

Textbook of  
**Internet of Things**  
**Software Development**  
Grade - X



**National Vocational & Technical Training Commission (NAVTTTC)**

**Textbook of**  
**Internet of Things**  
**Software Development**  
**Grade – X**



**National Vocational and Technical Training Commission**  
**H-9, Islamabad**

**Author:** **Muhammad Umair** Lecturer, Department of Electrical, Electronics and Telecommunication Engineering, New Campus, University of Engineering & Technology, Lahore.

**Reviewers:**

1. Dr. Ahmed Mustafa, Chief Instructor, P-TEVTA, Lahore
2. Dr. Hina Khalid, Assistant Professor, UET, Lahore
3. Mr. Muhammad Anees

**Designing:** Gul Awan Printers, Blue Area, Islamabad.

**Edition:** Test Edition,

**ISBN:**

**Publishers:** National Vocational & Technical Training Commission H-9, Islamabad.

Website: [www.navttc.gov.pk](http://www.navttc.gov.pk),

All rights are preserved with the National Vocational and Technical Training Commission. No part of this book can be copied, translated, reproduced or used for guide books, key notes, helping books etc. without permission of NAVTTC.

# PREFACE

This book has been written to meet the requirements of Matric Tech to train the students in the Internet of Things (IoT) trade. This book is specific for IoT software development. Matric Tech in IoT has been introduced first time in the history of Pakistan. This textbook is the first effort to describe the topics related to IoT software development. A key attempt has been made to make the book interesting and useful. The chapters cover basic details understandable to the students of Matric Tech. Each chapter includes assessments in form of MCQs, short questions and long questions.

The book starts with the concepts of Python scripting and provide concepts of Operating Systems for IoT. Afterwards, we explain Databases and Cloud Systems. A separate chapter is written on Computer Security. The last chapter is on soft skills.

The focus while reading should be on understanding rather than crammed. It should be read conceptually. Perform all the activities and tasks to gain hands on experience on IoT. The sequence of chapters can be adjusted as per convenience of the tutor. Make sure you never skip a prerequisite of any chapter.

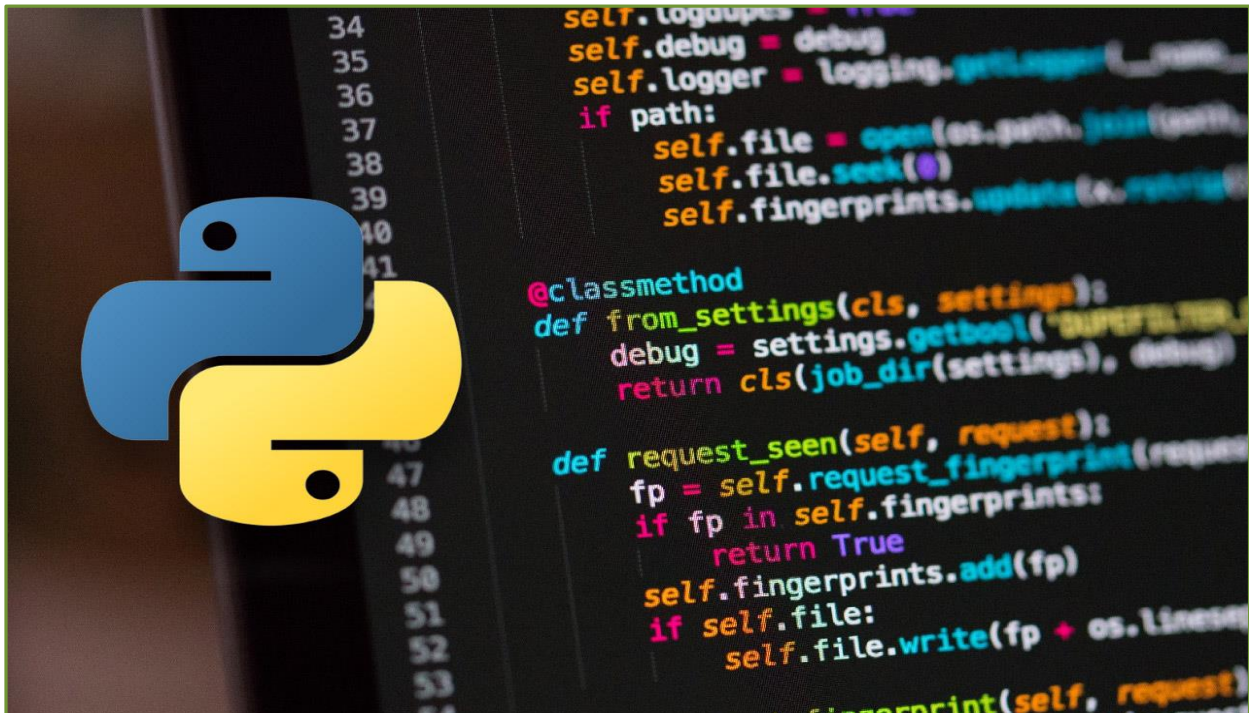
Any improvements and suggestions for the betterment of this book will be highly acknowledged.

**Executive Director**  
**National Vocational & Technical Training Commission**  
**(NAVTTC)**

## **TABLE OF CONTENTS**

<b>Sr. No.</b>	<b>Chapter Name</b>	<b>Page Number</b>
<b>1.</b>	<b>Python Scripting</b>	<b>1</b>
<b>2.</b>	<b>Operating Systems for IoT</b>	<b>35</b>
<b>3.</b>	<b>Introduction to Database</b>	<b>43</b>
<b>4.</b>	<b>IoT Cloud Deployment</b>	<b>64</b>
<b>5.</b>	<b>Basics of Datascience</b>	<b>77</b>
<b>6.</b>	<b>IoT Security</b>	<b>94</b>
<b>7.</b>	<b>Soft Skills</b>	<b>106</b>
	<b>Glossary</b>	<b>122</b>
	<b>About the Author</b>	<b>124</b>

# Chapter 1: Python Scripting



## After Studying this chapter, you will be able to:

- describe Python, describe structure of Python program.
- know life cycle of Python program, difference between script and a program.
- install Python interpreter.
- describe coding conventions of Python.
- use Python's interactive prompt.
- write and execute stored program in Python.
- understand data types and variables.
- understand mathematical operators available in Python.
- unary operators, binary operators.
- understand precedence of operators, implement arithmetic expression in Python.
- describe conditional statements and its types.
- understand nested decision statements, implement decision controls in Python.
- define concept of loop, describe FOR Loop in Python, describe WHILE Loop in Python.
- understand the handling of control variables for a loop, understand breaking a loop.
- implement loop in a Python program, understand nested loop.
- describe concepts of arrays, explain indexing and accessing an array.
- describe concepts of list, implementing arrays and lists in Python program.
- describe built-in functions and its types.
- understand return types, understand parameters.
- usage of built-in function.



## 1.1 Python

**Python** is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is designed to be highly readable. It uses English keywords frequently. Some of the key advantages of learning Python are:

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it.
- **Python is Interactive:** You can interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications.

### 1.1.1 Characteristics of Python

The Following are important characteristics of Python.

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to bytecode for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++ and Java.

### 1.1.2 Hello World using Python

The Following is a conventional Python Hello World program.

```
print ("Hello, Python!");
```

### 1.1.3 Advantages of Python

Python is one of the most widely used language over the web. Advantages of Python are:

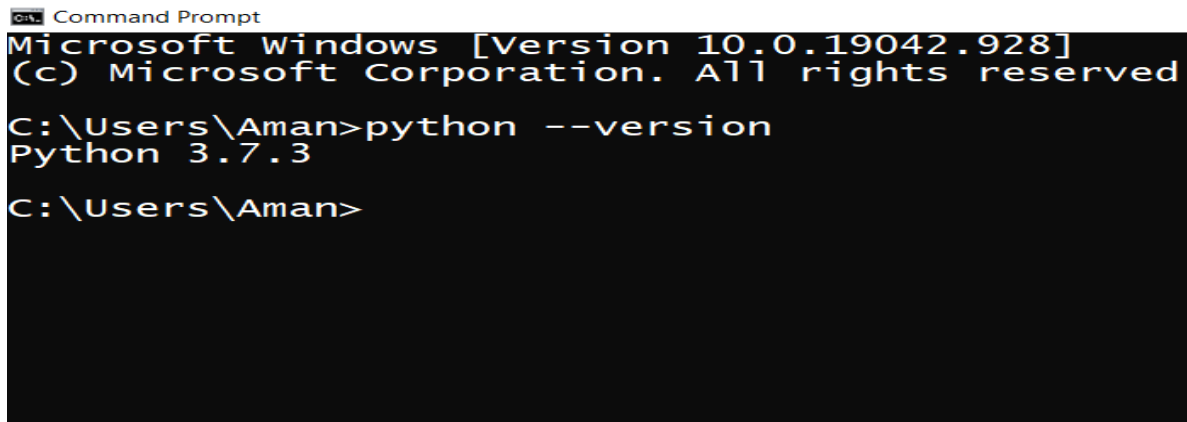
- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax.
- **Easy-to-read:** Python code is more clearly defined.
- **Easy-to-maintain:** Python's source code is easy-to-maintain.

- **A broad standard library:** Python libraries are very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive mode:** Python has support for an interactive mode which allows interactive testing and debugging of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **Scalable** – Python provides a better structure and support for large programs.

Python is available for a wide variety of platforms including Linux and Mac OS X.

## 1.2 Local Environment Setup

Open a terminal window and type "python --version" to find out if it is already installed and which version is installed.



```
Command Prompt
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aman>python --version
Python 3.7.3

C:\Users\Aman>
```

Fig. 1.1 Checking the version of Python

### Teacher's Note

Colab is a Google Research product, which allows developers to write and execute Python code through a browser



### 1.2.1 Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org/>. You can download Python documentation from <https://www.python.org/doc/>.

### 1.2.2 Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python. If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

### 1.2.3 Installation in Unix and Linux

The following are the simple steps to install Python on Unix/Linux machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- run `./configure` script
- `make`
- `make install`

This installs Python at standard location `/usr/local/bin` and its libraries at `/usr/local/lib/pythonXX` where XX is the version of Python. Python 3 is a newer version of the Python programming language which was released in December 2008. This version was mainly released to fix problems that exist in Python 2.

### 1.2.4 Installation in Windows

The following are the steps to install Python on Windows

- Open a Web browser and go to <https://www.python.org/downloads/>.
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.

- Run the downloaded file. This brings up the Python install wizard, which is easy to use. Accept the default settings and wait until the install is finished.

### 1.2.5 Setting up Path

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables. The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs. The *path* variable is named as PATH in Unix or Path in Windows.

#### Setting path in Unix/Linux

The following are the steps to add the Python directory to the path for a particular session in Unix:

- **In the csh shell:** type `setenv PATH "$PATH:/usr/local/bin/python"` and press Enter.
- **In the bash shell (Linux):** type `export PATH="$PATH:/usr/local/bin/python"` and press Enter.
- **In the sh or ksh shell:** type `PATH="$PATH:/usr/local/bin/python"` and press Enter.

#### Setting path in Windows

To add the Python directory to the path for a particular session in Windows:

**At the command prompt:** type `path %path%;C:\Python` and press Enter.

### 1.2.6 Running Python

There are three different ways to start Python:

#### Interactive Interpreter

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or a shell window.

Enter **python** in the command line.

Start coding right away in the interactive interpreter as the following:

```
$python # Unix/Linux
or
python% # Unix/Linux
or
C:> python # Windows/DOS
```

## Running Script from the Command-line

A "program" in general, is a sequence of instructions written so that a computer can perform certain task. A "script" is a code written in a scripting language. A scripting language is a type of programming language in which we can write code to control another software application.

A Python script can be executed at command line by invoking the interpreter on your application, as in the following:

```
$python script.py # Unix/Linux  
  
or  
  
python% script.py # Unix/Linux  
  
or  
  
C: >python script.py # Windows/DOS
```

## 1.2.7 Integrated Development Environment for Python

Python can be run from a Graphical User Interface (GUI) environment if you have a GUI application on your system that supports Python.

- **Unix:** IDLE is the very first Unix IDE for Python (see Figure 1.2).

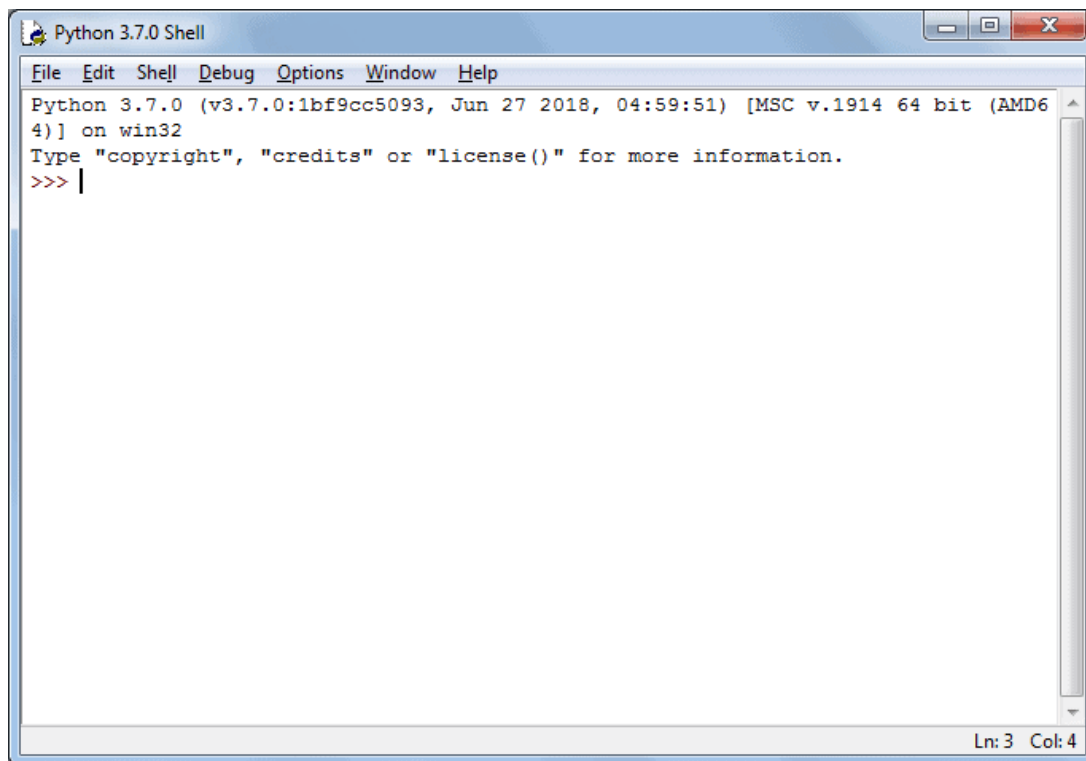


Fig. 1.2 IDLE for Python

- **Windows:** PythonWin is the first Windows interface for Python and is an IDE with a GUI.

The Python language has many similarities to Perl, C, and Java.

## 1.2.8 First Python Program

Python programs can be executed in different modes of programming.

### Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt:

```
$ python
Python 2.4.3 (#1, Nov 11 2022, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Type the following text at the Python prompt and press the Enter:

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in `print ("Hello, Python!");`. However, in Python version 2.4.3, this produces the following result:

```
Hello, Python!
```

### Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Python files have extension `.py`. Test the following source code in a `test.py` file:

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, run this program as follows:

```
$ python test.py
```

It produces the following result:

```
Hello, Python!
```

There is another way to execute a Python script. The following is the modified `test.py` file:

```
#!/usr/bin/python
```

```
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows:

```
$ chmod +x test.py      # This is to make file executable
$ ./test.py
```

It produces the following result:

Hello, Python!

## Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9). Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language.

The following are the naming conventions for Python identifiers:

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is private.
- Starting an identifier with two leading underscores indicates a strongly private identifier.
- If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

## Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For instance:

```
if True:
```

```
    print "True"
```

```
else:
```

```
    print "False"
```

However, the following block generates an error:

```
if True:
print "Answer"
```

```
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block.

The following example has various statement blocks:

```
#!/usr/bin/python

import sys

try:
    # open file stream
    file = open(file_name, "w")
except IOError:
    print "There was an error writing to", file_name
    sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
    file_text = raw_input("Enter text: ")
    if file_text == file_finish:
        # close the file
        file.close
        break
    file.write(file_text)
    file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
    print "Next time please enter something"
    sys.exit()
try:
    file = open(file_name, "r")
except IOError:
    print "There was an error reading file"
    sys.exit()
file_text = file.read()
file.close()
print file_text
```

## Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment:

```
#!/usr/bin/python
```

```
# First comment
print "Hello, Python!" # second comment
```

It produces the following result:

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression as follows:

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows:

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
```

The following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comment:

```
'''
This is a multiline
comment.
'''
```

## Types of Operators

Python language supports the following types of operators:

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

## Python Arithmetic Operators

Assume variable *a* holds 10 and variable *b* holds 20, then:

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	$a + b = 30$



- Subtraction	Subtracts right hand operand from left hand operand.	$a - b = -10$
* Multiplication	Multiplies values on either side of the operator.	$a * b = 200$
/ Division	Divides left hand operand by right hand operand.	$b / a = 2$
% Modulus	Divides left hand operand by right hand operand and returns remainder.	$b \% a = 0$
** Exponent	Performs exponential (power) calculation on operators.	$a^{**}b = 10$ to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. However, if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity).	$9//2 = 4$ and $9.0//2.0 = 4.0$ , - $11//3 = -4$ , - $11.0//3 = -4.0$

## Python Comparison Operators

These operators compare the values on either side of them. They are also called relational operators.

Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	$(a == b)$ is not true.
!=	If values of two operands are not equal, then condition becomes true.	$(a != b)$ is true.

<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

## Example

Assume a variable *a* holds 10 and variable *b* holds 20, then:

```
#!/usr/bin/python

a = 21
b = 10
c = 0

if ( a == b ):
    print "Line 1 - a is equal to b"
else:
    print "Line 1 - a is not equal to b"

if ( a != b ):
    print "Line 2 - a is not equal to b"
else:
    print "Line 2 - a is equal to b"

if ( a <> b ):
    print "Line 3 - a is not equal to b"
else:
    print "Line 3 - a is equal to b"
```

```

if ( a < b ):
    print "Line 4 - a is less than b"
else:
    print "Line 4 - a is not less than b"

if ( a > b ):
    print "Line 5 - a is greater than b"
else:
    print "Line 5 - a is not greater than b"

a = 5;
b = 20;
if ( a <= b ):
    print "Line 6 - a is either less than or equal to b"
else:
    print "Line 6 - a is neither less than nor equal to b"

if ( b >= a ):
    print "Line 7 - b is either greater than or equal to b"
else:
    print "Line 7 - b is neither greater than nor equal to b"

```

When you execute the above code, it produces the following result:

```

Line 1 - a is not equal to b
Line 2 - a is not equal to b
Line 3 - a is not equal to b
Line 4 - a is not less than b
Line 5 - a is greater than b
Line 6 - a is either less than or equal to b
Line 7 - b is either greater than or equal to b

```

## Python Assignment Operators

Assume a variable *a* holds 10 and a variable *b* holds 20, then:

Operator	Description	Example
=	Assigns values from right side operands to left side operand	<code>c = a + b</code> assigns value of <code>a + b</code> into <code>c</code>
<code>+=</code> Add AND	It adds right operand to the left operand and assign the result to left operand	<code>c += a</code> is equivalent to <code>c = c + a</code>

<code>-=</code> Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	<code>c -= a</code> is equivalent to <code>c = c - a</code>
<code>*=</code> Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	<code>c *= a</code> is equivalent to <code>c = c * a</code>
<code>/=</code> Divide AND	It divides left operand with the right operand and assign the result to left operand	<code>c /= a</code> is equivalent to <code>c = c / a</code>
<code>%=</code> Modulus AND	It takes modulus using two operands and assign the result to left operand	<code>c %= a</code> is equivalent to <code>c = c % a</code>
<code>**=</code> Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	<code>c **= a</code> is equivalent to <code>c = c ** a</code>
<code>//=</code> Floor Division	It performs floor division on operators and assign value to the left operand	<code>c //= a</code> is equivalent to <code>c = c // a</code>

## Example

```
#!/usr/bin/python

a = 21
b = 10
c = 0

c = a + b
print "Line 1 - Value of c is ", c

c += a
print "Line 2 - Value of c is ", c

c *= a
print "Line 3 - Value of c is ", c

c /= a
print "Line 4 - Value of c is ", c

c = 2
c %= a
```

```

print "Line 5 - Value of c is ", c

c **= a
print "Line 6 - Value of c is ", c

c //= a
print "Line 7 - Value of c is ", c

```

When you execute the above program, it produces the following result:

```

Line 1 - Value of c is 31
Line 2 - Value of c is 52
Line 3 - Value of c is 1092
Line 4 - Value of c is 52
Line 5 - Value of c is 2
Line 6 - Value of c is 2097152
Line 7 - Value of c is 99864

```

## Python Bitwise Operators

Bitwise operator performs bit-by-bit operation. Assume if  $a = 60$  and  $b = 13$ , their binary values are 0011 1100 and 0000 1101 respectively. The following table lists the bitwise operators supported by Python language with an example. We use the above two variables ( $a$  and  $b$ ) as operands:

$a = 0011\ 1100$

$b = 0000\ 1101$

-----

$a \& b = 0000\ 1100$

$a | b = 0011\ 1101$

$a \wedge b = 0011\ 0001$

$\sim a = 1100\ 0011$

The following bitwise operators are supported in Python language:

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
Binary OR	It copies a bit if it exists in either operand.	(a   b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

## Example

```
#!/usr/bin/python

a = 60          # 60 = 0011 1100
b = 13         # 13 = 0000 1101
c = 0

c = a & b;     # 12 = 0000 1100
print "Line 1 - Value of c is ", c

c = a | b;     # 61 = 0011 1101
print "Line 2 - Value of c is ", c
```

```

c = a ^ b;          # 49 = 0011 0001
print "Line 3 - Value of c is ", c

c = ~a;           # -61 = 1100 0011
print "Line 4 - Value of c is ", c

c = a << 2;        # 240 = 1111 0000
print "Line 5 - Value of c is ", c

c = a >> 2;        # 15 = 0000 1111
print "Line 6 - Value of c is ", c

```

When you execute the above program, it produces the following result:

```

Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
Line 4 - Value of c is -61
Line 5 - Value of c is 240
Line 6 - Value of c is 15

```

## Python Logical Operators

There are following logical operators supported in Python language:

Operator	Description	Example
and Logical AND	If both operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.



## Python Operators' Precedence

The following table lists operators from highest precedence to the lowest:

Sr.No.	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus (method names for the last two are +@ and -@)
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>> << Right and left bitwise shift
6	& Bitwise 'AND'
7	^   Bitwise exclusive 'OR' and regular 'OR'
8	<= < > >= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators

11	<i>is is not</i> Identity operators
12	<i>in not in</i> Membership operators
13	<i>not or and</i> Logical operators

### 1.3 Decision Control Statements

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions. Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE. The following is the general form of a typical decision making structure in most of the programming languages:

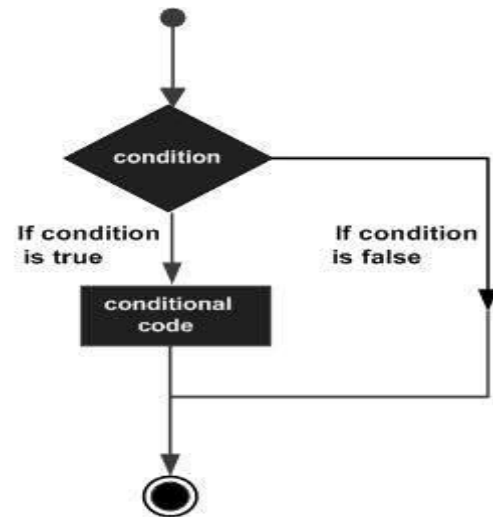


Fig. 1.3 Decision Control Statements in Python

Python assumes any non-zero and non-null values as TRUE, and if it is either zero or null, then it is assumed as FALSE. Python programming language provides the following types of decision making statements:

Sr.No.	Statement & Description
1	<u>if statements</u> An <b>if statement</b> consists of a boolean expression followed by one or more statements.

2	<u>if...else statements</u> An <b>if statement</b> can be followed by an optional <b>else statement</b> , which executes when the boolean expression is FALSE.
3	<u>nested if statements</u> You can use one <b>if</b> or <b>else if</b> statement inside another <b>if</b> or <b>else if</b> statement(s).

## Syntax

```
if expression:
    statement(s)
```

If the boolean expression evaluates to TRUE, then the block of statement(s) inside the *if* statement is executed. If boolean expression evaluates to FALSE, then the first set of code after the end of the *if* statement(s) is executed.

## Example

```
#!/usr/bin/python

var1 = 100
if var1:
    print "1 - Got a true expression value"
    print var1

var2 = 0
if var2:
    print "2 - Got a true expression value"
    print var2
print "Good bye!"
```

An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the *if* statement resolves to 0 or a FALSE value. The *else* statement is an optional statement and there could be at most only one **else** statement following **if**.

