



Diploma in
IT, Networking and Cloud

Module 3
Web Designing
Theory Manual

Table of Content

Table of Content	1
Learning Outcomes	29
Create simple Web Pages using HTML5	30
Introduction to the Internet, browsing, emailing	31
Overview of the Internet	31
Basics of Internet Architecture	31
Services on the Internet	33
Accessing Web Browser	35
Using Search Engines	44
Search Engines	69
Services Available on the Internet	72
Advantages of Internet	78
Structure and Working of E-Mail	80
Internet Applications	84
Introduction to HTML	87
What is HTML?	87
Structure of an HTML document	87
HTML Tags and Attributes	89
HTML Elements	89

Headings	93
Formatting	94
HTML Colors	99
HTML Image	101
Linking	104
Tables	107
Div and Span	110
Lists	112
Audio and Videos	119
HTML Forms and Input	121
Markup Validation Service	127
HTML5	128
Introduction to HTML5	128
Page Layout Semantic Elements	130
HTML5 Web Forms	134
SVG	137
HTML5 Media (Video & Audio)	138
Different editors used for Web Page Developing	141
HTML Editors	141
Common features of HTML Code Editors	141
Different editors used for Web Page Developing	142
Applications of HTML	168
How is its industry using HTML?	168

What is a Static Website?	168
What is a Dynamic Website?	170
Practical Application	172
Top 10 Uses of HTML	180
Able to create Styles of web pages using CSS	184
Introduction to CSS.	185
What is CSS?	185
Designing the HTML Page	185
Why is CSS required	186
CSS with HTML or XHTML	188
Introduction to CSS3	189
Important Properties of CSS3	191
Limitations of CSS	195
Limitation of CSS	196
Overcome the limitations of CSS	196
Advantages of CSS	198
Advantages of CSS	198
Web application using CSS	198
Three ways to Integrate CSS	201
Inline CSS	201
Internal CSS	202
External CSS	203
Class and ID as a selector	203

CSS Combinator	206
CSS Pseudo-classes	210
CSS Pseudo-elements	213
CSS Attribute Selectors	215
Merits and Demerits of all kinds of CSS	217
Inline CSS	217
Internal CSS	217
External CSS	219
Introduction to Bootstrap	219
Able To Create Own Account in Cloud and Hosting	227
Introduction to Cloud Computing	228
Cloud Overview	228
Essential Characteristics	229
Six Advantages of Cloud Computing	229
Cloud Computing Service Delivery Models	230
Cloud Computing Deployment Models	232
Cloud Computing Uses	233
Important Cloud Terminology	233
Amazon Web Services (AWS)	234
Access to AWS Services	234
AWS Global Infrastructure	236
Data Centers	236
Availability Zones	237

AWS Regions	238
Edge Locations	238
AWS Infrastructure Features	239
AWS Core Services	239
Compute	239
Storage	243
Networking	245
Database	249
Security	251
Industrial Faculty-Webinar, How Cloud and Webinar Works	256
Industrial Faculty-Webinar	256
How Cloud and Webinar Works	257
Significance of Cloud in Webinars	258
Able to configure embedded database with different web pages using mongodb	260
Purpose of database systems	262
What is Database?	262
What is DBMS?	263
What are the types of DBMS?	265
Centralised Database	266
Distributed Database	267
Personal Database	267
End User Database	268
Commercial Database	268

NoSQL Database	268
Operational Database	268
Relational Databases	269
Cloud Databases	270
Object-Oriented Databases	271
Graph Databases	272
DBMS vs. File System	274
Data Abstraction	275
Database Users	276
Naive Users	277
Application programmers	277
Sophisticated users	278
DBA [Database Administrators]	278
Data Independence (Logical & Physical)	279
Types of Data Independence	279
Levels of Database	279
Physical Data Independence	280
Logical Data Independence	281
Difference between Physical and Logical Data Independence	281
Importance of Data Independence	282
Instance & Schemes	283
Differences Between Schema and Instance	284
Sub-Schema	284

Three Layered of Architecture in DBMS	286
Types of DBMS Architecture	286
1-Tier Architecture	287
2-Tier Architecture	287
3-Tier Architecture	288
Different Levels of Abstraction	290
Physical level or Internal Level	291
Logical Level or Conceptual Level	291
View Level or External Level	292
Data Modelling	292
Relational Data Model	293
Entity-Relationship Data Model	294
Object-based Data Model	294
Semistructured Data Model	294
Hierarchical Model	294
Network Model	295
Flat Data Model	296
Record base Data Model	296
ER modeling	296
Component of ER Diagram	298
Logical Data Model	299
Entity Sets	300
Definition of Strong Entity	301

Definition of Weak Entity	303
Difference between Strong and Weak Entity	304
Relationship Sets	306
One-to-One Relationship	306
One-to-many relationship	307
Many-to-one relationship	307
Many-to-many relationship	308
Concept of Attributes	309
Introduction	309
Symbols used to represent attributes	310
Types of Attributes	310
Simple Attributes	311
Single Valued Attributes	312
Key Attributes	312
Derived Attributes	312
Multivalued Attributes	312
Example 1 – Student Book Relation	313
Example 2 – Product Order System	314
Example 3 –Employee Management	315
Concept of Relationship	316
Introduction	316
Types of Keys	317
Keys	317

Primary Key	317
Foreign Key	318
Candidate Key	319
Super Key	320
Compound Key	320
Alternate Key	321
Unique Key	321
Mapping Constraints	323
Introduction	323
Types of Mapping Constraints	323
One-to-One	324
One-to-many	324
Many-to-Many	325
Example of Employee Management System	326
Example of Movie Ticket Management	327
Entity Relationship Diagram	328
Introduction	328
Components of ER Diagram	328
Entity	329
Attribute	330
3. Relationship	333
Relational Model	336
Introduction	337

What is Relational Model?	337
IMPORTANT TERMINOLOGIES	337
Constraints in Relational Model	338
Insertion Anomaly in Referencing Relation:	340
Deletion/ Updation Anomaly in Referenced Relation:	340
Properties of Relation	341
Network Model	342
Introduction	343
The benefits of the network model include:	343
The drawbacks of the network model include:	343
Hierarchical Model	345
Introduction	345
Advantages	345
Disadvantages	346
Relational Database Management System	347
Introduction	347
Characteristics of physical database design	348
Transform the logical data model for target DBMS	348
Database Engine	349
Database Schema	349
Advantages of relational database management system	350
Disadvantages of RDBMS	350
Applications of RDBMS	351

Structure of DBMS	352
DBMS VS File System	354
DBMS VS RDBMS	355
1-Tier Architecture	355
2-Tier Architecture	356
3-Tier Architecture	356
Three Schema Architecture	358
Relational Algebra	360
Select Operation	360
Project Operation	361
Union Operation	363
Set Intersection	365
Set Difference	365
Cartesian Product	366
Rename Operation	367
Join Operation	367
Inner Join	368
Theta Join	368
EQUI Join	368
Left Outer Join	368
Right Outer Join	368
Division Operator	369
Relational Calculus	371

Introduction	371
Types of Relational Calculus	371
Tuple Relational Calculus	371
Examples	372
Domain Relational Calculus	374
Examples	374
RDBMS Technologies	376
Oracle Database Technology Features	376
MySQL Database Technology Features	376
MongoDB Database Technology Features	376
Microsoft SQL Server Database Technology Features	377
Relational Data Structure/Relational Model	378
What is a Relational Model?	378
Relational Model Concepts	378
Relational Integrity constraints	380
Types of integrity constraints	380
Domain constraints	381
Entity integrity constraints	381
Referential Integrity Constraints	382
Key constraints	383
Operations in the Relational Model	384
Insert Operation:	385
Update Operation	385

Delete Operation	385
Select Operation	386
Advantages of using the Relational model	386
Disadvantages of using Relational model	387
Key and Relational Data Manipulation	389
Keys	389
What are Keys?	389
Why do we need a Key?	389
Various Keys	390
Super key:	390
Primary Key	391
Alternate key	393
Candidate Key	394
Foreign key	395
Compound key	397
Composite key	398
Surrogate Key	398
Let's see the difference between keys	400
Primary Key and Foreign Key:	400
Primary and Candidate Key:	402
Super Key and Candidate Key	403
Primary key and Unique key	404
Data Manipulation Language (DML)	406

Introduction to DML	406
1. SELECT COMMAND	406
2. INSERT COMMAND	407
3. UPDATE COMMAND	408
4. DELETE COMMAND	409
Relational Algebra	410
Relational Algebra	410
Relational Algebraic Operations	411
Unary Relational Operations	411
SELECT (σ)	411
Projection(π)	412
Rename(ρ)	413
Union operation (\cup)	414
Intersection	415
Set Difference (-)	416
Cartesian product(\times)	417
Binary Relational Operations	418
JOIN	418
Inner Join:	419
Theta Join:	419
EQUI join:	420
NATURAL JOIN (\bowtie)	420
OUTER JOIN	421

Left Outer Join(A B)	422
Right Outer Join: (A B)	423
Full Outer Join: (A B)	424
Set Operations	426
Types of Set Operation	426
1. Union	426
2. Union All	429
3. Intersect	430
4. Minus	432
Fundamental Operations	434
SELECT (σ)	434
Projection(π)	434
Cartesian product(X)	436
Union operation (\cup)	437
Set Difference (-)	439
Relational Calculus	440
What is Relational Calculus?	440
Tuple Relational Calculus (TRC)	441
Domain Relational Calculus(DRC)	448
Difference between Relational Algebra and Relational Calculus:	453
Data Definition language	455
Introduction to DDL	455
1. CREATE COMMAND	455

2. DROP COMMAND	456
3. ALTER COMMAND	457
4. RENAME COMMAND	459
5. TRUNCATE COMMAND	459
Operators: Select, Project, Join, Rename etc	461
SELECT (σ)	461
Projection(π)	461
JOIN	463
Inner Join:	463
Theta Join:	463
EQUI join:	464
NATURAL JOIN (\bowtie)	464
OUTER JOIN	465
Left Outer Join(A \bowtie B)	465
Right Outer Join: (A \bowtie B)	467
Full Outer Join: (A \bowtie B)	468
Rename(ρ)	469
Introduction to MongoDB	471
Brief History of MongoDB	471
What is MongoDB?	471
MongoDB Features	471
Why Use MongoDB?	472
Difference between MongoDB & RDBMS	474

Where to Use MongoDB?	475
Advantages of MongoDB over RDBMS	477
Able to Design and Develop Dynamic Websites with PHP	478
Decisions and loops using HTML	479
Doing Repetitive task with looping	489
Difference Between while and do...while Loop:	494
Mixing Decisions and looping with Html:	496
PHP Conditions for HTML Code:	496
Other valid embedded syntax :	497
Embedded code in action :	498
Functions	500
What is function?	501
Advantages of functions:	501
Functions in PHP are divided into two types:	501
Predefined function:	502
User defined functions:	502
Define a Function:	502
Syntax:	502
Example:	502
Functions with Parameters:	503
Call by value:	504
Call by Reference:	504
Types of User defined functions:	506

Built in functions:	508
Strings	510
What is String?	511
Creating and Accessing Strings:	512
Searching & Replacing String	514
Formatting String :	516
String Related Library Function :	521
Arrays-Anatomy of Arras	524
What is an Array?	524
Advantages of arrays:	524
Declaration of Array:	524
Types of Arrays:	525
Creating an Index based Array:	526
Creating an Associative Array:	527
Creating Multi-Dimensional Array:	528
Looping with Index based array:	528
Looping with associated array:	529
Looping with associated array using foreach() :	530
Library functions for Arrays :	531
Working with file and Directories	543
Understanding File & Directory	543
Concept of file and directory	543
Reading the Contents of a Directory	543

PHP opendir() function	544
PHP readdir() function	545
PHP closedir() function	547
File Opening Modes	548
Opening, reading and closing a file	549
Opening a File	549
Reading a File	550
Closing a File	551
Copying, renaming and deleting a file	553
PHP copying a file	553
PHP renaming a file	554
PHP deleting a file	555
Working with directories	558
Reading the Contents of a Directory	558
Deleting the Directory and Its Contents	558
Creating New Directories	561
Building a text editor File: Uploading & Downloading	563
Building a text editor File	563
Upload a file to server	563
Download files in PHP	564
Using query string (URL rewriting)	567
Accessing a query string element in a PHP page	568
Using Hidden field	568

Using cookies	569
Using session	571
String matching with regular expression	575
What is a regular expression ?	575
Basics of regular expression	575
Pattern matching in PHP	577
Replacing text	577
Splitting a string with a Regular Expression	578
Splitting with Preg_Split()	578
Splitting without Losing	578
Splitting with an Invisible Delimiter	579
Generating Images with PHP	580
Embedding an Image in a Page	580
The GD Extension	580
Basics of computer Graphics	581
Creating Image	581
Creating An Image with the PHP GD Library	582
Manipulating Image	583
How to Draw Lines on an Image	584
How to Draw Shapes On An Image	585
How to Add Image Filters & Effects	586
Using text in Image	586
Database Connectivity with MySql	588

Introduction to RDBMS	588
What is a Database?	588
What is RDBMS?	588
RDBMS Terminology	588
What is a Table?	588
What is a Column?	588
What is a Row?	589
What is Data Redundancy?	589
What is Primary Key?	589
What is Foreign Key?	589
What is Compound Key?	590
What is a Referential Integrity?	590
What is an Index?	590
MySQL Database	591
Connection with MySql Database	591
Create a MySQL Database	592
Create MySQL Tables	593
Insert Data Into MySQL	593
Select Data From MySQL	594
Delete Data From MySQL	594
Update Data in MySQL	595
Able to make websites, web servers, game frameworks, desktop and CLI applications and IDE using Python	596

Introduction to Python	597
Overview of Python	597
Characteristics of Python	597
Platform Independent	598
Interpreted	598
Simple Syntax	598
High Level Language	598
Free and Open Source	598
Rich Library Support	599
Advantages over other languages	599
It is free	599
It Needs Less Coding	600
All kinds of Businesses can afford it	600
It is one of the most trending languages	600
History of Python	600
Python Timeline/History and IEEE rankings	601
Features of Python	603
Easy to Code and Understand	603
Expressive Language	603
Free and Open Source	604
Interpreted Language	604
Object-Oriented Language	604
Dynamically Typed Programming Language	605

Cross-Platform Language	605
Large Standard Library	606
Extensible Language	606
Setting Up Path	606
Python Installation for Windows	607
How to set Python Path in Windows	608
Python Installation for Linux	609
How to set Python Path in Linux	610
Basic Syntax Variable	610
Interactive Mode Programming	611
Script Mode Programming	612
Python Identifiers	612
Reserved Keywords	613
Lines and Indentation	613
Multi Line Statement	614
Quotation in Python	616
Comments in Python	616
Using Blank Lines	617
Input From the User	617
Multiple Statement on a Single Line	619
Multiple Statements Groups as Suites	619
Data Types Operator	621
Types of operators	621

Python Arithmetic Operators	622
Python Comparison Operator	623
Python Assignment Operator	623
Python Bitwise Operator	625
Python Logical Operator	626
Python Membership Operator	627
Python Identity Operator	627
Python Operator Precedence	628
Conditional Statements	632
Statement and Description	632
If statement	632
If else statement	633
Elif statement	634
Nested if statement	636
Single Statement Suites	638
Looping	639
While Loop in Python	639
For Loop in Python	640
Nested Loop in Python	642
Control Statements	644
Break Statement	645
Example of Break Statement in Python-	646
Continue Statement	646

Pass Statement	648
String Manipulation	650
Introduction	650
Create Strings	651
Index and Slice Strings	652
Index a String in Python	652
Slice a String in Python	653
Modify/Delete	653
String Operators	654
String Formatting Operators	657
Common Python String Methods	663
Regular Expressions / Misc String functions	668
List	673
Create a list in Python	675
How to access elements from a list?	679
Slice lists in Python	681
Change or add elements to a list	682
Delete or Remove elements from a list	684
Python List Methods	686
List Comprehension: Elegant way to create new List	688
Other List Operations in Python	689
Tuples	689
Creating a Tuple	691

Advantages of Tuple over List	692
Access Tuple Elements	693
Performing Operations - Modifying, Deleting Python Tuple	695
Tuple Methods	696
Other Tuple Operations	697
Functions and Methods	698
What Is a Function in Python?	698
Call a Function	700
Function Examples	716
Functions as Objects	718
Function Attributes	719
Dictionary	721
Create a dictionary	721
Access Items in a Dictionary	722
Change or Add elements in a dictionary	722
Delete or remove elements from a dictionary	723
Python Dictionary Methods	724
Dictionary Comprehension	725
Other Dictionary Operations	726
Built-in Functions with Dictionary	727
Functions	729
Introduction / Overview	729
What is lambda in Python?	729

Modules	735
What are modules in Python?	737
Python Module: Mechanism	738
import modules in Python?	738
Package	745
What are packages?	745
Importing module from a package	746
Input /Output	747
Python Input, Output and Import	748
Output formatting	748
Python Input	749
What is a file?	751
How to open a file?	751
How to close a file?	756
Perform Write operation	757
Perform Read operation	758
Renaming and deleting files in Python	760
Python File object methods	762
Exception Handling	773
Exception Handling: Error vs. Exception	773
What is Error?	773
What is Exception?	773
Handle Exceptions with Try-Except	774

Handling All Types of Exceptions with Except	775
Handling Multiple Exceptions with Except	776
Handle Exceptions with Try-Finally	777
Raise Exception with Arguments	778
Create Custom Exceptions	779
References	782

Learning Outcomes

After completing this module a student will be able to:

1. create simple web pages using HTML5.
2. create styles of web pages using CSS.
3. create your own account in cloud and hosting.
4. configure embedded databases with different web pages using MongoDB.
5. design and develop dynamic websites using PHP.
6. make websites, web servers, game frameworks, desktop and CLI applications, and IDE using Python.

