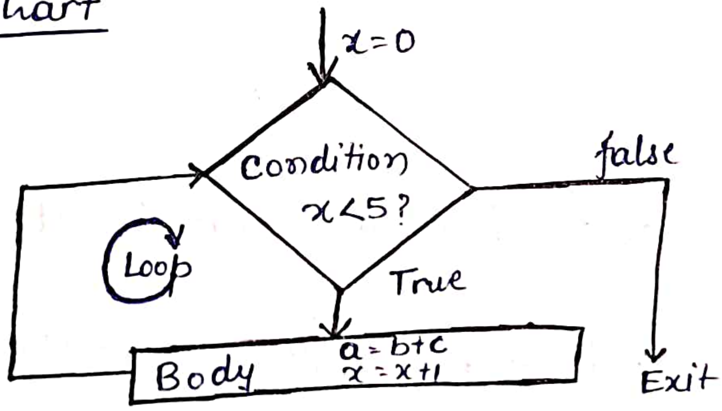
Loop :- A loop is a statement or set of statements which is executed/run multiple times until a given Condition is satisfied.
The statement or set of statements is called body of the loop.

Do-while loop

In computer language programming, in do-while loop:-

(i) do- means execute the body
(ii) While- means execute the body until a given condition is true else exit.

Flowchart



General sintex/structure of do-while loop

```
do {
   // statements } Body
  }
 While (condition):
```

Do-while looping in Python

- Python does not have do-while loop and there is no construct defined for it.
- Do_while loop working can be explained in following steps:-
  (i) Run a set of statements or body
  (ii) Check a condition and if condition is true, run the body again.

(iii) If condition is false, exit from the loop.

- In python, "do-while" loop is like "while" loop but it is run at least once.

Example - 1

```
        i = 1
    while  i < 6:
      Print (i)
      i = i+1
```

output Display

```
1
2
3
4
5
```

Example - 2

```
      i = 1
    While true :
     Print (i)
      i = i+1
     if (i>5):
       break
```

```
1
2
3
4
5
```

Break statement

- In python break statement is used to break out or exit a loop.
- It exits a loop even if loop condition is true.

Example   Break the loop when i is 3

```
  i = 1
 while (i<6):
   if  i==3
   print (i)
    i = i+1
```

Display

```
 1
 2
Break (i=3)
```

Continue Statement

Continue statement is used to jump a step in a loop.

Example   Ignore if i is 4 and continue in loop.

```
i = 0
While i<6
i+=1
if  i==4
Continue print(i)
```

Display

```
1
2
3
5
```

Pass Statement

- Pass statement is used as a place holder for two purposes-

  (i) To avoid compilation error

  (ii) To use a future code

- When pass statement is executed, nothing happens.
- It is used because empty code is not allowed in loops, function definitions, class definitions or in if statements.
- If empty code is used an error occurs and to avoid this pass code is used.

Example-1

```
for x in [0,1,2]:
    pass
```

Example-2

```
a = 5   b = 7
if b>a:
    Pass
```

OOPs

- OOPs full-form is object oriented programming system.
- It is a programming methodology using objects.
- Elements of OOPs are.

1. class
2. Object
3. Abstraction
4. Inheritance
5. Encapsulation
6. Polymorphism
7. Constructor

Four pillars of OOPs are :-

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Example of OOPs based programming languages are :-

1. Python
2. Java
3. Ruby
4. Javascript
5. C & C++
6. PHP
7. Cobol
8. Pearl

Class :- It is a collection of objects. eg. "Student."

Object :- It is anything with an identity. eg. "Ram".

Abstraction :- It means display only basic information and hide the details.

Encepsulation :- It is wrapping up of data and methods in a single unit.

Inheritance :- It is one class related with another class.

Polymorphism :- It means having many forms. It is any function or operator with more than one definitions.

Constructor :- It is used to initialize an object.

Class

- A class is a collection of objects.
- Class is created in python using keyword "Class". The syntax is –

```
Class    class_name :
```

- **Example** :- Create a class having name "student".
- **Code**

```
class student :
  Statements ....
```

A class cannot be empty. An empty class can be created as –

```
Class student :
    Pass
```

A class contains variables called attributes/properties inside it.

- **Example** :- Create a class having name "student" with variables (attributes) "name" and "age".
- **Code**

```
Class student :
    name = "   "
      age = 0
```

" " and 0 are the default values of attributes name and age.

Object

- An object is called an instance of a class.
- For example, If "student" is a class, then its objects can be "Student 1", "Student 2" etc.
- The syntax to create an object is

```
object_name = class_name ( )
```

Example :- Create a class having name "student" with attributes "name" and "age" and an object "Student 1".

Code:

```
Class student
    name = "   "
      age = 0
Student 1 = Student ( )
```

- A value can be given to object and its attributes as-

```
object_name. attribute_name = value
```

for example

```
Student 1.name = "Ram"
student 1.age  = 20
```

- The objects and their attribute values can be accessed as-

Student1.name - "Ram" - to access name attribute
Student1.age - 20 - to access age attribute

	Display
print (student1.name)	Ram
print (student1. age)	20

Constructor

- Constructor is a function used to create and initialize "objects" in a class.
- In other words, constructor is used to instantiate objects in a class.
- Syntax of constructor is-

```
def __init__( ):
```

- Rules of constructor

1. It must be named __init__()
2. The first argument must be "self" for example -

```
def __init__(self):
```

3. It must be defined inside a class.
4. Constructor can have any number of arguments or parameters. eg

```
def __init__(self, hame, age):
```

Types of constructors

1. Default Constructors
2. Non-parametrized constructor
3. Paramerized Constructors

Default constructors

When we do not define a constructor function, it is called a default constructor.

Example;
```
class student :
        name = " "
    Student 1 = Student ( )
    Student 1. name = "Ram"
    print (student 1. name)
```

Create class create attribute
create object
Value attribute
access

Display : Ram

Non-parameterized constructor

It consists of only one argument "self".

Example ;
```
Class student :
    def __init__ (self):
        Self.name = "Ram"
        self. age = 20

Student 1 = Student ()
Print (student 1. name)
Print (Student 1. age)
```

Display : Ram
20

Paramenterized Constructor

- It contains parameters or attributes.
- The first argument is fixed. ie "Self"
- The other arguments are defined by programmer.

Example ;
```
    class student :
def __init__ (self, name, age):
    Self. name = name
    Self. age = age
Student 1= Student ("Ram",20)
Student 2= Student (Shyam",22)
```

```
Print (f"{student1.name} {student1.age}")
Print (f"{student2.name} {student2.age}")
```

```
Student 1 = student ()
 Student 1.name = "Ram"
```

Code

```
Ram 20
Shyam 22
```

Create attribute "name"

" " "age"

" object and initialise their

" attributes eg-name = Ram

access the object attributes.