## Syntax

The syntax of the *if...else* statement is:

```
if expression:
    statement(s)
else:
    statement(s)
```

## Example

```
#!/usr/bin/python

var1 = 100
if var1:
    print "1 - Got a true expression value"
    print var1
else:
    print "1 - Got a false expression value"
    print var1

var2 = 0
if var2:
    print "2 - Got a true expression value"
    print var2
else:
    print "2 - Got a false expression value"
    print var2

print "Good bye!"
```

When the above code is executed, it produces the following result:

```
1 - Got a true expression value
100
2 - Got a false expression value
0
Good bye!
```

## The *elif* Statement

The **elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE. The **elif** statement is optional. However, there can be at most one **else** statement whereas there can be an arbitrary number of **elif** statements following an **if**.

## Syntax

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

Python does not provide switch or case statements as in other languages, but we can use `if..elif...` statements to simulate switch case as follows:

### Example

```
#!/usr/bin/python

var = 100
if var == 200:
    print "1 - Got a true expression value"
    print var
elif var == 150:
    print "2 - Got a true expression value"
    print var
elif var == 100:
    print "3 - Got a true expression value"
    print var
else:
    print "4 - Got a false expression value"
    print var

print "Good bye!"
```

The above code produces the following result:

```
3 - Got a true expression value
100
Good bye!
```

### Nested If

There may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, you can use the nested `if` construct. In a nested `if` construct, you can have an `if...elif...else` construct inside another `if...elif...else` construct.

### Syntax

The syntax of the nested `if...elif...else` construct may be:

```
if expression1:
    statement(s)
    if expression2:
        statement(s)
    elif expression3:
        statement(s)
    elif expression4:
        statement(s)
    else:
        statement(s)
else:
```

statement(s)

## Example

```
#!/usr/bin/python

var = 100
if var < 200:
    print "Expression value is less than 200"
    if var == 150:
        print "Which is 150"
    elif var == 100:
        print "Which is 100"
    elif var == 50:
        print "Which is 50"
    elif var < 50:
        print "Expression value is less than 50"
else:
    print "Could not find true expression"

print "Good bye!"
```

The above code produces following result:

```
Expression value is less than 200
Which is 100
Good bye!
```

## 1.4 Loops in Python

Generally, statements are executed sequentially. First statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times. Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or a group of statements multiple times. The following diagram illustrates a loop statement:

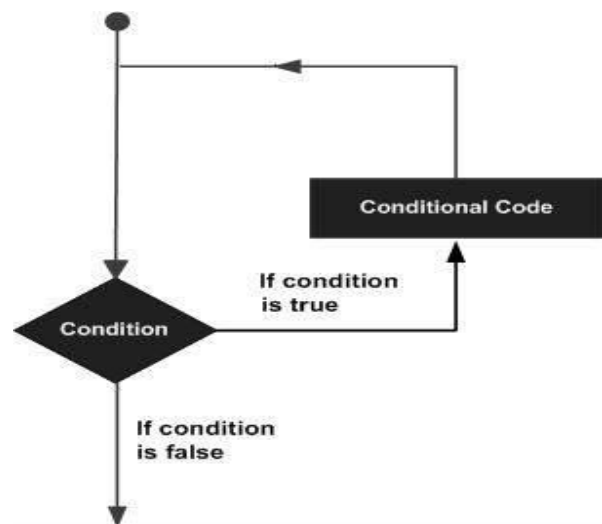


Fig. 1.4 Loops in Python

Python programming language provides the following types of loops to handle repetitions:

## While Loop

A **while** loop statement in Python programming language repeatedly executes a target statement if a given condition is true.

### Syntax

The syntax of a **while** loop is:

```
while expression:  
    statement(s)
```

Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and *true* is any non-zero value. The loop iterates while the condition is *true*. When the condition becomes *false*, program control passes to the line immediately following the loop. In Python, all statements indented by the same number of character spaces after a programming construct are part of a single block of code. Python uses indentation to group statements.

### Example

```
#!/usr/bin/python  
  
count = 0  
while (count < 9):  
    print 'The count is:', count  
    count = count + 1  
  
print "Good bye!"
```

When the above code is executed, it produces the following result:

```
The count is: 0  
The count is: 1  
The count is: 2  
The count is: 3  
The count is: 4  
The count is: 5  
The count is: 6  
The count is: 7  
The count is: 8  
Good bye!
```

## For Loop

*For* Loop can iterate over the items of any sequence of code.



## Syntax

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

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable *iterating\_var*. Next, the statements block is executed. Each item in the list is assigned to *iterating\_var*, and the statement(s) block is executed until the entire sequence is exhausted.

## Flow Diagram

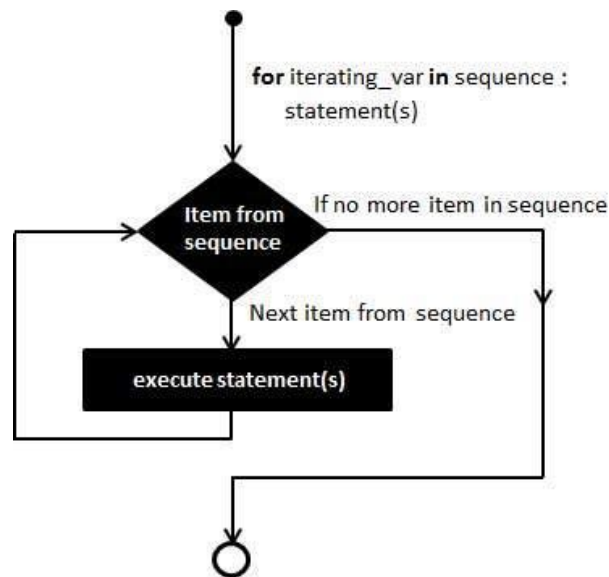


Fig. 1.5 For Loop in Python

## Example

```
#!/usr/bin/python

for letter in 'Python':    # First Example
    print 'Current Letter :', letter

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:      # Second Example
    print 'Current fruit :', fruit

print "Good bye!"
```

The above code produces the following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
```

```

Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!

```

## Nested Loops

Python programming language allows to use a loop inside another loop. The following section shows a few examples:

### Syntax

```

for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)

```

The syntax for a **nested while loop** statement in Python programming language is as follows:

```

while expression:
    while expression:
        statement(s)
    statement(s)

```

You can put any type of loop inside of any other type of loop. For example, a for loop can be inside a while loop or vice versa.

### Example

The following program uses a nested for loop to find the prime numbers from 2 to 100:

```

#!/usr/bin/python

i = 2
while(i < 100):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print i, " is prime"
    i = i + 1

print "Good bye!"

```

The above code produces the following result:

```

2 is prime
3 is prime
5 is prime

```

```

7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime
Good bye!

```

## Loop Control Statements

Loop control statements control the execution of statements inside a loop. Python supports the following control statements:

Sr.No.	Control Statement & Description
1	<u>break statement</u> Terminates the loop statement and transfers execution to the statement immediately following the loop.
2	<u>continue statement</u> Causes the loop to skip the remaining instructions and immediately retest its condition prior to reiterating.
3	<u>pass statement</u> The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

### Break Statement:

It terminates the current loop and resumes execution at the next statement, just like the traditional *break* statement in C. The *break* statement can be used in both *while* and *for* loops. If you are

using nested loops, the `break` statement stops the execution of the innermost loop and start executing the next line of code after the block.

## Syntax

The syntax of `break` statement in Python is as follows:

```
break
```

## Example

```
#!/usr/bin/python

for letter in 'Python':      # First Example
    if letter == 'h':
        break
    print 'Current Letter :', letter

var = 10                     # Second Example
while var > 0:
    print 'Current variable value :', var
    var = var -1
    if var == 5:
        break

print "Good bye!"
```

The above code produces the following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Good bye!
```

## Continue Statement

It returns the control to the beginning of the while loop. The **continue** statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the start of loop.

## Example

```
#!/usr/bin/python

for letter in 'Python':      # First Example
    if letter == 'h':
        continue
    print 'Current Letter :', letter

var = 10                     # Second Example
while var > 0:
    var = var -1
    if var == 5:
        continue
    print 'Current variable value :', var
print "Good bye!"
```

The above code produces the following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Current variable value : 4
Current variable value : 3
Current variable value : 2
Current variable value : 1
Current variable value : 0
Good bye!
```

## 1.5 Lists in Python

The most basic data structure in Python is the sequence. Each element in a sequence is assigned a number. The first index is zero, the second index is one, and so forth. Python has six built-in types of sequences, but the most common ones are lists and tuples. There are certain operations you can perform with all sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership. In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements. The list is a most versatile datatype available in Python which can be written as a list of comma-separated values

(items) between square brackets. Creating a list is as simple as putting different comma-separated values between square brackets. For example:

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

### 1.5.1 Accessing Values in Lists

To access values in lists, use the square brackets for slicing along with indices to obtain values available at those indices. For instance:

```
#!/usr/bin/python

list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
print "list1[0]: ", list1[0]
print "list2[1:5]: ", list2[1:5]
```

When the above code is executed, it produces the following result:

```
list1[0]: physics
list2[1:5]: [2, 3, 4, 5]
```

### 1.5.2 Updating Lists

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the *append()* method. For example:

```
#!/usr/bin/python

list = ['physics', 'chemistry', 1997, 2000];
print "Value available at index 2 : "
print list[2]
list[2] = 2001;
print "New value available at index 2 : "
print list[2]
```

The above code produces the following result:

```
Value available at index 2 :
1997
New value available at index 2 :
2001
```

### 1.5.3 Delete List Elements

To remove a list element, you can use either the *del* statement or the *remove()* method. For example:

```
#!/usr/bin/python

list1 = ['physics', 'chemistry', 2010, 2022];
print list1
del list1[2];
print "After deleting value at index 2 : "
print list1
```

The above code produces following result:

```
['physics', 'chemistry', 2010, 2022]
After deleting value at index 2 :
```

```
['physics', 'chemistry', 2022]
```

### 1.5.4 Basic List Operations

Lists respond to the + and \* operators as that of strings:

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

### Indexing, Slicing, and Matrixes

Indexing and slicing work the same way for lists as they do for strings.

Assuming following input:

```
L = ['spam', 'Spam', 'SPAM!']
```

Python Expression	Results	Description
L[2]	SPAM!	Offsets start at zero
L[-2]	Spam	Negative: count from the right
L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

## Built-in List Functions & Methods

Python includes the following built-in list functions:

Sr.No.	Function with Description
1	<u>cmp(list1, list2)</u> Compares elements of both lists.
2	<u>len(list)</u> Gives the total length of the list.
3	<u>max(list)</u> Returns item from the list with max value.
4	<u>min(list)</u> Returns item from the list with min value.
5	<u>list(seq)</u> Converts a tuple into list.

Python includes following list methods:

Sr.No.	Methods with Description
1	<u>list.append(obj)</u> Appends object <i>obj</i> to list
2	<u>list.count(obj)</u> Returns count of occurrence of <i>obj</i> in list
3	<u>list.extend(seq)</u> Appends the contents of <i>seq</i> to list
4	<u>list.index(obj)</u> Returns the lowest index in list that <i>obj</i> appears
5	<u>list.insert(index, obj)</u> Inserts object <i>obj</i> into list at offset index



6	<u>list.pop(obj=list[-1])</u> Removes and returns last object or <i>obj</i> from list
7	<u>list.remove(obj)</u> Removes object <i>obj</i> from list
8	<u>list.reverse()</u> Reverses objects of list in place

### Key Points

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- Python is processed at runtime by the interpreter. You do not need to compile your program before executing it.
- Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Python provides a better structure and support for large programs than shell scripting.
- Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.
- Python does not provide switch or case statements as in other languages.

## Exercise

### Choose the most suitable option.

1. Which of the following is not supported in Python?  
a. loops      b. if else      c. Switch case      d. None of these
2. Which of the following application is not suited for python?  
a. Machine Learning algorithm    b. Preprocessing data    c. Programming UART    d. None of these
3. Python is a  
a. scripting language    b. interactive language    c. High level language    d. All of these
4. Python supports  
a. Object Oriented Programming    b. Decision statements    c. Switch case    d. Both a and b
5. Python is  
a. case specific    b. indentation specific      c. Both a and b      d. None of these

### Answer the following questions.

1. Define high-level programming language.
2. Differentiate between a script and a program.
3. Describe structure of python.
4. Explain the life cycle of python programming.

### Practical Task

1. Install Python on Linux.
2. Install Python environment on Windows.
3. Swap the values of two variables using a python script.
4. Write a python script to solve a quadratic equation. Coefficients of equation will be input to the script.
5. Write a python script to find largest and smallest number from given numbers.
6. Write a Python script to print Fibonacci numbers.
7. Write a python script to print values of a list.

## Chapter 2: Operating Systems for IoT



After studying this chapter, you will be able to:

- describe IoT Operating Systems.
- name different IoT Operating Systems.
- characteristics of IoT OS.
- differentiate between conventional OS and IoT OS.
- define OSI architecture.
- describe IoT application layer protocols (MQTT, CoAP).
- describe IoT transport layer protocols (DTLS).
- describe IoT network layer protocols (IPv6).
- describe IoT physical layer protocols (Wi-Fi, Bluetooth, Zigbee) using MQTT broker and clients.

## 2.1 Operating Systems for IoT

An IoT OS is an operating system that is designed to perform within the constraints including restrictions on memory, size, power, and processing capacity. IoT operating systems are designed to enable data transfer over the internet.

The essence of the Internet of Things (IoT) is the ability of embedded systems to connect and communicate over a network. IoT OSes control systems in cars, traffic lights, digital televisions, ATMs, airplane controls, point of sale (POS) terminals, digital cameras GPS navigation systems, elevators and smart meters etc.

## 2.2 Examples of IoT Operating Systems

Some common IoT OSes are mentioned below:

<b>IoT System</b>	<b>Operating System</b>	<b>Features</b>	<b>Use cases</b>
Contiki		Open-source, free	Networked memory-constrained systems
FreeRTOS		Open-source, free, uses AWS IoT Core	Devices with tiny amounts of memory
Mbed OS		ARM-based, high-grade security	For portable code
MicroPython		Uses standard Python, easy to learn, C++	Rapid deployment
Embedded Linux		Linux kernel, free	Versatile - can be used for various use cases
RIOT		Open-source, full multithreading	Can be run as MacOS process
TinyOS		C language, open-source	Portability across similar devices
Windows 10 IoT		Proprietary, high-grade security	Ideal for heavy-duty industrial use cases

## 2.3 Characteristics of IoT Operating Systems

The architecture of IoT systems involves a large number of end-nodes (ex. sensors) connected to gateways, which are connected to cloud platforms. Figure 2.1 represents a simplified view of an IoT system.

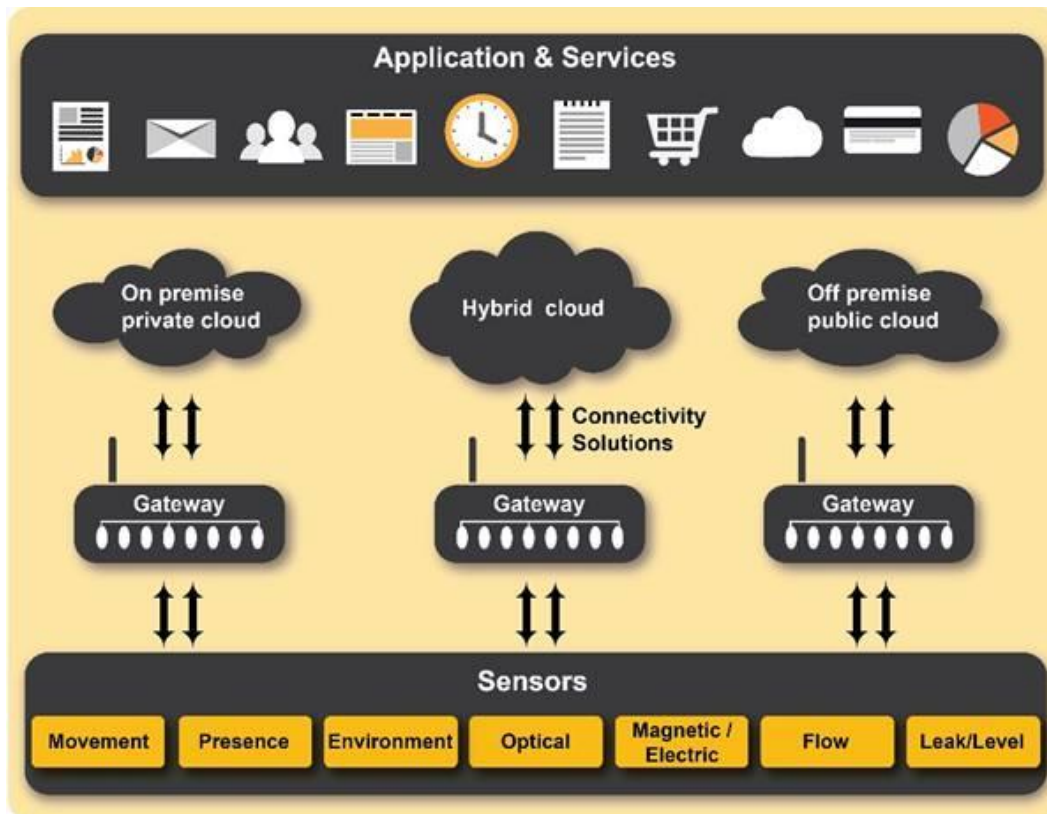


Fig. 2.1 Architecture of an IoT Operating Systems

The key characteristics and requirements that differentiate IoT OSEs are:

**Small memory footprint:** Sensor nodes are typically small and have limited memory available.

**Real-time capabilities:** IoT applications like medical devices, automotive controls, and security systems are critical and require precision in timing. An IoT OS has to perform similar to real time operating systems.

**Energy efficient operation:** Sensors nodes are characterized by low power and are often battery powered. Furthermore, it is commonly expensive to replace batteries etc. It is thus crucial for the IoT OS to be highly energy efficient.

**Hardware agnostic operation:** It is important that the OS support a variety of platforms to simplify interconnectivity and drive standardization and to lower costs of ownership.

**Network Connectivity & Protocol Support:** Continuous connectivity to the network is crucial for IoT device operation. This requirement is achievable by providing support for a variety of connectivity protocols like Wi-Fi, Cellular, Bluetooth, etc.

**Security:** It is imperative that the OS for IoT adhere to strict security expectations and meet stringent requirements imposed by deployments in critical settings.

**Eco-system & Application Development:** Gateways are responsible for providing data services: routing, data shaping, and decision making. In addition, they act as firewalls to protect downstream devices (sensors, actuators, and other **embedded systems**). IoT Gateways should perform their own operations on real-time data and take independent action with minimal human intervention. To perform these important functions, additional requirements need to be met by the OS at the gateway level.

**Protocol Support and data bridge:** In addition to supporting a variety of wired (USB, Serial, Ethernet, etc.) and wireless (Wi-Fi, BT, ZigBee, LoRA, etc.) protocols, the gateway OS should also support web protocols that typically have low transmission overhead like CoAP, MQTT, or UDP in addition to HTTP.

**Data aggregation, local processing and storage:** The gateway OS should support decision making processes in real-time and minimize the amount of data sent to the cloud.

**Security:** Gateway OS Security should afford protections at both the hardware and network levels. Support for encryption, SSL/TLS certification management, user authentication, VPN connectivity, firewall, and application whitelisting is essential.

**Reliable communication to cloud based IoT platforms:** Capabilities to back up transmissions and to manage data and devices in the event of network disruption is a key attribute of IoT OS at the gateway.

**End-device management:** IoT OS should support remote upgrades, provisioning, and two-way communication to the cloud and the devices. Simultaneously, support for web-based tools to configure the gateways and end-devices is needed. It is therefore necessary to make OS choices by carrying out a rigorous assessment of current and future needs of the IoT system.

### 2.3.1 Why a Separate OS for Internet of Things Devices?

The physical size of IoT devices is becoming increasingly smaller. Physical parameters put significant constraints on the hardware and the software. An IoT OS is considered successful when it can be embedded within an internet-connected device, run software, and process data on the host IoT device. Developing an IoT operating system is a software engineering marvel as programmers deal with extreme challenges.

## 2.4 Layered OSI Reference Model for IoT Protocols

It is important to understand the foundations of IoT to solve interoperability challenges. The OSI network model makes it easier to implement a communication by separating it into these seven layers and thus it helps in understanding IoT communications and related standards. The seven layers of the OSI network model form a logically composed ordered set of subsystems. We cover several components of an IoT solution and IoT communication technologies and refer to the OSI model. The following is a break-down of the seven layers:

### 2.4.1 Layers in the OSI Model

**Application layer:** represents the processes on the level of applications and users. In the scope of IoT it's where the link with the business application and the end user is the closest, enabling access to network services. Examples of application layer protocols are CoAP, MQTT, XMPP, AMPQP and HTTP.

**Presentation layer:** translates data in a form that can be used by applications. As it formats and encrypts data for communication, it essentially solves compatibility issues in the communication between the application and the network. An example is the Transport Layer Security (*TLS*) series of cryptographic protocols.

**Session layer:** is where the connections between local and remote applications are initiated, managed and terminated. It enables to establish network sessions between processes on several network stations.

**Transport layer:** is about host-to-host data transport or transmission. Transport layer ensures that data transfers between hosts are completed. TCP and UDP are two examples of protocols in the transport layer.

**Network layer:** is the layer of routing technologies to transfer data packets between different nodes across various networks. It includes IPv4 and IPv6 protocols.

**Datalink layer:** is the OSI layer where data transfer between two directly connected nodes in a network takes place. Bluetooth and Zigbee are examples of IoT datalink layer protocols.

**Physical layer:** It is the layer that includes the essential physical structure needed to make IoT possible: from cables and radio frequency links to essential transmission specifications, network topologies and the communication protocols and hardware on a device.

	IOT STACK	WEB STACK
<b>TCP/IP</b>	<i>IOT applications</i> <i>Device Management</i>	<i>Web applications</i>
<b>Data Format</b>	<i>Binary, JSON, CBOR</i>	<i>HTML, XML, JSON</i>
<b>Application Layer</b>	<i>CoAP, MQTT, XMPP, AMPQP</i>	<i>HTTP, DHCP, DNS, TLS/SSL</i>
<b>Transport Layer</b>	<i>UDP, DTLS</i>	<i>TCP, UDP</i>
<b>Internet Layer</b>	<i>IPv6/IP Routing</i> <i>6LOWPAN</i>	<i>IPv6, IPv4, IPSec</i>
<b>Network/Link Layer</b>	<i>IEEE 802.15.4 MAC</i> <i>IEEE 802.15.4 PHY / Physical Radio</i>	<i>Ethernet (IEEE 802.3), DSL, ISDN, Wireless LAN (IEEE 802.11), Wi-Fi</i>

Fig. 2.2 OSI Layered Model for IoT Systems

### Teacher's Note

Concept of opensource and freeware OS should be explained to students by the instructor.



## Key Points

- An IoT OS is designed to perform within the constraints that are particular to Internet of Things devices, including restrictions on memory, size, power and processing capacity.
- IoT operating systems are a type of embedded OS but by definition are designed to enable data transfer over the internet.
- The functions of sensor nodes and their form factors and target applications vary considerably and create context for the suitability of any specific operating system for IoT implementations.
- Sensor nodes are typically small and have limited memory available. This restricts the memory footprint of the OS.
- IoT applications like medical devices, automotive controls, and security systems are critical and require precision in timing.
- It is crucial that the OS be highly energy efficient.
- An OS supports a variety of platforms to simplify interconnectivity and drive standardization and to lower costs of ownership.
- It is imperative that the OS for IoT adhere to strict security expectations and meet stringent requirements imposed by deployments in sensitive and critical settings
- The gateway OS should also support web protocols that typically have low transmission overhead like CoAP, MQTT, or UDP in addition to HTTP.
- The gateway OS should support decision making processes in real time and minimize the amount of data going to the cloud
- The application layer represents the processes on the level of applications and users.
- Physical layer decides how the networks are organized and where the raw data originates and needs to be communicated.

## Exercise

### Choose the most suitable option.

1. Which of the following is not an IoT OS?  
a. Windows 10 IoT Core   b. Contiki   c. RIoT   d. None of these
2. Which of the following is not an application layer protocol for IoT?  
a. HTTP   b. MQTT   c. TCP   d. Both a and c
3. Which of the following is IoT transport layer protocol  
a. UDP   b. TCP   c. DTLS   d. All of these
4. Following protocols are used in IoT networking layer  
a. IPv6   b. 6LowPAN   c. Both a and b   d. None of these
5. CoAP is \_\_\_\_\_ layer protocol  
a. application   b. transport   c. network   d. Link

### Give short answer to the following questions.

1. Describe requirements for an IoT OS.
2. Describe OSI layered model.
3. How is a IoT OSI model different from traditional OSI model.
4. Describe characteristics of an IoT OS.
5. Name some IoT OSes and enlist their key features.

## Chapter 3: Introduction to Database



### After Studying this chapter, you will be able to:

- describe the concept of data as information.
- describe database.
- importance of a database.
- explain applications of a database.
- describe some known database systems.
- define RDBMS and understand concepts of tables, rows and columns.
- understand the concept of entity and attributes.
- understand the concept of relations and record.
- Use entity relationship (ER) modeling to describe data.
- describe schema, understand concept of constraints, primary key, foreign key, create schema.
- know the installation process of MySQL database.
- Explain the usage of visual browser for MySQL.
- understand the installation process of visual browser for MySQL.
- create a physical schema using visual browser and export/ import schema using visual browser.
- define query.
- use basic SQL queries in a database to:
  - CREATE
  - READ
  - UPDATE

## 3.1 Data

Data is a raw and unorganized fact that requires to be processed to make it meaningful. Data can be simple and unorganized. Data comprises of facts, observations, perceptions numbers, characters, symbols and images etc. Data is always interpreted, by a human or machine, to derive meaning. Therefore, data is meaningless. Data contains numbers, statements, and characters in a raw form. For instance, when students get admission, to colleges or universities, they have to fill out an admission form. The form consists of raw facts i.e., student's name, father's name and address etc. The idea of collecting this data is to sustain the records of the students.

### 3.1.2 Information

Information is a set of data which is processed in a meaningful way according to the given requirement. Information is processed, structured, or presented in a given context to make it meaningful and useful. Information possesses context, relevance, and purpose. It also involves manipulation of raw data. Information assigns meaning and improves the reliability of the data. It reduces uncertainty.

## 3.2 Database

A database is a collection of data, usually stored in an electronic form. A database is typically designed so that it is easy to store and access information. A good database is crucial to any company or organisation. This is because the database stores all pertinent details about the company such as employee records, transactional records and salary etc.

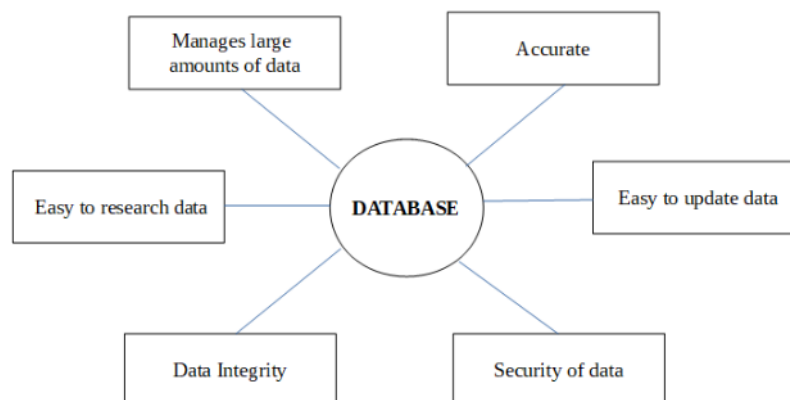


Fig. 3.1 Essential components of a database

### 3.2.1 Importance of Database

The various important aspects of a database are:

#### **Manages large amounts of data**

A database stores and manages a large amount of data. This would not be possible using any other tool such as a spreadsheet.

#### **Accurate**

A database is accurate as it has all sorts of built-in constraints and checks etc. This means that the information available in a database is guaranteed to be correct.

#### **Easy to update data**

In a database, it is easy to update data using various Data Manipulation Languages (DML) available.

#### **Security of data**

Databases have various methods to ensure security of data. They require user logins before accessing a database and various access specifiers.

#### **Data integrity**

Data integrity is ensured using various constraints for data. Data integrity in databases makes sure that the data is accurate and consistent in a database.

#### **Easy to search data**

It is very easy to access and search data in a database. This is done using Data Query Languages (DQL) which allow searching of any data in the database.

## 3.3 Database Management

Database Management System (DBMS) is an automatic system helping users to control, create, update and maintain a database. The two main components of a Database Management System are: The query processor and the data manager.

### 3.3.1 The Importance of Database Management

A database management system plays an important role in processing and controlling information. Specifically, a database management system has the following functions:

- Provide a data creation environment: Database management system provides the language of data definition to describe and declare the data type.

- Provide ways of updating and exploring data: the database management system provides actions to describe the requirement and actions of updating and exploring the database. Data actions include updating (create, edit, delete data (CRUD)) and exploring data (search, filter data).
- Provide tools to control access on the database to ensure the implementation of some basic requirements of a database system. It includes:
  1. Assure security, find out and prevent illegal access.
  2. Maintain data consistency.
  3. Arrange and control access.

### 3.4 Applications of a Database

Applications of Database Management Systems are:

- **Telecom:** Database keeps track of the information regarding calls made, network usage, customer details etc. Without a database system, it is difficult to maintain the huge amount of data that keeps updating every millisecond.
- **Industry:** A manufacturing unit, warehouse and distribution centre need a database to keep records of ins and outs. For example, a distribution centre should keep a track of the product units that are supplied into the centre as well as the products that got delivered out from the distribution centre.
- **Banking System:** Databases are required for storing customer information, tracking day to day credit and debit transactions, generating bank statements etc. All this work is done with the help of Database management systems.
- **Sales:** Database Management Systems are used to store customer information, production information and invoice details.
- **Airlines:** We make reservations to travel through airlines. This reservation information along with the flight schedule is stored in databases.
- **Education sector:** Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance and fee details etc.
- **Online shopping:** Amazon, ebay and flipkart etc. store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query.

## Examples of Databases

Below are some of the best Free database software:

- Microsoft SQL
- MySQL
- PostgreSQL
- MongoDB
- MariaDB
- SQLite

## 3.5 RDBMS

RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model. The relational model means that the logical data structures-the data tables, views, and indexes-are separate from physical storage structures.

### Table

Data in an RDBMS is stored in database objects which are called as tables. Table is basically a collection of related data entries, and it consists of numerous columns and rows. Remember, a table is the most common and simplest form of data storage in a relational database. The following program is an example of a CUSTOMERS table:

ID	NAME	AGE	ADDRESS	SALARY
1	Faizan	32	Lahore	2000.00
2	Ali	25	Karachi	1500.00
3	Numan	23	Quetta	2000.00
4	Arsalan	25	Islamabad	6500.00
5	Ayesha	27	Bhopal	8500.00
6	Komal	22	Multan	4500.00
7	Amir	24	Indore	10000.00

## Field

A field is a column in a table that is designed to maintain specific information about every record in the table. Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.

## Record or a Row

A record is also called as a row of data which is an individual entry that exists in a table. For example, there are 7 records in the above CUSTOMERS table. The following is a single row of data or record in the CUSTOMERS table:

5	Ayesha	27	Bhopal	8500.00
---	--------	----	--------	---------

## Column

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

For example, a column in the CUSTOMERS table is ADDRESS, which represents location description and would be shown as:

ADDRESS
Lahore
Karachi
Quetta
Islamabad
Bhopal
Multan
Indore

## NULL value

A NULL value in a table is a value in a field that appears to be blank, which means a field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces. A field with a NULL value is the one that has been left blank during a record creation.



## SQL Constraints

Constraints are the rules enforced on data columns in a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in a database. Constraints can either be column level or table level. Column level constraints are applied only to one column, whereas table level constraints are applied to the entire table. The following are some of the most used constraints available in SQL:

- **NOT NULL Constraint**: Ensures that a column cannot have a NULL value.
- **DEFAULT Constraint**: Provides a default value for a column when none is specified.
- **UNIQUE Constraint**: Ensures that all the values in a column are different.
- **PRIMARY Key**: Uniquely identifies each row/record in a database table.
- **FOREIGN Key**: Uniquely identifies a row/record in any another database table.
- **CHECK Constraint**: The CHECK constraint ensures that all values in a column satisfy certain conditions.
- **INDEX**: Used to create and retrieve data from the database quickly.

## Data Integrity

The following categories of data integrity exist with each RDBMS:

- **Entity Integrity**: There are no duplicate rows in a table.
- **Domain Integrity**: Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- **Referential integrity**: Rows cannot be deleted, which are used by other records.
- **User-Defined Integrity**: Enforces some specific business rules that do not fall into entity, domain or referential integrity.

## Database Normalization

Database normalization is the process of efficiently organizing data in a database. There are two reasons of normalization:

- Eliminating redundant data, for example, storing the same data in more than one table.
- Ensuring data dependencies make sense.

It reduces the amount of space a database consumes and ensures that data is logically stored.

## 3.6 Installing MySQL on Windows

### Download MySQL Installer

MySQL installer is the easiest way to install MYSQL. MySQL installer provides you with an easy-to-use wizard that helps you to install MySQL with the following components:

- MySQL Server
- All Available Connectors
- MySQL Workbench with Sample Data Models
- MySQL Notifier
- Tools for Excel and Microsoft Visual Studio
- MySQL Sample Databases
- MySQL Documentation

To download MySQL installer, go to the following link <http://dev.mysql.com/downloads/installer/>.

### Install MySQL via MySQL Installer

To install MySQL using the MySQL installer, double-click on the MySQL installer file and follow the steps below:

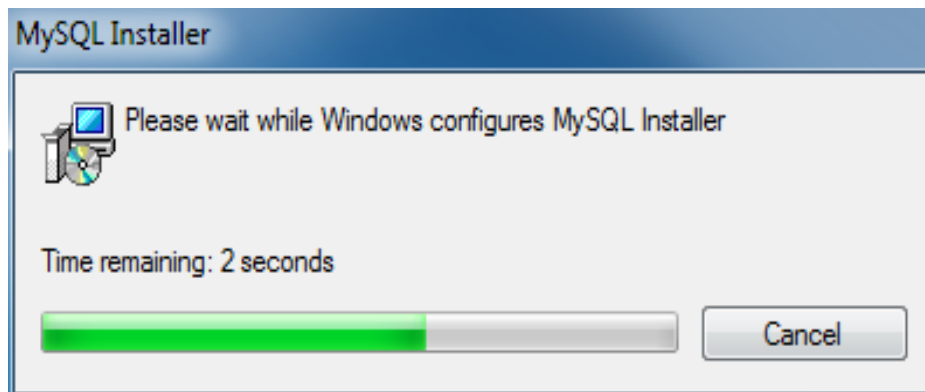


Fig 3.2. Install MySQL Step 1: Windows configures MySQL Installer

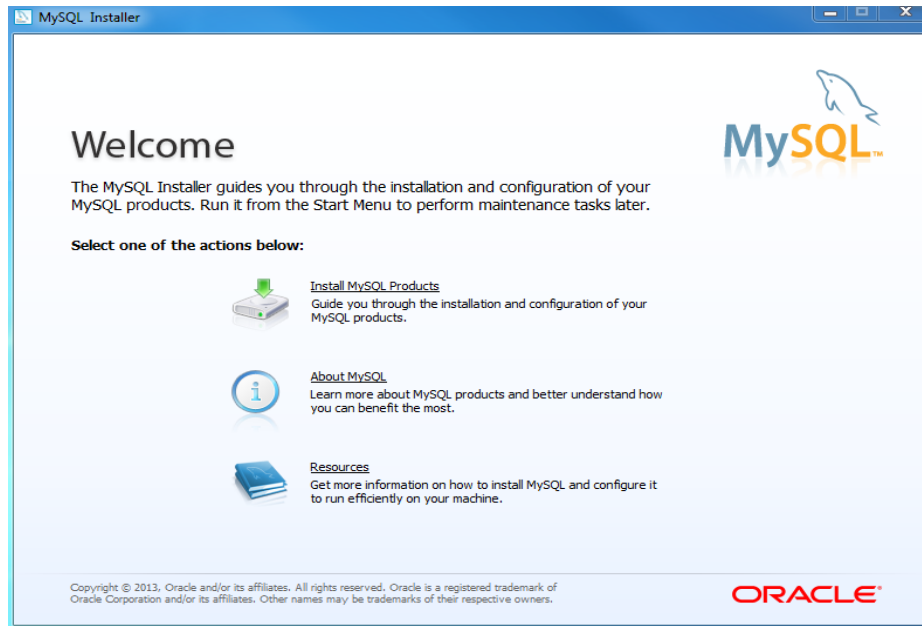


Fig. 3.3 MySQL Installer: Welcome Wizard

Install MySQL Step 1: Welcome Screen: A welcome screen provides several options. Choose the first option: Install MySQL Products

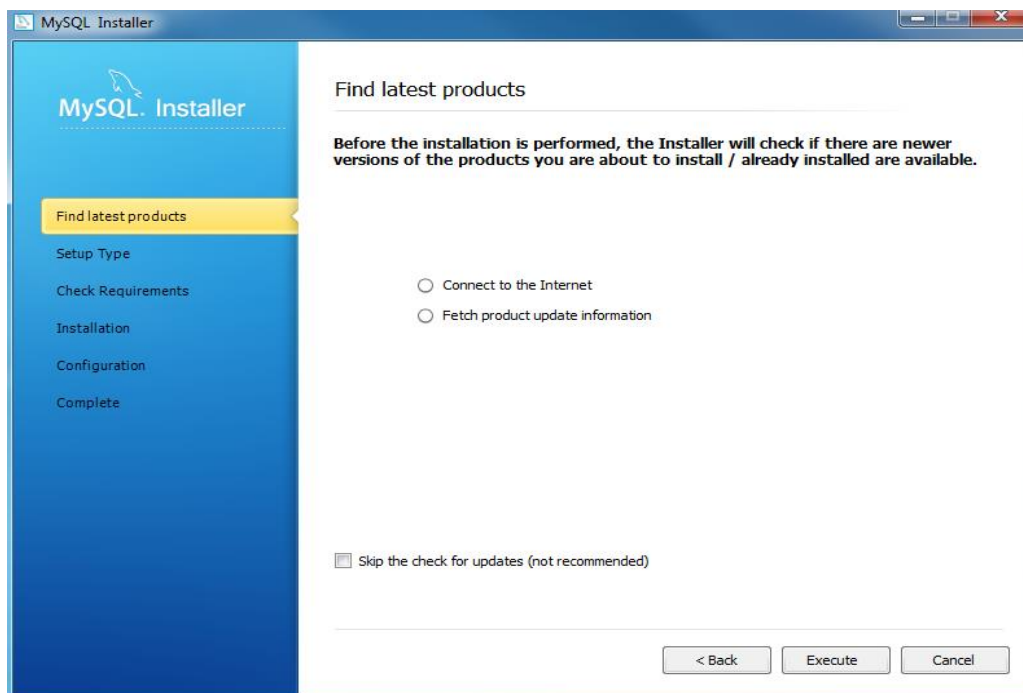


Fig. 3.3 MySQL Installer: Find Latest Products

Install MySQL Step 3: Download the latest MySQL products: MySQL installer checks and downloads the latest MySQL products including MySQL server, MySQL Workbench, etc.

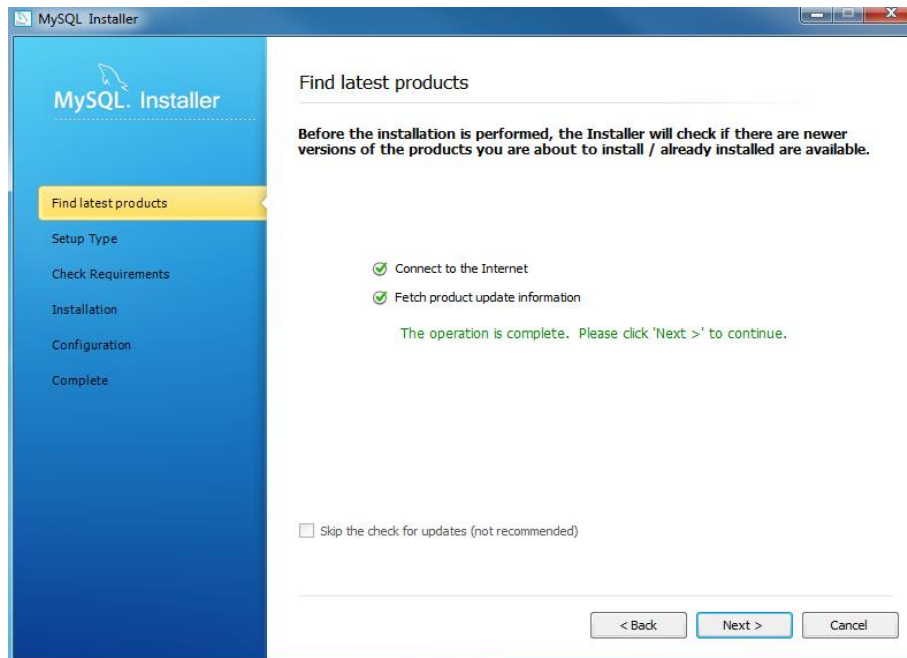


Fig. 3.4 MySQL Installer: Find Latest Products

Install MySQL Step 4: Click the Next button to continue.

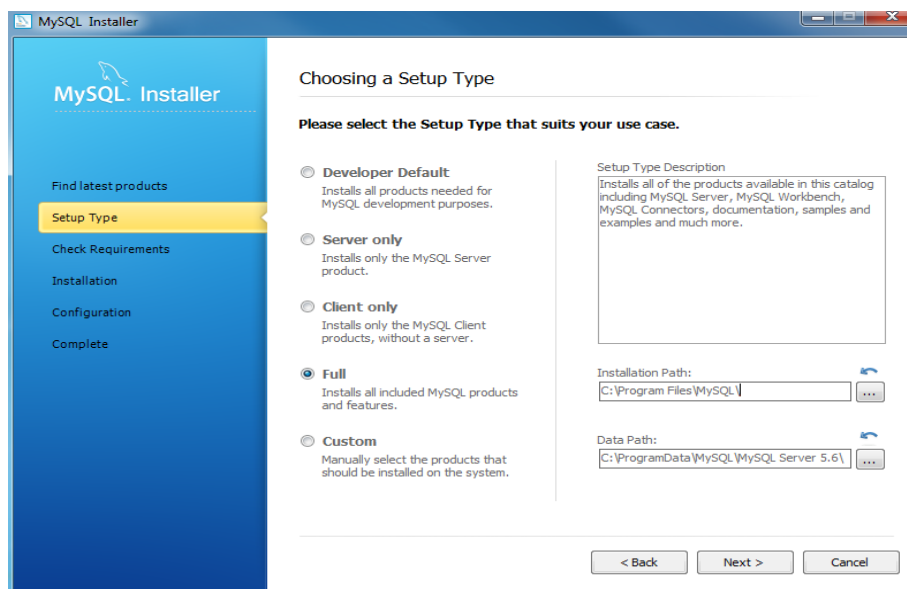


Fig. 3.5 MySQL Installer: Choosing a Setup Type

Install MySQL Step 5: Choosing a Setup Type: there are several setup types available. Choose the full option to install all MySQL products and features.

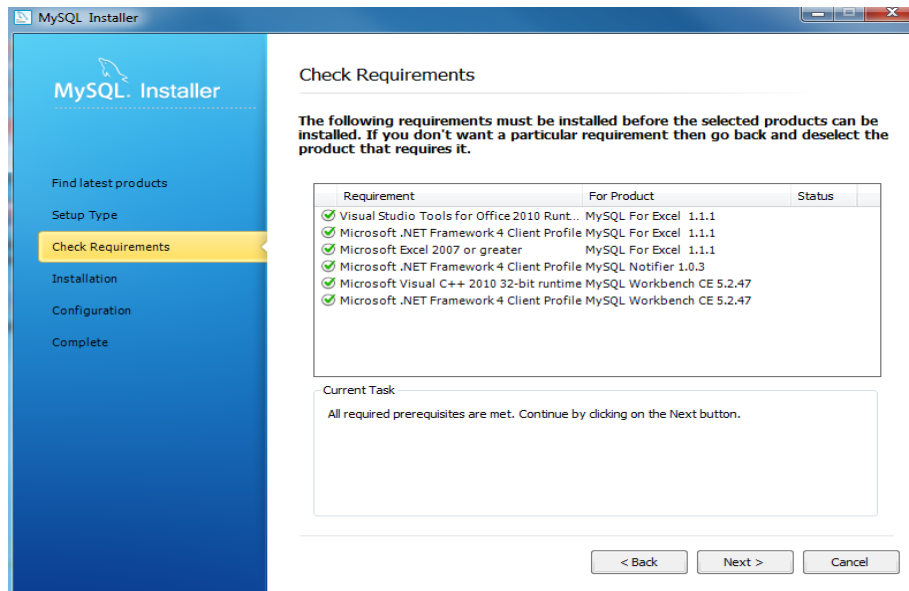


Fig. 3.6 MySQL Installer: Check Requirements

Install MySQL Step 6: Checking Requirements

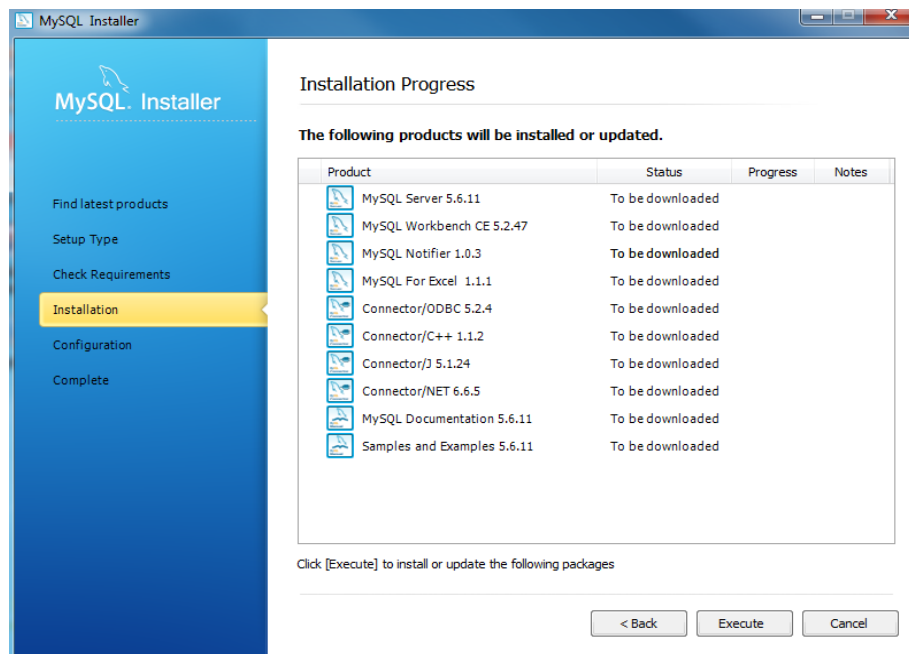


Fig. 3.7 MySQL Installer: Installation Progress

Install MySQL Step 7: Installation Progress: MySQL Installer downloads all selected products. It will take a while, depending on which products you selected and the speed of your internet connection.

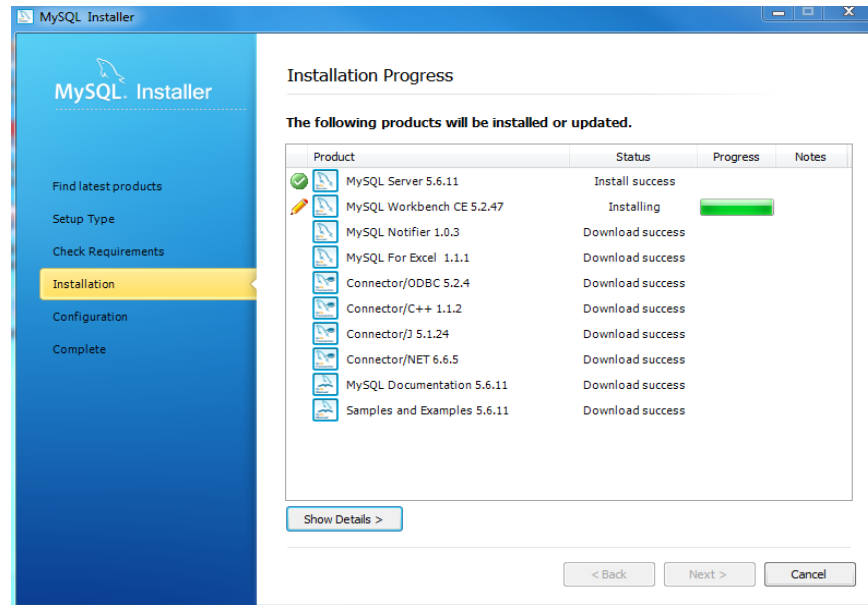


Fig. 3.8 MySQL Installer: Installation Progress

Install MySQL Step 7: Installation Progress: downloading Products in progress.

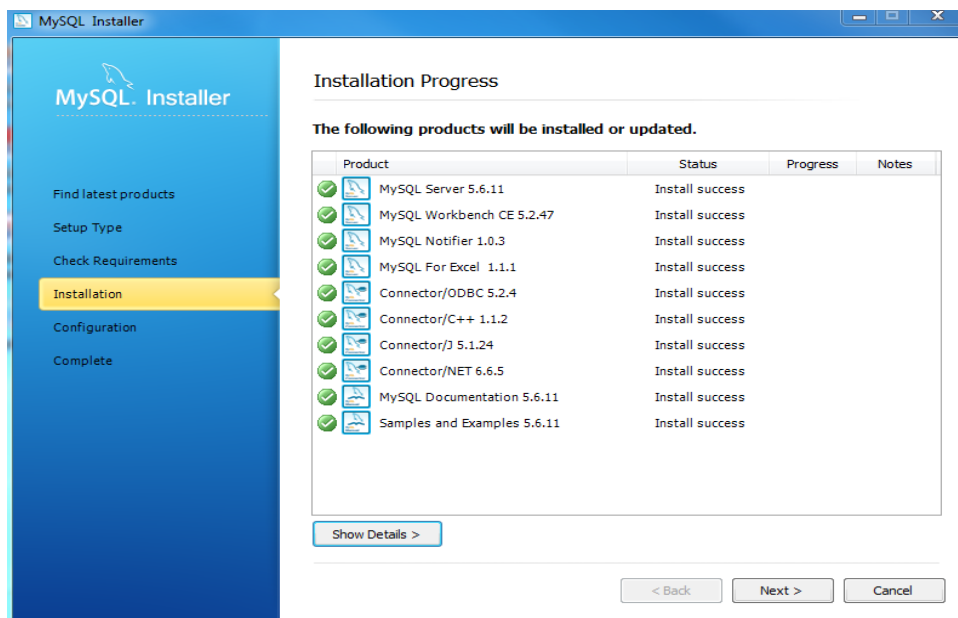


Fig. 3.9 MySQL Installer: Installation Progress

Install MySQL Step 7: Installation Progress: Complete Downloading. Click the **Next** button to continue.

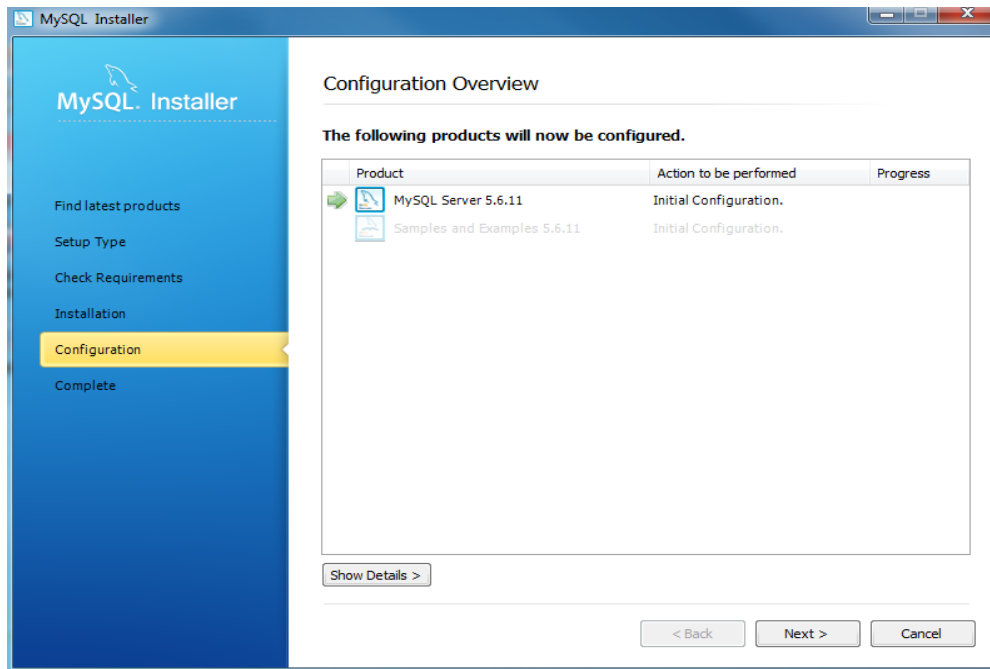


Fig. 3.10 MySQL Installer: Configuration Overview

Install MySQL Step 8: Configuration Overview. Click the **Next** button to configure MySQL Database Server.

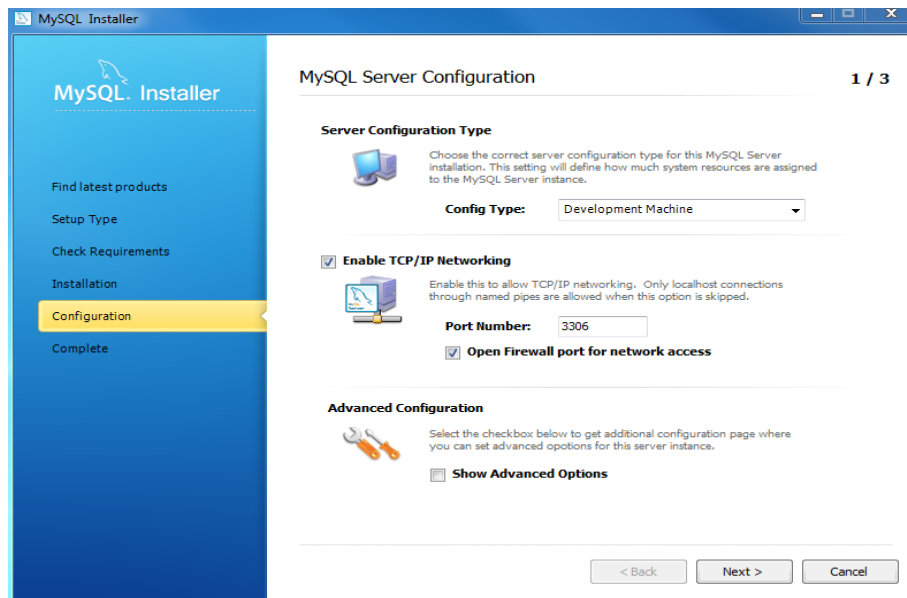


Fig. 3.11 MySQL Installer: Server Configuration

Install MySQL Step 8.1: MySQL Server Configuration: choose Config Type and MySQL port (3006 by default) and click Next button to continue.

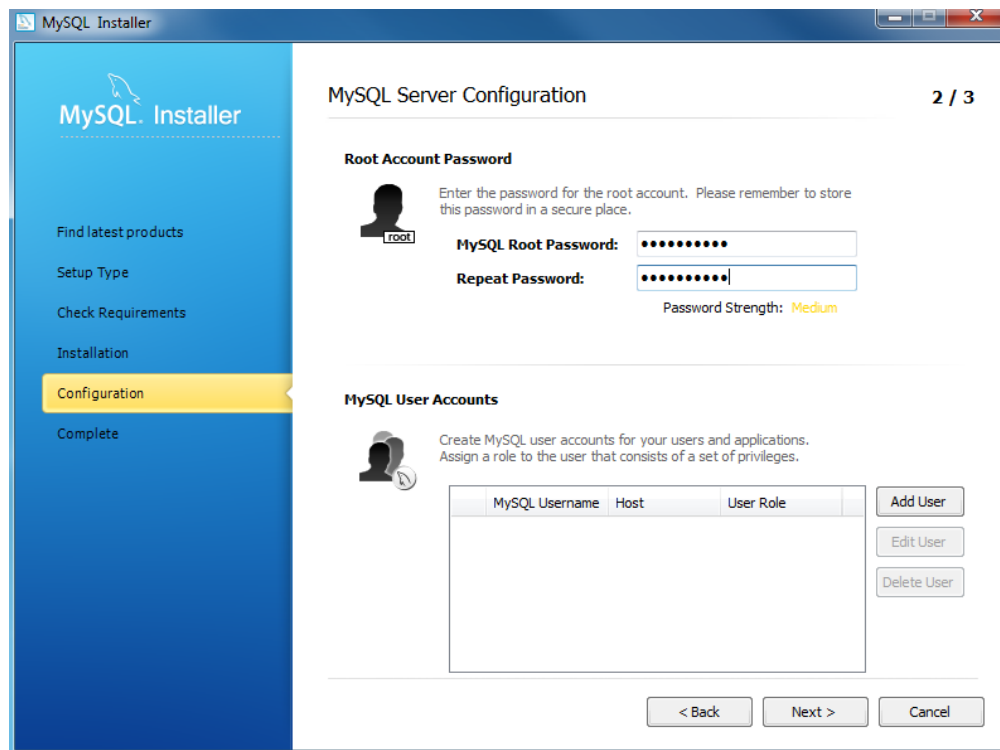


Fig. 3.12 MySQL Installer: Server Configuration

Install MySQL Step 8.2: MySQL Server Configuration: choose a password for the root account. Please note the password download and keep it securely if you are installing MySQL database server on a production server. If you want to add a more MySQL user, you can do it in this step.

Install MySQL Step 8.3: MySQL Server Configuration: choose Windows service details including Windows Service Name and account type, then click Next button to continue.

### Teacher's Note

Concept of DDL and DML should be explained to students.



Install MySQL Step 8.4: MySQL Server Configuration: In Progress: MySQL Installer is configuring MySQL database server. Wait until it is done and click the Next button to continue.

Install MySQL Step 8.5: MySQL Server Configuration: Done. Click the Next button to continue.

Install MySQL Step 8.6: Configuration Overview: MySQL Installer installs sample databases and sample models.

Install MySQL Step 8.8: Installation Completes: the installation completes. Click the **Finish** button to close the installation wizard and launch the MySQL Workbench.

## 3.5 Creating a MySQL Database Schema

### Using a GUI

Use these instructions to create a database using MySQL Workbench.

1. Download and install MySQL.
2. Open the MySQL Workbench as an administrator (Right-click, **Run as Admin**).
3. Click on **File>Create Schema** to create the database schema.

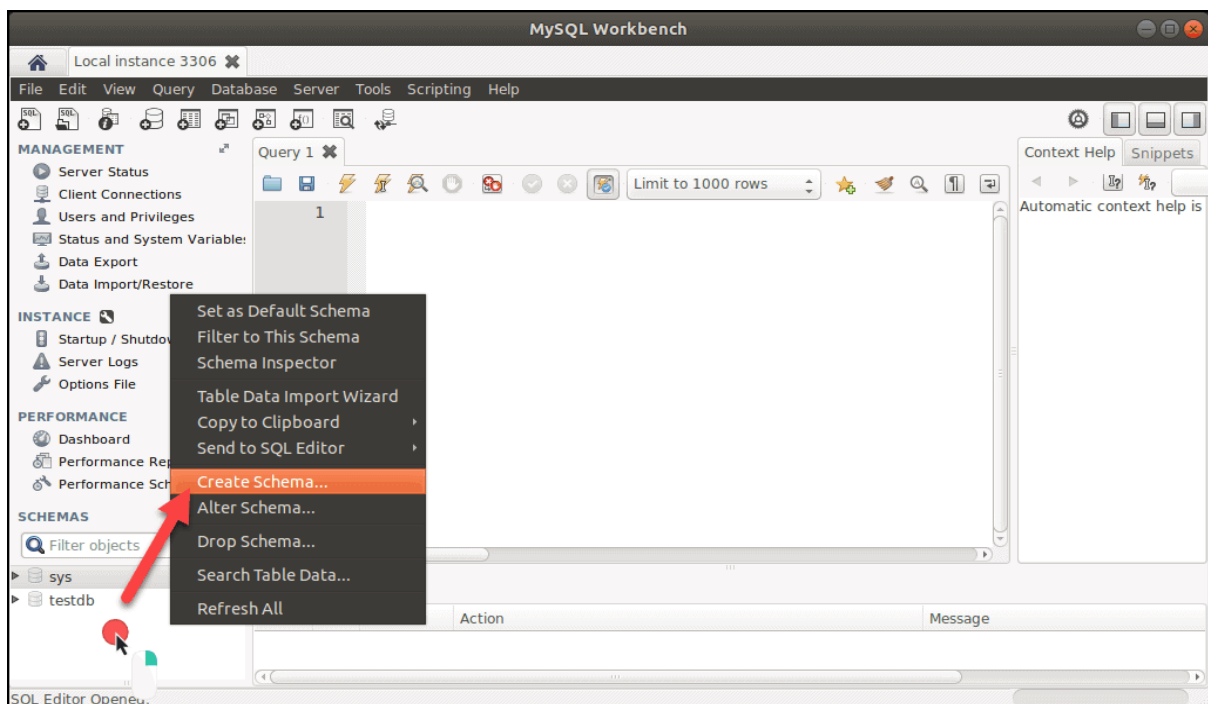


Fig. 3.12 MySQL Workbench: Create Schema

4. Enter a name for the schema and click **Apply**.
5. In the **Apply SQL Script to Database** window, click **Apply** to run the SQL command that creates the schema.
6. Click **Finish**.
7. From the **Server** menu, select **Users and Privileges** to add a user account.
8. From the **Server** menu, select **Options File** and click the **Networking** tab. Find the `max_allowed_packet` entry (should be at the top) and change it to at least `256M`. Click **Apply**.
9. Finally, to apply this change you need to restart the database. From the **Server** menu, select **Startup/Shutdown** and click **Stop Server**, followed by **Start Server**.

You now have finished preparing the RapidMiner Server database.

## Using the command line

Use the following instructions to install MySQL via command line.

1. Download and install MySQL.
2. Change the `max_allowed_packet` variable to at least `256M` to allow storing larger models in the repository.
3. From the command line, create the database:
 