Create simple Web Pages using HTML5

In this section, we will read about:

- Introduction to the Internet, browsing, emailing
- Introduction to HTML
- Different editors used for Webpage Development
- Application of HTML

Introduction to the Internet, browsing, emailing

Overview of the Internet

The Internet is defined as an electronic communications network that connects computer networks and organizational computer facilities around the world. - Merriam-webster

We can say that the internet is a global network of interconnected computer networks and other electronic devices worldwide, which uses standard Internet Protocol (TCP/IP). It is possible to access almost any information, communicate with anyone else in the world using the internet. Networking functionality of internet can be described as below:

- Every computer on the internet is identified by a unique IP address.
- IP Address is a unique set of numbers (such as 110.22.33.114) which identifies a computer location.
- A special computer DNS (Domain Name Server) is used to give a name to the IP Address so that users can locate a computer by name.

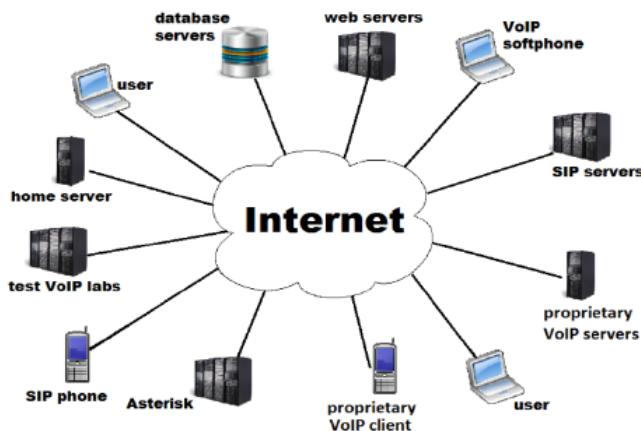


Image 1: Internet - users and services are pushed to the edge of the network.
Reference: <https://www.researchgate.net>

Basics of Internet Architecture

The concept of the Internet was originated in 1969 and has undergone several technological & Infrastructural changes as discussed below:

- 1969: The origin of the Internet devised from the concept of the Advanced Research Project Agency Network (ARPANET).
- ARPANET was developed by the United States Department of Defense.
- The basic purpose of ARPANET was to provide communication among the various bodies of government.
- Initially, there were only four nodes, formally called Hosts.
- In 1972, the ARPANET spread over the globe with 23 nodes located in different countries and thus became known as the Internet.

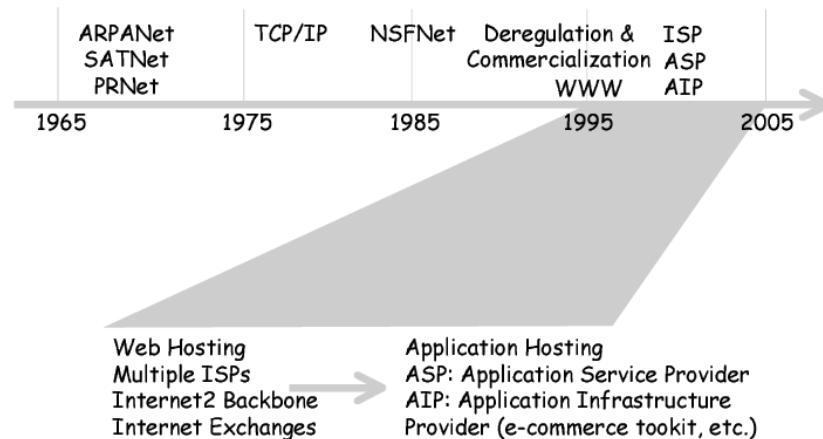


Image 2: Evolution of the Internet
Reference: <https://www.researchgate.net>

Internet architecture consists of three layers

IP

In order to communicate, we need our data to be encapsulated as Internet Protocol (IP) packets. These IP packets travel across a number of hosts in a network through routing to reach the destination. However, IP does not support error detection and error recovery and is incapable of detecting the loss of packets.

TCP

TCP stands for "Transmission Control Protocol". It provides end to end transmission of data, i.e., from source to destination. It is a very complex protocol as it supports the recovery of lost packets.

Application Protocol

The third layer in internet architecture is the application layer which has different protocols on which the internet services are built. Some of the examples of internet services include email (SMTP facilitates email feature), file transfer (FTP facilitates file transfer feature), etc.

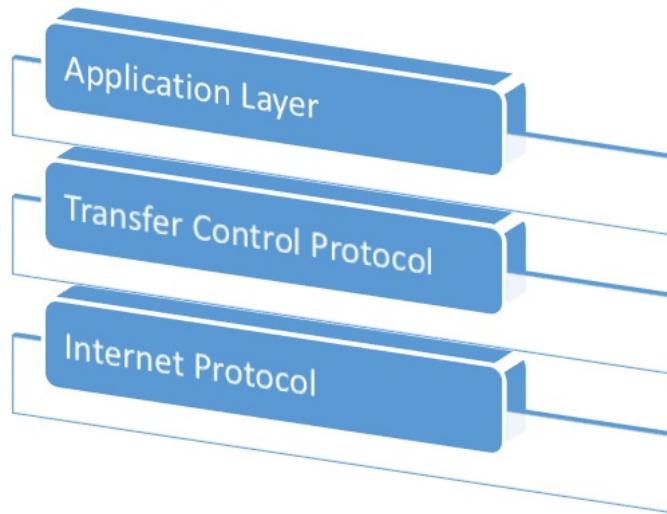


Image 3: Internet architecture
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_internet.htm

Services on the Internet

Internet Services allows us to access a huge amount of information such as text, graphics, sound and software over the internet.

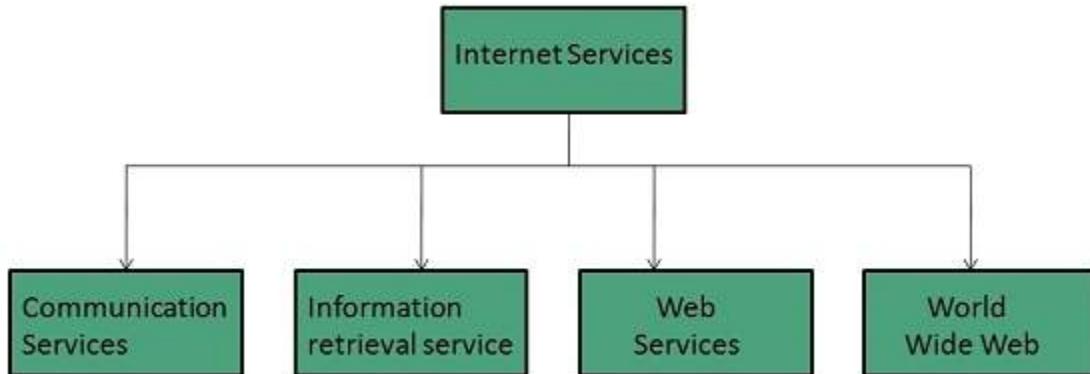


Image 4: categories of Internet Services
Reference:https://www.tutorialspoint.com/internet_technologies/internet_services.htm

Communication Services

There are various Communication Services available that offer exchange of information with individuals or groups

- Electronic Mail -Used to send electronic messages over the internet.
- Telnet-Used to log on to a remote computer that is attached to the internet
- Newsgroup-Offers a forum for people to discuss topics of common interests
- Internet Relay Chat (IRC)-Allows people from all over the world to communicate in real-time.
- Mailing Lists-Used to organize a group of internet users to share common information through e-mail.
- Internet Telephony (VoIP)-Allows internet users to talk across the internet to any PC equipped to receive the call.
- Instant Messaging-Offers real-time chat between individuals and groups of people. Eg. Yahoo messenger, MSN messenger.

Information Retrieval Services

There exist several Information retrieval services offering easy access to information present on the internet.

-
- File Transfer Protocol (FTP)-Enable the users to transfer files.
 - Archie-It's updated database of public FTP sites and their content. It helps to search a file by its name.
 - Gopher-Used to search, retrieve, and display documents on remote sites.
 - Very Easy Rodent Oriented Netwide Index to Computer Achieved (VERONICA)-VERONICA is a gopher based resource. It allows access to the information resource stored on gopher's servers.

Web Services

Web services allow the exchange of information between applications on the web. Using web services, applications can easily interact with each other.

The web services are offered using the concept of Utility Computing.

World Wide Web (WWW)

WWW is also known as W3. It offers a way to access documents spread over several servers over the internet. These documents may contain texts, graphics, audio, video, hyperlinks. The hyperlinks allow the users to navigate between the documents.

Accessing Web Browser

A web browser is a software application that enables a user to display and interact with text, images, videos, music, and other information that could be on a website.

There are several ways to access a web page like using URLs, hyperlinks, navigating tools, search engines, etc.

Using URLs

URL refers to "Uniform Resource Locator". Each and every website can be recognized using a unique address called "Uniform Resource Locator" or simply a URL. Once you provide the URL of a specific page in the address bar, the web browser will find the corresponding page and display the result to the user.

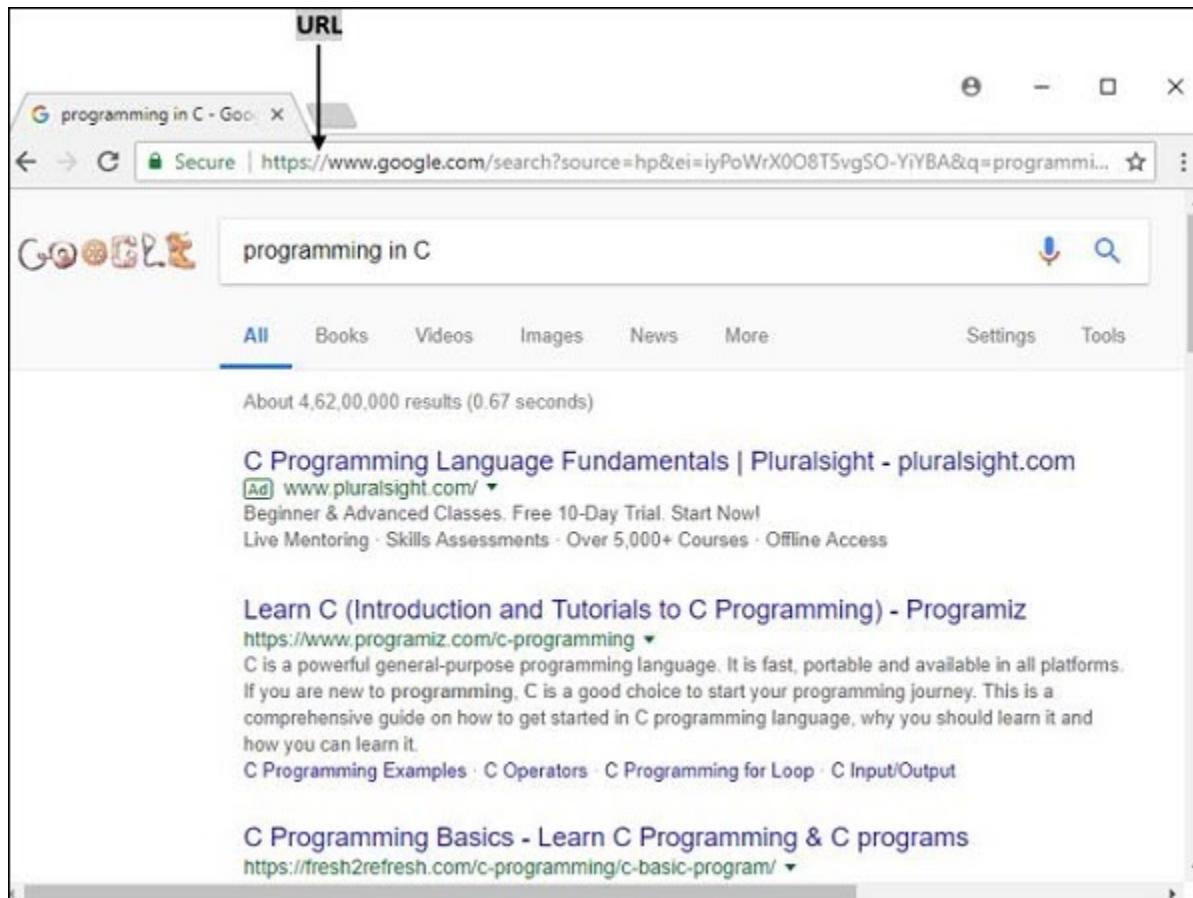


Image 5: URL

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Using Hyperlinks

"Hyperlink" is a part of a web page that is linked to a URL. A hyperlink can be text, image, button, arrow, etc. By clicking on a hyperlink you can move to a different URL specified in the link from the current URL. Hyperlinked text is an underlined blue color text which is represented using a hand symbol.

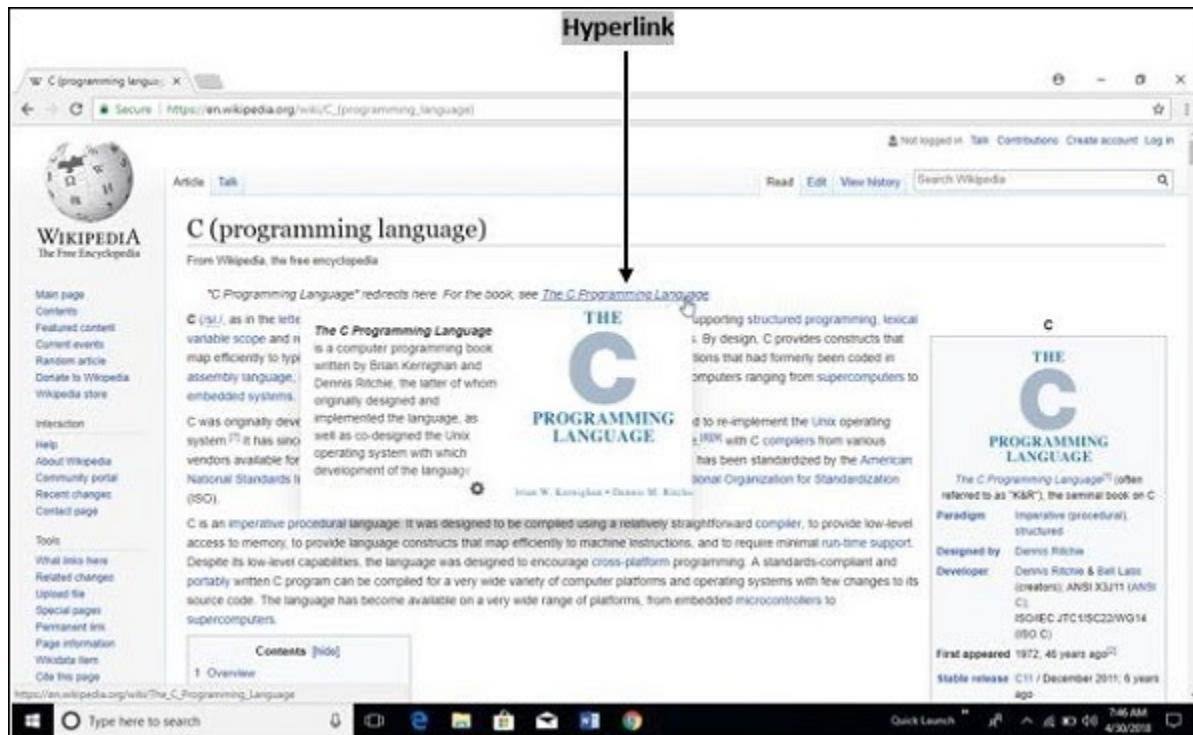


Image 6: Hyperlink

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Using Browsers Navigation Tools

Web browsers offer a variety of tools to help you move around the web. These tools help you to quickly go back and forth through web pages.

Back Button – Helps to move back to the previous page from the current page.

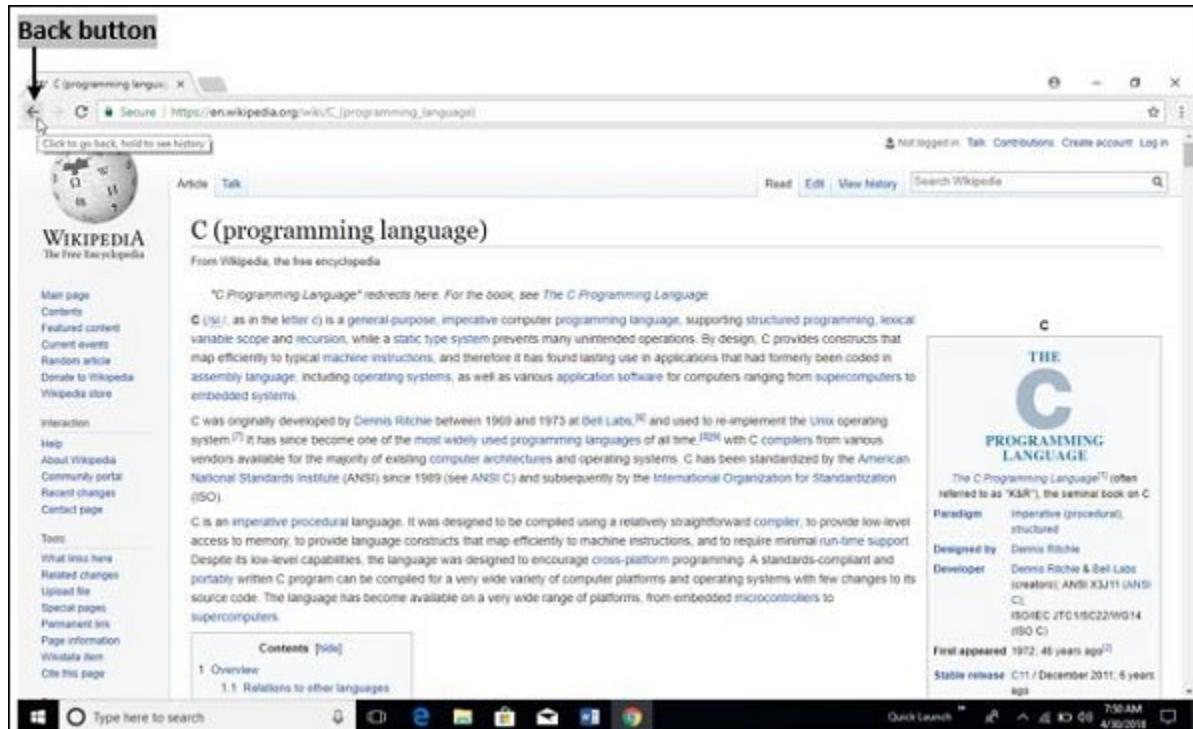


Image 7:Back button

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Forward Button – Helps to move to the next page from the current page.