Inheritance

- In python, we can use the codes of an existing class in a new class. This is alled inheritance.
- The existing class is called base class or parent class or super class.
- The new class is called derived (inherited) class or child class or sub class respectively.
- Syntax

```
Class parent_class-name:
    # Body
Class child-class-name (Parent_class_name):
    # Body
```

Example ; Write code to create a parent class 'student' and child class "Boys" with empty code body.

Code

```
Class student:                # Create parent
  Pass                        # Empty code
Class Boys (student):         # create child
  Pass                        # empty code
```

Example Write code to create a parent class 'student' and child class 'Boys' with empty code body.

Code

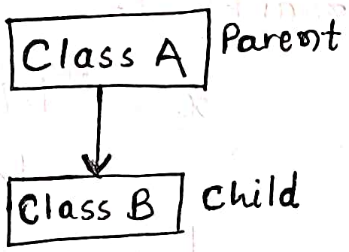
```
  class student :            # create parent
      Pass                   # Empty code
    class Bays (student) :   # Create child
      Pass                   # Empty code
```

Benefits of inheritance;

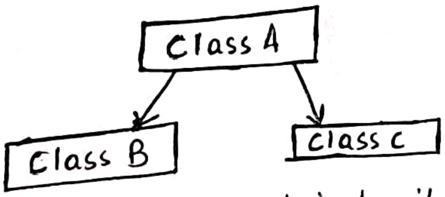
1. It provides real-world relationships.
2. It provides reusable codes we don't need to write the Same code again.
3. It provides simple and understandable programming structure.
4. It is transitive. If B inherits from A, then all the sub classes inside B also inherit from A.

Types of inheritance

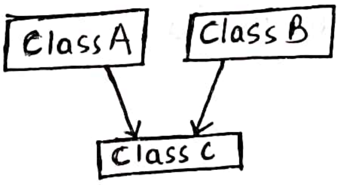
1. Single level inheritance
2. Multi level inheritance
3. Hierarchical inheritance
4. Multiple inheritance



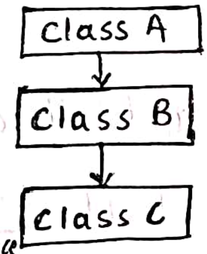
Single level inheritance



Hierarchical inheritance



Multiple inheritance



Multilevel Inharitance

Example Program

```
Class addition:                 # create parent class 'addition'
      def add (self,a,b)        # create function add () to
          print (a+b)           # add a&b and display
 Class do (addition):           # Create child class -'do'
          Pass                  # empty code
 number = do()                  # create objects-no. in child do
 numbers. add (5,3)             # Inheritance
```

Output : 8 — indent.

Python strings

- Strings in python are array of characters. for example -

  Polytechnic Academy

- String in python are written inside single or double quotation marks. for example

  'Polytechnic Academy' or "Polytechnic Academy" are same.

- A string can be displayed by print( ) function. ie-

```
Print ('Polytechnic Academy')
```

- A string can be also assigned to a variable and printed.

Eg.
```
x = 'Polytechnic Academy'
Print (x)
```

- String can be displayed on multiple lines using quotation marks. Eg.

```
x = """Polytechinic Academy ↵
       Patna - 13 """
Print (x)
```

Display :
```
Polytechnic Academy
Patna - 13
```

String Methods

1. Replace ( )

It is used to replace characters by another characters in a string.

Example ;x="Hello"
```
Print (x.replace('Hello','Hi'))
Print (x. replace ('H', 'J'))
Print (x. replace ('l', 'm'))
```

Hi → Jello → Hemmo

2. Join ()

Joins all the elements of a string using a seperator.

Example1 ; Join all the elements in a tuple using hash '#' as saperator.

```
x = ('Ram','Shyam', 'Batram)
x = '#'.Join(x)
Print (x) → Ram # Shyam # Balram
```

Example 2 ; Join elements of a dictionary

```
x = {'name: Ram', 'age' : 20}
x = "#".Join(x)
Print (x)  disp→ name #age
```

3. Split ()

It splits strings into substrings using a seperator.

Syntax : string var. split ('seperator')

Example

```
x = "Hellog World !"
Print (x.split (,))
```

Display : ['Hello','World !']

4. Reverse- reversed ()

It reverses the order of a list.

Example :

```
x = ["a","b","c","d"]
y = reversed (x)
from z in y:
    Print (z)
```

Display

```
d
c
b
a
```

5. Uppercase - upper ()

Sets string characters in capital letters.

Example

```
x = "Hello"
Print (x.upper(x))
```

HELLO

6. Lowercase - lower()

Sets string characters in small letters.

Example.
```
x = "Hello"
Print (x. lower())
```

hello

7. Count ()

Gives the number of times a character occurs in a string.

Syntax: string. count (Value, start, end)

- Start and end are optional.
- Default value is 0th position to end.

Example
```
x = " My state is Bihar. I love Bihar."
y = x. count ("Bihar")
Print (y)
 Z = x. count ("Bihar", 10,20)
Print (Z)
```

8. Find ()

It returns the position value.

Syntax: String find ("character")

Example
```
x ="hello"
y = x. find ("e")
Print (y)
```

9. Length - len ()

It returns the number of characters in a string.

Syntax : len (string)

Example;
```
x = "Hello World!"
y = len (x)
Print (y)
```

Python Function

function : A function is a block of codes or set of statements which only runs when it is 'called'.

Example   print ()

x = "Hello"
Print (x) ←
"Called"

Working

A function :-

(i) takes inputs,

(ii) do some function and

(iii) returns or gives result

Example : Print ("Hello") takes the input "Hello" and prints it on monitor.

Uses of function

1. To perform a certain function.
2. To reuse the block of codes. ie Write the codes once and use them many times.

Types of functions

1. Built-in functions or pre-defined functions. eg print ()
2. User defined functions

Built-in functions

Built-in functions are pre-defined functions in OOPs based programming languages.

Python built-in functions are -

abs ()
all ()
any ()
ascii ()

bin ()
bool ()
bytearray ()
bytes ()

Collable ()
chr ()
Compile ()
complex ()

delatter ()
dict ()
dir ()
dirmod ()

eval ()
exec ()
filter ()
float ()
format ()

getatte ()
globals()
len()
list()
range()

help()
hex ()
max()
mix ()
reversed()

id()
input()
int()
iter()
round ()

next()
print()
tuple()
types()

Slice()
Sorted()
Str()
Sum()
Super()

Description

1. abs () - returns absolute value of a number

   Ex -
```
x = abs (3+5j)
print (x)
```
   Output : 5.83095

2. all () - Returns true if all the elements in an iterable (list, tuple, dictionary, string etc) are true, otherwise returns false.
```
x = [1,2,3]
y = all (x)
Print (y)
```
   O/p ; True

3. Any () - Returns true if any item in an iterable is true, otherwise returns false.

4. ascii () - Returns ascii value american standard code for information incharge

5. bin () - Retuns the binary value

6. bool () - Returns boolean value

7. Complex () - Returns a comblex number

8. divmod () - returns the remainder and quotient of a division.