```
create database rapidminer_server;
```
4. Execute the following query to create a new user and grant privileges to the database `rapidminer_server`:
 

```
grant all privileges on rapidminer_server.* to
rmUser@localhost identified by 'pswd';
```

With privileges granted, you have finished creating the RapidMiner Server database.

## MySQL Workbench: Exporting Schema

In MySQL Workbench version 8.0 or above, you can just follow the next steps

1. Go to **Server** tab
2. Go to **Database Export**

This opens up something like this:

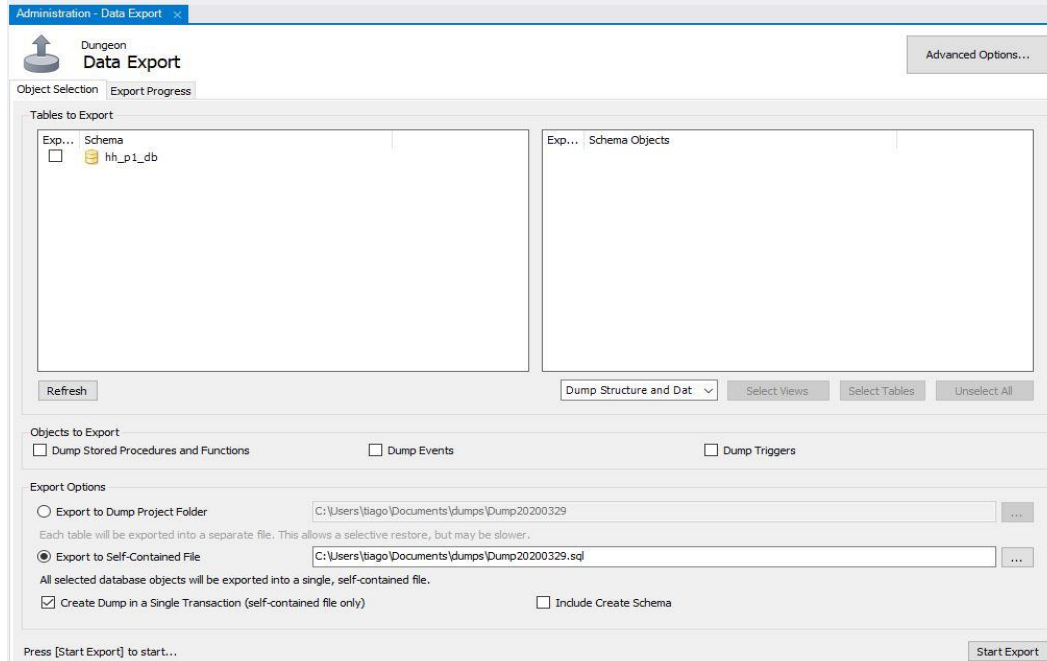


Fig. 3.13 MySQL Workbench: Export Schema

3. Select the schema to export in the **Tables to export**
4. Click on **Export to Self-Contained file**
5. Check if **Advanced Options...** are exactly as you want the export
6. Click the button **Start Export**

### 3.7 SQL Queries

SQL stands for Structured Query Language. SQL commands (queries) are the instructions used to communicate with a database to perform tasks, functions, and queries with data.

SQL commands can be used to search the database and to create tables, add data to tables, modify data, and delete tables.

The following is a list of basic SQL commands:

#### **SELECT and FROM**

The **SELECT** part of a query determines which columns of the data to show in the results.

The example below shows three columns **SELECT**ed **FROM** the “student” table and one calculated column. The database stores the studentID, FirstName, and LastName of the student. We can combine the First and the Last name columns to create the FullName calculated column.

```
SELECT studentID, FirstName, LastName, FirstName + ' ' + LastName AS
FullName
FROM student;
```

```
+-----+-----+-----+-----+
| studentID | FirstName      | LastName  | FullName                |
+-----+-----+-----+-----+
|          1 | Monique        | Davis     | Monique Davis          |
|          2 | Teri           | Gutierrez | Teri Gutierrez         |
|          3 | Spencer        | Pautier   | Spencer Pautier        |
|          4 | Louis          | Ramsey    | Louis Ramsey           |
|          5 | Alvin          | Greene    | Alvin Greene           |
|          6 | Sophie         | Freeman   | Sophie Freeman         |
|          7 | Edgar Frank "Ted" | Codd      | Edgar Frank "Ted" Codd |
|          8 | Donald D.      | Chamberlin | Donald D. Chamberlin  |
|          9 | Raymond F.     | Boyce     | Raymond F. Boyce      |
+-----+-----+-----+-----+
```

```
9 rows in set (0.00 sec)
```

## CREATE TABLE

CREATE TABLE creates a table in the database. You can specify the name of the table and the columns that should be in the table.

```
CREATE TABLE table_name (
    column_1 datatype,
    column_2 datatype,
    column_3 datatype
```

```
);
```

## WHERE

**(AND, OR, IN, BETWEEN, and LIKE)**

The WHERE clause is used to limit the number of rows returned.

As an example, first we will show you a SELECT statement and results *without* a WHERE statement. Then we will add a WHERE statement that uses all five qualifiers above.

```
SELECT studentID, FullName, sat_score, rcd_updated FROM student;
```

```
+-----+-----+-----+-----+
| studentID | FullName          | sat_score | rcd_updated          |
+-----+-----+-----+-----+
|      1 | Monique Davis    |      400 | 2017-08-16 15:34:50 |
|      2 | Teri Gutierrez   |      800 | 2017-08-16 15:34:50 |
|      3 | Spencer Pautier  |     1000 | 2017-08-16 15:34:50 |
|      4 | Louis Ramsey     |     1200 | 2017-08-16 15:34:50 |
|      5 | Alvin Greene     |     1200 | 2017-08-16 15:34:50 |
|      6 | Sophie Freeman   |     1200 | 2017-08-16 15:34:50 |
|      7 | Edgar Frank "Ted" Codd |     2400 | 2017-08-16 15:35:33 |
|      8 | Donald D. Chamberlin |     2400 | 2017-08-16 15:35:33 |
|      9 | Raymond F. Boyce |     2400 | 2017-08-16 15:35:33 |
+-----+-----+-----+-----+
```

```
9 rows in set (0.00 sec)
```

## UPDATE

To update a record in a table you use the UPDATE statement.

Use the WHERE condition to specify which records you want to update. It is possible to update one or more columns at a time. The syntax is:

```
UPDATE table_name
SET column1 = value1,
    column2 = value2, ...
WHERE condition;
```

The following is an example updating the Name of the record with Id 4:

```
UPDATE Person
SET Name = "Elton John"
WHERE Id = 4;
```

## Key Points

- Data is a raw and unorganized fact that required to be processed to make it meaningful.
- Information is a set of data which is processed in a meaningful way according to the given requirement.
- Information is processed, structured, or presented in a given context to make it meaningful and useful.
- A database is a collection of data, usually stored in electronic form.
- A database is typically designed so that it is easy to store and access information.
- The Database Management System is an automatic system helping the user to control information, create, update and maintain the database.
- RDBMS stands for Relational Database Management System.
- RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The data in an RDBMS is stored in database objects which are called as tables.
- Every table is broken up into smaller entities called fields.
- A record is also called as a row of data is each individual entry that exists in a table.
- A column is a vertical entity in a table that contains all information associated with a specific field in a table.

## Exercise

### Choose the most suitable option.

1. Which of the following is not a SQL query?  
a. Update   b. Insert   c. delete   d. Manipulate
2. \_\_\_\_\_ query is used to add new rows in a table?  
a. Update   b. Insert   c. delete   d. Manipulate
3. \_\_\_\_\_ query is used to create table?  
a. Make   b. Create   c. Implement   d. Add
4. Which of the following is a relational database  
a. MYSQL   b. SQLite   c. RDBMS   d. None of these
5. A record is also called as a row of \_\_\_\_\_ is each individual entry that exists in a table.  
a. Information   b. data   c. Knowledge   d. values

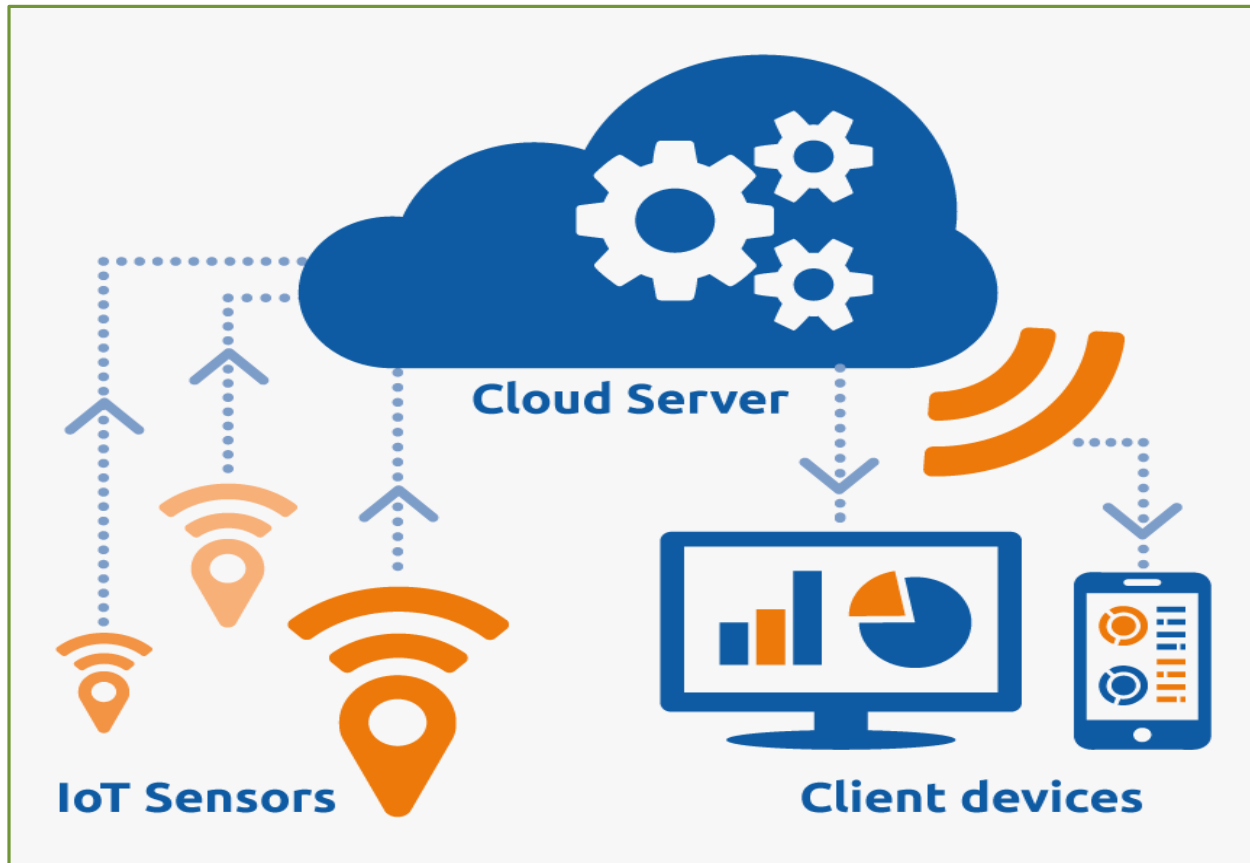
### Give short answer to the following questions.

1. Differentiate between data and information.
2. Define Database.
3. Describe the importance of database.
4. Enlist some applications of database.
5. Define RDBMS.
6. Define tables, rows and columns.
7. What are foreign key constraints.
8. What do you mean by a primary key?
9. Define Schema.
10. What is MySQL workbench?

### Practical Tasks

1. Install MySQL Workbench on a Windows based PC.
2. Create a schema on MYSQL Workbench.
3. Export Schema from MySQL workbench to a .sql file.

## Chapter 4: IoT Cloud Deployment



After Studying this chapter, you will be able to:

- define cloud.
- explain different types of cloud.
- understand characteristics of a cloud.
- describe advantages and disadvantages of a cloud.
- define cloud service providers.
- explain different types of cloud models.
- describe usage of cloud service providers in cloud.
- define cloud server.
- understand the process to create an account and login on a cloud server provider's dashboard.
- understanding of virtualization and virtual machines.
- create a virtual machine.
- understand of different configurations available on the dashboard.
- install an OS on virtual machine.
- define inbound and outbound rules to open ports for SSH and FTP.



## 4.1 Cloud Computing

"The cloud" refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centres all over the world. Companies using cloud computing do not have to manage physical servers themselves.

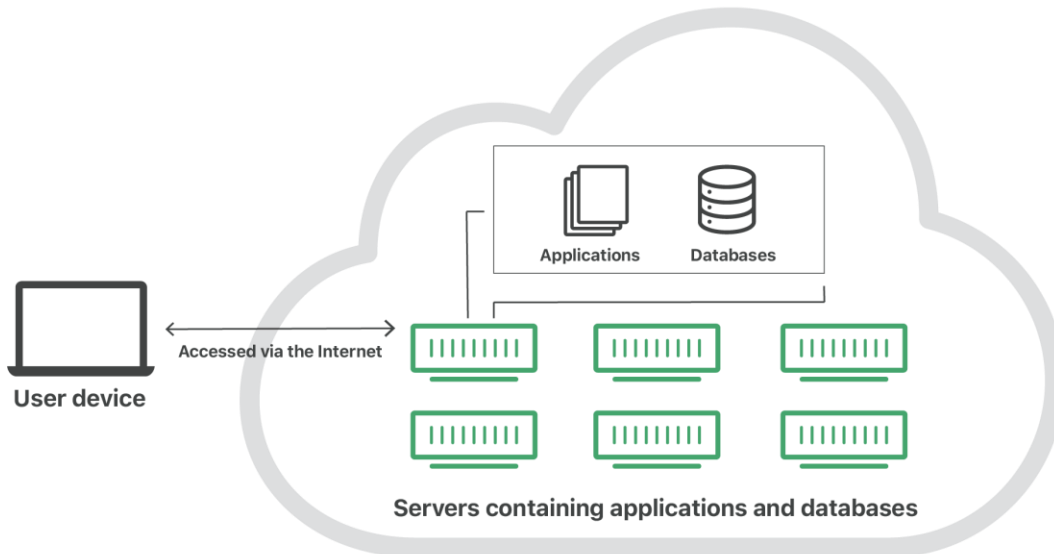


Fig. 4.1 The Cloud

The cloud enables users to access the same files and applications from almost any device, because the computing and storage takes place on servers in a data centre, instead of locally on the user device. This enables a user to log in to their Instagram account on a new phone after their old phone breaks and still find their old account in place, with all their photos, videos, and conversation history. It works the same way with cloud email providers like Gmail or Microsoft Office 365, and with cloud storage providers like Google Drive.

Switching to cloud computing decreases costs. For instance, businesses no longer need to update and maintain their own servers. This especially makes an impact for small businesses that may not have been able to afford their own internal infrastructure but can outsource their infrastructure needs affordably via the cloud. The cloud can also make it easier for companies to operate internationally, because employees and customers can access the same files and applications from any location.

## 4.2 Cloud Computing Working Principle

Cloud computing is possible because of virtualization. Virtualization allows for the creation of a simulated, digital-only "virtual" computer that behaves as if it were a physical computer with its own hardware. The technical term for such a computer is virtual machine. When properly implemented, virtual machines on the same host machine are sandboxed from one another, so they do not interact with each other at all, and the files and applications from one virtual machine are not visible to the other virtual machines.

Virtual machines also make more efficient use of the hardware hosting them. By running many virtual machines at once, one computer can host many servers, and a data centre becomes a host of data centres (refer to Figure 4.1). Cloud vendors generally back up their services on multiple machines and across multiple regions. Users access cloud services either through a browser or through an app.

## 4.3 Different Types of Cloud Deployments

The most common cloud deployments are:

- **Private cloud:** A private cloud is a server, data centre, or distributed network fully dedicated to one organization.
- **Public cloud:** A public cloud is a service run by an external vendor that may include servers in one or multiple data centres. Public clouds are shared by multiple organizations. Using virtual machines, individual servers may be shared by different companies. It is called "multitenancy" because multiple tenants are renting server space within the same server.
- **Hybrid cloud:** Hybrid cloud deployments combine public and private clouds, and may even include on-site servers. An organization may use their private cloud for some services and their public cloud for others.
- **Multi-cloud:** Multi-cloud is a type of cloud deployment that involves multiple public clouds. In other words, an organization with a multi-cloud deployment rents virtual servers and services from several external vendors.

## 4.4 Advantages of Cloud Computing

Cloud computing is trending technology. The following are some important advantages (also shown in Figure 4.2) of Cloud Computing:

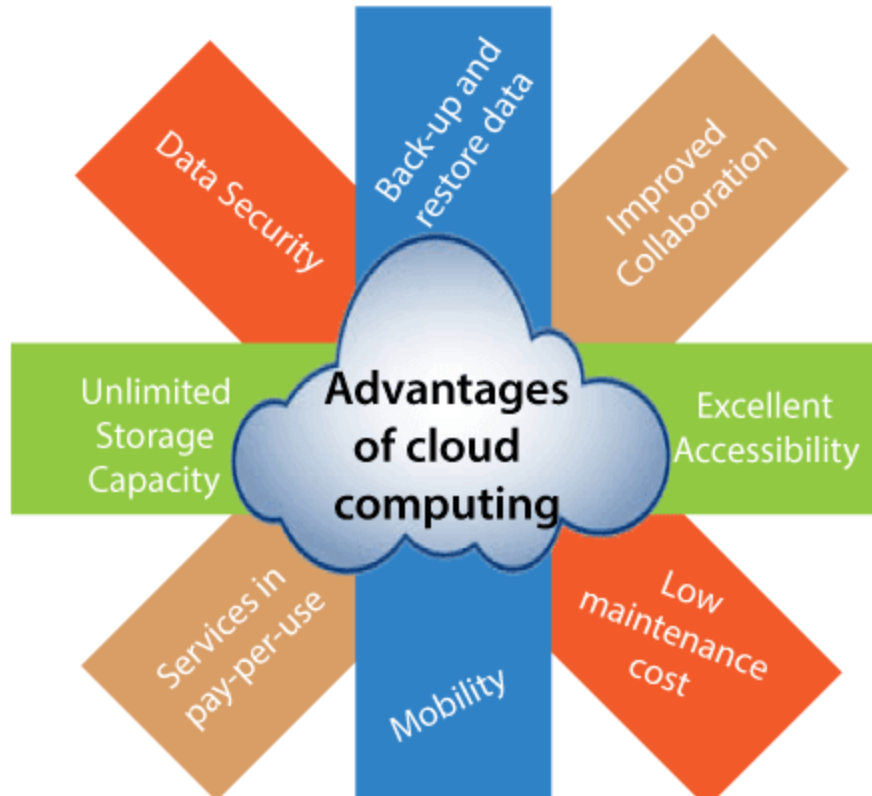


Fig. 4.2 Advantages of Cloud Computing

#### 1) Back-up and restore data

Once the data is stored in the cloud, it is easier to get back-up and restore that data.

#### 2) Improved collaboration

Cloud applications improve collaboration by allowing groups of people to share information quickly and easily in the cloud via shared storage.

#### 3) Excellent accessibility

Cloud allows us to access store information anywhere, anytime quickly and easily in the whole world, using an internet connection.

#### 4) Low maintenance cost

Cloud computing reduces both hardware and software maintenance costs.

#### 5) Services in the pay-per-use model

Cloud computing offers Application Programming Interfaces (APIs) to the users for access services on the cloud and pays the charges as per the usage of service.

### 6) Unlimited storage capacity

Cloud offers us a huge amount of storing capacity for storing our important data such as documents, images, audio, video, etc. in one place.

### 7) Data security

Data security is one of the biggest advantages of cloud computing. Cloud offers many advanced features related to security and ensures that data is securely stored.

## 4.5 Disadvantages of Cloud Computing

A list of the disadvantage of cloud computing is given below:

### 1) Internet Connectivity

In cloud computing, every data (image, audio, video, etc.) is stored on the cloud, and we access this data through cloud by using the internet connection. If you do not have good internet connectivity, you cannot access these data.

### 2) Vendor Lock-in

Organizations may face problems when transferring their services from one vendor to another. As different vendors provide different platforms, that can cause difficulty moving from one cloud to another.

### 3) Limited Control

Cloud infrastructure is completely owned, managed, and monitored by the service provider. Therefore, the cloud users have less control over the function and execution of services within a cloud infrastructure.

## 4.6 Cloud Service Providers

A cloud service provider is a third-party company offering a cloud-based platform, infrastructure, application, or storage services.

### Service Models of Cloud Computing

**Software-as-a-Service (SaaS):** SaaS applications are hosted on cloud servers, and users access them over the Internet. Examples of SaaS applications include Salesforce, MailChimp, and Slack.

**Platform-as-a-Service (PaaS):** In this model, companies don't pay for hosted applications; instead, they pay for the things they need to build their own applications. PaaS vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet. PaaS examples include Heroku and Microsoft Azure.

**Infrastructure-as-a-Service (IaaS):** In this model, a company rents the servers and storage they need from a cloud provider. They then use that cloud infrastructure to build their applications. IaaS providers include DigitalOcean, Google Compute Engine, and OpenStack.

**Function-as-a-Service (FaaS):** FaaS, also known as serverless computing, breaks cloud applications down into even smaller components that only run when they are needed. FaaS or serverless applications still run on servers, as do all these models of cloud computing. But they are called "serverless" because they do not run on dedicated machines.

## 4.7 Setting up a Cloud Server

A cloud server is a virtual server running in a cloud computing environment.

### 4.7.1 Create and deploy

1. Log in to the [Azure portal](#).
2. Click **Create a resource** > **Compute**, and then scroll down to and click **Cloud Service** (as shown in Figure 4.3).

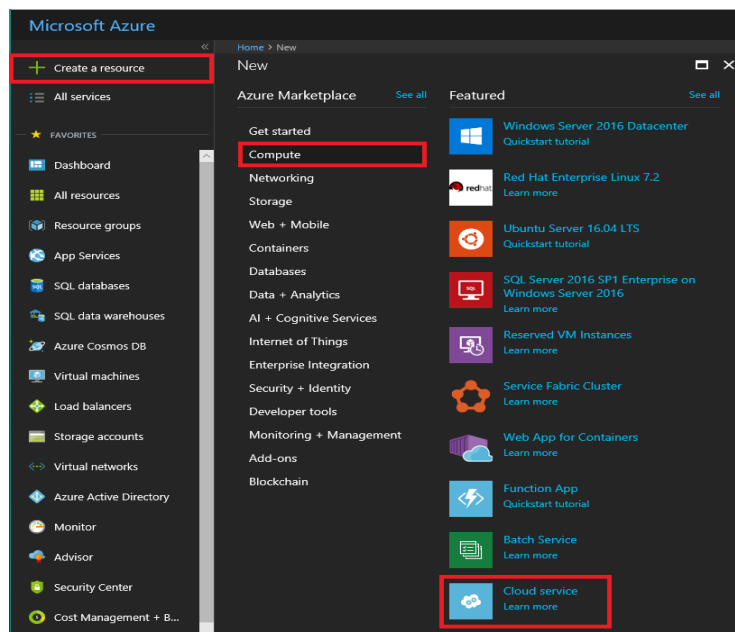


Fig. 4.3 Screen shot of Microsoft Azure Portal

3. In the new **Cloud Service** pane, enter a value for the **DNS name**.
4. Create a new **Resource Group** or select an existing one.
5. Select a **Location**.

6. Click **Package**. This opens the **Upload a package** pane (as shown in Figure 4.4). Fill in the required fields. If any of your roles contain a single instance, ensure **Deploy even if one or more roles contain a single instance** is selected.
7. Make sure that **Start deployment** is selected.
8. If you do not have any certificates to add, click **Create**.

The screenshot shows the 'Upload a package' pane in Microsoft Azure. The left pane, titled 'Cloud service (classic)', contains the following fields:

- DNS name:** A text input field with a placeholder 'Enter the name' and a '.cloudapp.net' domain suffix.
- Subscription:** A dropdown menu showing 'Visual Studio Ultimate with MSDN'.
- Resource group:** A text input field showing 'ssl-example-rg'.
- Location:** A dropdown menu showing 'West US'.
- Package:** A button labeled 'Package (Optional) Select a package'.
- Certificates:** A section labeled 'Certificates (Optional) Add certificates' with a lock icon.
- Pin to dashboard:** A checkbox.
- Create:** A blue button at the bottom.

The right pane, titled 'Upload a package', contains the following fields:

- Storage account:** A dropdown menu showing 'sslexamplestorage91274'.
- Deployment label:** A text input field showing 'ssl-example-label' with a green checkmark.
- Package:** A dropdown menu showing 'MultipleRoles.cspkg' with a green checkmark and a blue icon.
- Configuration:** A dropdown menu showing 'ServiceConfiguration.Cloud.cscfg' with a green checkmark and a blue icon.
- Environment:** A section with two tabs: 'Production' and 'Staging'.
- Deploy even if one or more roles contain a single instance:** A checkbox that is currently unchecked.
- Start deployment:** A checkbox that is currently checked.
- OK:** A blue button at the bottom.

Fig. 4.4 Microsoft Azure: Upload a Package Pane

## Upload a certificate

If your deployment package is configured to use certificates, you can upload the certificate now.

1. Select **Certificates**, and on the **Add certificates** pane, select the TLS/SSL certificate .pfx file, and then provide the **Password** for the certificate,

2. Click **Attach certificate**, and then click **OK** on the **Add certificates** pane.
3. Click **Create** on the **Cloud Service** pane (refer to Figure 4.5). When the deployment has reached the **Ready** status, you can proceed to the next steps.

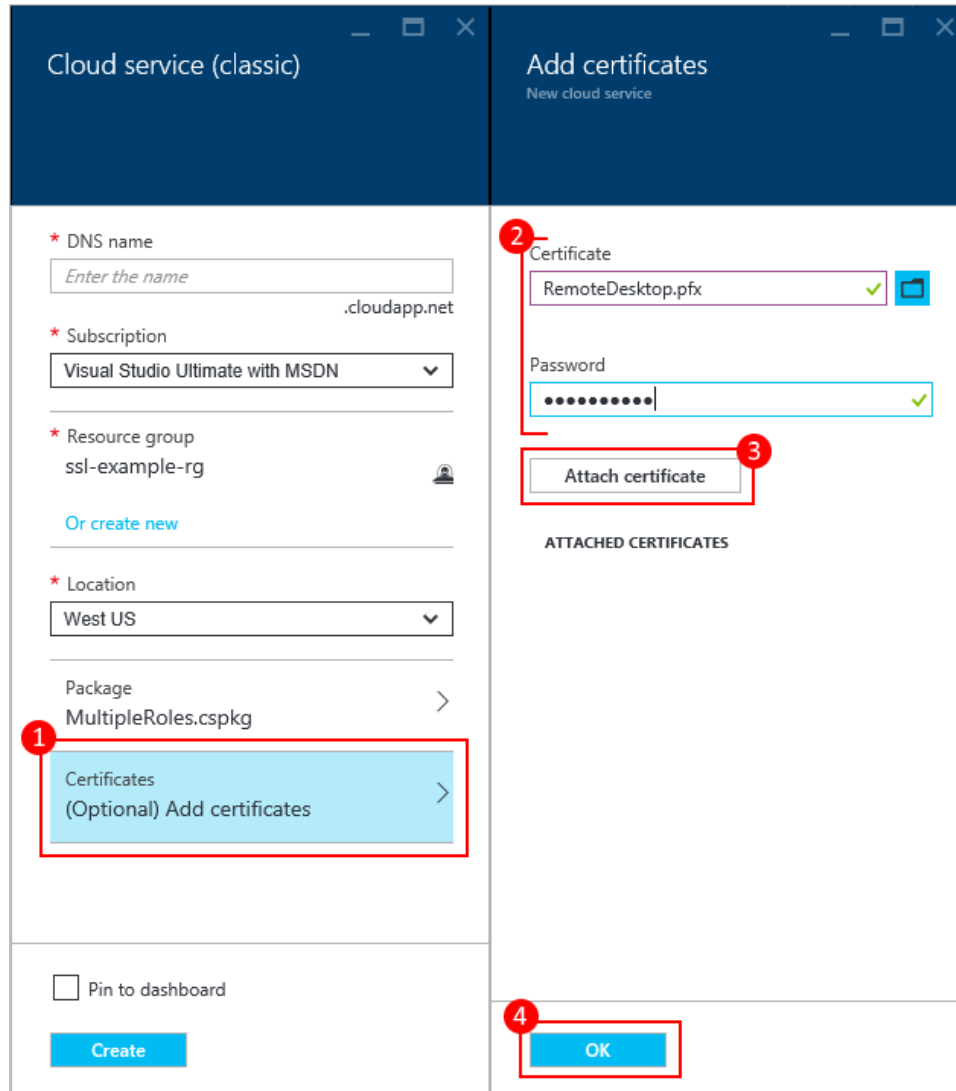


Fig. 4.5 Microsoft Azure: Adding Certificates

### Verify your deployment completed successfully

1. Click the cloud service instance. The status should show that the service is **Running** (as shown in Figure 4.6).
2. Under **Essentials**, click the **Site URL** to open your cloud service.

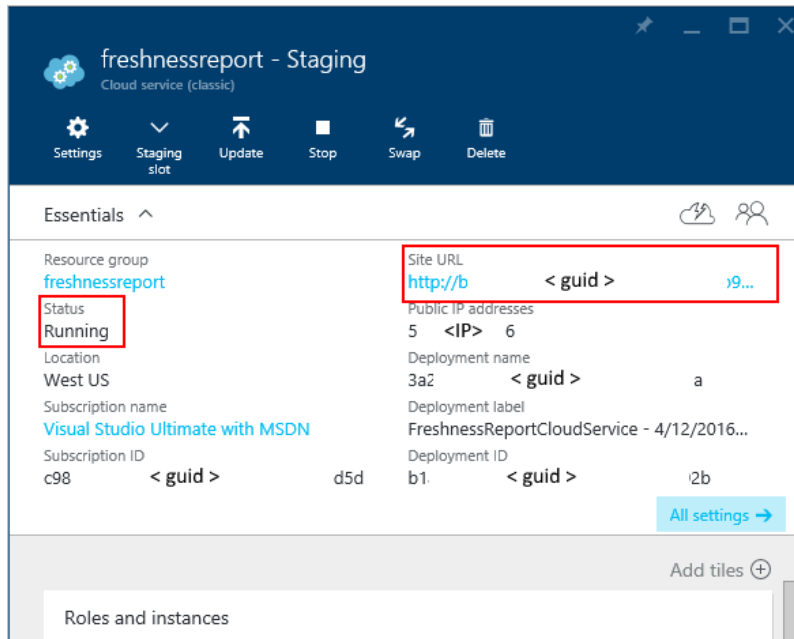


Fig. 4.6 Microsoft Azure: Checking Status

## 4.7.2 Installing Virtual Machine on Cloud

### Sign-in to Azure

Sign-in to the Azure portal at <https://portal.azure.com>.

### Create virtual machine

1. Type **virtual machines** in the search.
2. Under **Services**, select **Virtual machines**.
3. In the **Virtual machines** page, select **Create** and then **Virtual machine**.
4. In the **Basics** tab, under **Project details**, make sure the correct subscription is selected and then choose to **Create new** resource group. Type *myResourceGroup* for the name.

**Project details**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Resource group \* ⓘ  [Create new](#)

Fig. 4.7 Virtual Machine: Creating a Project



5. Under **Instance details**, type *myVM* for the **Virtual machine name** and choose your **Region**. Choose *Windows Server 2019 Datacenter* for the **Image** and *Standard\_DS1\_v2* for the **Size**.

The screenshot shows the 'Instance details' configuration page in the Azure portal. The fields are as follows:

- Virtual machine name \***: myVM
- Region \***: (US) East US
- Availability options**: No infrastructure redundancy required
- Image \***: Windows Server 2019 Datacenter - Gen1
- Azure Spot instance**:
- Size \***: Standard\_DS1\_v2 - 1 vcpu, 3.5 GiB memory

Fig. 4.8 Virtual Machine: Instance Details

6. Under **Administrator account**, provide a username and a password.

The screenshot shows the 'Administrator account' configuration page in the Azure portal. The fields are as follows:

- Username \***: azureuser
- Password \***: [masked]
- Confirm password \***: [masked]

Fig. 4.9 Virtual Machine: Administrator Account

7. Under **Inbound port rules**, choose **Allow selected ports** and then select **RDP (3389)** and **HTTP (80)** from the drop-down.

The screenshot shows the 'Inbound port rules' configuration page in the Azure portal. The configuration is as follows:

- Public inbound ports \***:  Allow selected ports
- Select inbound ports \***: HTTP (80), RDP (3389)

A warning message is displayed below the configuration:

**⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.**

Fig. 4.10 Virtual Machine: Inbound Port Rules

## Teacher's Note

Teacher should deploy azure cloud in front of students.

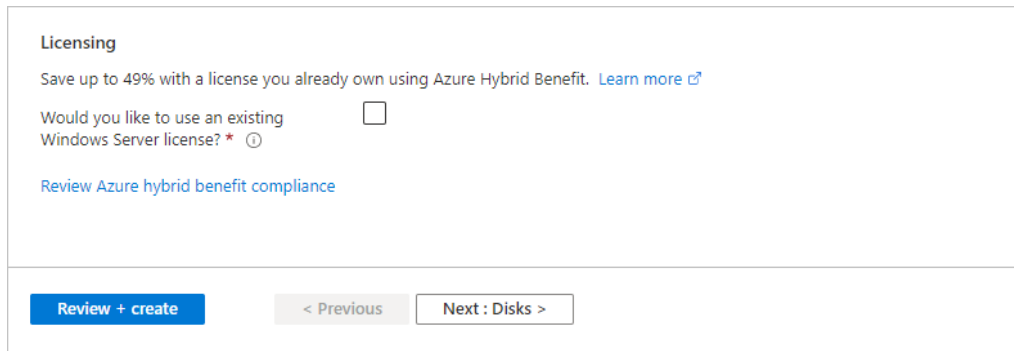


Fig. 4.11 Virtual Machine: Licensing

9. After validation runs, select the **Create** button at the bottom of the page.
10. After deployment is complete, select **Go to resource**.

## Connect to a Virtual Machine

Create a remote desktop connection to the virtual machine. The following steps show the steps to connect to your VM from a Windows computer.

1. On the overview page for your virtual machine, select the **Connect** button then select **RDP**.

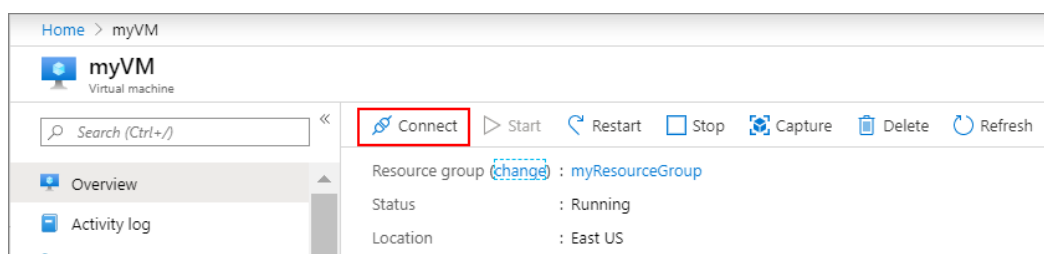


Fig. 4.12 Connecting to a Virtual Machine

2. In the **Connect with RDP** page, keep the default options to connect by IP address, over port 3389, and click **Download RDP file**.
3. Open the downloaded RDP file and click **Connect** when prompted.

4. In the **Windows Security** window, select **More choices** and then **Use a different account**. Type the username as **localhost\username**, enter the password you created for the virtual machine, and then click **OK**.
5. Click **Yes** or **Continue** to create the connection.

### Key Points

- "The cloud" refers to servers that are accessed over the Internet, and the software and databases that run on those servers.
- The cloud enables users to access the same files and applications from almost any device, because the computing and storage takes place on servers in a data center, instead of locally on the user device.
- Cloud computing is possible because of a technology called virtualization.
- Virtualization allows for the creation of a simulated, digital-only "virtual" computer that behaves as if it were a physical computer with its own hardware.
- A private cloud is a server, data center, or distributed network wholly dedicated to one organization.
- A public cloud is a service run by an external vendor that may include servers in one or multiple data centers.
- Hybrid cloud deployments combine public and private clouds, and may even include on-premises legacy servers.
- Multi-cloud is a type of cloud deployment that involves using multiple public clouds.
- A cloud service provider is a third-party company offering a cloud-based platform, infrastructure, application, or storage services.
- Instead of users installing an application on their device, SaaS applications are hosted on cloud servers, and users access them over the Internet.
- PaaS vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet.

## Exercise

### Choose the most suitable option.

1. "The cloud" refers to \_\_\_\_\_ that are accessed over the Internet.
  - a. Servers    b. Websites    c. Applications    d. Gateways
  
2. Cloud Deployment is made possible by \_\_\_\_\_.
  - a. Demonteization    b. Virtualization    c. Contextualization    d. Manipulation
  
3. \_\_\_\_\_ cloud is a service run by external vendor.
  - a. Private    b. Public    c. Virtual    d. All of these
  
4. \_\_\_\_\_ is a type of cloud deployment that involves using multiple public clouds.
  - a. Multi-cloud    b. IoT Cloud    c. private cloud    d. None of these
  
5. \_\_\_\_ vendors offer everything necessary for building an application.
  - a. IaaS    b. SaaS    c. IaaS    d. DaaS

### Give short answer to the following questions.

1. Define Cloud Computing.
2. Describe types of cloud deployment.
3. Describe different service models of cloud.
4. Describe advantages and disadvantages of cloud.
5. What are cloud service providers.
6. Enlist some cloud service providers.
7. Explain virtualization.
8. Describe inbound and outbound rules.
9. What is a resource group on azure?

### Practical Tasks

1. Deploy a Cloud server on azure.
2. Install virtual machine on azure cloud.

## Chapter 5: Basics of Data Science



### After Studying this chapter, you will be able to:

- define data science.
- understand the basic concepts, attributes, & features of data.
- explain relationship between data science & IoT.
- data preprocessing.
- concept of mean, median, mode and standard deviation/variance.
- define data visualization.
- explain components of data visualization.
- understanding of graphs and its types.
- describe heat map.
- identify trends in data.
- know about tools available in Python to visualize different type of data.
- process the import of CSV file in Python.
- calculate mean, median, mode, standard deviation, variance.
- define machine learning.
- define association rule and reinforcement learning.
- explain types of regression & classification.
- define supervised and unsupervised learning.
- understand training and testing of machine learning algorithm on a data set.
- understand and apply K nearest neighbor (KNN) on a data set.

## 5.1 Data Science

Data is commonly referred to as the “oil of the 21st century”. Data has incalculable benefits in business, research and our everyday lives. Your route to work, your most recent Google search for the nearest coffee shop, your Instagram post about what you ate, and even the health data from your fitness tracker are important in different ways.

## 5.2 Lifecycle of Data Sciences

Data science involves a plethora of disciplines and expertise areas to produce a holistic and refined look into raw data. Data scientists rely on artificial intelligence, especially its subfields of machine learning and deep learning, to create models and make predictions using algorithms and other techniques. Data science has a five-stage lifecycle that consists of:

1. Capture: Data acquisition, data entry, signal reception, data extraction
2. Maintain: Data warehousing, data cleansing, data staging, data processing, data architecture
3. Process: Data mining, clustering/classification, data modelling, data summarization
4. Communicate: Data reporting, data visualization, business intelligence, decision making
5. Analyse: Exploratory/confirmatory, predictive analysis, regression, text mining, qualitative analysis

## 5.3 IoT and Data Science

Data Science is crucial to the growth of IoT. The innovation of IoT provides proof of data science’s indispensable role within the modern tech sector. The IoT industry already has a data science talent shortage, and as the industry continues to expand, its hunger for data science talent will continue to expand.

### 5.3.1 Importance of Data Science in IoT

IoT is fundamentally concerned with computers and machines using networks to “talk” to each other, a process which occurs entirely through the exchange of data. Therefore, if data is the fuel that IoT runs on, data science algorithms turn that fuel into something useful. For example, virtual assistants such as Amazon’s Alexa use machine learning to power their speech recognition functions. The sophisticated hardware and networking setups that Alexa devices rely upon would be of little value if data scientists were not around to enable their core voice-command capabilities. IoT devices leverage a wide variety of data science capabilities to power the key functions.

## 5.4 Data Pre-processing

Raw data is often incomplete and has inconsistent formatting. The adequacy or inadequacy of data preparation has a direct correlation with the success of any project that involve data analytics. Data pre-processing involves transforming raw data to well-formed data sets so that data mining analytics can be applied. Pre-processing involves both data validation and data imputation. The goal of data validation is to assess whether the data is complete and accurate. In machine learning (ML) processes, data pre-processing is critical for ensuring large datasets are formatted in such a way that the data they contain can be interpreted and parsed by learning algorithms.

## 5.5 Mean, Median, Mode and Standard Deviation

Mean, median, and mode are different measures in a numerical data set. They try to summarize a dataset with a single number to represent a "typical" data point from the dataset.

**Mean:** The "average" number; found by adding all data points and dividing by the number of data points.

Mean = Sum of all observations / Number of observations

Mean =  $(12 + 34 + 45 + 50 + 24) / 5$

Mean =  $165/5 = 33$

**Median:** The middle number; found by ordering all data points and picking out the one in the middle (or if there are two middle numbers, taking the mean of those two numbers).

**Example:** Consider the data: 56, 67, 54, 34, 78, 43, 23. What is the median?

**Solution:**

Arranging in ascending order, we get: 23, 34, 43, 54, 56, 67, 78. Here, n (no.of observations) = 7

So,  $(7 + 1) / 2 = 4$

$\therefore$  Median = 4<sup>th</sup> observation

Median = 54

**Mode:** The most frequent number i.e., the number that occurs the highest number of times.

For example, in the data: 6, 8, 9, 3, 4, 6, 7, 6, 3 the value 6 appears the most number of times.

Thus, mode = 6.

**Standard Deviation:** Standard deviation is the average amount of variability in your dataset. It tells you, on average, how far each value lies from the mean. A high standard deviation means that values are generally far from the mean, while a low standard deviation indicates that values are clustered close to the mean.

## 5.6 Data Visualization

Data visualization is the graphical representation of information and data. Data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions. The following are the types of graphs and charts used to visualize the data:

- Bar Chart/Graph (see Figure 5.1).
- Pie Chart.
- Line Graph or Chart.
- Histogram Chart.
- Area Chart.
- Dot Graph or Plot.
- Scatter Plot.

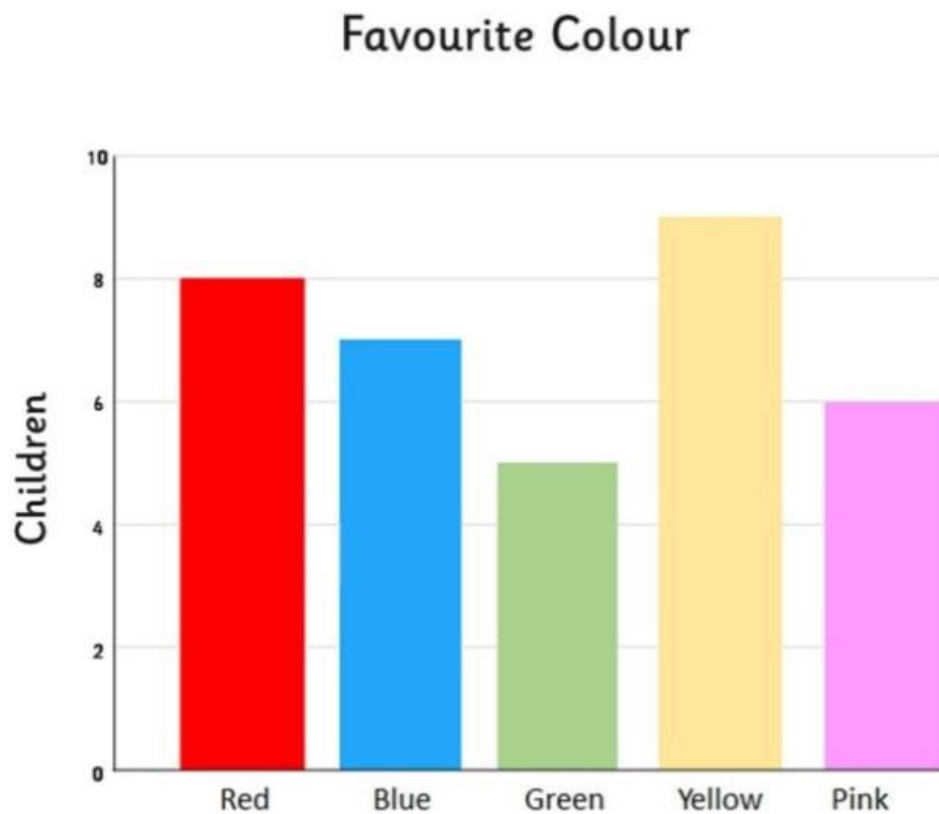


Fig. 5.1 Bar Plots



### 5.6.1 Heatmap

A heatmap is a graphical representation of data that uses a system of color-coding to represent different values (refer to Figure 5.2). Heatmaps are used in various forms of analytics but are most used to show user behaviour on specific webpages.

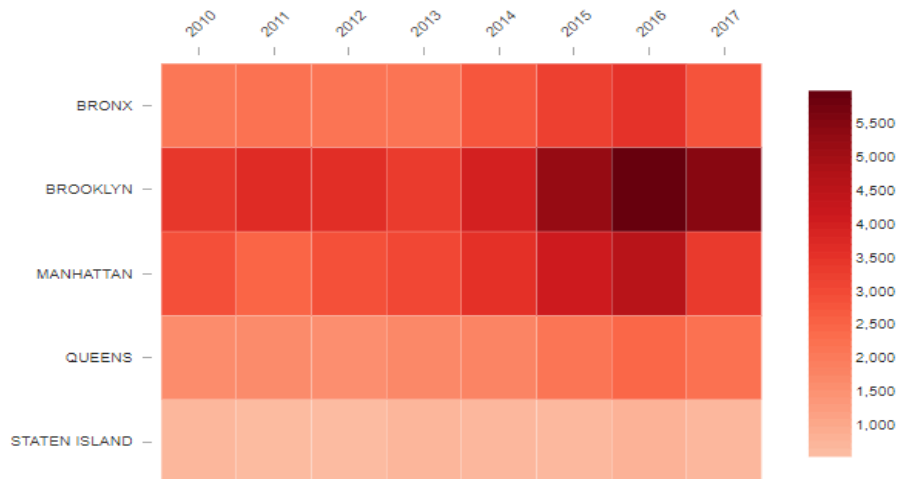


Fig. 5.2 Heat Map

### 5.6.2 Spotting trends

A trending quantity is a number that is generally increasing or decreasing in a pattern. Consider the following data about US life expectancy from 1920-2000:

Year	Life expectancy
1920	55.38
1930	59.57
1940	63.24
1950	68.07
1960	69.86
1970	70.86
1980	73.91

Year	Life expectancy
1990	75.4
2000	76.9

In this case, the numbers are steadily increasing decade by decade, so this an **upward trend**.

### 5.6.3 Visualization Tools in Python

The following are a few popular plotting libraries:

- **Matplotlib:** low level, provides lots of freedom.
- **Pandas Visualization:** easy to use interface, built on Matplotlib.
- **Seaborn:** high-level interface, great default styles.
- **Plotly:** can create interactive plots.
- **Orange:** Build data analysis workflows visually.

### Importing Datasets

In this Section, we will use two datasets which are freely available. The Iris and Wine Reviews dataset can both be loaded using pandas `read_csv` method.

:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Iris dataset head

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	winery
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian
4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sweet Cheeks

Wine Review dataset head

## Matplotlib

It is a low-level library with a MATLAB like interface which offers lots of freedom at the cost of having to write more code.

To install *Matplotlib*, *pip* and *conda* can be used as follows:

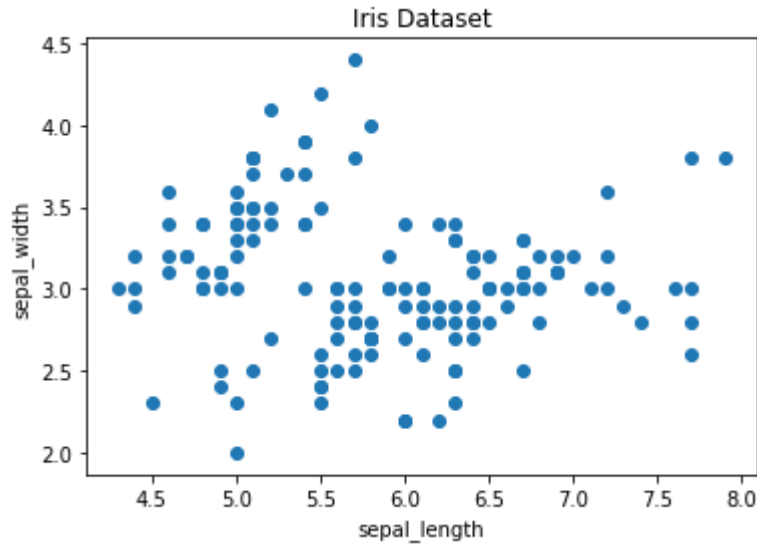
```
pip install matplotlib
or
conda install matplotlib
```

*Matplotlib* is specifically good for creating basic graphs like line charts, bar charts, histograms and many more. It can be imported by typing:

```
import matplotlib.pyplot as plt
```

## Scatter Plot

To create a scatter plot in Matplotlib we can use the `scatter` method. We will also create a figure and an axis using `plt.subplots` so that we can give our plot a title and labels.



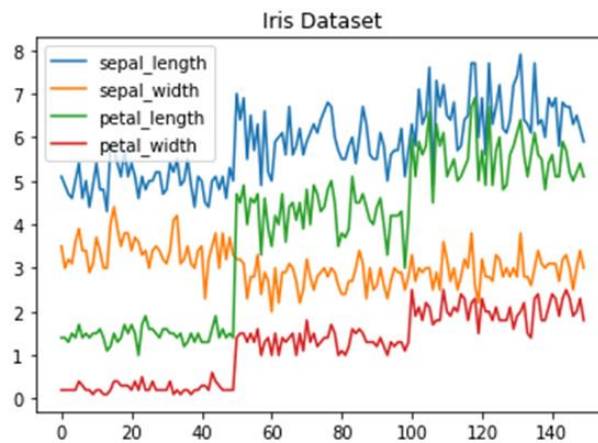
Matplotlib Scatter plot

### Do you know?

Conda is an open-source, cross-platform, language-agnostic package manager and environment management system.

### Line Chart

In Matplotlib we can create a line chart by calling the `plot` method. We can also plot multiple columns in one graph, by looping through the columns we want and plotting each column on the same axis.



Line Chart

## 5.7 Machine Learning (ML)

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. The process of learning begins with observations or data, such as examples, direct experience, or instructions, to look for patterns in data and make better decisions in future. The primary aim is to allow the computers learn automatically without human intervention.

### 5.7.1 Types of Machine Learning

There are the following types of machine learning algorithms:

1. Association Rule
2. Classification
3. Regression
4. Unsupervised Learning
5. Reinforcement Learning

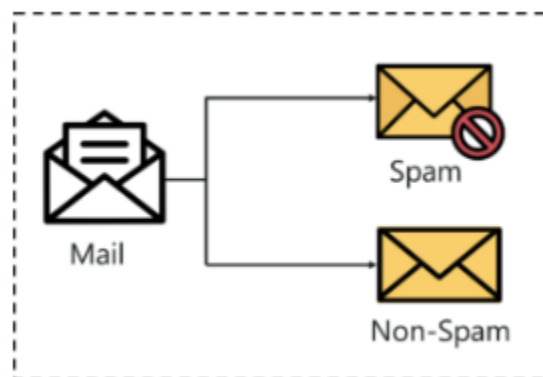
#### Association Rule

Association rule finds interesting associations and relationships among large sets of data items. This rule shows how frequently an itemset occurs in a transaction. A typical example is Market Based Analysis. Market Based Analysis is one of the key techniques used by large relations to show associations between items. It allows retailers to identify relationships between items that people buy together frequently (Refer to the following table).

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

## Classification

Classification is a process of categorizing a given set of data into classes. It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classification predictive modelling is the task of approximating the mapping function from input variables to discrete output variables.



Classification

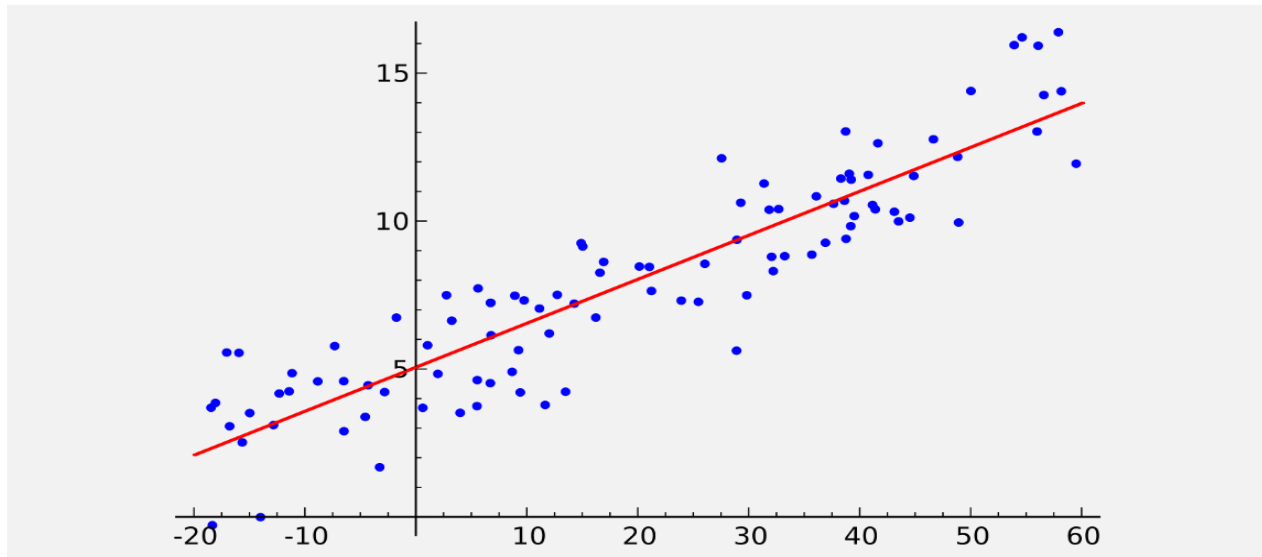
Heart disease detection can be identified as a classification problem, this is a binary classification since there can be only two classes i.e., has heart disease or does not have heart disease. The classifier, in this case, needs training the data to understand how the given input variables are related to the class. And once the classifier is trained accurately, it can be used to detect whether heart disease is there or not for a particular patient.

## Regression

Regression is a method of modelling a target value based on independent predictors. This method is mostly used for forecasting and finding out cause and effect relationship between variables.

### Teacher's Note

Give real-life examples of association rule, classification, and regression.



Linear Regression

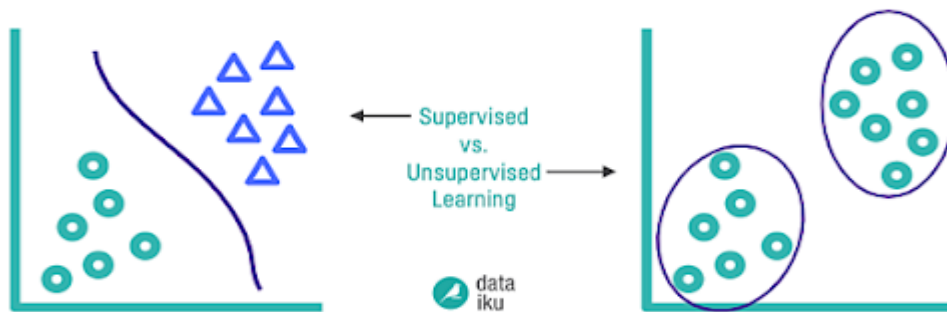
Linear regression is a type of regression analysis where the number of independent variables is one and there is a linear relationship between the independent variable and dependent variable. The red line in the above graph is referred to as the best fit straight line. Based on the given data points, we try to plot an optimal line that models the points. The line can be modelled based on the linear equation shown below.

$$y = a_0 + a_1 * x \quad \#\# \text{ Linear Equation}$$

The aim of the linear regression algorithm is to find the best values for  $a_0$  and  $a_1$ .

## Unsupervised Learning

Supervised learning refers to using a set of input variables to predict the value of a labeled output variable. It requires labeled data (think of this like an answer key that the model can use to evaluate its performance). Whereas, unsupervised learning refers to inferring underlying patterns from an unlabeled dataset without any reference to labeled outcomes or predictions.



Unsupervised Learning

There are several methods of unsupervised learning, but clustering is the most used unsupervised learning technique. Clustering refers to the process of automatically grouping together data points with similar characteristics and assigning them to “clusters.”

## Reinforcement Learning

Reinforcement learning is the training of machine learning models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning, an artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward. Although the designer sets the reward policy—that is, the rules of the game—he gives the model no hints or suggestions for how to solve the game. It is up to the model to figure out how to perform the task to maximize the reward, starting from totally random trials and finishing with sophisticated tactics and superhuman skills.

### 5.7.2 Training and Testing

Effective machine learning algorithms require quality training and testing data to make accurate predictions. Different datasets serve different purposes in preparing an algorithm to make predictions and decisions based on real-world data. In this section, we’ll compare training data vs. test data vs. validation data and explain the place for each in machine learning.

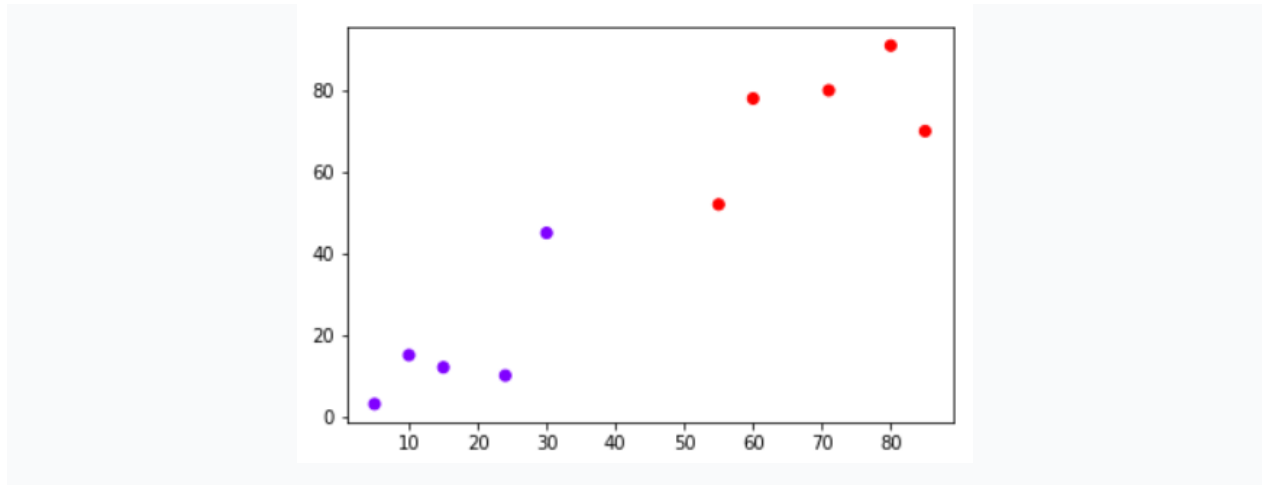
- **Training data:** This type of data builds up the machine learning algorithm. The data scientist feeds the algorithm input data, which corresponds to an expected output. The model evaluates the data repeatedly to learn more about the data’s behaviour and then adjusts itself to serve its intended purpose.
- **Validation data:** During training, validation data infuses new data into the model that it hasn’t evaluated before. Validation data provides the first test against unseen data, allowing data scientists to evaluate how well the model makes predictions based on the new data. Not all data scientists use validation data, but it can provide some helpful information to optimize hyperparameters, which influence how the model assesses data.
- **Test data.** After the model is built, testing data validates that it can make accurate predictions. If training and validation data include labels to monitor performance metrics of the model, the testing data should be unlabelled. Test data provides a final, real-world check of an unseen dataset to confirm that the ML algorithm was trained effectively.



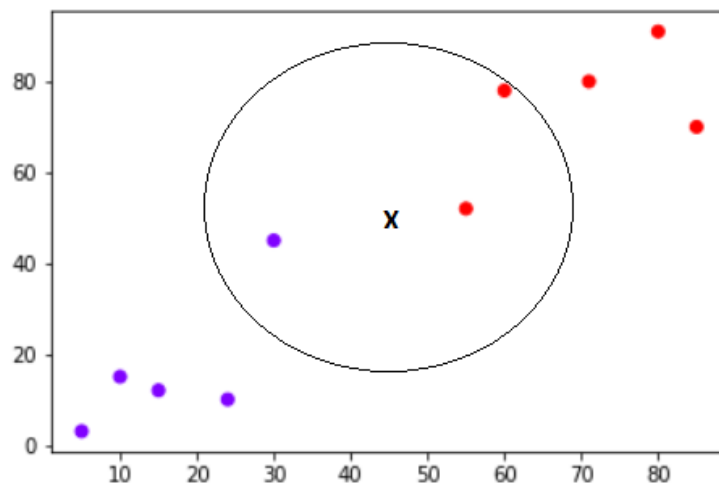
### 5.7.3 KNN Algorithm

The intuition behind the KNN algorithm is one of the simplest of all the supervised machine learning algorithms. It simply calculates the distance of a new data point to all other training data points. The distance can be of any type e.g., Euclidean or Manhattan etc. It then selects the K-nearest data points, where K can be any integer. Finally, it assigns the data point to the class to which most of the K data points belong.

Suppose you have a dataset with two variables, which when plotted, looks like the one in the following figure:



Your task is to classify a new data point with 'X' into "Blue" class or "Red" class. The coordinate values of the data point are  $x=45$  and  $y=50$ . Suppose the value of K is 3. The KNN algorithm starts by calculating the distance of point X from all the points. It then finds the 3 nearest points with least distance to point X. This is shown in the figure below. The three nearest points have been encircled.



The final step of the KNN algorithm is to assign new point to the class to which majority of the three nearest points belong. From the figure above we can see that the two of the three nearest points belong to the class "Red" while one belongs to the class "Blue". Therefore, the new data point will be classified as "Red".

## Implementation in Python

As we know K-nearest neighbors (KNN) algorithm can be used for both classification as well as regression. The following is the Python script to use KNN as classifier as well as regressor:

### KNN as Classifier

First, start with importing necessary python packages:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Next, download the iris dataset from its weblink as follows:

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

Next, we need to assign column names to the dataset as follows:

```
headernames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

Now, we need to read dataset to pandas dataframe as follows:

```
dataset = pd.read_csv(path, names = headernames)
dataset.head()
```

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Data Pre-processing is done with the help of the following script lines:

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
```

Next, we divide the data into train and test split. The following code splits the dataset into 60% training data and 40% of testing data:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40)
```

Next, data scaling is done as follows:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Next, we train the model with the help of KNeighborsClassifier class of sklearn as follows:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 8)
classifier.fit(X_train, y_train)
```

At last, we need to make prediction. It can be done with the help of the following script:

```
y_pred = classifier.predict(X_test)
```

Finally, we print the results as follows:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print(result1)
result2 = accuracy_score(y_test, y_pred)
print("Accuracy:", result2)
```

## Output

Confusion Matrix:

```
[[21 0 0]
```

```
[ 0 16 0]
```

```
[ 0 7 16]]
```

Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	21
Iris-versicolor	0.70	1.00	0.82	16
Iris-virginica	1.00	0.70	0.82	23
micro avg	0.88	0.88	0.88	60
macro avg	0.90	0.90	0.88	60
weighted avg	0.92	0.88	0.88	60

Accuracy: 0.8833333333333333

### Key Points

- Data preprocessing involves transforming raw data to well-formed data sets so that data mining analytics can be applied.
- Mean, median, and mode are different measures of centre in a numerical data set.
- The "average" number; found by adding all data points and dividing by the number of data points.
- The middle number; found by ordering all data points and picking out the one in the middle (or if there are two middle numbers, taking the mean of those two numbers).
- The most frequent number—that is, the number that occurs the highest number of times.
- The standard deviation is the average amount of variability in your dataset.
- A heatmap is a graphical representation of data that uses a system of color-coding to represent different values.
- Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
- Association rule mining finds interesting associations and relationships among large sets of data items.
- Classification is a process of categorizing a given set of data into classes,
- Regression is a method of modelling a target value based on independent predictors. This method is mostly used for forecasting and finding out cause and effect relationship between variables.
- Unsupervised learning refers to inferring underlying patterns from an unlabeled dataset without any reference to labeled outcomes or predictions.