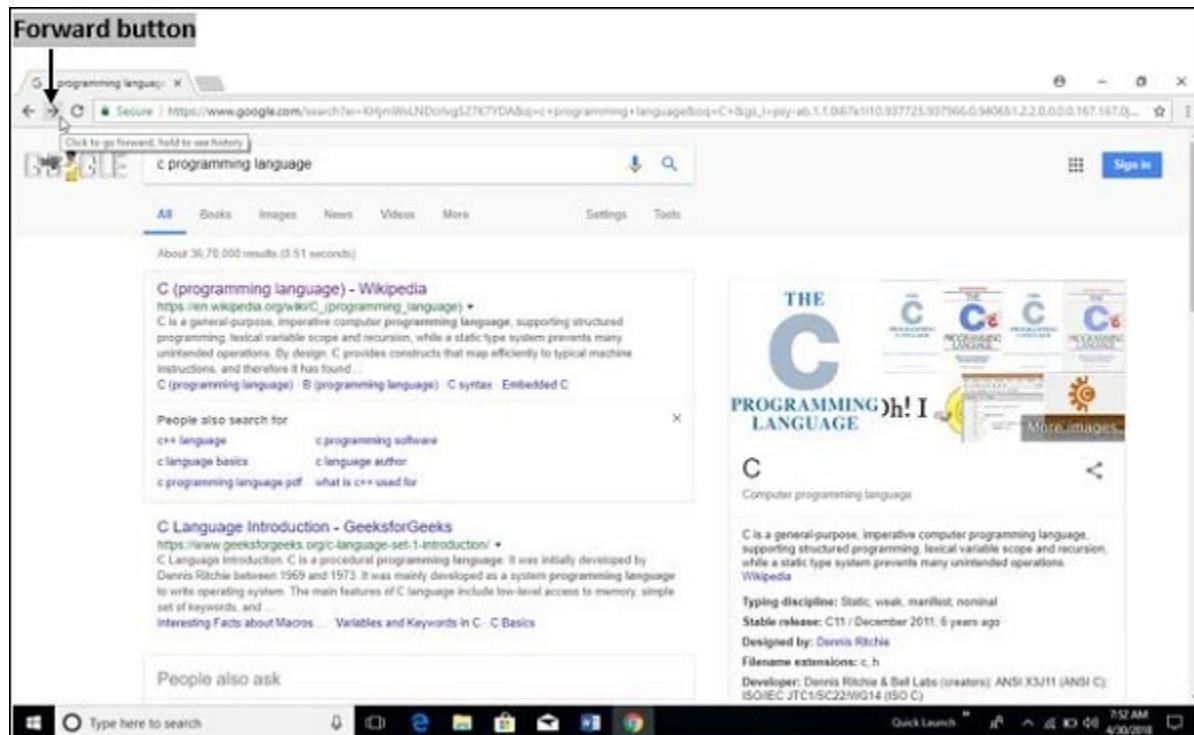


Image 8:Forward button

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Refresh Button – Helps to refresh a current page.

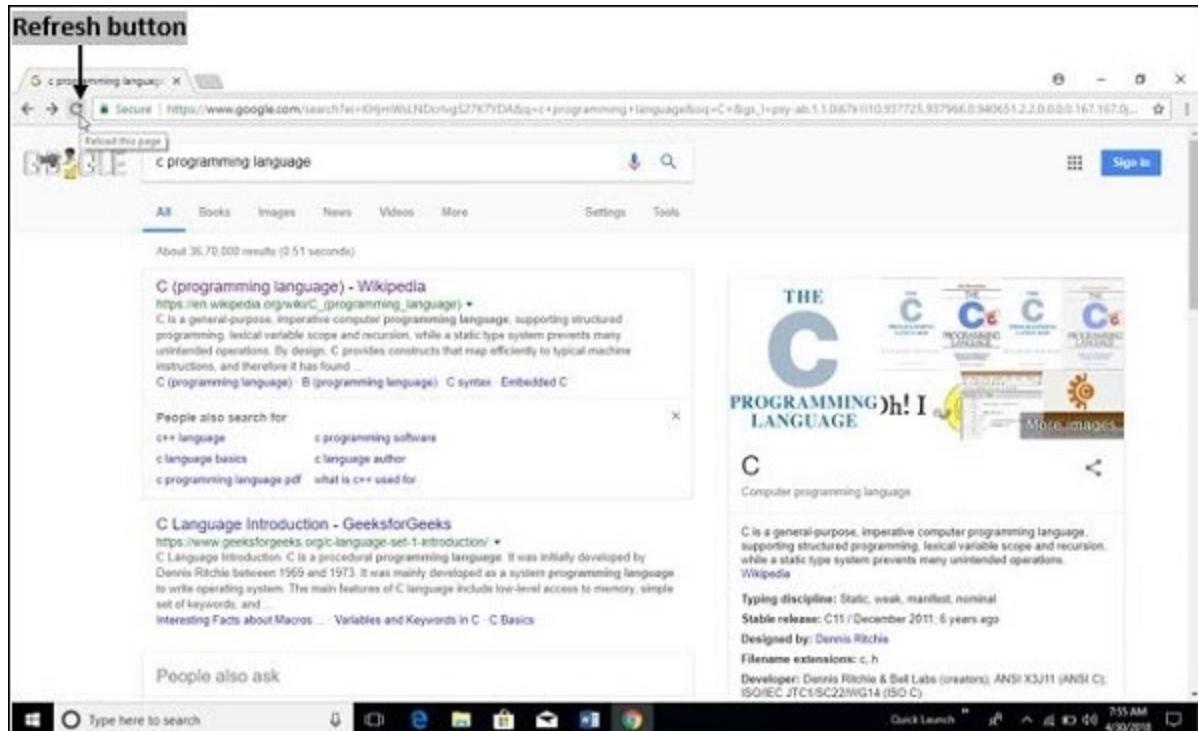


Image 9: Refresh

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Close Button – Helps to close a web page.

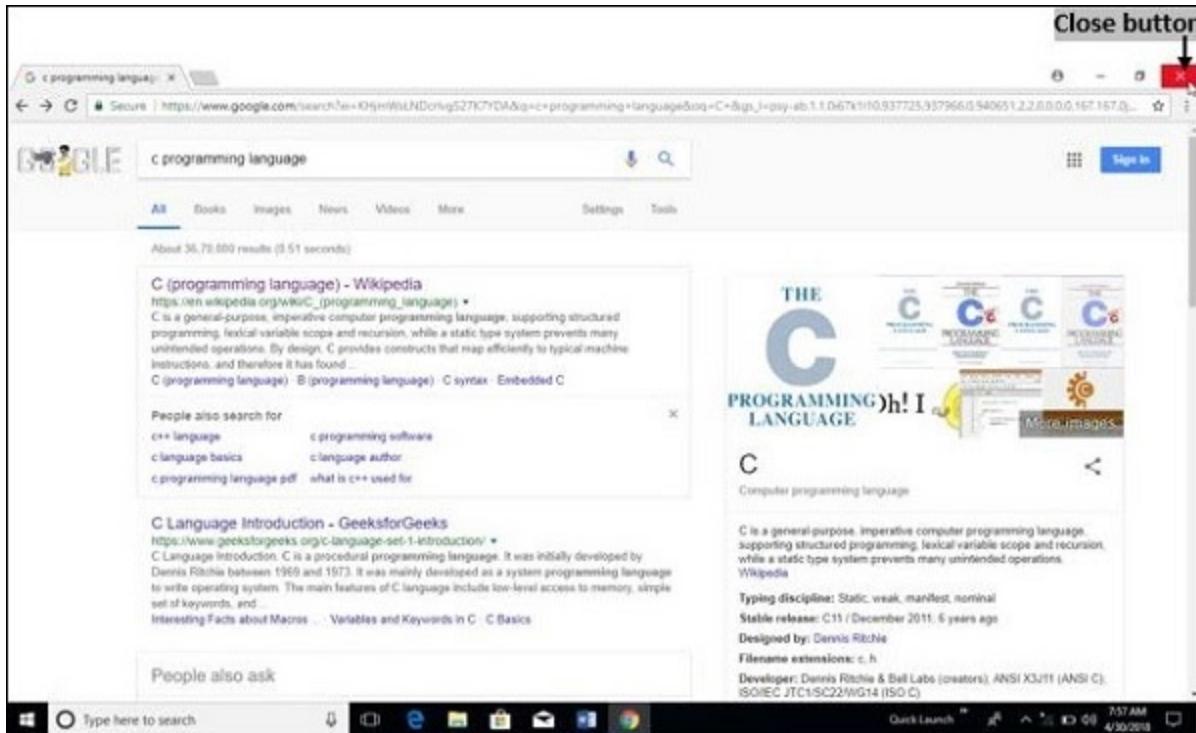


Image10: Close Button
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Using Bookmark

Web browsers allow you to bookmark pages that you visit most frequently. This helps you to go to a web page directly by selecting from a list of bookmarks instead of typing the URL multiple times. This is displayed as an icon with a star symbol in the top right corner of the page.

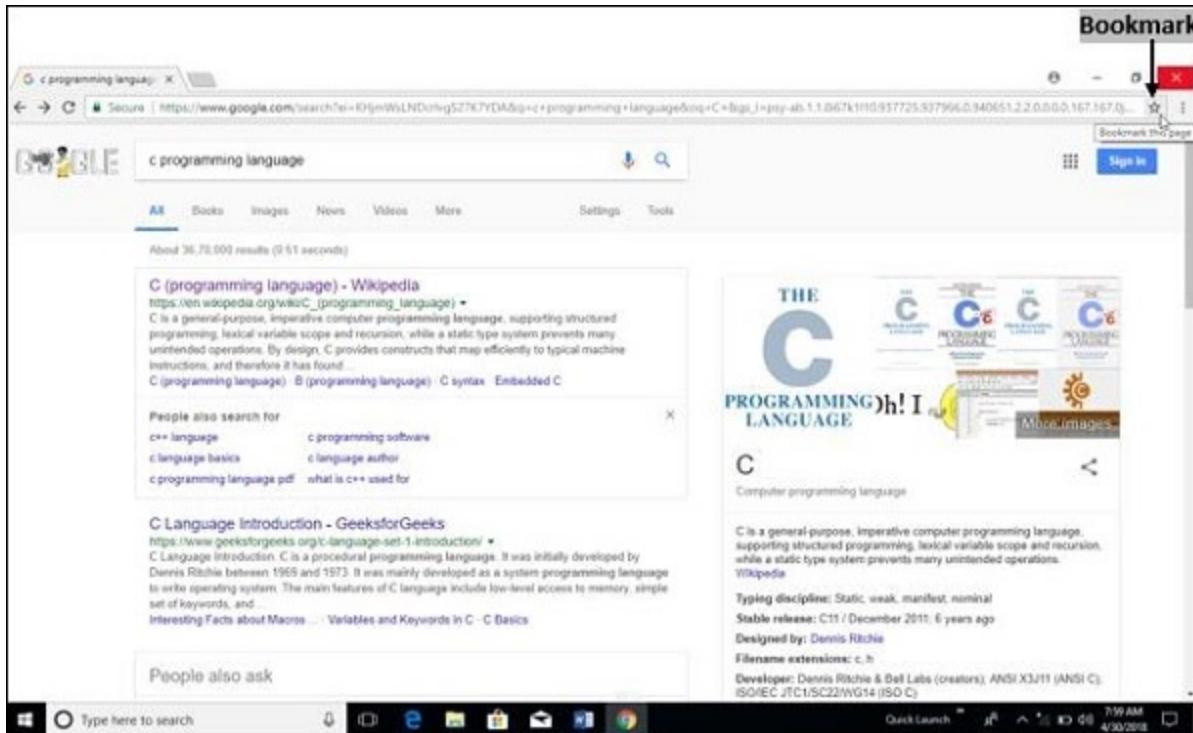


Image 11: Using a bookmark
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Using History

When you type any URL in the address bar, the browser saves that URL automatically, thus creating a history list for the current session. You can choose the URL you want from the history list instead of typing it again.

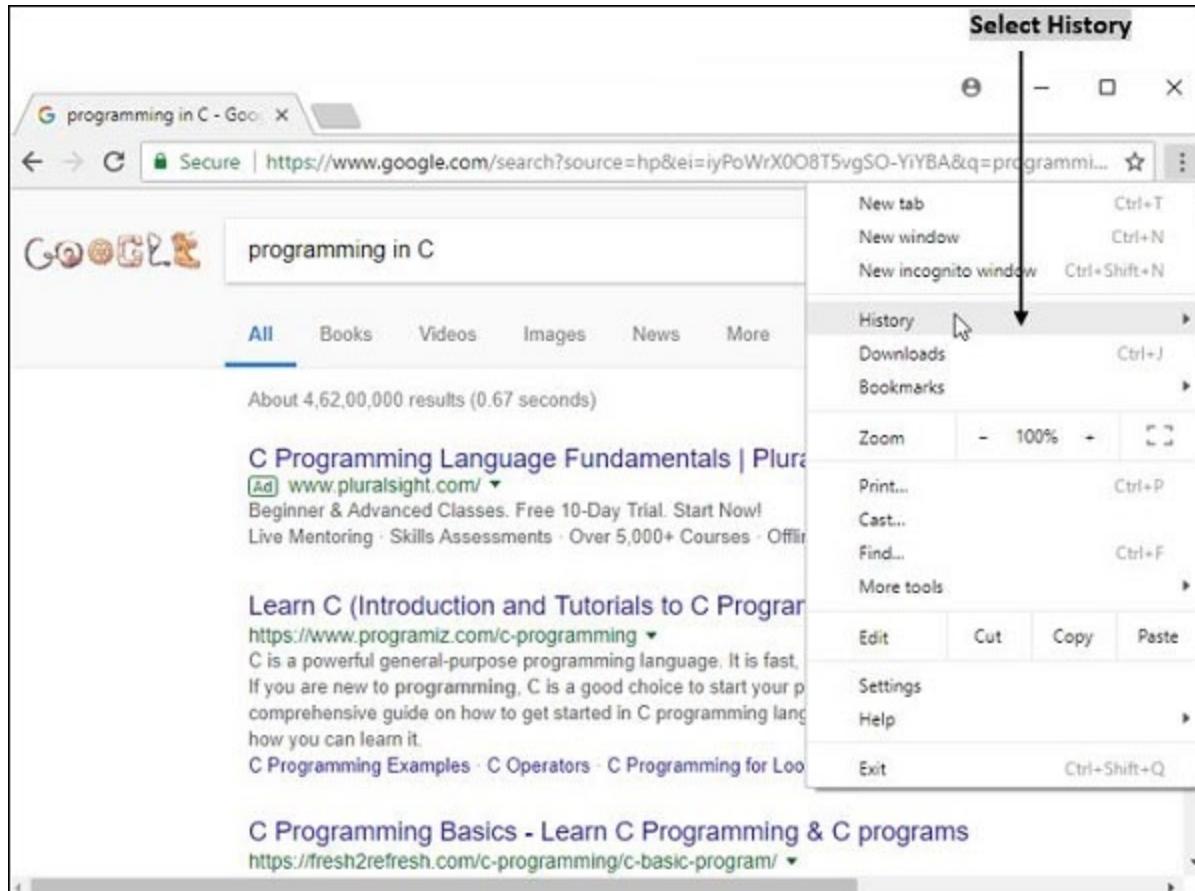


Image 12: Using History

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

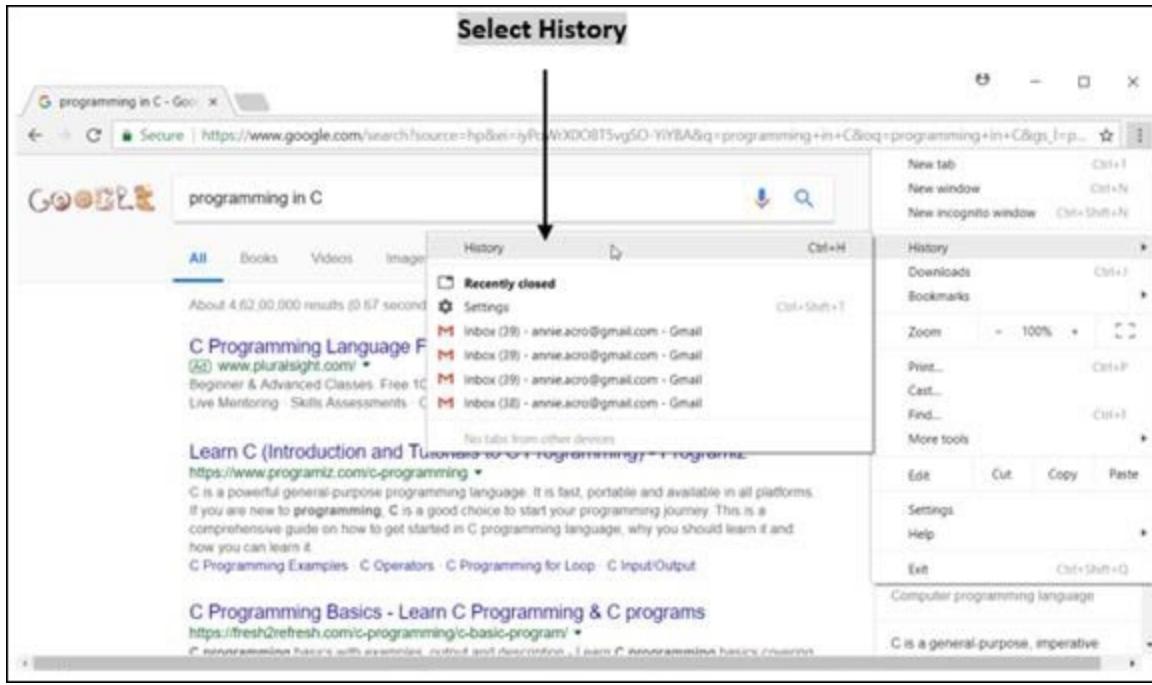


Image 13: Select History

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Using Search Engines

A search engine is an application that allows you to search for content on the web. It displays multiple web pages based on the content or a word you have typed. The most popular search engines include Google, Yahoo, Ask, etc. Below are the steps to use a search engine.