9. filter () - filters items in an ~~fato~~ iterable.

10. float () - returns a floating point number.

11. help() - returns built in help system. ~~hexa decimal value of a number~~.

12. hex() - returns hexa decimal value of a number.

13. Input() : allows user input.

14. int () - Returns integer value

ex. x = int(3.5)
    Print (x)   Output : 3

15. len() = Returns length ie number of items.

ex x = ['a', 'b', 'c']
    y = len (x)
    Print (y)   output : 3

16. max() = returns the largest number/item.

Ex. x = max (5,10)
     print (x)   output; 10

17. min() = returns the smallest item.

Ex, x = min (5,10)
     Print (x)   output ; 5

18. Oct() - Returns the octal value

19. Pow() - Returns the value of x^y

Ex - x = Power(4,3)
     Print (x)   Output ; $4^3 = 64$

20. range() - creates a sequence of numbers starting from 0.

x = range (6)
for n in x: Output ;
    Print (n)

0
1
2
3
4
5

21. Reversed () - Reverses the order of items in an iterable.

22. Round () - rounds a number.

x = round (5.76543,2)
print (x) .   output 5.77

23. Sorted () - Returns a sorted list

x = ('C','a','b') , y = sorted (x) , print(y) output ; 'a','b','c'.

Argument in a function

- Argument is information written inside parentheses or brackets().
- Multiple arguments can be written by seperating them with commas.
- Example;

```
def- function_1():                          # create fn-1
    print ("Hello")                         # fn_1. function
                          Argument
def - function -2(name)                     # create fn-2
    print (name +  "Sir")                   # fn-2 function

function_ 1 ( )                             Display
function - 2(Shyam)                         Hello
function- 3( Vikash)                        Shyam Sir
                                            Vikash Sir
```

Lambda Function

- A lambda function is a small anonymous function. Anonymous means it has no name.
- Comparison between lambda function and general function.

  (i) Lambda function has no name, ie it is anonymous.

  (ii) It contains 0,1 or more arguments like general functions.

  (iii) It contains only one statement or expression in the body.

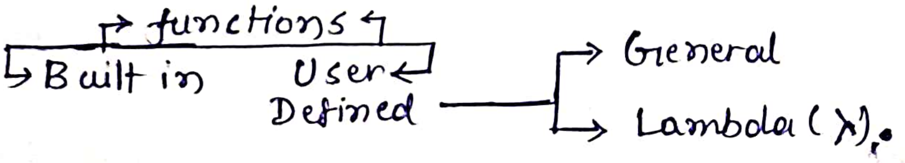
- Syntax and structure

```
fixed          user          user
lambda         arguments : Expression
```

Here, (i) lambda - is keyword

(ii) Argument - 0, 1 or more parameters eg. x, y, name, age....

(iii) Expression - the statement or body eg $x+y$



Example

1. x= lambda: Print ("Hello") Display : Hello
   x( )

2. x = lambda name : Print (name)
   x("Hello") Display : Hello

3. x = lambola a:a+10 Display : 15
   print (x(5))

4. x = lambda a,b : a+b
   print (x(5.6)) Display: 11

5. x = lambda a,b,c : a+b+c
   print (x (1, 2,3)) Display 6

Uses of lambda function

- To define a single statement function
- It can be used as an argument in another function.
- It is specially used with higher-order functions.
  eg. filter() , map ( ) , reduce( ) etc.

Advantages

1. Simple and Compact structure
2. It runs as soon as it is defined called IIFE (immediately invoked function execution)
3. It can be as an argument in other functions.

Disadvantages

1. It runs only one expression.

UNIT-3 "Cloud Features"

Cloud Compting

- Cloud is defined as a global network of servers. cloud is not a physical thing.
- Cloud computing is defined as a system which provides computing services over the internet.
- The computing services are –

(i) To store and manage data

(ii) Run Applications

(iii) Deliver content

(iv) Services such as streaming videos, web mail, social media etc.

(v) IoT services

Benefits of cloud computings :–

1. Low cost – No hardware or software required.
2. High speed services – A larger number of computing services, provided in minutes.
3. Global Scale – Delivers right amount of data and only when required.
4. Increased productivity – Main focus is on important goals.
5. Better Reliability – Data backup, disaster recovery etc.
6. Security – protection of data.

Types of cloud computing :–

1. Platform as a service. (Paas)
2. Software as a service. (Saas)
3. Infrastructure as a service. (Iaas)
4. Public cloud – cloud services for the public.
5. Private cloud – Cloud services for private network inside a building.
6. Hybrid cloud – It shares cloud services between public & private networks.
7. Community cloud – Shares cloud services between organizatio eg. gov, institution.

IoT cloud

IoT cloud is a cloud computing platform to create, install and monitor IoT projects in an easy way.

Some popular IoT cloud computing platforms are -

1. Amazon Web Services (AWS)
2. Microsoft Azure
3. IBM Watson
4. Google IoT platform
5. Oracle IoT
6. Cisco IoT cloud connect

Benefits of cloud in IoT

1. Scalability

- It is the biggest benefit.
- Without IoT cloud, upscaling of IoT networks requires -
  - i). Hardware
  - ii). Set up the configuration
  - iii). Lot of time and money
  - iv). Heavy maintenance
- These problems are eliminated by simply using more cloud space and increasing the storage capacity.
- Similary, downscaling can also be done.

2. Data Mobility

- The data stored on cloud server can be used at any place.
- Same IoT applications are smart factories, automatic cars, sensor networks, smart cards, smart phones, smart watches etc.
- IoT cloud allows to manage, process, analyze and update the data and devices remotely and in real time.
- IoT cloud provides real time data access.

3. Security / Reliability

- Data security is a main issue in IoT systems.
- IoT cloud provides reliable -
  (i). Authentication and
  (ii). Encryption.

4. Cost-effectiveness

- Cloud storage expense comes under operating expense.
- On-premise storage is capital expense.
- On-premise network requires greater investment for hardware, maintenance and IT solutions.
- IoT cloud service providers have flexible plans to suit our need.

5. Less time-to-market

It takes less time and effort to setup IoT system and also lowers the overall cost.

Types of cloud

1. Public Cloud

- It is created for open use by the public
- Public cloud provid services for anyone on the internet.
- It is useful for big infrastructure IoT systems.
- It provides reduced capital cost and operational cost.
- It is used by educational institudions, industries, gov. institutions business or enterprises and is open for all.

2. Private Cloud

- It is created for exclusive (privet) use by a single organization.
- It is used by institutions, industries, business, enterprises. and is meant for private use by the employees only.

3. Hybrid Cloud

- It combines two or more private, community or public clouds.
- It may also combine on-premise infrastructure and private cloud based services.

Cloud Services

1. Saas

- Saas stands for software as a service.
- It is also known as cloud application services.
- It is most commonly used services.
- Saas provides only application (app) to the user.
- The apps run directly through a web portal and no need to download and install.
- Responsibilities of CSP in saas are software control, maintain updation and infrastructure.
- Exmple of saas providers—

  Google Workspace, Dropbox, salesforce, cisco Webex.