## Exercise

### Choose the most suitable option.

1. \_\_\_\_ calculates the distance of a new data point to all other training data points.  
a. KNN   b. Association rule   c. SVM   d. Decision tree
2. KNN can be used for \_\_\_\_\_.  
a. Classification   b. Regression   c. Association   d. Both a and b
3. \_\_\_\_\_ is the training of machine learning models to make a sequence of decisions.  
a. Reinforcement learning   b. Association rule   c. SVM   d. Decision tree
4. \_\_\_\_\_ data builds up the machine learning algorithm.  
a. Training Data   b. Testing Data   c. Validation Data   d. None of these
5. \_\_\_\_\_ data tests the machine learning algorithm.  
a. Training Data   b. Testing Data   c. Validation Data   d. Both b and c

### Give short answer to the following question.

1. Define Data Science.
2. What is the relationship between IoT and Data Sciences?
3. Define Mean, Median and Mode.
4. Define Standard Deviation.
5. What do you mean by data visualization?
6. Name different types of graphs used for data visualization.
7. Define Machine Learning.
8. Differentiate between Supervised and Unsupervised learning.
9. Differentiate between Classification and Regression.
10. Describe Reinforcement learning.

### Practical Tasks

1. Plot the IRIS data using python.
2. Apply KNN on IRIS data using python and print the results.

## Chapter 6: IoT Security



### After Studying this chapter, you will be able to:

- define IoT Security.
- describe threats to IoT.
- describe security attacks on IoT.
- elaborate Wi-Fi attack.
- understand Wi-Fi encryption.
- understand the concept of ACL/MAC filtering in Wi-Fi router.
- apply AES/TKIP on IoT gateway.
- apply MAC address filtering.
- know end to end communication.
- describe micro services.
- understand devices:
  1. users account
  2. privileges setting
  3. update firmware
- describe isolation and its importance.
- understand *ssl*.
- install *ssl* certificate on client.
- understand physical security and techniques (deterrence, delay and detect).
- understand hashing and encryption.
- know about installing and using crypto module library.
- knowabout hashing function used in securing MQTT
- enlist security ambiguities in MQTT protocol
- enlist attacks and threat to MQTT protocol
- secure MQTT with encryption

## 6.1 IoT Security

IoT security keeps IoT systems safe. IoT security tools protect from threats and breaches, identify and monitor risks and help fix vulnerabilities. IoT security ensures the availability, integrity, and confidentiality of an IoT solution.

### 6.1.1 IoT threats and vulnerabilities:

As IoT continues to expand, the number of threats will continue to increase. Being able to identify and understand different types of threats and vulnerabilities associated with the Internet of Things can significantly reduce the risk of a data breach. The following are the main vulnerabilities in IoT:

#### 1. Lack of physical hardening

Most IoT devices are remotely deployed. There is no way to properly secure devices that are constantly exposed to the broader physical attack surface. Devices without a secure location and the inability for continual surveillance allow potential attackers to gain valuable information about their network's capabilities. For example, hackers can facilitate the removal of a memory card to read its contents and access private data and information that may allow them to access other systems.

#### 2. Insecure data storage and transfer

Whenever data is transferred, received, or stored through networks, the potential for a breach or compromised data also increases. Hence, it is important to ensure the secure transfer and storage of data through robust network security management tools.

#### 3. Lack of visibility and device management

Many IoT devices remain unmonitored, untracked, and improperly managed. As devices connect and disconnect from the IoT network, monitoring them can be very difficult. Lack of visibility into device status can prevent organizations from detecting or even responding to potential threats. These risks can become life-threatening when we investigate the healthcare sector.

#### 4. Botnets

Botnets are a series of internet-connected devices that are created to steal data, compromise networks, or send spam. Botnets contain malware that allows the attacker to access the IoT device and its connection to infiltrate an organization's network. They are most prominent in appliances that were not initially manufactured securely.

## 5. Weak passcodes

Inconsistent management of passcodes throughout the workplace enables hackers to compromise your entire business network. If employees do not adhere to advanced password management policies, the potential for a password-oriented attack increases. Practicing good password hygiene is essential to ensure your business is following standard security practices.

## 6. Insecure ecosystem interfaces

Application programming interfaces (APIs) are software intermediaries that allow two applications to talk to each other. With the connection of the two servers, APIs can introduce a new entrance for attackers to access a business's IoT devices and breach a network's router, web interface, server, etc.

### 6.2 Types of Wireless Network Attacks

There are several different types of Wi-Fi attacks that hackers use to eavesdrop on wireless network connections to obtain passwords and banking credentials. The main Wi-Fi attacks are described below:

#### **Fake Wi-Fi Access Points and Man-in-the-Middle Attacks**

Visitors to hotels, coffee shops and malls often connect to free Wi-Fi. Customers often choose the Wi-Fi access point based on the SSID without checking it is the wireless network set up by a particular establishment for customer use. Criminals can easily set up fake Wi-Fi access points, often using the name of the establishment in the SSID. A SSID called 'Free Airport Wi-Fi' would be enough to get many people to connect. When customers connect to these rogue Wi-Fi networks, they can still access the Internet. However, once connected to that network, everything they do online will be monitored by cybercriminals. Sensitive information entered online, such as email addresses and passwords, credit card numbers, or banking credentials, can be stolen. Fake access points are among the most common wireless network attacks. They are easy to conduct, require little technical skill, and are very effective. Evil twin attack is an example of this type of attack as shown in Figure 6.1.



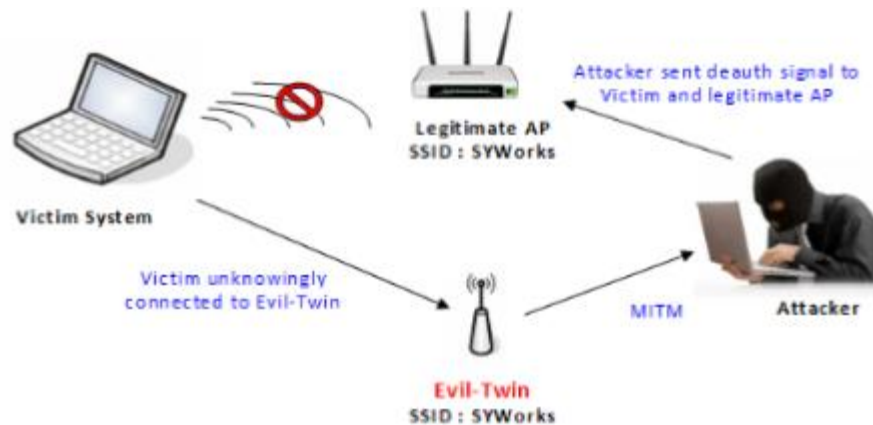


Fig. 6.1 Evil twin attack

## Packet Sniffing: Interception of Unencrypted Traffic

Hackers can use packet sniffers to intercept traffic on unencrypted Wi-Fi networks. Packet sniffing is one of the most common wireless attacks. These common wireless network attacks are easy on older routers, such as those using WEP encryption.

## Wardriving

War driving is a technique used to identify and map vulnerable access points. The name comes from the fact that attackers drive around a neighbourhood and use a laptop with a GPS device, antenna to identify and record the location of wireless networks. This technique is effective, since many Wi-Fi networks used by businesses extend beyond the confines of the building.

## Warshipping

Warshipping is a more efficient method of attacking Wi-Fi networks as it allows attacks to be conducted remotely, even if the attacker is not within range of a Wi-Fi.

## MAC Spoofing

Many businesses use MAC filtering to prevent specific devices from connecting to their Wi-Fi networks. While this is useful for preventing individuals from taking advantage of free Wi-Fi for customers, this method of blocking users can be easily bypassed. It is easy to spoof a MAC address and bypass this filtering control.

## Jamming Signals

An attacker can disrupt the network connection by jamming the signal, there are functioning tools for this purpose also called as creating noise.

## Misconfiguration Attacks

If a router is set up using the default configuration, weak credentials, weak encryption algorithms, then the attacker can easily break into the network.

## Unauthorised/Ad-hoc Connection Attacks

An attacker can enable an Ad-hoc connection in a user's system utilizing Trojan, malware. The attacker can compromise the connection operating in Ad-hoc mode since this mode does not provide stronger encryption to the connection.

## 6.3 Wireless Encryption

Wireless encryption secures a wireless network with an authentication protocol. It requires a password or network key when a user or device tries to connect. If the wireless network is not secure, unauthorized users could access your network and obtain personal information or use your internet connection for malicious or illegal activity. The following information provides details about different types of wireless encryptions that are commonly supported on most Wi-Fi enabled devices:

### Wired Encryption Privacy or Wired Encryption Protocol (WEP)

- **64-bit:** This configuration requires a 10 character password when you use a hexadecimal (zero to nine and A-F) digits or eight characters when you use ASCII characters.
- **128-bit:** This configuration requires a 26 character password when you use hexadecimal digits or 14 characters when you use ASCII characters.

### Wi-Fi Protected Access (WPA and WPA2)

- **TKIP:** Temporal Key Integrity Protocol
- **PSK:** Pre-shared Key or Personal mode. 256-bit encryption that requires a 64 hexadecimal digit password or an 8-63 ASCII character passphrase.
- **EAP:** Extensible Authentication Protocol

### Wi-Fi Protected Setup (WPS)

- **PIN (Personal Identification Number) Method:** A PIN is required from either a sticker label or the Web interface of the WPS device. This PIN is entered in the access point or client WPS device to make the connection.

- **PBC** (Push button configuration) Method: Simply push a button, either a physical or virtual button, on both WPS devices to make the connection.

## 6.4 MAC Filtering

MAC filtering is a security method based on access control. In Mac filtering, each address is assigned a 48-bit address which is used to determine if we can access a network. It helps in listing a set of allowed devices that you need on your Wi-Fi and the list of denied devices that you don't want on your Wi-Fi. It helps in preventing unwanted access to the network. In this way, we can blacklist or whitelist certain computers based on their MAC addresses. We can configure the filter to allow connection only to those devices included in the whitelist.

## 6.5 Important Concepts

### End-to-End Communication

End-to-end communication refers to the communication between two end nodes residing in a same or a separate network as shown in Figure 6.2:

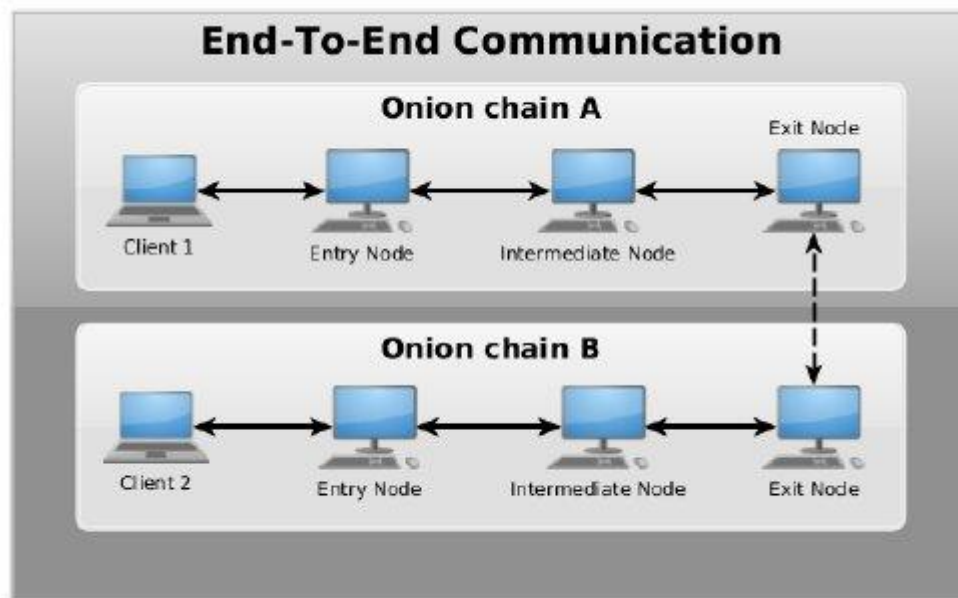


Fig. 6.2 End-end communication

### Microservices

Microservices are an architectural and organizational approach to software development where a software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams.

## User Account

A user account is a location on a network server used to store a computer username, password, and other information. A user account allows or does not allow a user to connect to a network, another computer, or other shares.

## Privileges Settings

The role's privileges determine the user's controls in the user console, information they can access, and tasks they can perform.

## Isolation

Network segmentation (often referred to as network isolation) is the concept of taking your network and creating silos within it called **VLANS** (virtual local area networks) that separates assets in the networked environment based on the function of the asset within the organization or some other schema.

## Secure Socket Layer

Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a server and a client typically a web server and a browser, or a mail server and a mail client (e.g., Outlook). It is more widely known as compared to TLS.

## Hashing algorithm

A hash can simply be defined as a number generated from a string of text. Other literature can also call it a message digest. In essence, a hash is smaller than the text that produces it. It is generated in a way that a similar hash with the same value cannot be produced by another text. From this definition, hashing is the process of producing hash values for the purpose of accessing data. In principle, hashing will take arbitrary input and produce a string with a fixed length. Hashing has the following attributes:

- A given known input must always produce one known output.
- Once hashing has been done, it should be impossible to go from the output to the input.
- Different multiple inputs should give a different output.
- Modifying an input should mean a change in the hash.

A hash algorithm is a function that can be used to map data of random size to data of fixed size. Hash values, hash codes and hash sums are returned by functions during hashing. The following are different types of hashing algorithms used in computing:

- **MD4:** It has a length of 128 bits and has influenced many posterior designs.
- **SHA algorithm:** It has a length of 160 bits.
- **RIPMEND:** It is a cryptographic hash algorithm designed by Hans Dobbertin. It has a length of 160 bits.

### Purpose of hashing

- Hashing can be used to compare a large amount of data.
- It is easy to find a record when the data is hashed.
- Hashing algorithms are used in cryptographic applications.
- Hashing is used to generate random strings to avoid duplication of data.
- Geometric hashing is widely used in computer graphics to find closet pairs and proximity problems in planes.

### Encryption

Encryption is a process of encoding simple text and other information that can be accessed by the sole authorized entity if it has a decryption key. It protects your sensitive data from being accessed by cybercriminals. It is the most effective way of achieving data security in modern communication systems. For the receiver to read an encrypted message, it should have a password or a security key that is used in decryption. Asymmetric encryption is also known as public-key encryption.

- **Symmetric encryption:** Uses the same secret key to encrypt and decrypt the message. The secret key can be a word, a number, or a string of random letters. Both the sender and the receiver should have the key.
- **Asymmetric encryption:** It deploys two keys, a public key known by everyone and a private key known only by the receiver. The public key is used to encrypt the message and a private key is used to decrypt it.
- **Hybrid encryption:** It is a process of encryption that blends both symmetric and asymmetric encryption.

### Purpose of encryption

The main idea of encryption is to protect data from an unauthorized person who wants to read or get information from a message that was not intended for them. Encryption enhances security when sending messages through the Internet or through any given network. The following are the key elements of security:

- **Confidentiality:** Encrypted message cannot be read or changed by another person.
- **Encrypt:** It transforms data in such a way that only specific individuals can transform the message.
- **Authentication:** The origin of the message received can be traced thus facilitating authentication.

## 6.6 Attacks in MQTT

MQTT is an application layer protocol to transfer data among many IoT devices. MQTT was designed for light-weight communications between constrained resource devices such as mobile phones and servers. Publisher, subscriber, and broker are the basic elements for establishing communication among devices in IoT. Initially, a publisher device sends request message i.e., CONNECT, to connect with the broker. After the request is received by the broker, the broker sends the acknowledgment, CONNACK, to the publisher device. After receiving acknowledgment from the broker, the publisher device sends or publishes the message on a specific topic to the broker, and finally the receiving devices subscribe to the messages from the broker.

There are several attacks against the MQTT protocol. The malicious devices get an access to the network, and it prevents the services offered by the broker during publishing and subscribing the messages. The MQTT attacks carried out are as follows:

### A. Denial-of-service attack

An attacker can initiate a DoS attack in the broker by frequently sending multiple connection requests hence making the broker busy as in flooding attack. If multiple connection requests reach at the same time, then the buffer will be drained and the broker will not be able to handle all new incoming requests. When the broker receives flood request messages, it starts to acknowledge with CONNACK message. During DoS attack, there is a rapid rate of increase in the number of CONNECT and CONNACK packets which halts the broker service.

### B. Man-in-the-middle attack.

Man-in-the-middle (MitM) attack interrupt the messages between two points to modify the content. This is done between a broker and the sensor by modifying the sensor data. MQTT was designed for light-weight communications between constrained resource devices such as mobile phones and servers. MQTT provides some security safeguards. This protocol enables a two-way hand shake by allowing client authentication. If SSL/TLS is available on the constrained resource devices then this mechanism allows for encryption of data in the message. When SSL/TLS is not

available, the user name and password that authenticate the client are in the clear instances. This two-way handshake is vulnerable to man-in-the-middle attacks.

### C. Intrusion

A network intrusion is an unauthorized activity on a computer network. Intrusion is detected based on the defenders having a clear understanding of how attack can work. In some cases, such an unused activity uses network resources for other uses, and nearly always threatens the safety of the network or its data or both.

## 6.6.1 How to Encrypt MQTT Payloads with Pytho

Encrypting the MQTT payload rather than the link has the advantage that the data is encrypted end to end and not just between the broker and the client. It also means that the intermediate brokers don't need to support SSL and that you don't need to obtain and install certificates. As a demonstration, we have modified the simple pub-sub script to use payload encryption. The following is a screen shot of the script with annotations:

### Do you know?

Andy Stanford-Clark (IBM) and Arlen Nipper (then working for Eurotech, Inc.) authored the first version of the MQTT protocol in 1999. It was used to monitor oil pipelines within the SCADA industrial control system.

```
import time
import paho.mqtt.client as paho
from cryptography.fernet import Fernet
broker="broker.hivemq.com"
#define callback
def on_message(client, userdata, message):
    #time.sleep(1)
    print("receive payload ",message.payload)
    if message.payload==encrypted_message:
        print("\npublished and received messages are the same")
        decrypted_message = cipher.decrypt(message.payload) #decrypted mes:
        print("\nreceived message =",str(decrypted_message.decode("utf-8")))

client= paho.Client("client-001") #create client object client1.on_pub:
#####
client.on_message=on_message
#####encryption
cipher_key = Fernet.generate_key()
cipher = Fernet(cipher_key)
message = b'on'
encrypted_message = cipher.encrypt(message)
out_message=encrypted_message.decode()# turn it into a string to send
##
print("connecting to broker ",broker)
client.connect(broker)#connect
client.loop_start() #start loop to process received messages
print("subscribing ")
client.subscribe("house/bulb1")#subscribe
time.sleep(2)
print("publishing encrypted message ", encrypted_message)
client.publish("house/bulb1", out_message)#publish
time.sleep(4)
client.disconnect() #disconnect
client.loop_stop() #stop loop
```

**Note 1** points to the import of Fernet.

**Note 2** points to the generation of a cipher key.

**Note 3** points to the encryption of the message.

**Note 4** points to the publishing of the encrypted message.

**Note 5** points to the comparison of the received payload with the encrypted message.

**Note 6** points to the decryption of the received message.

Fig. 6.3 Pub-sub script with payload encryption



## Running the Script

The output of the script is as follows:

```
>>>
connecting to broker broker.hivemq.com
subscribing
publishing encrypted message b'gAAAAABZqDnlrJ_zsTeoQZrN9
yZu0isXJkLW3RPosDRRtEN4piW7mNSijLy83xIFHC22pIj9s4nlgLIbYm
5cvXhrjVCQInUh2g=='
receive payload b'gAAAAABZqDnlrJ_zsTeoQZrN9yZu0isXJkLW3R
PosDRRtEN4piW7mNSijLy83xIFHC22pIj9s4nlgLIbYm5cvXhrjVCQInU
h2g=='

published and received messages are the same

received message = on ←----- This is the message we sent
```

Fig. 6.4 Output of the Pub-sub script

### Key Points

- IoT security tools protect from threats and breaches, identify and monitor risks and can help fix vulnerabilities.
- Botnets are a series of internet-connected devices that are created to steal data, compromise networks, or send spam.
- Hackers can use packet sniffers to intercept traffic on unencrypted Wi-Fi networks.
- War driving is a technique used to identify and map vulnerable access points.
- Warshipping is a more efficient method of attacking Wi-Fi networks as it allows attacks to be conducted remotely, even if the attacker is not within range of a Wi-Fi network.
- An attacker can set up fake access points/hotspots with the same SSID as that of a public wi-fi AP; thus, he can set traps for the users who connect to these AP's.
- Wireless encryption secures your wireless network with an authentication protocol.
- MAC filtering is a security method based on access control. In this, each address is assigned a 48-bit address which is used to determine whether we can access a network or not.
- A network intrusion is an unauthorized activity on a computer network. Intrusion is detected based on the defenders having a clear understanding of how attack can work.



## Exercise

### Choose the most suitable option.

1. \_\_\_\_ are a series of internet-connected devices that are created to steal data, compromise networks, or send spam.
  - a. Botnets
  - b. Intruders
  - c. Wardrivers
  - d. Warshippers
2. Hackers can use \_\_\_\_ to intercept traffic on unencrypted WiFi networks.
  - a. Packet Sniffers
  - b. Spoofers
  - c. Spam
  - d. Both a and b
3. \_\_\_\_ is a technique used to identify and map vulnerable access points.
  - a. Warshipping
  - b. Wardriving
  - c. Spoofing
  - d. Snifing
4. \_\_\_\_ allows attacks to be conducted remotely, even if the attacker is not within range of a WiFi network.
  - a. Warshipping
  - b. Wardriving
  - c. Spoofing
  - d. Snifing
5. \_\_\_\_ is a security method based on access control.
  - a. IP filtering
  - b. MAC filtering
  - c. Encyption
  - d. Authentication

### Give short answer to the following question.

1. Define IoT Security.
2. What are the vulnerabilities in an IoT network?
3. What are the common security attacks in IoT?
4. What are the Wi-Fi attacks?
5. Describe different Wi-Fi encryption techniques.
6. Explain MAC address filtering.
7. Define End-to-End communication.
8. Describe microservices.
9. Describe Hashing.
10. Define SSL and SSL certificates.