Step 1 – Launch your web browser.

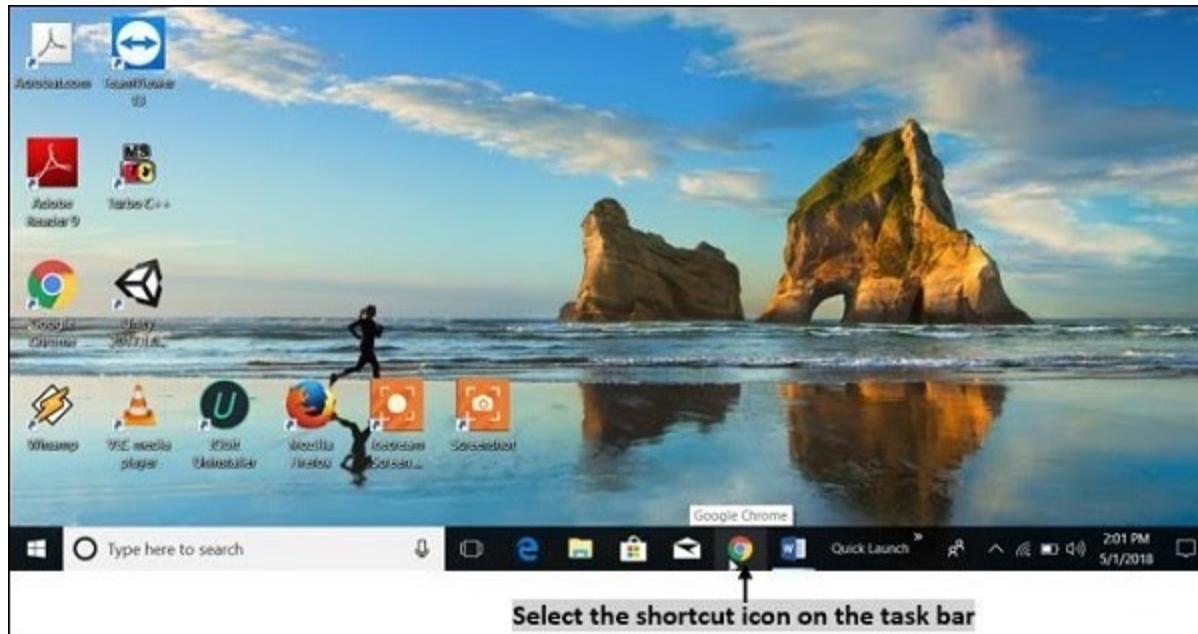


Image 14: Launch Web browser

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 2 – In "Address bar/Location", type the search engine you want to use and press enter.

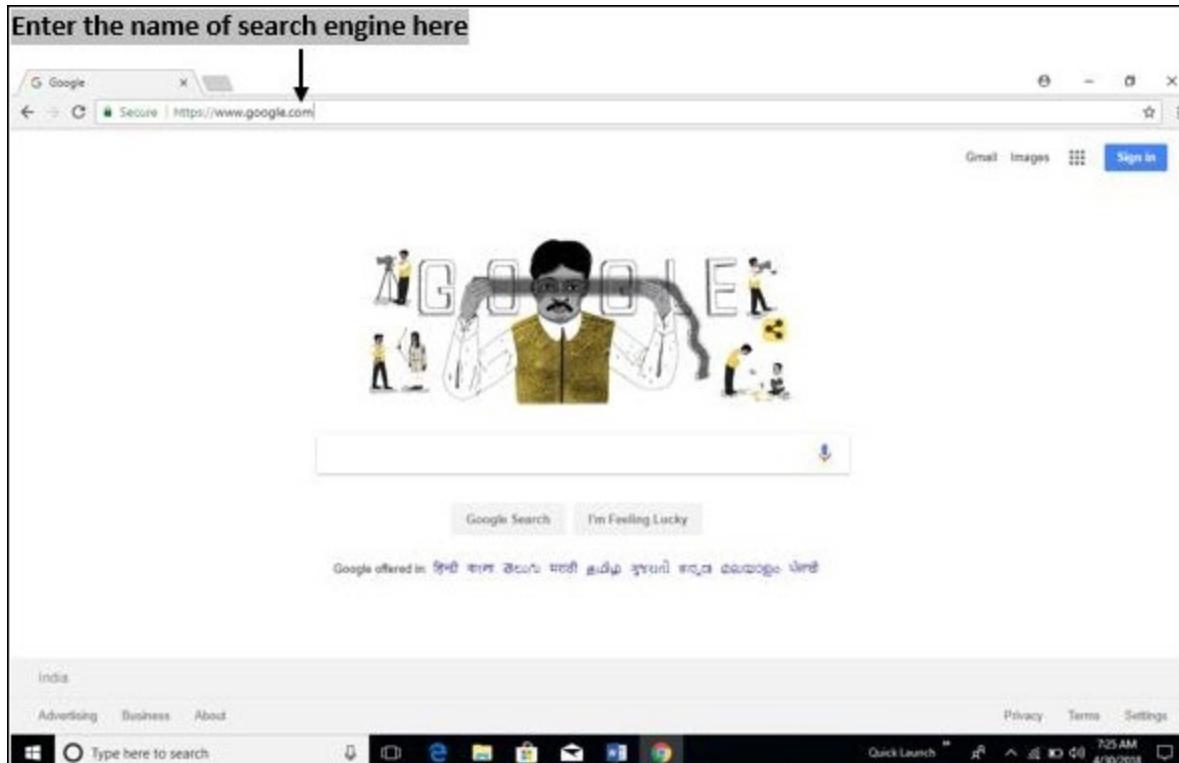


Image 15: Enter the name of search engine
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 3 – Type the content you want to search in the "search text box" and press enter.

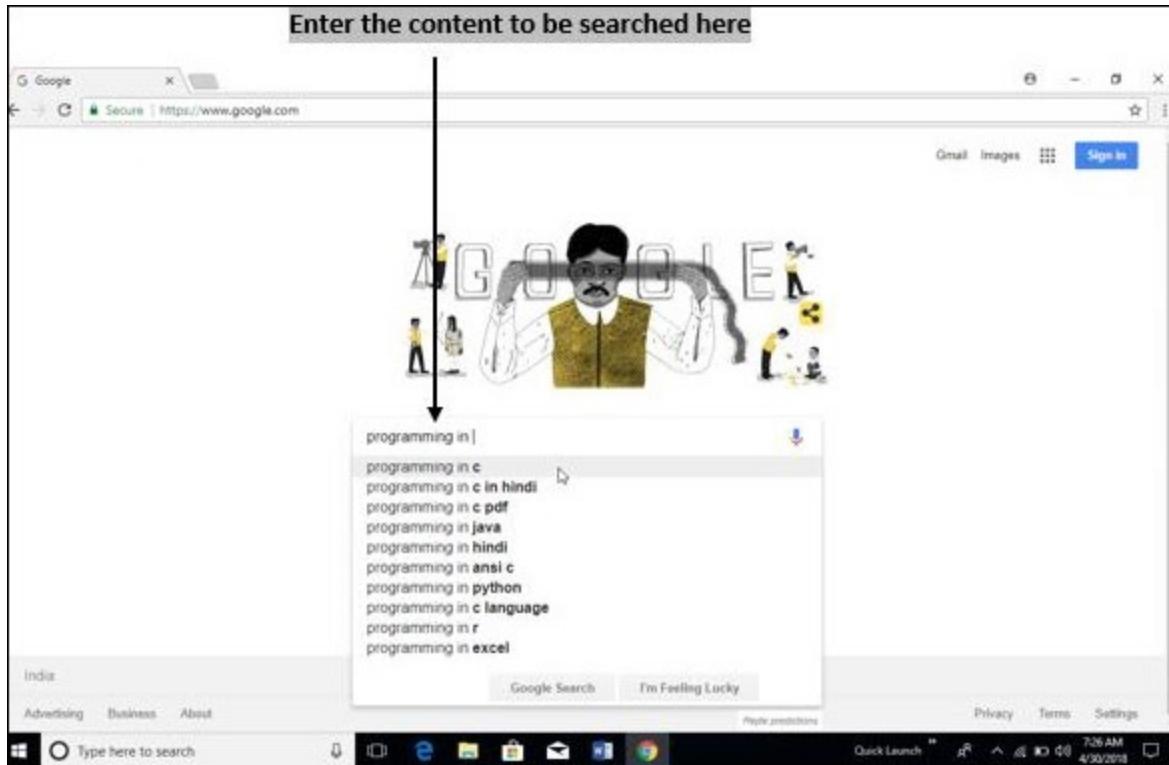


Image 16: Enter the content to be search
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 4 – It displays a list of web pages from which you can select the content/web page you want.

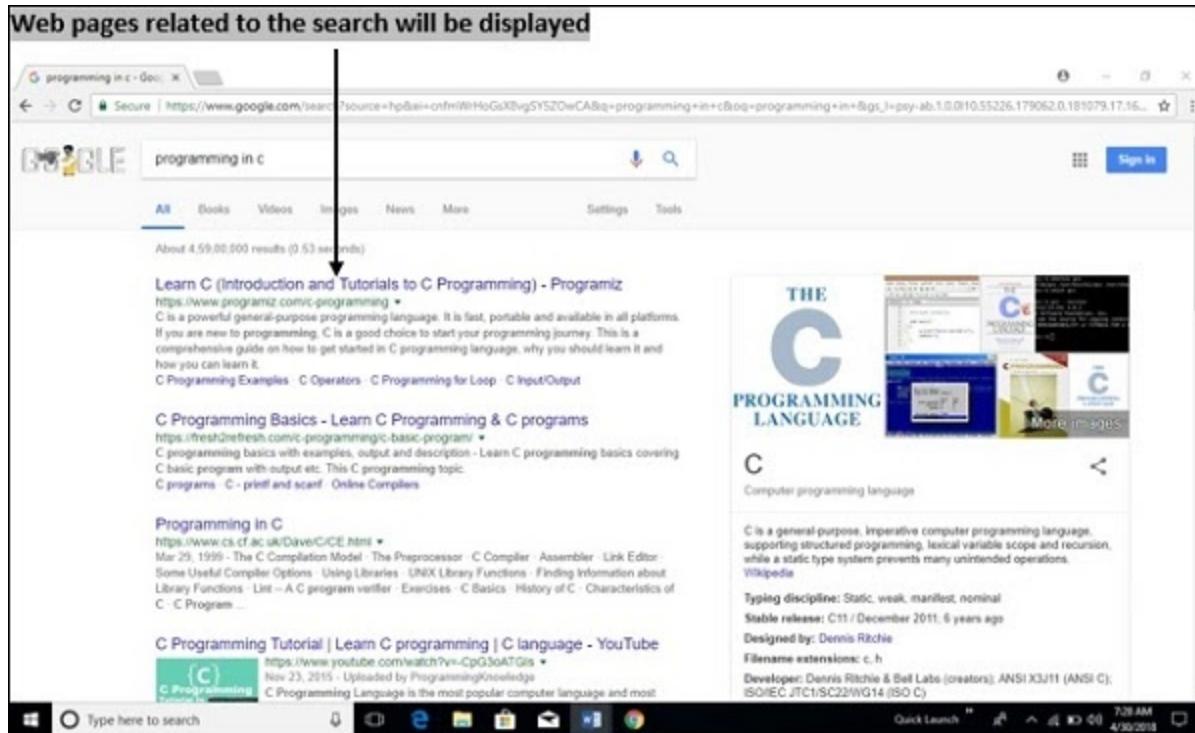


Image 17: Webpages related to the search will be displayed
 Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Using Favorites Folder

Web browsers allow you to bookmark the pages that you visit most frequently. The Favorites folder is called Bookmarks. This helps you to go to the web page directly by selecting from the list of bookmarks instead of typing the URL again and again. Adding and removing the pages to the favorite folder include the following steps.

Step 1 – Click the star icon present at the top right corner of the page.