Characteristics

- Managed from a central location.
- Hosted on a remote server.
- Accessed on internet.
- User not responsible for hardware or software update.

Advantages

- Saves time and money to install, manage and upgrade software.

Applications

1. Start-ups or small companies
2. Short-term projects
3. Less need application eg. tax software.

2. Paas

- Paas stands for Platform as a service.
- It is also known as cloud platform service
- It provides the users a platform to develop or create a software.

Characteristics

1. Easy to scale ~~dow~~ up or scale-down resouces.
2. Provides a variety of services.
3. Suitable for users who want to develop costomized applications.

Advantages

1. Simple and cost-effective to develop app.
2. Scalable
3. Highly available
4. Developers can customize apps.

Limitations

1. Data Security
2. Problem of integration of databases with Web services.
3. Runtime errors

Example;

1. AWS Elastic Beanstalk
2. Microsoft Azure
3. Heroku
4. Force.com
5. Google Application Engin
6. Openshift.

3. IaaS

- IaaS stands for Infrastructure-as-a-service.
- It is also known as cloud infrastructure service.
- It provides infrastructure to the user or developer such as data stores, servers, data centres and networking.
- Examples of IaaS are connecting cars, wearables, TVs, Smartphones, fitness devices, robots, ATMs etc.

Characteristics

1. Resource available in the service.
2. Cost depends on consumption.
3. Highly Scalable
4. Complete control of infrastructure.
5. Dynamic and flexible.

Advantages

1. Most flexible.
2. Hardware cost according to need.
3. Users have full control of infrastructure.

4. Highly Scalable

Iaas Providers

1. Digital Ocean
2. Linode
3. Rockspace
4. AWS
5. Cisco Metacloud
6. Microsoft Azure
7. Google compute engine (GCE)

Cloud Connectivity

Cloud connectivity represents an internet connection between a private network and a public cloud.

Types of cloud connectivity;

i). Using Public Internet

- It is conventional type cloud connectivity by simply using public internet.
- It is cheap but having following disadvantages:—

i). Security is not good

ii). Higher Latency

iii). Less Reliability

iv). Lower throughput

v). Poor cloud performance

- It can be of two types;

(i) Public Internet Based - Internet and cloud service both share the bandwidth.

(ii) Public internet and cloud prioritisation - a part of bandwidth is reserved for cloud service.

ii). Direct cloud internet

- It provides direct end-to-end private and dedicated cloud connectivity.
- It is modern type and has following advantages;—

(i) Better Security
(ii) Low Latency
(iii) More Reliable
(iv) Higher Throughput
(v) Cost effective
(vi) Good performance

- Examples of service providers –
  1. Amezon. AWS direct contact
  2. Google cloud dedicated interconnect
  3. Microsoft Azure Express Route
- It can be of following types –

(i) Directly ethernet cloud connect - dedicated connectivity via ethernet.

(ii) Wave cloud connect - also called optical or wavelength cloud connect.

(iii) MPLS VPN cloud connect - multi-protocol Label switching Virtual Private Network

(iv) SD WAN cloud connect software defined WAN.

Data storage on cloud

- Data storage on cloud called cloud storage.
- It is ~~a~~ made of computer data storage in which digital data is stored on servers.
- Cloud storage provides - store data, access data & manage data.
- Advantages;
  1. No need of own data centre
  2. Low moving cost
  3. Scalable easy to scale up or scale down
  4. Security
  5. Flexible - how to store and access data.
- Disadvantages;
  1. Latency Problem - due to slow internet.
  2. Data manage control limitations.
  3. No internet no data.

- Modes of Cloud storage,

1. Public - Data is stored in CSP's server.
2. Private - Data is stored in users data centre.
3. Hybrid - Mix of public and private cloud storage.
4. Multicloud - Data is stored in more than one CSP's servers.

- Types of cloud storage;-

1. Object ; Each place of data is an object and stores it in bundles.
2. File ; Organizes data in files and folders.
3. Block ; organizes data in block - suitable for large volume of data.

Arduino

- Arduino is an open source platform of electronic embedded System projects.
- Arduino designs and manufactures microcontroller based electronic boards and its software.
- Arduino features are;-

(i) Straight - forward design
(ii) Simple and powerful
(iii) Design according to user's need
(iv) Useful for students and professionals.
(v) Do not need much technical knowledge

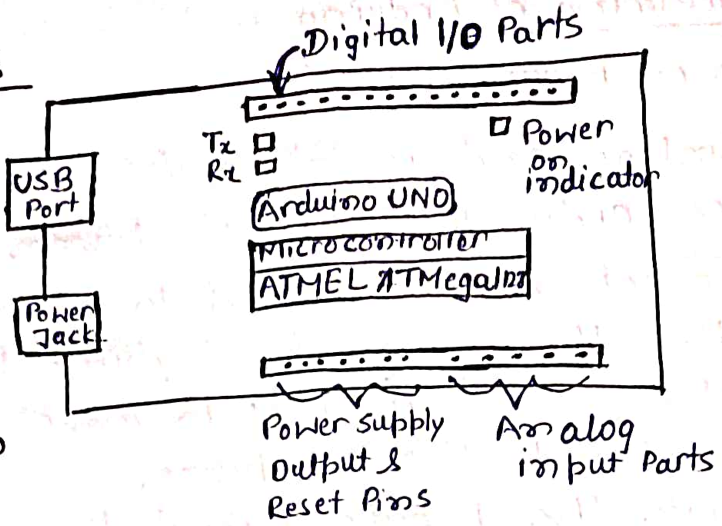
- Applications;

1. Smart homes
2. Public Utility Automation
3. IoT
4. Defence (Radars)
5. Medical
6. Laboratories
7. Aerospace
8. Automatic Vehicle control.

Arduino Architecture

Arduino boards are of different types,such as-

1. Arduino UNO
2. Lily Pad
3. Red board
4. Arduino Leonardo
5. Arduino Mega
6. Arduino Shild



Architecture of Arduino board are;

1. USB Port ;- To connect arduino board with computer.
   - It is used to provide power supply.
   - Also programming of Arduino board.
2. Power Jack - To provide power supply directly from AC Power outlet using an AC to DC adaptor.
3. Micro controller - Microcontrolar is an single chip comptutar. In integrates CPU, memory and I/O in a single chip. It is the heart of embedded project. Arduino uses ATmega series of microcontrollers of ATMEL company.
4. Digital I/O Ports - To connect digital input and output devices to the arduino board. eg. switch, digital sensors, LED, DC motors etc.
5. Analog input ports - To connect analog input devices eg. analog sensors.
6. Power Supply & Rest pins - To reset the board and to provide power supply to external components.
7. LED indicators - for power on, transmit data (Tx), Receive data (Rx).