### Practiacal Tasks

1. Apply AES/TKIP on IoT gateway
2. Apply MAC address filtering.
3. Use MQTT on application layer
4. Apply encryption on MQTT

## Chapter 7: Soft Skills



### After studying this chapter, you will be able to:

- know basic soft skills.
- understand the importance of soft skills in daily life.
- apply soft skills for academic and professional success.
- know model of communication.
- know importance of active listening and responding.
- understand effective communication.
- identify obstacles in communication.
- know the importance of teamwork in a professional environment.
- understand the concept of teamwork and leadership.
- know the concept of better time management.
- observe time management in daily life.
- understand professional and personal time management.
- know the concepts of attitude and behavior.
- understand the impact of positive and negative attitude in professional life.

## 7.1 Introduction to Soft Skills

### 7.1.1 Basic Soft Skills:

Soft skills are personality traits, social graces, personal habits, friendliness, and optimism that mark people to different degrees. Soft skills complement hard skills, which are the technical requirements of any job.



Fig. 7.1 Basic Soft Skills

### 7.1.2 Importance of Soft Skills

Soft skills enable students with a strong conceptual and practical framework to develop and manage teams. They play an important role in the development of the students' overall personality by enhancing their career prospects. Training in soft skills provides strong practical orientation to the students and help them in building and improving their communication skills, effective use of English, business correspondence, presentations, team building, leadership, time management, group discussions, interviews and interpersonal skills. Soft Skills are required to improve one's attitudes, values, beliefs, motivation, desires, willingness to embrace new ideas, goal orientation, persuasion, futuristic thinking, comparison, diplomacy, and various skills sets and etiquette to be able to deal with different situations diligently and responsibly.



Fig. 7.2 Important Soft Skills

### 7.1.3 Soft Skills for Academic & Professional Success:

Soft skills are increasingly important indicators of success both at school and in adult life. Careers that focused on social skills grew more rapidly rather than high math low social skill occupations. New technologies result in an increase in the importance of skills. Machines are better than humans at performing routine tasks that follow explicit rules, but people are much better at tasks that require flexibility, creativity, and judgment qualities that do not require an explicit understanding of rules.

Effective teamwork requires a complex and context-dependent understanding of one's team members and their likely responses to a wide range of scenarios. It includes communication skills, interpersonal skills, group dynamics, teamwork, body language, etiquettes, selling skills, presentation skills, confidence building etc.

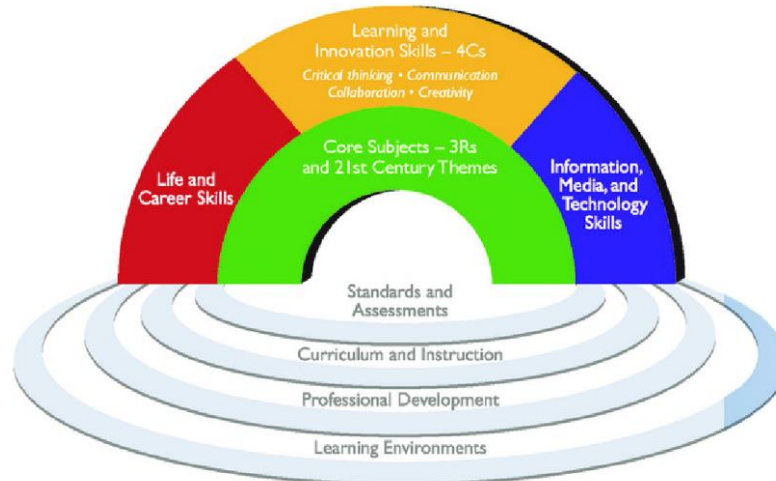


Fig. 7.3 Framework for 21<sup>st</sup> Century

## 7.2 Communication Skills

Communication skills refer to the skills to interact with others.

### 7.2.1 Model of Communication

Models of communication simplify the communication process by providing a visual representation of the various aspects of a communication encounter. The following are the three models of communication:

- i. Linear or Transmission model of Communication.
- ii. Interaction model of Communication.
- iii. Transaction model of Communication.

#### Linear or Transmission model of Communication

The linear or transmission model of communication describes communication as a linear, one-way process in which a sender intentionally transmits a message to a receiver. Although the receiver is included in the model, this role is viewed as more of a target or end point rather than part of an ongoing process.

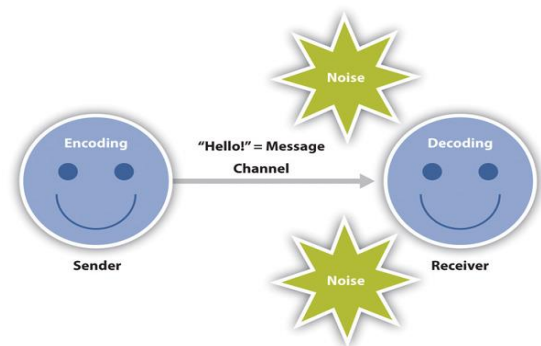


Fig. 7.4 Linear or Transmission model of Communication

## Interaction model of Communication

The interactive or interaction model of communication describes communication as a process in which participants alternate positions as sender and receiver and generate meaning by sending messages and receiving feedback within physical and psychological contexts. Rather than illustrating communication as a linear, one-way process, the interactive model incorporates feedback, which makes communication a more interactive, two-way process.

The interactive model takes physical and psychological context into account. Physical context includes the environmental factors in a communication encounter. Psychological context includes the mental and emotional factors in a communication encounter.

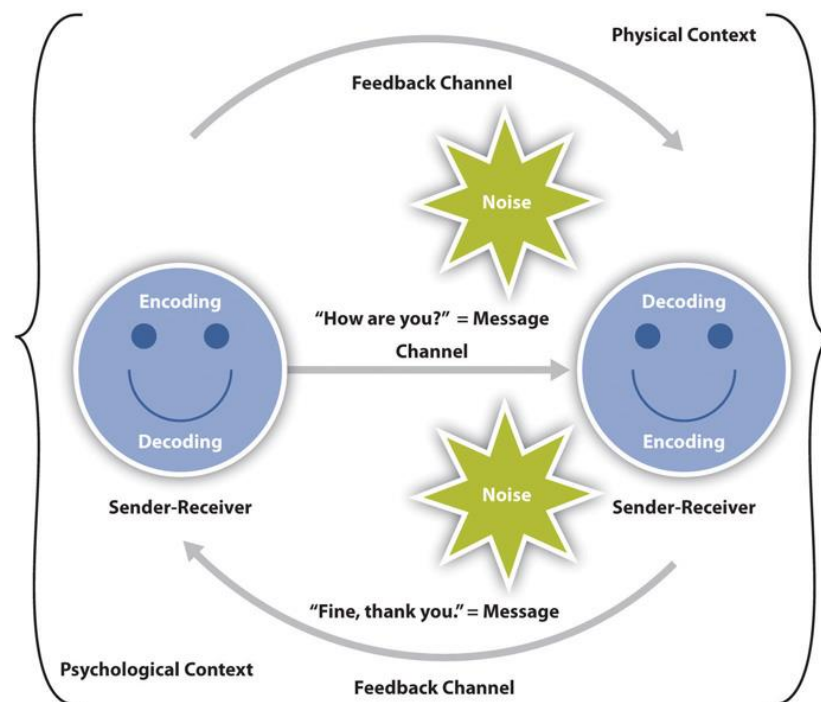


Fig. 7.5 Interactive model of Communication

## Transaction model of Communication

The transaction model of communication describes communication as a process in which communicators generate social realities within social, relational, and cultural contexts. In this model, we do not just communicate to exchange messages; we communicate to create

relationships, form intercultural alliances, shape our self-concepts, and engage with others to create communities.

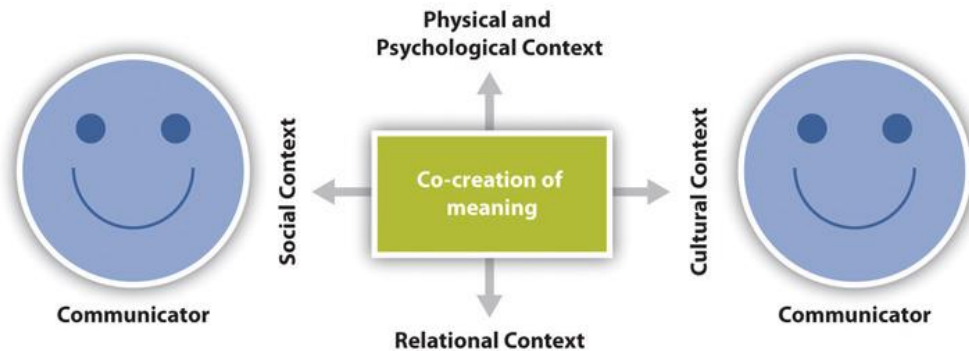


Fig. 7.6 Transaction model of Communication

The roles of sender and receiver in the transaction model of communication differ significantly from the other models. Instead of labelling participants as senders and receivers, the people in a communication encounter are referred to as communicators.

## 7.2.2 Importance of Active Listening and Responding

Effective communication consists of both speaking and listening. Active listening is when you are fully aware and concentrate on what is being said rather than passively hearing what the speaker is trying to convey. The goal of active listening is to acquire information, listen to understand people and situations before responding to it. The following are several benefits of being an active listener:

1. It helps you build connections
2. It helps you build trust
3. It helps you identify and solve problems
4. It helps you increase your knowledge and understanding of various topics
5. It helps you avoid missing critical information

## 7.2.3 Effective Communication

Effective communication is a process of exchanging ideas, thoughts, knowledge, and information to serve a purpose.



- **Characteristics of Effective Communication**
- **Clear Message:** The message which the sender wants to convey must be simple, easy to understand and systematically framed to retain its meaningfulness.
- **Correct Message:** The information communicated must not be vague or false in any sense.
- **Complete Message:** Communication is the base for decision making. If the information is incomplete, it may lead to wrong decisions.
- **Precise Message:** The message sent must be short and concise to facilitate straightforward interpretation and take the desired steps.
- **Reliability:** The sender must be sure that whatever he is conveying is right by his knowledge even if the receiver has trust on the sender.
- **Consideration of the Recipient:** The medium of communication and other physical settings must be planned, keeping in mind the attitude, language, knowledge, education level and position of the receiver.

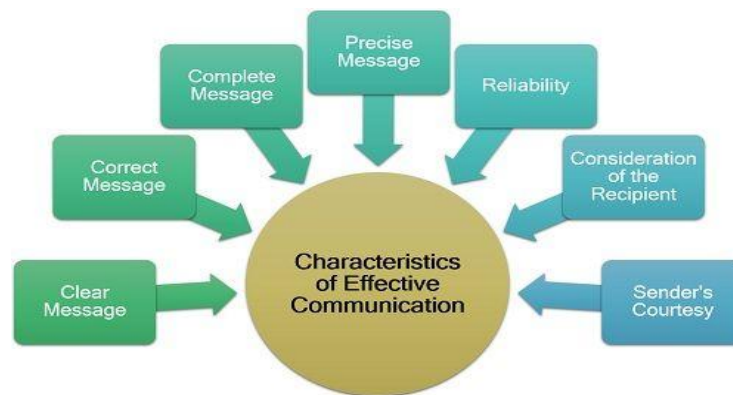


Fig. 7.7 Characteristics of Effective Communication

### Effective Communication Skills

- **Observance:** A person must possess sharp observing skills to gain more knowledge and information.
- **Clarity and Brevity:** The message must be drafted in simple words, and it should be clear and precise to create the desired impact.
- **Listening and Understanding:** The most crucial skill in a person is he must be a good, alert and patient listener.



- **Emotional Intelligence:** A person must be emotionally aware and has the ability to influence others.
- **Self-Efficacy:** One must have faith in himself and his capabilities to achieve the objectives of communication.
- **Self-Confidence:** Being one of the essential communication skills, confidence enhances the worthiness of the message being delivered.
- **Respectfulness:** Delivering a message with courtesy and respecting the values, beliefs, opinions and ideas of the receiver is the essence of effective communication.
- **Non-Verbal Communication:** To connect with the receiver in a better way, the sender must involve the non-verbal means of communication. These include gestures, facial expressions, eye contact, postures, etc.
- **Selection of the Right Medium:** Choice of the correct medium for communication is also a skill. It is necessary to select an appropriate medium according to the situation, priority of the message and the receiver's point of view, etc.



Fig. 7.8 Effective Communication Skills

## 7.2.4 Obstacles in Communication

There are certain obstacles which hinder the process of communication, making it less useful for the sender as well as the receiver. These barriers are categorized under three groups:



Fig. 7.9 Barriers in Effective Communication

### Barriers Involving Words

Words play an essential role in communication. Any disturbance or distraction in the way a message is presented may lead to miscommunication. The following are different types of communication barriers related to words:

- ✓ Language
- ✓ Ambiguity and Overuse of Abstractions
- ✓ Disorganized Message
- ✓ Information Overload

### Barriers Involving People's Background

People belong to different backgrounds, i.e., culture, education level, gender, etc. These attributes majorly affect the efficiency of the communication process. It involves the following challenges:

- ✓ Attitudinal Differences
- ✓ Demographic Differences
- ✓ Lack of Common Experience or Perspective
- ✓ Jumping to Conclusions

## Physical Barriers

These barriers can be experienced directly but are challenging to overcome. These include:

- ✓ Physical Distance
- ✓ Noise

## 7.3 Leadership and Teamwork

Teamwork means that people try to cooperate, using their individual skills and providing constructive feedback, despite any personal conflict between individuals. Teamwork is selfless. It focuses on the end goal. Thus, the foundation for teamwork is a common goal.

### 1. Teamwork is efficient work

- Split difficult tasks into simpler ones, then work together to complete them faster
- Develop specialized skills, so that the best person for each task can do it better and faster
- It leads to better productivity, reduced costs and greater profitability

### 2. Teams self-monitor

In teamwork, many people have responsibility for the same goal. Most significantly, teammates observe and depend on the quality of each other's work. When one team member's performance dips, the others have the knowledge and motivation to help them improve.

### 3. Teams innovate faster

When a team tackles a problem, the project benefits from multiple perspectives, skillsets, and experiences all at once. A team approach can therefore lead to faster, deeper innovation.

### 4. Teammates learn from each other

Working together make you learn each other's strengths and correct each other's mistakes.

## Group Activity

All participants will be playing in teams of four, and they need to be at their computers. Send each person a one-word clue. All four clues together will reveal a common theme. For example, 'bowl', 'vet', 'walk', and 'lead' are all connected by the theme 'dog'. Using only email to communicate, the group must coordinate to find the answer and be the first team to email it to you. Completing this challenge underlines the importance of collaboration.

### 7.3.2 Concept of Teamwork and Leadership

Teamwork always leads to greater productivity. Leadership and teamwork have a direct impact on the ability of an organization. Good leadership provides a clear vision for the team. Both leadership and teamwork are driven by critical soft skills that need to be exercised in a balanced manner.

Organizational leadership is a composite of skills and behaviours enabling a person to exercise an interpersonal influence on a group of people. The leader's vision and purpose are thus achieved by directing and motivating the team to accomplish the desired set of goals envisioned by the leader.

Teams are an essential component of successful organizations today and building and motivating teams are necessary pursuits to attain that success. Teams require continuous nurturing and interaction to maintain high performance throughout their temporary lives.

## 7.4 Time Management

### 7.4.1 Concept of Better Time Management

“Time management” is the process of organizing and planning how to divide your time between specific activities. Good time management enables you to work smarter not harder. Failing to manage your time damages your effectiveness and causes stress. Time Management refers to making the best use of time.



Fig. 7.10 Time Management

## 7.4.2 Time Management in Daily Life

The ability to manage your time effectively is important. Good time management leads to improved efficiency and productivity, less stress, and more success in life.

The following are the ways to manage time effectively:



Fig. 7.11 Time Management Tips

1. Set goals correctly
2. Prioritize wisely
3. Set a time limit to complete a task
4. Take a break between tasks
5. Organize yourself
6. Remove non-essential tasks/activities
7. Plan ahead

## 7.4.4 Personal and Professional Time Management

Time Management plays a very important role not only in organizations or the lives of entrepreneurs but also in our personal lives. The importance of time management comes down to how much it impacts your personal and professional life. Time management is organizing your day so that you find the best use for every moment. Excellent time management allows you to create a healthy balance in your workflow and family life. The following are the excellent ways of managing time:

- Effective Planning
- Setting goals and objectives
- Get organized
- Setting deadlines
- Delegate Tasks
- Prioritizing activities as per their importance
- Spending the right time on the right activity

## 7.5 Attitude, Behaviour and Customer Care

**Attitude** is a feeling, belief, or opinion of approval or disapproval towards something. **Behavior** is an action or reaction that occurs in response to an event or internal stimuli (i.e., thought). Attitude's structure can be described in terms of three components.

- **Affective component:** It involves a person's feelings / emotions about the attitude object. For example: "I am scared of spiders".
- **Behavioural component:** It refers to behavioural responses. For example: "I will avoid spiders and scream if I see one".
- **Cognitive component:** It involves a person's belief / knowledge about an attitude object. For example: "I believe spiders are dangerous".

This model is known as the ABC model of attitudes. One of the underlying assumptions about the link between attitudes and behaviour is related to consistency. This means that we often or usually expect the behaviour of a person to be consistent with the attitudes that they hold. This is called the principle of consistency.

## Attitude Strength

The strength with which an attitude is held is often a good predictor of behaviour. The stronger the attitude the more likely it should affect behaviour. Attitude strength involves:

**Importance / personal relevance** refers to how significant the attitude relates to self-interest.

The **knowledge** aspect of attitude strength covers how much a person knows about the attitude object.

### 7.5.2 Impact of Positive and Negative Attitude in Professional Life

Having a “positive attitude” means a person believes everything happens for the best in the end.

A person with a “negatives attitude” tends to believe their best days are in the past. A person with a negative attitude pays attention to other people's shortcomings.

### How Positivity Impacts a Workplace

When there are positive attitudes in a workplace, there is a feeling that anything can be accomplished. Colleagues support each other and work in tandem, and a host of other positive outcomes materialize, such as:

- Increased productivity
- Greater probability of collaboration and teamwork
- Improved morale
- Ability to overcome adversity
- Willingness to think creatively and try new things
- Willingness to share information and ideas
- Lower turnover
- Increased sense of camaraderie

Positivity can be contagious, where everyone feels like they are in the same team, the effort is collective, and everyone's ideas are valued.

### Impact of Negativity

A negative attitude does not just put others in a bad mood it also has a tangible impact on how a team works. The following are some of the effects of an unchecked negative attitude:

- Unwillingness to work collaboratively
- Unwillingness to try new things
- Reduced energy levels
- Depressive feelings



### Key Points

- Soft skills relate to how you work. Soft skills include interpersonal skills, communication skills, listening skills, time management and empathy, among others.
- Soft skills are an essential part of improving one's ability to work with others and can have a positive influence on furthering your career.
- Soft skills relate to academic success are study skills, listening skills, organization skills, and writing skills. Soft skills relate to professional success are attitude, communication, work ethic, teamwork, leadership qualities, time management, decision making and conflict resolution.
- The three most well-known models for communication are Linear, Interactional, and Transactional.
- Active listening and responding avoids misunderstandings, as people have to confirm that they do really understand what another person has said. It tends to open people up, to get them to say more. It helps people avoid conflicts, because people become more attuned to concerns and don't feel as though they're being dismissed.
- Effective Communication is defined as the ability to convey information to another effectively and efficiently.
- These barriers can be categorized into four main types of constraints to effective communication: Physical, psychological, organizational, as well as semantic barriers.
- Collaboration within a group can help solve difficult problems. Brainstorming is a good opportunity for the team to exchange ideas and come up with creative ways of doing things.
- Teamwork is the ability to work cooperatively with others to achieve group objectives. The essence of leadership is accomplishing worthy goals through the combined efforts of others, and teamwork capabilities are crucial.
- Better time management enables you to work smarter not harder so that you get more done in less time, even when time is tight and pressures are high
- Managing your time wisely improves work-life balance and increases happiness. Good time management also reduces stress and allows you to achieve your goals faster and easier.



## Exercise

### Choose the most suitable option.

- Which of the following are the 3 models of communication?  
a. Linear   b. Interaction   c. Transaction   d. All of these
- The linear or transmission model of communication describes communication as a \_\_\_\_\_ process  
a. One-way   b. Two-way   c. 3-way   d. Both a and b
- \_\_\_\_\_ is the ability to work cooperatively with others to achieve group objectives.  
a. Teamwork   b. Talent   c. Hardwork   d. Working individually
- \_\_\_\_\_ relate to academic success are study skills, listening skills, organization skills, and writing skills.  
a. Technical skills   b. Communication skills   c. Soft skills   d. Attitude
- \_\_\_\_\_ involves a person's belief / knowledge about an attitude object. For example: "I believe spiders are dangerous".  
a. Cognitive component   b. Behavioral component   c. Affective component   d. None of these

### Give short answer to the following question.

- Describe three advantages of team work.
- Describe three disadvantages of negative mindset.
- What are the three components of attitude?

## GLOSSARY

**Context Dependent:** Dependent upon the circumstances.

**Cognitive:** Relating to knowledge.

**Cross Platform:** A software that can run on several computing platforms.

**Cybercriminal:** Cybercriminals are individuals or teams of people who use technology to commit malicious activities on digital systems or networks with the intention of stealing sensitive company information or personal data and generating profit.

**Data Breach:** A data breach is a security violation, in which sensitive, protected, or confidential data is copied, transmitted, viewed, stolen, or used by an individual unauthorized to do so.

**Eavesdropping:** Secretly capturing data packets.

**Embedded Systems:** Embedded systems refer to microprocessor-based systems capable of doing computation, sensing and communication.

**Foreign Key:** A foreign key refers to dependency of a column in a table to another value in another table.

**GNU:** GNU is an extensive collection of free software, which can be used as an operating system or can be used in parts with other operating systems.

**Host Machine:** A host machine refers to the computer on which you are installing any OS/software.

**Indexing:** Indexing in Python is a way to refer the individual items within an iterable by its position.

**Interpersonal:** Relating to relationships or communication between people.

**Interoperability:** It means whether two different nodes can share information.

**Integrity:** Data Integrity refers to accuracy and consistency of data.

**Inbound Rules:** Rules for the incoming traffic to a cloud.

**Library:** A built-in piece of code.

**Mobility:** Mobility is the ability of a device to be operated while moving.

**OOP:** Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic.

**Outbound Rules:** Rules for the outgoing packets from a cloud.

**Opensource:** Opensource means the source code of a software is available publicly.

**Outlier:** A sample of data which lies outside the main trend.

**Portability:** In operating systems, portability refers to possibility of running an OS in to different platforms.

**Pattern:** A trend in data.

**Privileges:** In databases, privileges refer to user rights.

**Package:** A package is any software setup or any windows image.

**Passcode:** a combination of numbers or movements on the screen that allow you to use an electronic device.

**Query:** In databases, query refers to a command to a database to fetch or edit data in the database.

**Real-time:** A real-time OS has definitive response time for every task.

**Redundancy:** Repetition.

**Scripting Language:** A scripting language or script language is a programming language for a runtime system that automates the execution of tasks that would otherwise be performed individually by a human operator.

**Slicing:** Python slicing is a computationally fast way to methodically access parts of your data.

**Schema:** A schema is an architecture of a database in terms of its tables and table structures.

**Scatter Plot:** A plot of the whole data.

**Surveillance:** Close observation, especially of a suspected spy or criminal.

**Social Skills:** Communication skills to socialize with people.

**SCADA:** Supervisory control and data acquisition (SCADA) is a system of software and hardware elements that allows industrial organizations to: Control industrial processes locally or at remote locations.

## ABOUT THE AUTHOR



**Muhammad Umair** is a researcher at the *Sensors, Cloud and Services (SCS) Lab*, School of Computer Sciences, The University of Sydney, Australia. Muhammad Umair is also a Lecturer at the Department of Electrical, Electronics and Telecommunication Engineering, New Campus, UET Lahore. He completed his B.Sc. Electrical Engineering and M.Sc. Electrical Engineering from University of Engineering & Technology (UET) Lahore in 2014 and 2017, respectively. He has worked as a Research Officer at Internet of Things (IoT) lab at Al-Khwarizmi Institute of Computer Sciences, UET Lahore. He has also worked at Sultan Qaboos IT Research lab as a Research Officer. His survey on Social IoT platforms is the most cited survey for SIoT applications. He has designed graduate level courses on IoT.

# قومی ترانہ

پاک سر زمین شاد باد! کشورِ حسین شاد باد!  
تو نشانِ عزمِ عالی شان ارضِ پاکستان  
مرکزِ یقینِ شاد باد!

پاک سر زمین کا نظام قوتِ اخوتِ عوام  
قوم، ملک، سلطنت پائندہ تابندہ باد!  
شاد باد منزلِ مراد!

پرچمِ ستارہ و ہلال رہبرِ ترقی و کمال  
ترجمانِ ماضی، شانِ حال جانِ استقبال  
سایہ خدائے ذوالجلال!



**National Vocational & Technical Training Commission (NAVTTC)**

Plot No.38, Sector H-9/4, Kirthar Road, Islamabad.

Tel: +92-51-9207518

Website: [www.navttc.gov.pk](http://www.navttc.gov.pk)