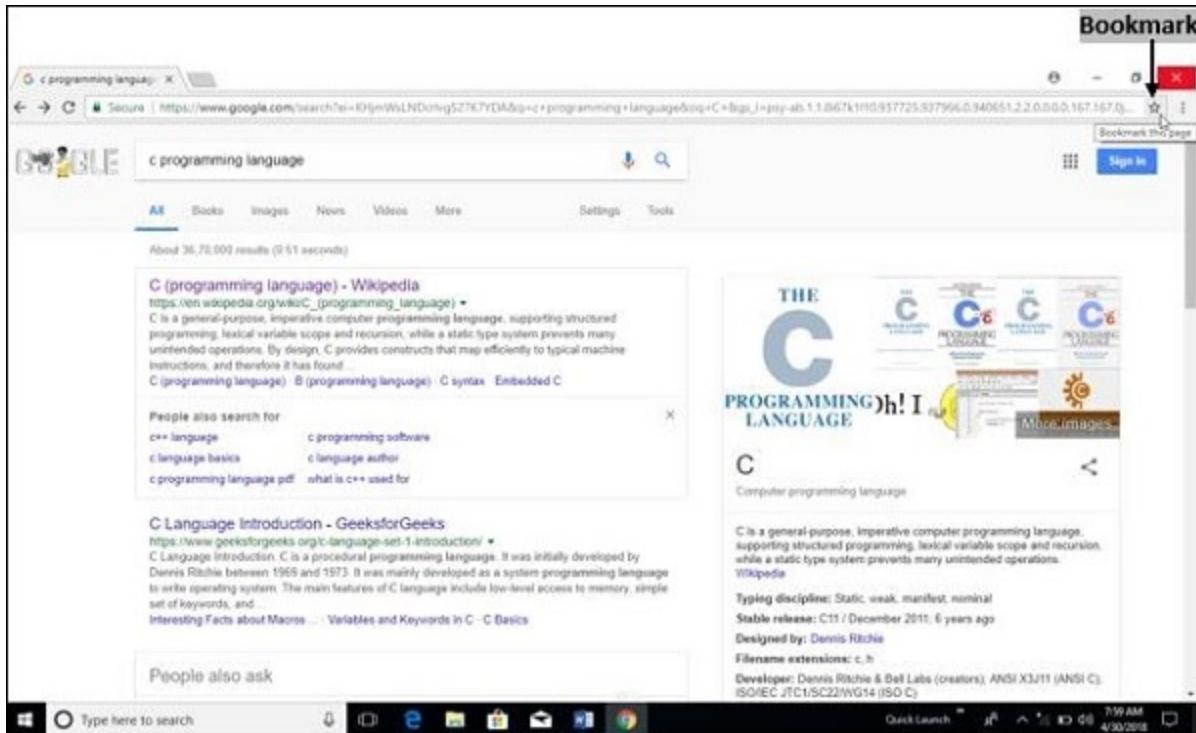


Image 18: Bookmark

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

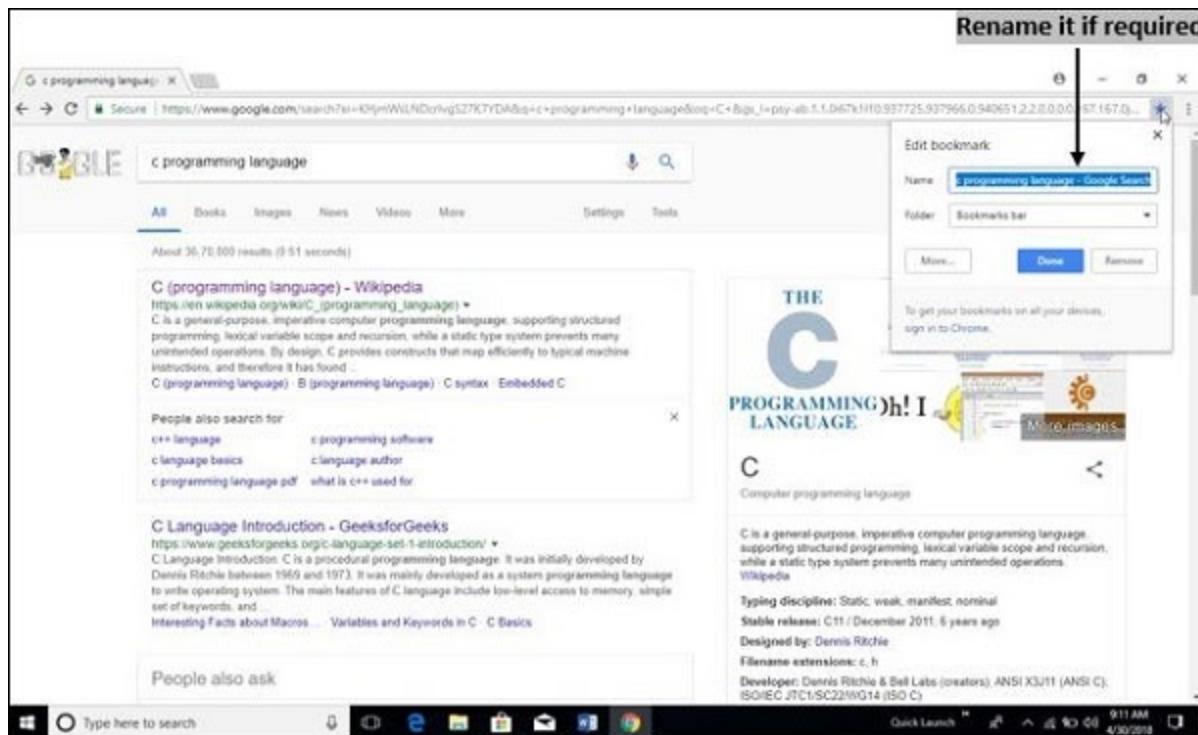


Image 19: Rename

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

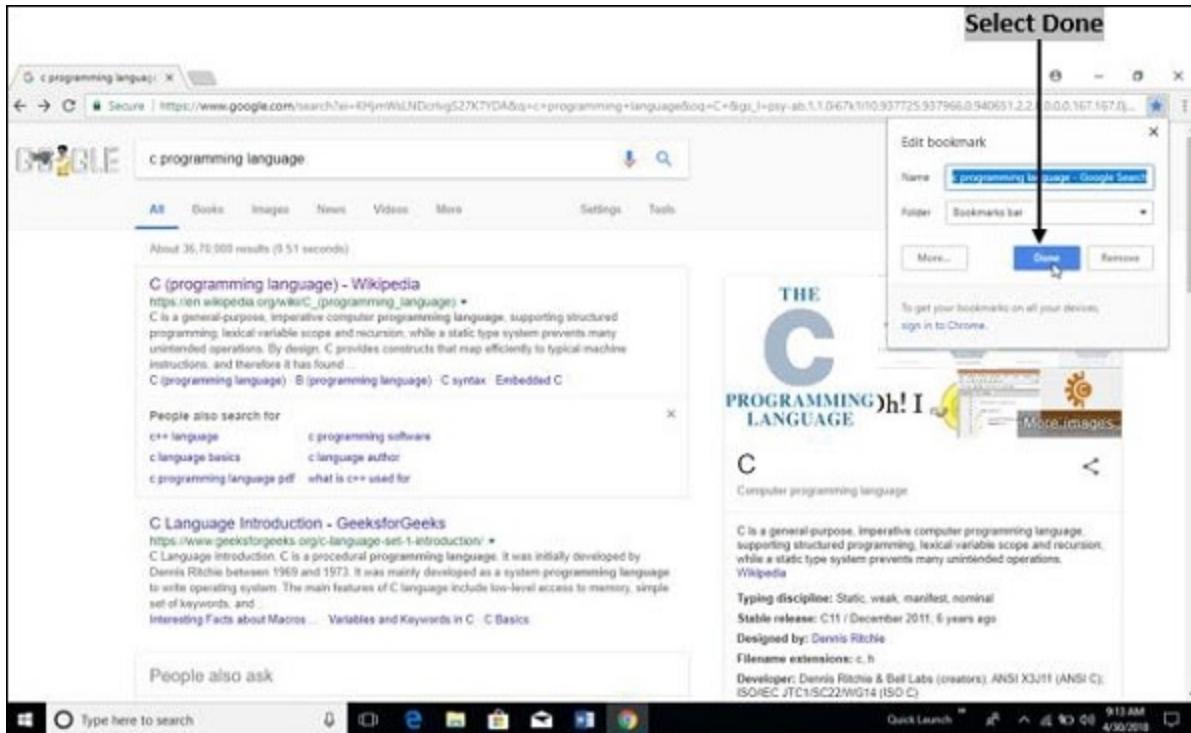


Image 20: Bookmark Done

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 2 – In order to add the web page, type the page you want to add as favorite and click "Done" button or you can click the three vertical dots (⋮) icon on the top right corner of the screen and select "Bookmark this page" option from the displayed menu to bookmark the current page.

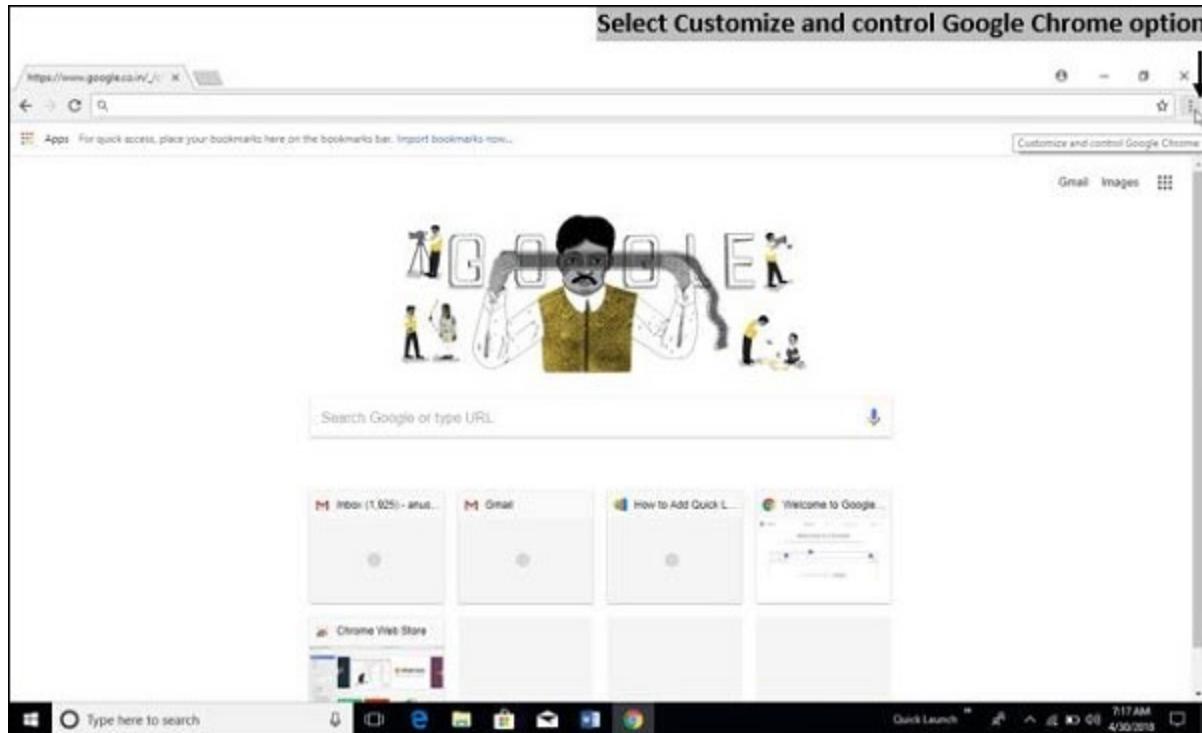


Image 21: Customized
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

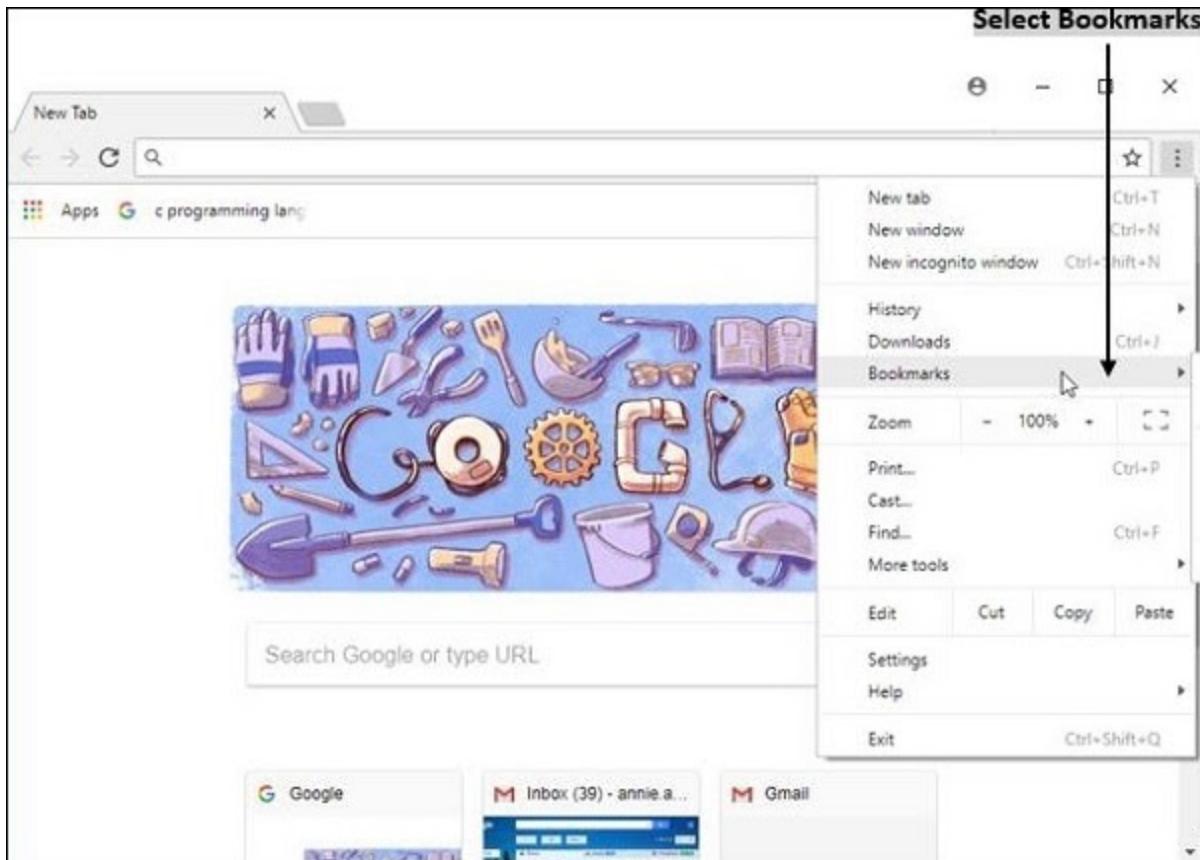


Image 22: Select Bookmark

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

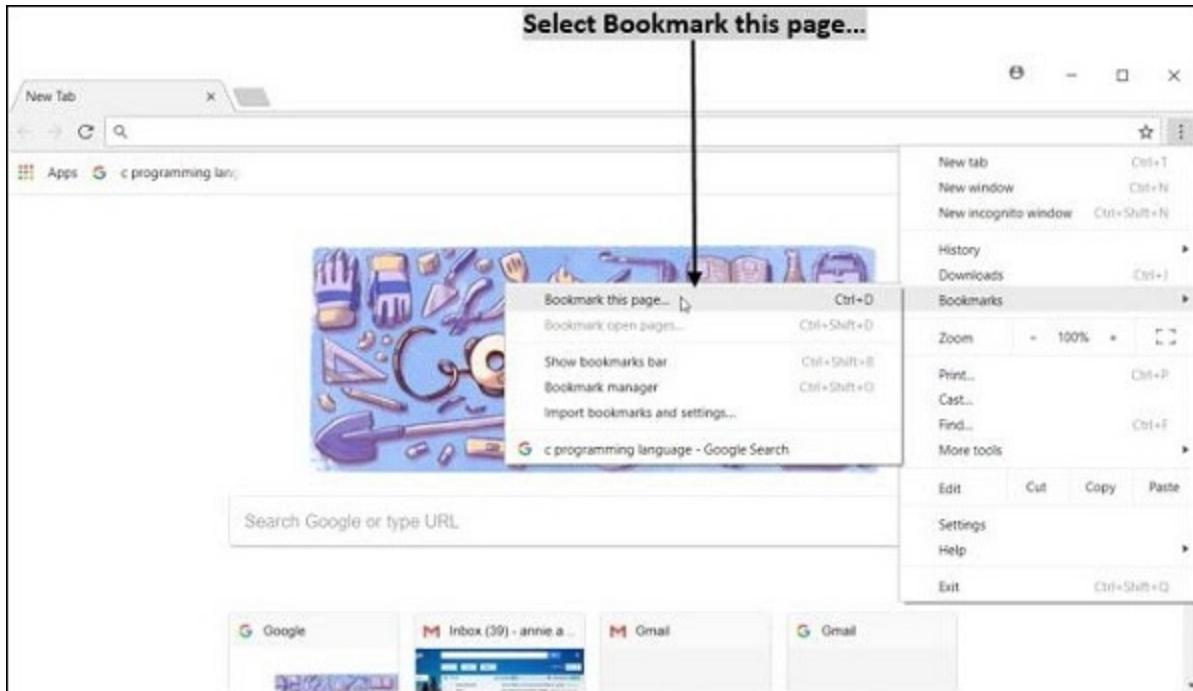


Image23: Select bookmark this page
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 3 – In order to remove the web page, select a page and press "Remove" button or you can click the three vertical dots icon(⋮) on the top right corner of the screen and select the web page you want to delete and "Right-click" and click "Delete" option.

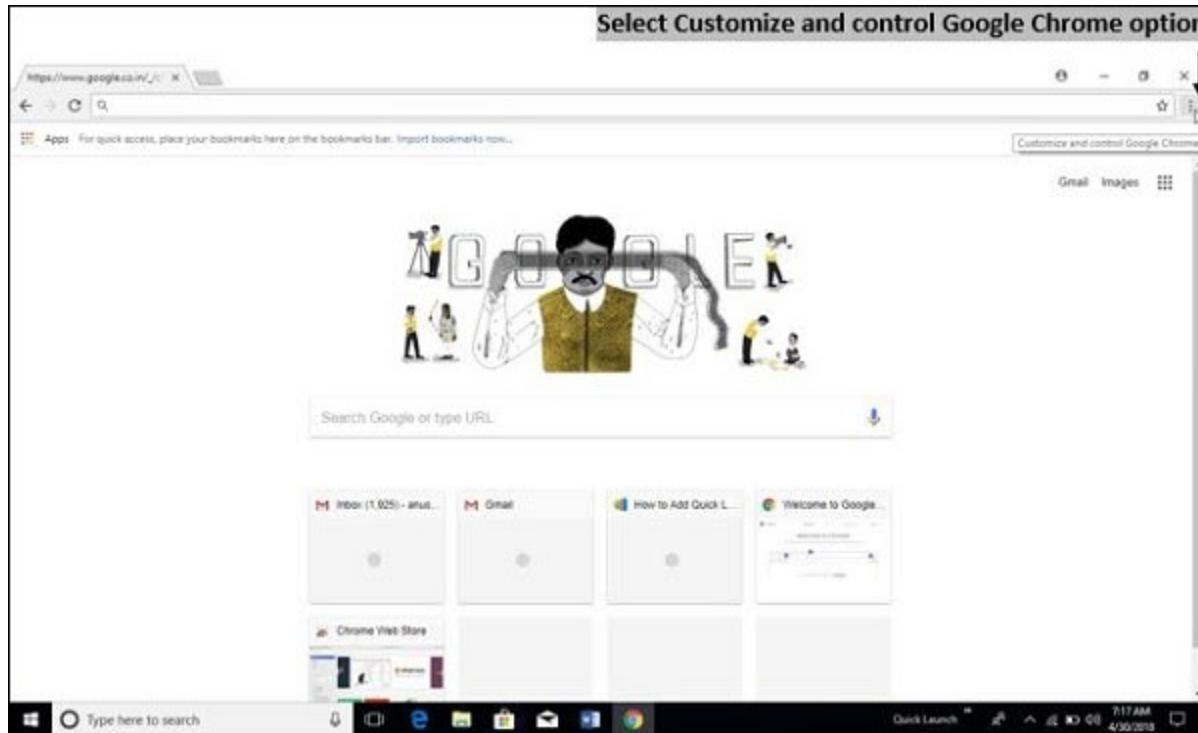


Image 24: Select Customize and control Google Chrome option
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