Programming of Arduino

Steps of programming;-

1. Download and install arduino IDE software in computer.
2. open arduino IDE in computer.
3. Connect Arduino board to computer via USB cable.
4. Select the board in Arduino IDE.
5. Select our serial port to which Arduino board is connected to the computer.
6. Write the program in Arduino IDE, an arduino program is called sketch.
7. Upload the program in Arduino board.

Arduino IDE

- IDE - Integrated Development Environment
- It is available on www.arduino.cc.
- Arduino programs or codes are created in Arduino IDE.
- Arduino codes are called sketch.
- Arduino codes are based on C/C++.
- Arduino IDE is written or created using "Java".

Raspberry Pi

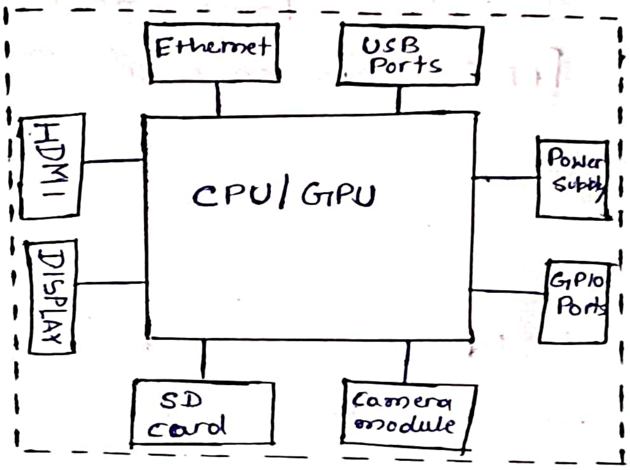
- Raspberry Pi is a computing and digital technology platform.
- It is of Raspberry foundation - a UK based charity mission.
- Raspberry Pi is a low cost credit card size small single board computer (SBC).
- The Raspberry Pi boards available are;-

  (i) Raspberry Pi in 2012
  (ii) Raspberry Pi 2
  (iii) Raspberry Pi Zero
  (v) Raspberry Pi 3
  (V) Raspberry Pi 4
  (vi) Raspberry Pico
- Features of Raspberry Pi are;-

1. GPIO Pins - General Purpose I/O (GPIO) Pins to connect sensors actuators and motors etc.
2. Low Power consumption.

3. OS support - Raspbian, Linux, Ubuntu, Windows 10, IoT care.
4. Compact size eg. 8.56 x 5.4 x 1.7 cm.
5. Video and graphics support.
6. Wireless connectivity - WiFi bluetooth suitable for IoT applications.
7. Expandable - Support for external micro SD cards USB pen drives external hard disc drive.

- Architecture



1. CPU :- 32 bit processor commonly used in mobiles.
2. GPU :- Graphic processing unit to play HD Video & video games.
3. Memory :- 1GB RAM
4. Micro SD card - To store OS and user data.
5. GPIO Pins - To connect sensors, LED's, actuators, motors etc.
~~6~~. HDMI - High definition media input.
7. Display - eg. a monitor
8. Ethernet, USB ports, Power supply and camer module

- Applications

1. Education - In school and colleges to empower learning of computing.
2. Smart home automation - To control hom appliances through smart phone.
3. Media centres - Using GPU to play HD video.
4. IoT Internet of things - Using wireless connection & sensors to monitor temperature and humidity.
5. Gaming
6. Robots and drones operate through smart phones.

Programming

Programming language are mostly used for Raspberry Pi are—

1. Scratch - It is a block based programming language.
2. Python - Using Python IDE. It is a poweful language.
3. C/C++ - C or C++ object oriented programming languages.

[IDE - Integrated Development Environment]

UNIT - 4

Networking and Application

Network :- A network is a collection of two or more objects that are linked together for a purpose. E.g. communication.

Wired Network :-

- Wired network is a network in which devices are connected via cables.
- "Wired" actually means "via cables".
- The cables used in wired network can be twisted-pair cable, co-axial cable or optical fibre cable (OFC).
- Example - Ethernet - A wired network of computers in LAN.

Short-range wireless network

- Wireless network is a network in which connecting medium between devices in air.
- The devices communicate through EM waves or radio waves or light - visible or IR.
- Short range wireless networks are wireless networks having range in metres.
- Examples -
  1. BLE - Bluetooth Low Energy (15-30 m)
  2. ZigBee (10-100m)
  3. RFID - Radio Frequency Identification (200m)
  4. NFC - Near Field Communication (10cm - 1m)
  5. LoRaWAN - Low Range Wide Area Network
  6. Z-Wave - 30m (indoors). 100 m (outdoors)

M2M communication

- M2M stands for machine-to-machine.
- M2M refers to a connected network of machines such as home appliances, smart meters, ATMs, vending machines etc.

- M2M is a subset of IoT. The new name of M2M is IoT. M2M is synonymous etc.
- In M2M communication, two machines communicate i.e exchange data without human interaction.
- The purpose of M2M communication is
  (i) data exchange
  (ii) remote monitoring and control
- In M2M, machines may communicate via wired or wireless means.
- Wired means - twisted pair wire, co-axial cable or power lines.
- Wireless means - wireless network. Eg. cellular mobile network, WiFi etc.

Application

1. Veding machines - send inventory report
2. ATMs - get message to dispense cash
3. Industrial telematics eg. smart meters.
4. Manufacturing industry
5. Home appliances eg. washing machines
6. Healthcare device management
7. Smart utility management

M2M	IOT
1. It is related to machines.	1. It is related to everything.
2. Homogeneous . ie things are machine only.	2. Heterogeneous . ie variety of things.
3. Emphasis is more on hardware	3. On software
4. Generally uses on premise data storage	4. Cloud storage
5. Protocols used - zigbee, bluetooth, modbus etc.	5. HTTP, MQTT etc.

Cellular Networks

- Cellular networks provide long-range wireless communication.
- Cellular network standards are 2G, 3G, 4G and 5G. These are called generations of standards of mobile telephone.
- 2G and 3G are legacy standards ie not used now.

Note: Some long-range wireless communication standards made especially for IoT are NB IoT and LTE-M.

- A cellular network is a network of cells of hexagonal shape.
- Cellular network provides communication by deviding a geographic area in a number of cells.

Generations of cellular networks

- The first wireless communication was done by marconi in 1895.
- Cellular mobile communication was invented in 1947 & commercial use started in 1971.

1G (1st generation cellular technology)

- 1G started first in 1979 in Japan.
- It was an analog technology.
- It was used for only voice communication.
- Frequency band - 800 MHz/900MHz.
- Demerits
  - (i) Poor voice quality
  - (ii) Poor battery life
  - (iii) large size mobile phone
  - (iv) Less security
  - (V) Limited coverage area
  - (Vi) No roaming.