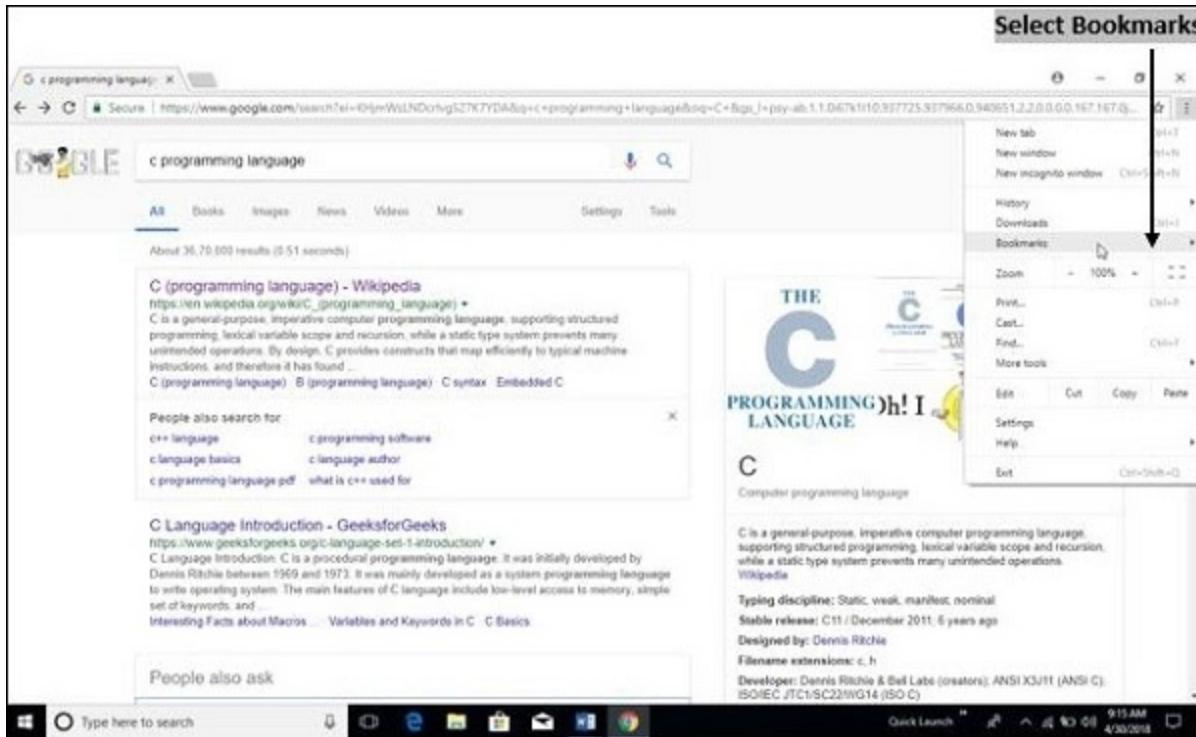
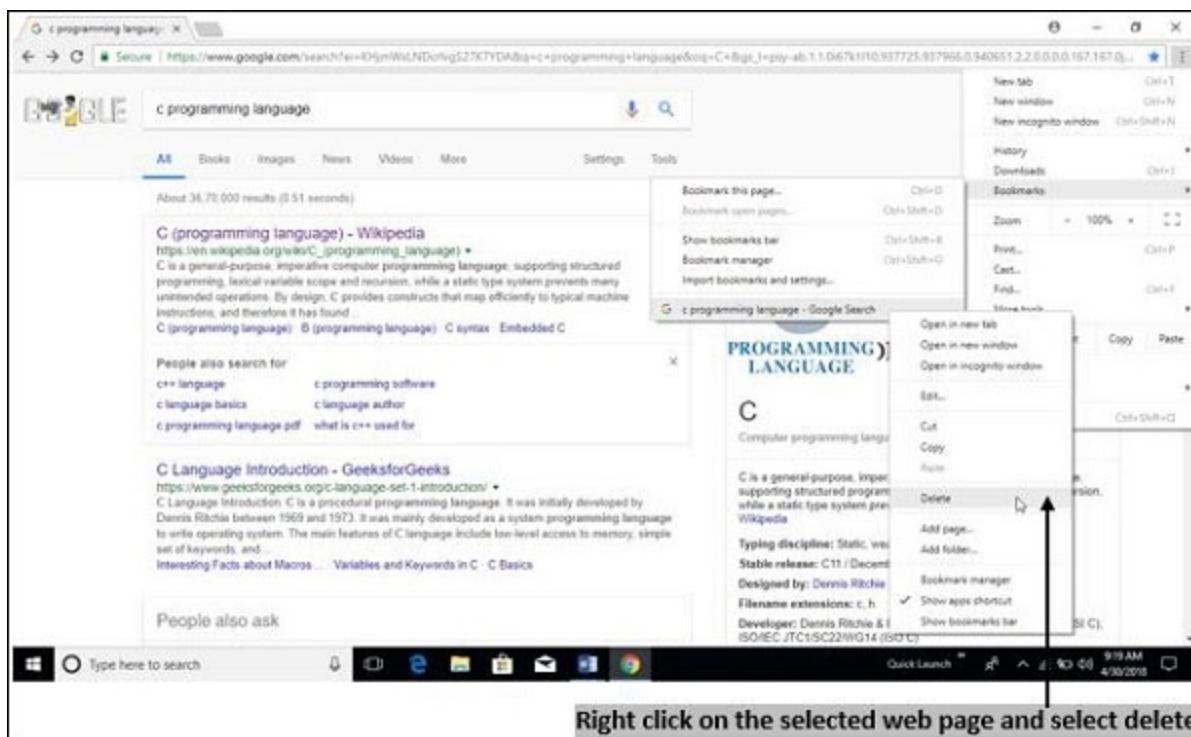
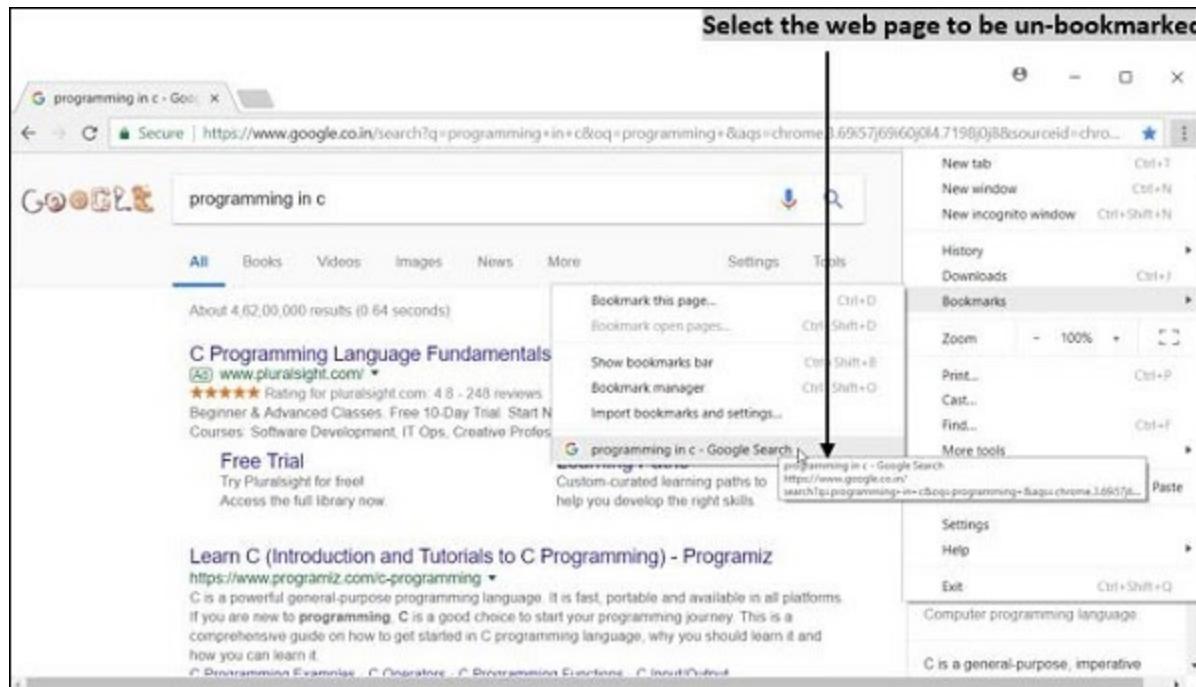


Image 25: Select Bookmarks

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser



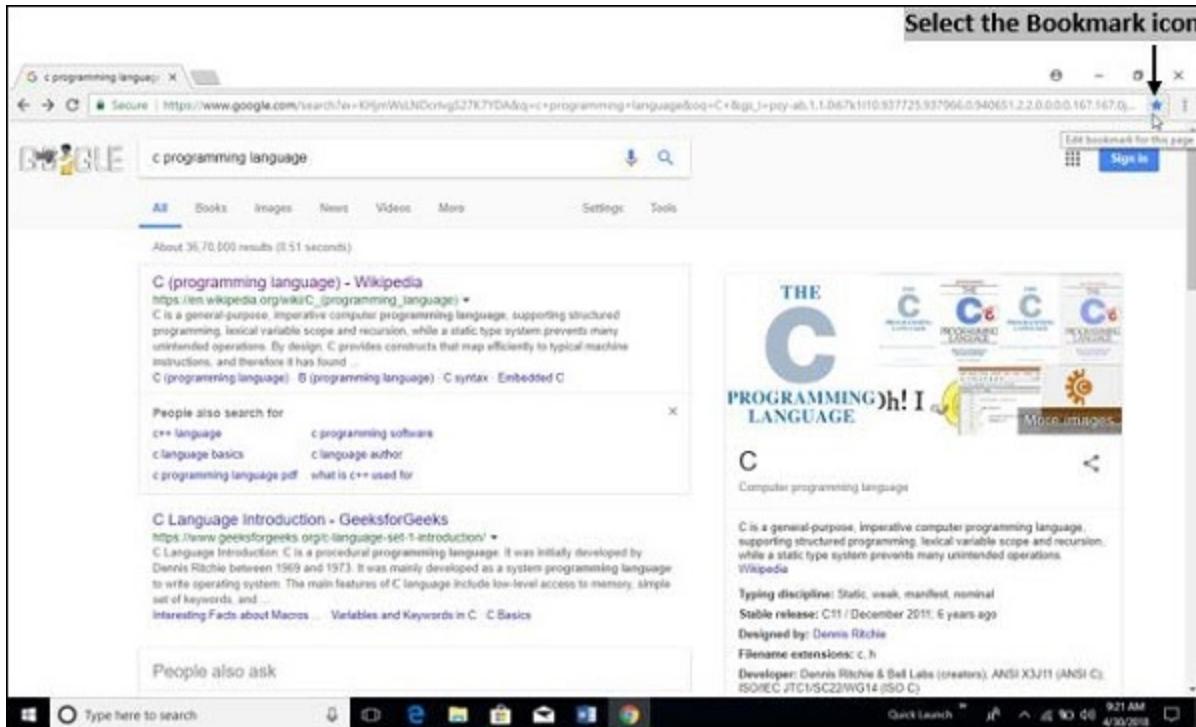


Image 26: Select the Bookmark icon
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

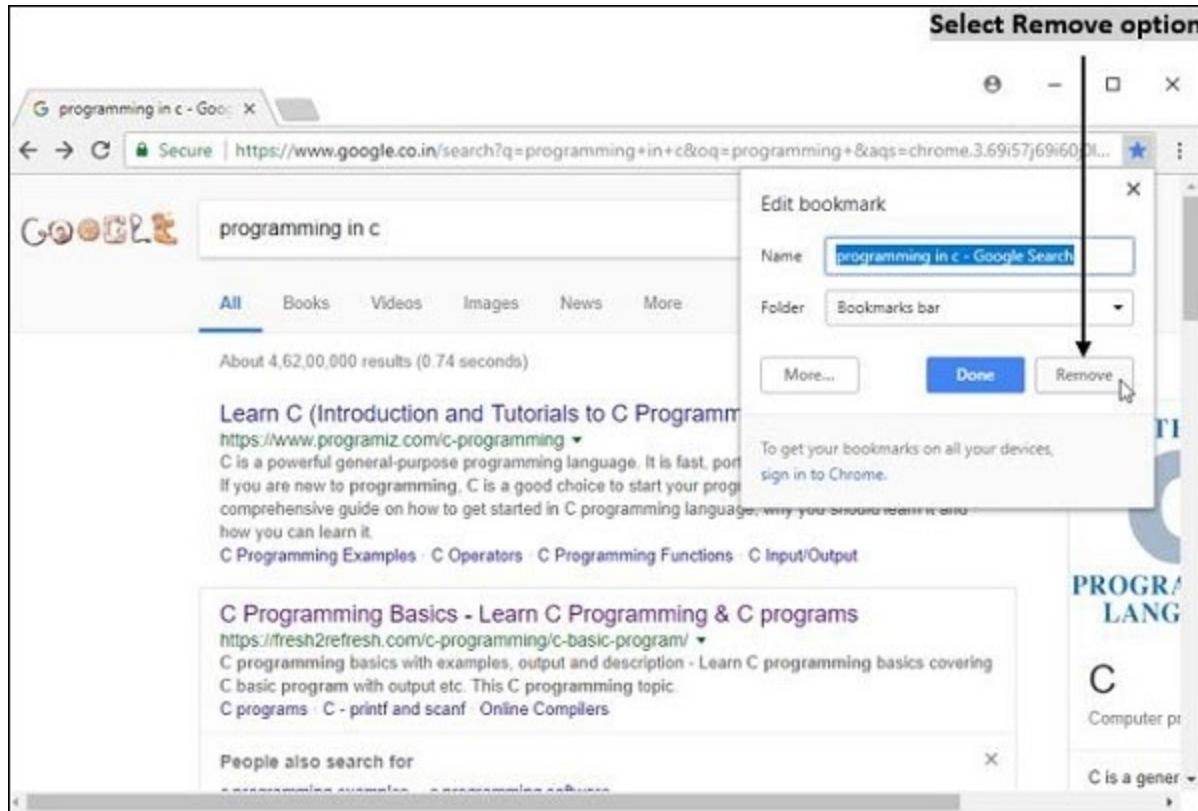


Image 27: Select Remove option
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Downloading Web Pages

Downloading is saving a file or document or web page on your hard disk. It consists of the following steps.

Step 1 – Open a web browser and navigate to the webpage which you want to download.

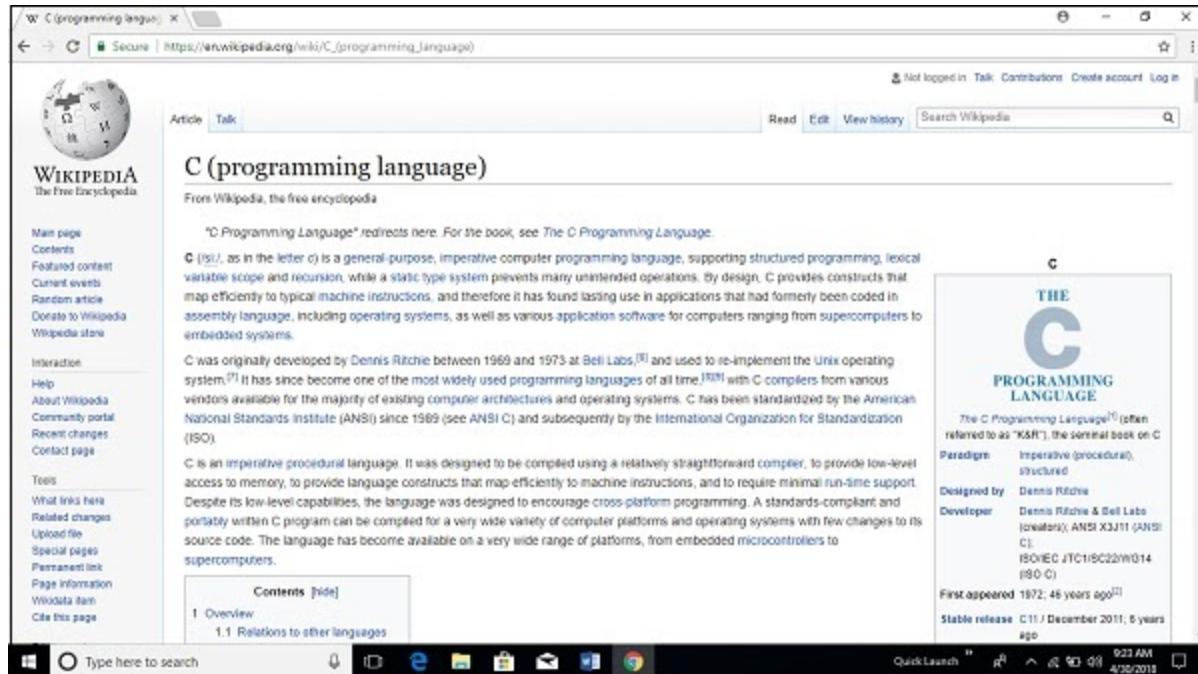


Image 28: Downloading web pages
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 2 – Right-click on the file and choose Save as.



Image 29:Save as a webpage

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

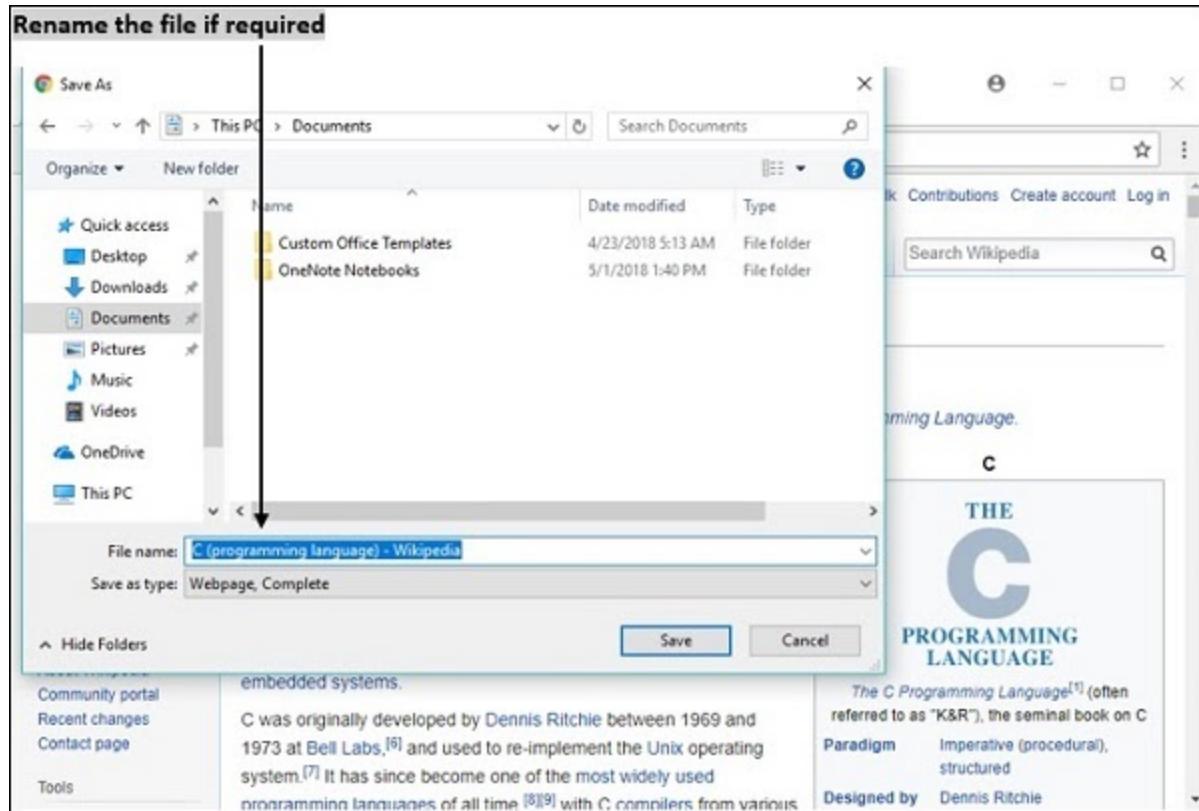


Image 30: Rename

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 3 – Choose where you want to save the file, then click Save.

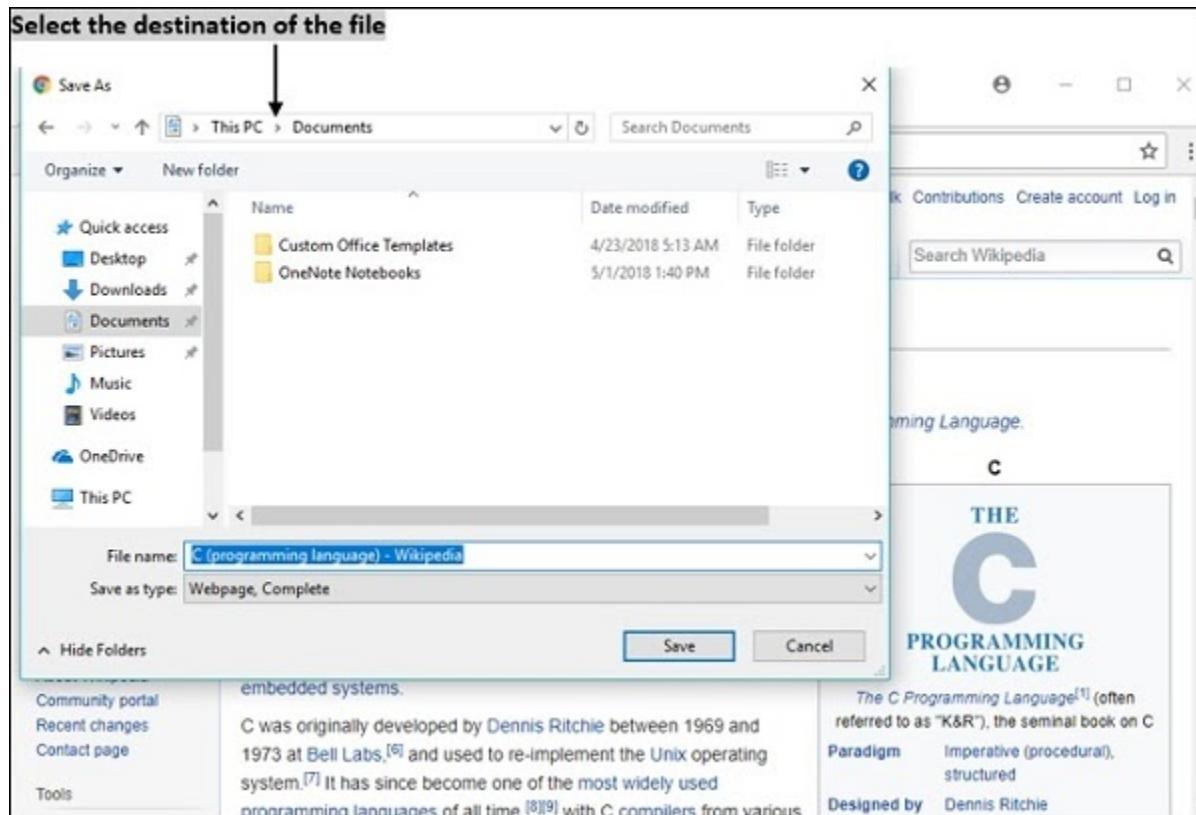


Image 31: Select the Destination File

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

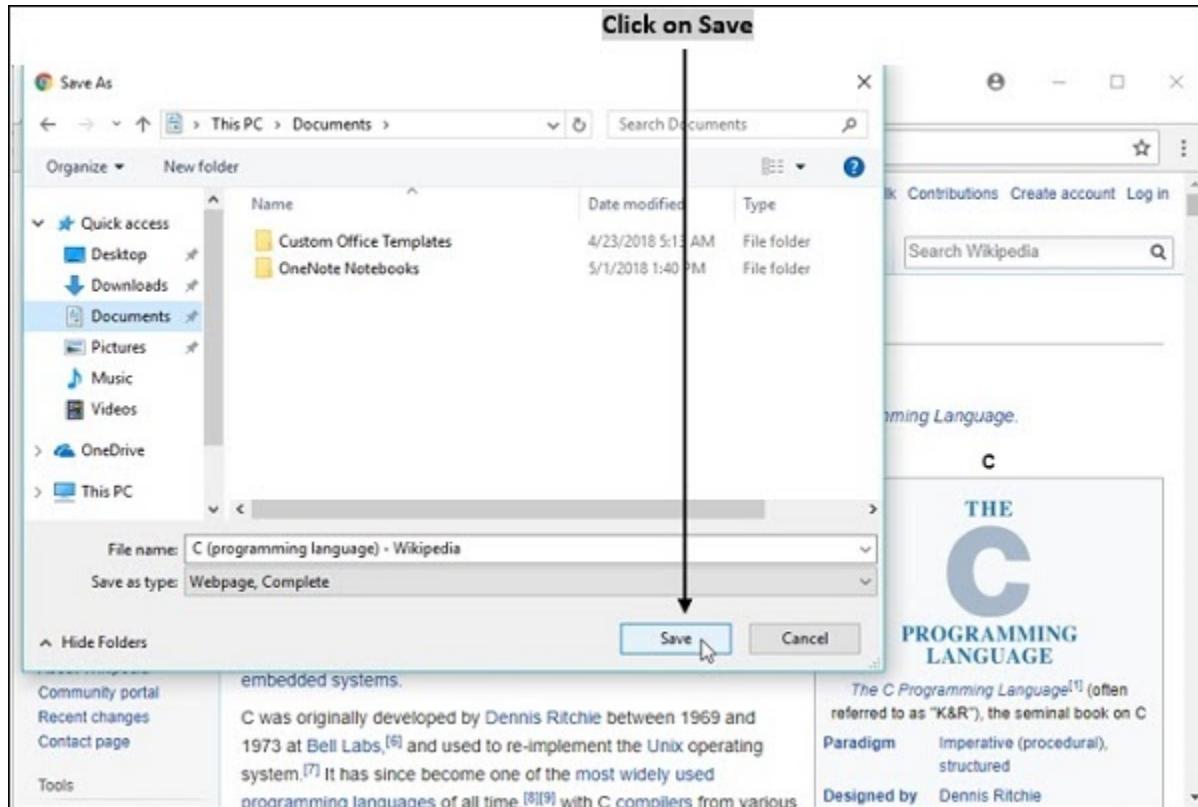


Image 32: Save web page
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 4 – When the file is downloaded, you'll see it at the bottom of your Chrome window. Click the file name to open it.



Image 33: Downloaded File

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Pause, Resume or Cancel

At the bottom of the screen, you can see the downloading file. Click the arrow next to the file name at the bottom of your screen. Click Pause, Resume or Cancel, whatever action you want to perform.

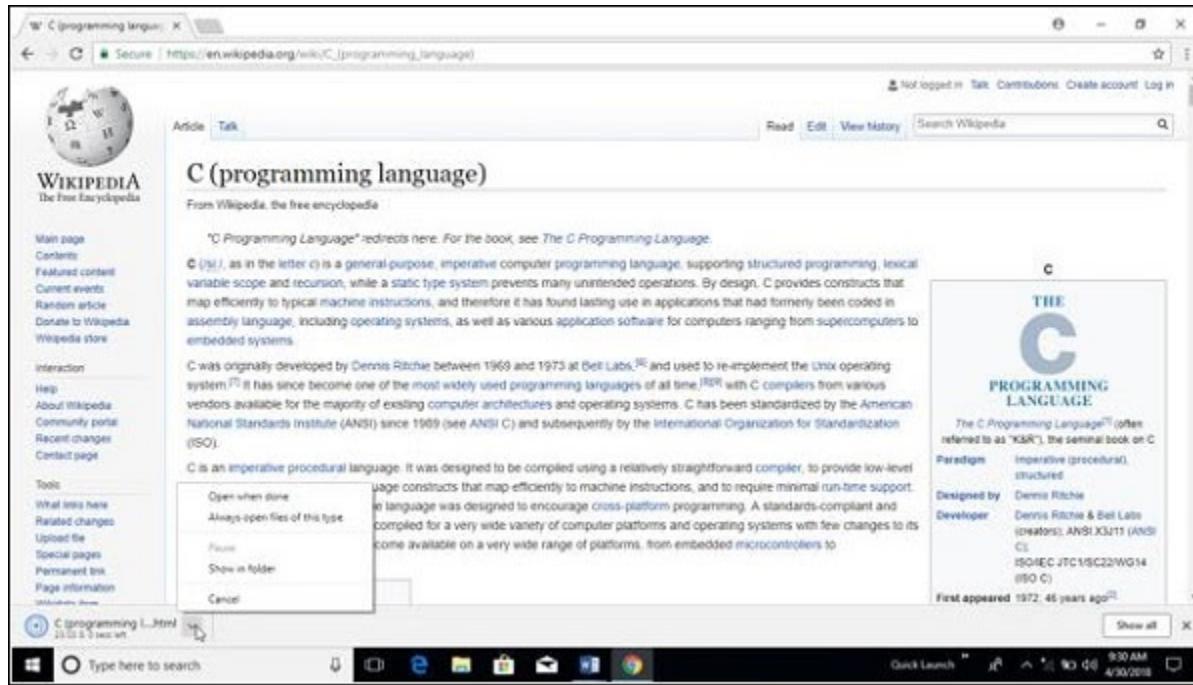


Image 34: Pause Resume Cancel

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Printing Web Pages

Printing is creating a hard copy of a document which can be a web page or any other content. It includes the following steps –

Step 1 – After launching a web browser, open the page, image, or file you want to print.

Step 2 – Click on three vertical dots icon (⋮) on the top right corner of the screen or use a keyboard shortcut: Ctrl + P.

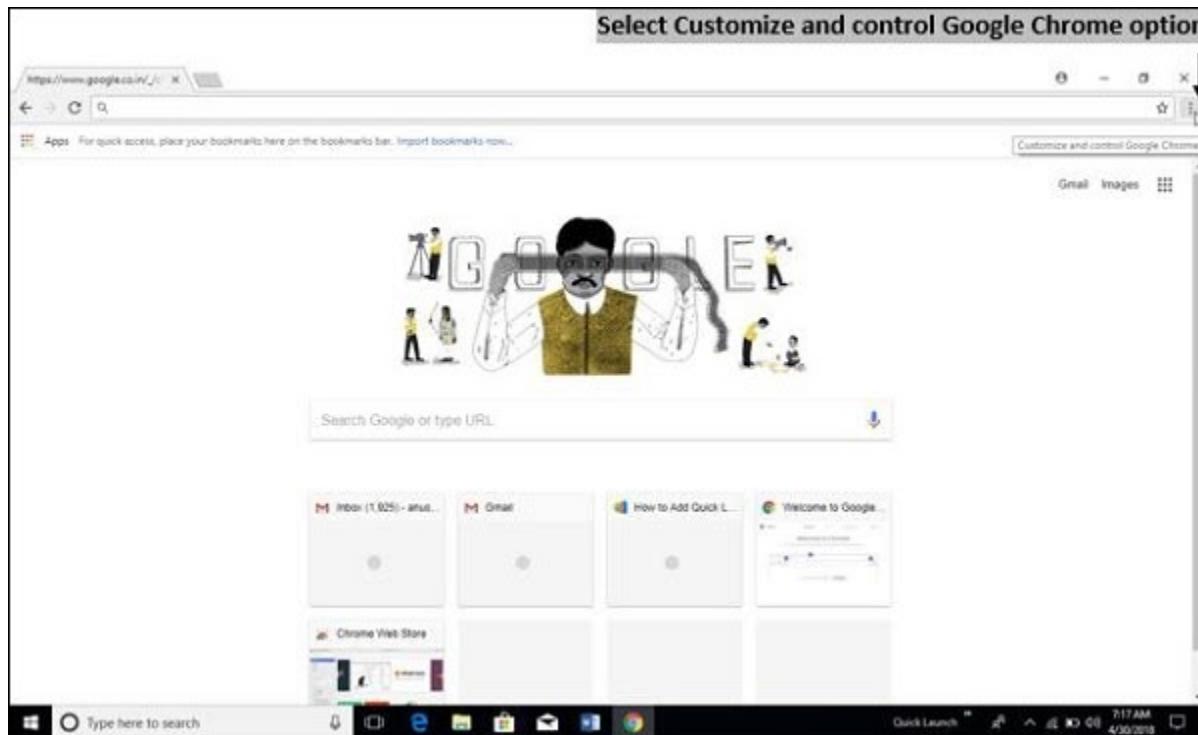


Image 35: Select Customize Control
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

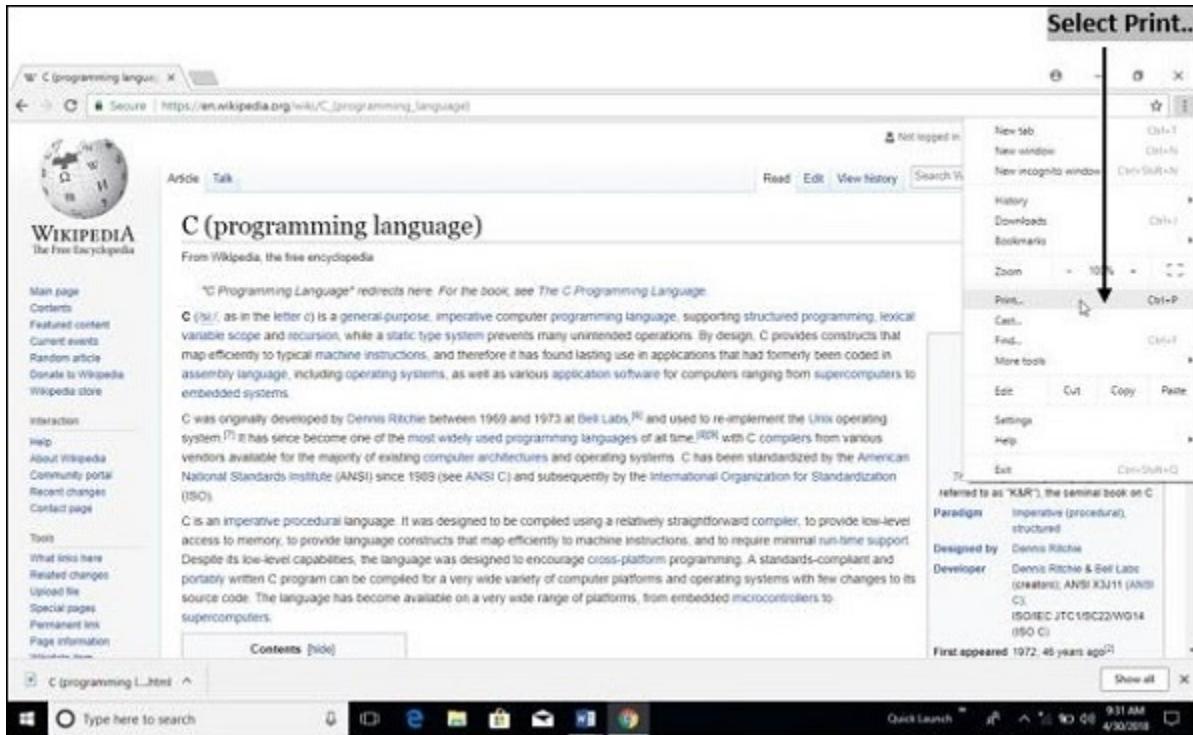


Image 36: Select Print
Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Step 3 – In the window that appears, select destination and change any print settings you want; when ready, click Print.