2G (2nd generation)

- 2G started in 1991 in finland.
- Main features :-
  - (i) It was a digital technolgy.
  - (ii) It provided SMS service.
  - (iii) Roaming facility.
  - (iv) Increased security i.e encryption.
  - (v) Interinet facility.
- Communication standards used:-
  - GSM - Global System for mobile
  - CDMA - Code Division Multiple access
- Frequency bands used:-
  - 900 MHz / 1800 MHz / 1900 MHz
- Av. Data speed - 100 kbps = 0·1 mbps

3G (3rd generation)

- 3G started in Europe and Japan in 2001 & 2002 resp.
- It started in India on 11 Dec, 2008 by MTNL.
- Main features
  1. Multimedia ie audio, video and image downloading.
  2. WiFi, video conferencing and voice data service.
  3. Maps and navigation
- Communication Standards -
  - i. GSM
  - ii. GPRS - General Pocket Radio Service (2·5G)
  - iii. EDGE - Enhanced data rates for GSM evolution (2·75G)
- Average data speed 3Mbps.

4G (4th Generation)

- 4G started in 2011.
- Main features
  (i) Mobile internet access and smart phones.
  (ii) Telephone over internet
  (iii) High Definition (HD)Mobile TV
  (iv) Online Gaming.
  (v) Video conferencing
- Frequency bands
  2100MHz/1800MHz/850MHz/900MHz/2300MHz/2500MHZ
- VoLTE - Voice over LTE
- 4G standards
  WIMAX, LTE (Long Time Evolution)
- Av. download speed - 14 MbPS (max.150mbPS)

5G (5th Generation)

- 5G started in 2022.
- Features :-
  1. Network capacity - 10^4 times than 4G
  2. Peak data speed - 10 GbPS
  3. Cell edge data speed - 100MbPS.
- Applications :-
  1. Smart homes and smart cities
  2. Autonomous driving
  3. Ultra high speed data network
  4. Health care applications
  5. IoT

Frequency Bands

1. Low Band - 600-850 MHz
   - Lowest speed 5G
   - Largest coverage area

2. Mid band - 2-6GHz

- Speed greater than low-band
- Coverage - smaller than low band

3. High band - 25-39 GHz

- Highest speed
- Smallest coverage area

Low Power Wide Area Network (LPWAN)

- LPWAN is a category of technology made for low-power long range wireless communication.
- It was developed due to problems faced in 2G-GSM and CDMA, for IoT communications.
- LPWAN is ideal for large-scale use of low power IoT devices such as wireless sensors.
- Communication protocols for IoT is basically of two way
  - (i) Short-Range Network
  - (ii) LPWAN - long range

Popular LPWANs

1. Unlicensed - WiFi ; Bluetooth, LoRa, Sig Fox, 802.15.4
2. Licensed - LTE-A, LTE-M, NB-IOT, LTE-A, LTE advanced

Importance of LPWAN :—

1. Provide large coverage area at low cost, not possible with cellular network.
2. To provide long battery life of IoT devices.
3. To decrease the production cost
4. To provide more reliable connection to IoT devices in remote area.
5. To decrease the requirement of infrastucture.
6. To provide cheaper wireless technology.

Applications

1. Almost 5 Billion devices connected in LPWAN up to 2022.
2. Smart Meters
3. Smart meters/cities eg. smart parking, smart waste management.
4. Ideal for IoT.

Sigfox

- Sigfox is first LPWAN technology introduced in 2012.
- It is an ultra narrowband technology used for IoT and M2M communication.
- Main features
  1. Extremely low power consuption.
  2. Data speed - 100 bps
  3. Range - 10 km (Urban)
     50 km (Rural)
  4. Low cost
  5. Very long battery life - up to 20 years
  6. Operating frequency - 868 MHz (EU - European Union)
     902 MHz (USA)
  7. Ultra Norrowband (UNB) - Do not disturbs With other communication systems.
- Applications

Sigfox is used for IoT and M2M communication unication

Some common applications are ;-

i). Smart city application.eg. smart energy meters, street lighting etc

ii). Asset tracking

iii). Automotives

iv). Medical healtcare etc

Architecture & Working

Its basic elements are IoT Sensor nodes or end devices, base station, Sigfox cloud and ~~d~~ network server and costomer's IT.

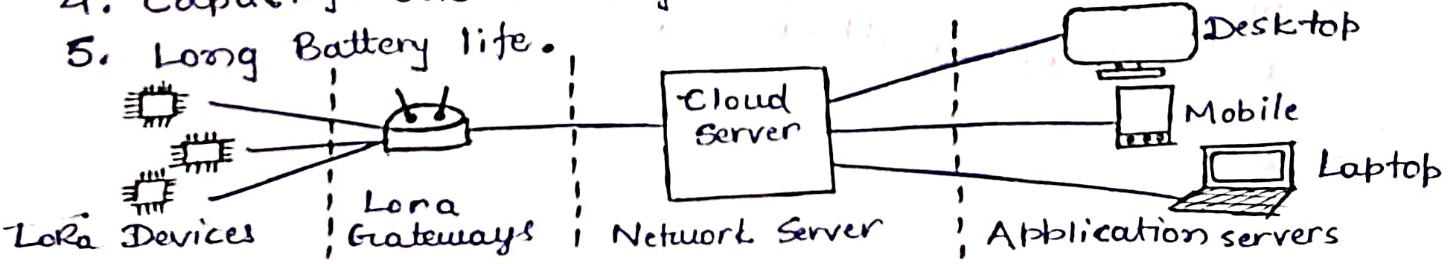
Working

- IoT nodes send data to base station by radio waves.
- The base station, also called gateway, receives the data by the nodes.
- The base station sends the nodes data to sigfox cloud.
- The cloud server sends data to either costomer or anyother node.

LoRa

- LoRa stands for Long Range radio (Wireless) and is mainly used for machine-to-machine (M2M) and IoT networks.
- LoRa is a long range, low power communication tapology.
- Features of LoRa :-

1. Range - 2-5 km Urban, 15 km suburban
2. Operating frequency - Ism (Industrial scientific Medical) band of 868/915 MHz. India-865-867 MHz.
3. LPWAN (Low Power Wide Area Network) statement - IEEE 802.15.4 called LoRaWAN.
4. Capacity - One LoRa gateways connects to 1000's of Nodes.
5. Long Battery life.



The element of LoRa communications are :-

(i). LoRa Devices
(ii). LoRa Gratways
(iii) Notwork server
(iv) Application servers

LoRa Devices - LoRa devices is the physical layer which which are chips connected through LoraWAN which is the MAC (media access control) layer. LoRaWAN is a softwareloaded in the LoRa chips. LoRa devices are end IoT devices that send and receive on the LoRa wireless Network.

LoRa Grateways - It works as a relay. It's task is to receive messages from end devices (LoRa devices) and forward them to the network server and vice-versa. Grateways sends messages via Wi-Fi, Ethernet or cellular mobile to the network server.

Network Server : It manages and maintains the LoRa Network, when required it provides data of LoRa devices to application servers.

Application Server : It is the application layer which provides user interface to LoRa network.

Application of LoRa

Lora provides IoT application in :-

1. Agriculture : Smart agriculture, Preserving three health, coop and soil monitoring.
2. Buiddings : Smart buildings, low power long range connectivity to view building functions.

3. Cities :- Smart cities with inhanced performance, optimize resources, reduce waste, consumption and costs.

4. Industry :- Smart factories, machines, infrastructure, managements etc.

5. Logistics :- Supply chain monitoring (end-to-end).

6. Utilities :- Smart services for water, electricity, gas and heating.

Sigfox	LoRaWAN (LoRa)
1. Bandwidth -100Hz (ultra narrow Band)	1. Bandwidth 250 KHz
2. Data - 100 bts (bits persec)	2. Data rate 20 kbps
3. Coverage range; Urbon 10km, Rular 50km	3. Coverage range; Urbon 2-5 km, Rular 15 km
4. Energy Consumption - V. low	4. Energy Consumption - V. low
5. Device Network - Star topology	5. Device network - Star topology
6. Open-Source (Unlicensed)	6. Open source (unlicensed)
7. Battery life - V. long (upto 20yrs)	7. Battery life - Very long

RFID

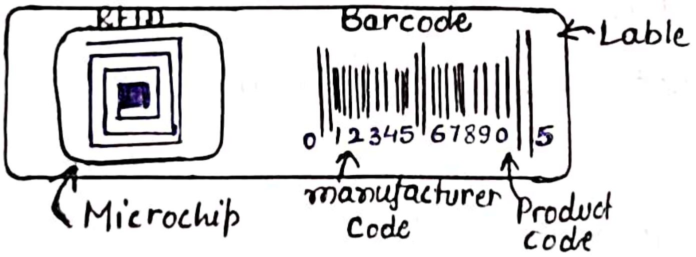
- RFID stands for Radio Frequency Identification.
- RFID is a wireless communication technology.
- Functions of RFID are -
  - (i) To identify tags attached to physical object.
  - (ii) To read the information stored in the tags.
  - (iii) To modify (write) information in the tags.
- Kevin Ashton was working on RFID technology when he coined the name Internet of Things.
- RFID is a major technology in IoT and is used to collect raw data.

Application of RFID

1. Inventory management
2. Asset tracking
3. Personnel tracking
4. I-cards
5. Supply chain management
6. Indian Railways (IR) kavach system
   - traffic safety control system for tracking the trains.
7. Smart shoping
8. Healthcare & Pharmaceuticals
9. Agriculture
10. Controlling home appliances.

Barcode

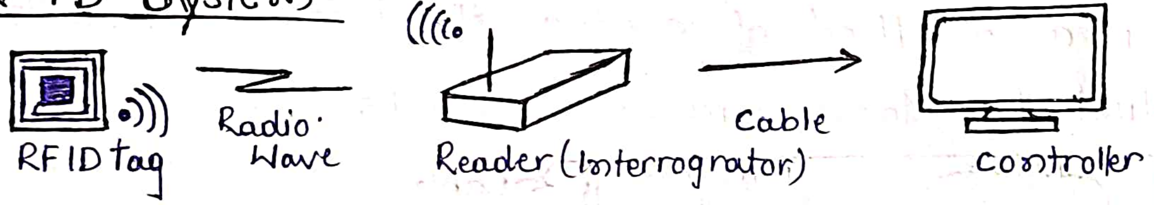
- A barcode is a ractangular block containing black and white lines or bars of different widths.
- It is a method of coding the information about a product. Eg. Manufacturer, product type etc.



- The bars may represent numericals or alphabets.
- The most common coding format used is UPS (Universal Product code) format.
- A barcode system consists of three parts -
  (i) Printer (ii) Scanner (iii) Computer
- Printer is used to print barcode lablel or directly print the barcode on product.
- The scanner is an optical laser light operated devices. It is used to read the barcode and is also called barcode reader.
- Computer receive data from the scanner and uses it for any purpose such as printing the bill.

	Barcode	RFID
1. Technology	Optical Laser	Radio Frequency (RF)
2. Reads	Only one tag at a time	1000's of tags at a time
3. Read Range	Same Feet	1000's of feet
4. Read speed	Slow	Very Fast (ms)
5. Read Requirements	Line of sight (No obstruction)	Can read through objects eg metal & Water.
6. Cost	Low	High
7. Type	Read-only	Read/Write both
8. Disturbance	obstruction, dirty or torn lables	Almost negligible
9. Durability	Less	More
10. Working	Manual	Generally Automatic
11. Data capacity	Low	High
12. Installation.	Simple & Easy.	Required spacial equpments or complex to install.

RFID System



The basic component of RFID system are :-

1. RFID Tag

- It is a label consisting of a semiconductor microchip, an antenna and a battery.
- It is also known as smart label or transponder.
- Its function is to give response when interrogated by the reader. (transmitter + responder)

2. Reader

- It consist of a RF module, control module and an an antenna.
- It's function is to interrogate the RFID tag and is also known as interrogator.

3. Controller

- It is a computer, it is also known as host or workstation.
- It's function is to receive data from the reader which it collects from the RFID tag.

Working of RFID system

- The reader sends the radio signals to the RFID tag. ie interrogates the RFID tag.
- The RFID tag responds and sends data to the reader such object type, location etc.
- The reader sends this data to the controller and the controller uses it for desired purpose.

Types of RFID system

RFID System can be of two types:-

(i) Passive RFID system (ii) Active RFID system

Passive RFID system

- In this type, the RFID tags are battery less and are activated by the radio signal received from the reader.
- These type of tags are called passive tags and are also known as transponders.
- These are mostly used type. These are cheaper, smaller and easier to produce.

Active RFID System

- In this type, the RFID tags contain battery or solar cell.
- These are called beacons which are self responding and send data to the reader after evevery few seconds.

RFID system Features

- Data Speed 4 mbps
- Frequency bonds - 125 kHz, 13.56 MHz, 902-928 MHz.
- Power - Ultra low Power
- Topology - P-2-P - Peer to Peer
- Range - 200 m
- Security - RC4 standard (less secure but very fast than AEs standard)
- Low cost
- Common application - tracking

Components of RFID system

1. Data

- RFID data is programmed into RFID tag.
- RFID tag transmits this data which can be transferred to internet through a gateway (reader).
- RFID data size can be 16 KB - 64 KB.
- RFID data speed is 4Mbps.
- RFID data type can be header, domain manager, object class and serial number.

Example 01.0000A89.000016F.0001b9DC0

Header formate

Domain Manager eg. company

object class eg. object typ

serial

2. Tags

RFID Tags are of two types, ie Passive tags and Active tags.

Passive tags;

- These are battery less tags and only contain microchip and antenna.
- These tags are transponders i.e respond only when Interrogated by the RFID reader.
- These tags are mostly used and are simple and chaip.

Active tags;

- These are battery or solar powered tags.
- These tags are like becans which continuously transmit data after every few seconds.
- Microchip, antenna, battery etc.

3. Antennas

- Antennas are provided in RFID tags and the reader.
- In RFID reader, the antenna can be seperate.
- Different types of antennas are used such as pannel, portal shalf, ground or desktop dype.
- Mostly pannel type antennas are used.

4. Connectors

Connectors are end points of Cables used to connect reader to antenna and the controller.

The connectors can be of following types –

1. RS 232 9-pin connector
2. RJ 45 Ethernet (LAN) Connector
3. USB type A or B
4. RP-TNC, SMA and N-type connectors for coaxial cables.

5. Cables

1. Co-axial cables to connect antenna
   - (a) RP-TNC - N type
   - (b) RP-TNC - RP-TNC
   - (c) RP-T~~N~~C - SMA

   etc.
2. Cables to connect reader to computer or host.
   1. Ethernet (LAN) twisted-pair cable (RJ45)
   2. USB cable (type A or B)
   3. RS232 cable

6. Readers

- Reader is also called interrogator.
- It communicates with the RFID tag throgh radio waves and sends information to a computer. It is a gateway.
- It can be of two types.

  (i) Read-only - This type of reader only reads the RFID tag.

  (ii) Read/Write - This type can read as well as write data in tag.
- Both readers can be handheld on fixed-mount type.

7. Encoders and Printers for smart labels

- Encoders are used to write initial data in tags.
- The encoding format can be Hexadecimal or ASCII codes.
- Printers are then used to print smart lables.
- RFID Printers can be –

a). Desktop Printer :- Most common type used in offices &

production unit. Easy to use.

b) Hand Held - Portable printers used in field onsite. Used in inventory management, asset tracking etc.

c). Industrial - Used in manufacturing plants and warehouse for mass printing.

d). In-line - Used in production-line, on move. Eg. Automobile indus

e). Software based - Printing through a software installed in a computer.

8. Controller Software

The software installed on controller (computer) is used to manage the readers, the data received from the tags and passes it to the backend data base system.

RFID Advantages over barcodes :-

1. Laser Range :- Range of RFID is longer than barcodes. RFID range is 100's of meters and barcode range is few feet.
2. More Storage Capacity :- RFID tags store more data/info.
3. No line of sight (LoS limitations) :- RFID uses radio Waves Hence there is no LoS limitations. RFID tags can be read through many materials like metals and water.
4. More Durability :- RFID tags are more durable than barcode.
5. More Capacity :- RFID reader can read 1000's of tags at a time, Barcode can read only one at a time.
6. Read / Write :- RFID readers can read / Write both in RFID tags, Barcodes can be read only.

Disadvantages

- RFID is more expensive than barcodes.
- RFID needs spacial treatment.

UNIT-5

Framework Selection for IoT app development

- IoT framework is a platform to develop IoT applications.
- It provides all the solutions to IoT app development.
- IoT framework is a set of tools and technologies used to build, desplay and manage IoT apps and devices.
- Examples; AWS IoT, Microsoft Azure IoT, Google Cloud IoT, Cisco IoT, Homekit, Android Things etc.
- IoT frameworks are of two types:—
  (i) Proprietary (ii) Open source - free source code

Selection of IoT framework

- No IoT framework is best.
- selection of IoT framework depends on the type of IoT app development.
- It is necessary for the success of IoT project.
- Selection of IoT framework can be done according to following factors :-

1. Future proof
2. Security
3. Pricing model
4. Time-to- value
5. Technology

Future Proof

- IoT platform which can envolve with the market.
- It should provide regular updates according to new standards and protocols of IoT.
- It should be secure and reliable.

Security

It is necessary to throughly study the security features in a given IoT framework.

Pricing

- Understand the pricing module.
- Speak to the company team.
- Look for hidden costs.

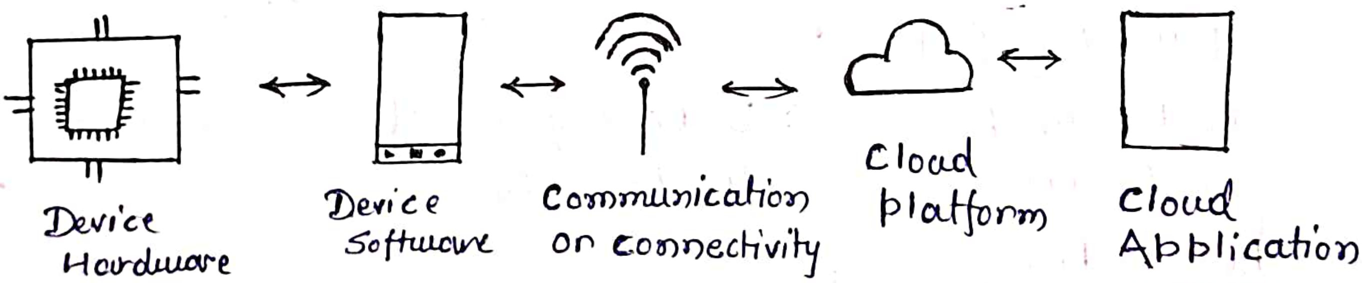
Time to value

- Platform which provide end-to-end solutions or complete Solutions.
- It should provide full suite (package) so that we dont need any other platform for a part of project.
- It reduces the time of IoT app development also marks it reliable.

Technology

- The success of IoT projects depends on technologies used.
- for example - IoT cloud is a powerful technology, than on premises technology.

Stages of IoT app development



1. Device Hardware:
   - It is the first stage of IoT app development.
   - It defines the physical parts of IoT.
   - Factors of hardware selection are size, deployment, cost, useful lifetime, reliability etc.
   - Example:, Sensors, microprocessors, microcontrollers etc.
   - Industrial and automatic sensors must be more reliable.
   - Smartwatches need oxymeters, pedometers & pulse monitor.
   - Smart vehicle need GPS, altitude monitorsetc.

2. Device Software

- It is an important stage.
- Device side software is called firmwork, It is the software which is used to runs the hardware.
- Firmwarke requirements are -
  - (i) Connection loss don't lead to data loss.
  - (ii) Compression of data
  - (iii) Data encryption for data security
  - (iv) Reliable downlaads of images.

3. Connectivity

- This stage deals with selection of right method of connectivity.
- Example: WiFi, 2G, 5G, Bluetooth, LoRa, NB IoT, Zigbee etc.
- For office or home Bluetooth is better, it is free.
- For defices in motion or remote location, 2G, 5G willbe better, These are paid services.

4. IoT cloud

- Cloud platform provide necessary tools to create IoT application.
- Important factors are uptime, data security, debugging, stability, security, connectivity, migration, scalability.
- Example of IoT platforms -
  Azure IOT suite, Google Cloud, AWS, Home kit, Android Things etc.

5. IoT application

- It includes designing a prototype IoT app and its testing.
- Features of IoT apps are -
  - Device management
  - Dashbording
  - Device to mobile communications
  - Updates etc.