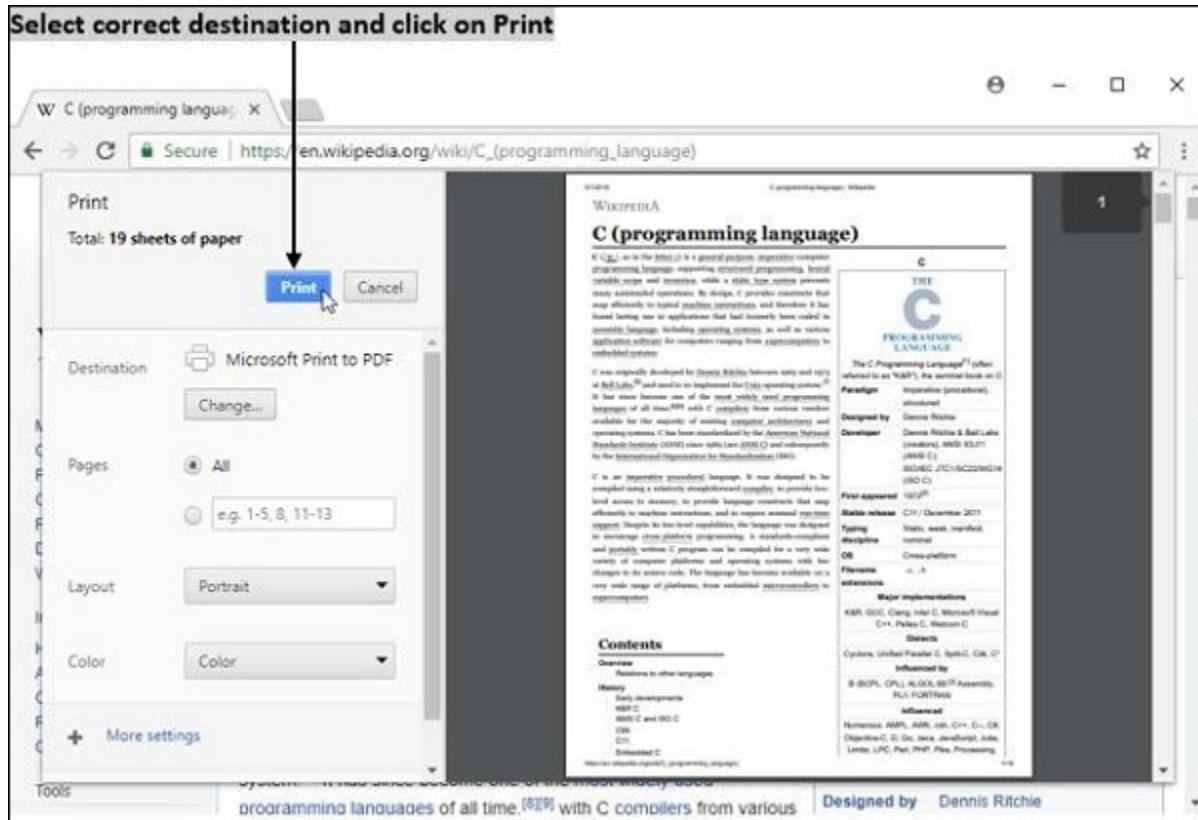


Image 37: Printing

Reference: https://www.tutorialspoint.com/computer_concepts/computer_concepts_accessing_web_browser

Search Engines

Search Engine refers to a huge database of internet resources such as web pages, newsgroups, programs, images, etc. It helps to locate information on the World Wide Web.

Users can search for any information by passing a query in the form of keywords or phrases. It then searches for relevant information in its database and returns it to the user.



Image 38: Search Engine

Reference:https://www.tutorialspoint.com/internet_technologies/search_engines.htm

Search Engine Components

Generally, there are three basic components of a search engine as listed below:

- Web Crawler
- Database
- Search Interfaces

Web crawler

It is also known as a spider or bots. It is a software component that traverses the web to gather information.

Database

All the information on the web is stored in a database. It consists of huge web resources.

Search Interfaces

This component is an interface between the user and the database. It helps the user to search through the database.

Search engines work by crawling hundreds of billions of pages using their own web crawlers. These web crawlers are commonly referred to as search engine bots or spiders. A search engine navigates the web by downloading web pages and following links on these pages to discover new pages that have been made available

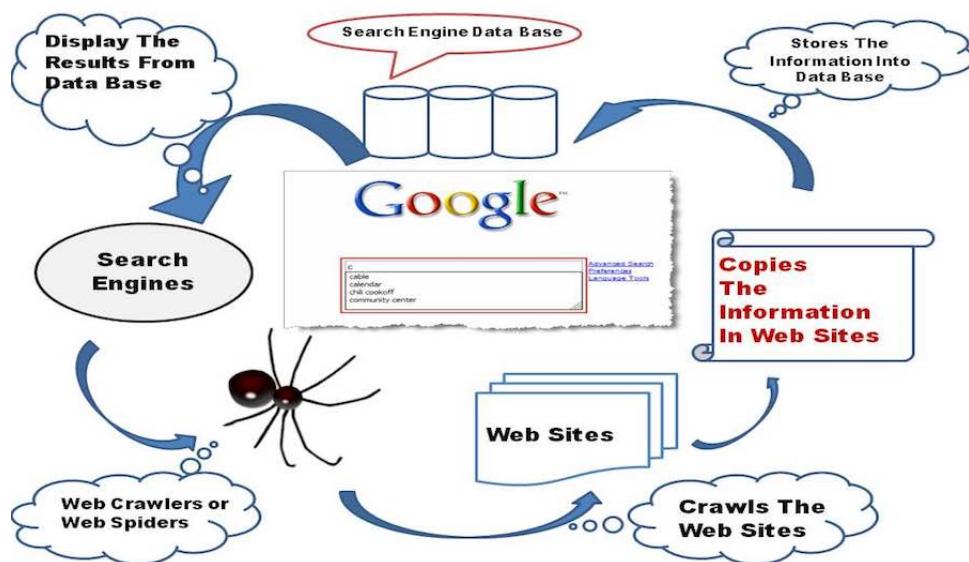


Image 39:How search engines Works?
Reference:<https://fossbytes.com/search-engine-works-makes-life-easier/>

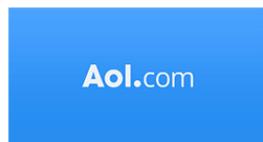


Image 40:Popular Web Browsers
Reference:<https://wildstonesolution.com/top-search-engines>

Services Available on the Internet

- Data Transfer
- Internet banking
- E-commerce
- E-Learning
- E-Governance
- Browsing and Chatting
- E-Mail

Data Transfer:

Data transfer is the process of using computing techniques and technologies to transmit or transfer electronic or analog data from one computer node to another. Data is transferred in the form of bits and bytes over a digital or analog medium, and the process enables digital or analog communications and its movement between devices. Data transfer is also known as data transmission.

Internet Banking:

Traditionally, customers used to access banking services through Retail/ corporate branches. But in this digital era, Online Banking has taken a vital role. The online banking is also called as internet banking, virtual banking or e-banking. This is a value-added application to connect the core banking system and provide the self-service bank facilities for customers online.



Image 41:Internet banking
Reference:https://www.brainkart.com/article/Services-Available-on-the-Internet_36837/

Features

- A bank customer can perform transactional and non-transactional tasks through online banking
- Viewing account balances, transactions, statements of customer
- Viewing images of paid cheques request for cheque books

-
- Funds transfers between the customer's linked accounts
 - Paying third parties, including bill payments and third-party fund transfers
 - Register utility billers and make bill payments

Advantages

- Permanent online access for banking transactions.
- Access anywhere using the application via mobile or computer
- Less time consuming, easy to use and safe
- Customer can manage their funds instantly and accurately

E-commerce:

E-commerce application is a transaction of buying or selling goods and services online. Electronic commerce attraction technologies such as mobile commerce, electronic funds transfer, supply chain management, Internet marketing, online transaction processing, electronic data interchange (EDI), inventory management systems, and automated data collection systems.

E-commerce businesses may also employ some or all of the followings:

- Online shopping web sites for retail sales direct to consumers
- Providing or participating in online marketplaces, which process third-party business-to-consumer or consumer-to-consumer sales
- Business-to-business buying and selling;
- Gathering and using demographic data through web contacts and social media
- Business-to-business (B2B) electronic data interchange
- Marketing to prospective and established customers by e-mail or fax (for example, with newsletters)
- Engaging in retail for launching new products and services

- Online financial exchanges for currency exchanges or trading purposes.

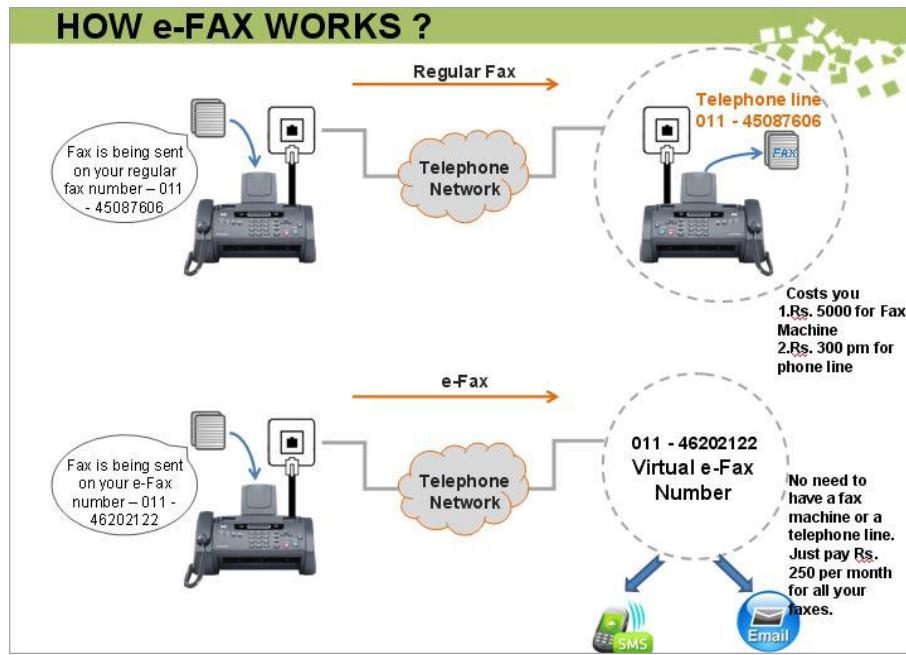


Image 42:eFax works

Reference:https://www.brainkart.com/article/Services-Available-on-the-Internet_36837/

e-Marketing: (Electronic Marketing)

E-Marketing is the process of marketing a product or service using the internet. It also includes marketing done via email and wireless media. It is also called Digital Marketing.



Image 43:e-marketing
Reference:https://www.brainkart.com/article/Services-Available-on-the-Internet_36837/

Professionals working in e-marketing must design and implement Internet marketing plans. But they also must have a broad understanding of what makes these plans effective. Those working in e-marketing must be able to carry out many tasks:

- Following business market trends
- Consulting with companies about digital marketing needs
- Resolving issues businesses have in reaching customers oriented problems
- Creating e-marketing objectives for business challenges
- Developing marketing strategies competitive business model.
- Choosing cost-effective marketing methods

-
- Launching digital marketing campaigns and monitoring results

E-Learning :

A learning system based on electronic resources is known as E-learning. The use of computers and the Internet forms the major component of E-learning. E-learning can also be termed as a network-enabled transfer of skills and knowledge and the delivery of education is made to a large number of recipients at the same or different times.

E-Governance:

E-governance is the application of Information and Communication Technology (ICT) for delivering government services, exchange of information, communication transactions, integration of various stand-alone systems and services between government-to-citizen (G2C), Government-to-business (G2B), Government-to-Government (G2G), Government-to-Employees (G2E) as well as back-office processes and interactions within the entire government framework. Through e-governance, government services will be made available to citizens in a convenient, efficient and transparent manner. The three main target groups that can be distinguished in governance concepts are government, citizens and businesses/ interest groups. In e-governance, there are no distinct boundaries.

It classifies four basic models they are :

- Government-to-Citizen (customer)
- Government-to-Employees
- Government-to-Government
- Government-to-Business

Examples of e-Governance:

- Aadhaar Card is a 12-digit unique identity number issued to all Indian residents based on their biometric and demographic data.
- Inspector General of Registration portal - Tamil Nadu – www.tnreginet.gov.in used for Land and legal registrations



Image 44:e-governance
Reference:https://www.brainkart.com/article/Services-Available-on-the-Internet_36837/

Online Chatting:

Online chat refers to any kind of communication via the Internet that offers a real-time transmission of text messages from sender to receiver. The chat messages are generally short in order to enable other participants to respond quickly. Online chat may address point-to-point communications as well as multicast communications from one sender to many receivers and voice and video chat and web conferencing service.

Advantages of Internet

Communication:

The main advantage of the internet is faster communication than any other device. It's an instant process. Communication in the form of video calls, emails, etc. is possible using the internet. Thus, there is no specific region that can be accessed. It is accessible all over the world. Hence, because of this global issues are reduced since video conferencing is possible where everyone across the world can be in a single place and can solve a problem.

Information:

The internet is a source of knowledge. All kinds of information are present in it. It is easily accessed and can be searched more to get more additional knowledge. Information like educational related, government laws, market sales, stocks and shares, new creations, etc. are gathered from a single place.

Learning:

The internet has now become a part of education. Education like homeschooling is easily carried out using the internet. Teachers can upload their teaching videos on the internet and are accessed by people across the world which is helpful for all students. The marks are also released on the internet since releasing marks for the whole institution in notice boards will create chaos.

Entertainment:

The internet is now the most popular form of entertainment. Movies, songs, videos, games, etc. are available on the internet for free. Social networking is also possible using the internet. Hence, there are tons of entertainment that are available online on the internet.

Social network:

Social networking is the sharing of information with people across the world. Apart from being an entertainment website, it has many uses. Any job vacancy, emergency news, ideas, etc. can be shared on the website and the information gets passed on quickly to a wide area. Also, social networking websites are used for easy communications. Example: Facebook and Twitter.

E-commerce:

All business deals can be carried on the internet like transactions of money etc. This is called E-commerce. Online reservations, online ticket booking for movies etc. can be done easily. It

saves us lots of time. Online shopping is now the latest trend in the internet world where products from dresses to household furniture is available at doorstep.



Image 45:Service available in the internet
Reference:<https://indiacelebratings.com/advantages-disadvantages-of-internet/>

Structure and Working of E-Mail

Electronic Mail (email or e-mail) is a method of exchanging messages between people using electronic devices. Email first entered limited use in the 1960s and by the middle of 1970s had taken the form now recognized as email. Email operates across computer networks, which is primarily called the Internet.

Earlier email systems required the sender and the recipient to both be online at the same time, in common with instant messaging. Today's email systems are based on a store-and-forward model. Email servers accept, forward, deliver, and store messages.

The structure of the E-mail address is username@domain name

An example of E-mail address is raman@gmail.com

An E-mail address consists of two parts separated by @ symbol. The first part Raman is the user name that identifies the address and the second part gmail.com is the domain name of the E-mail server.

Sample Email Application

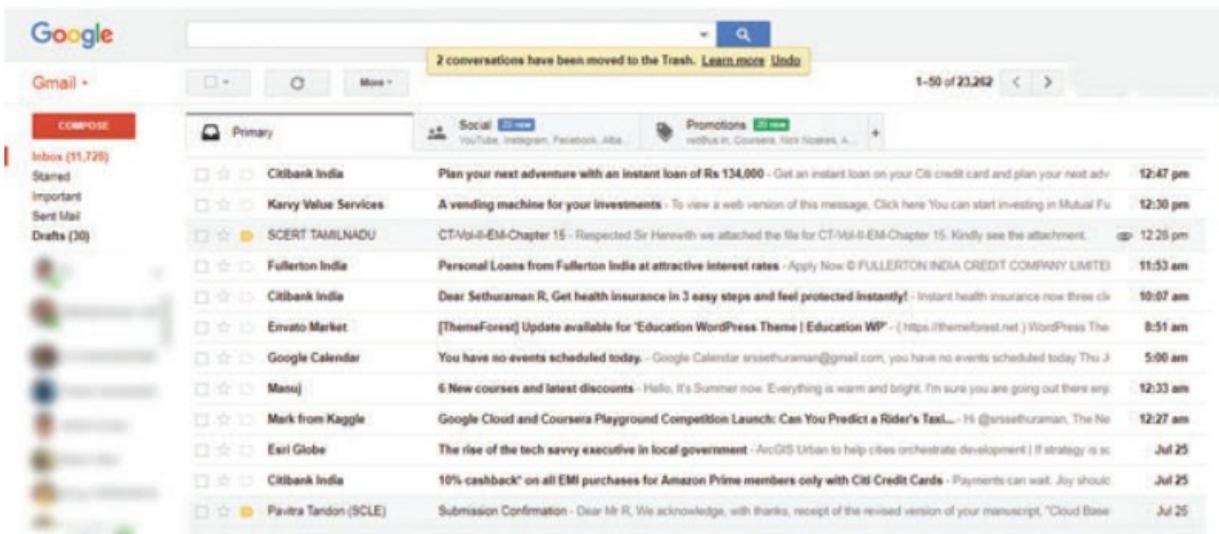


Image 46: Sample Email
Reference: https://www.brainkart.com/article/Structure-and-Working-of-E-Mail_36840/

How Email works on the Internet :

To send Internet e-mail, requires an Internet connection and access to a mail server. The standard protocol used for sending Internet e-mail is called SMTP (Simple Mail Transfer Protocol). The SMTP protocol is used to both send and receive email messages over the Internet.

When a message is sent, the email client sends the message to the SMTP server. If the recipient of the email is local the message is kept on the server for accessing by the POP, IMAP or other mail services for later retrieval.

If the recipient is remote (i.e. at another domain), the SMTP server communicates with a Domain Name Server (DNS) to find the corresponding IP address for the domain being sent to. Once the

IP address has been resolved, the SMTP server connects with the remote SMTP server and the mail is delivered to this server for handling.

If the SMTP server sending the mail is unable to connect with the remote SMTP server, then the message goes into a queue. Messages in this queue will be retried periodically. If the message is still undelivered after a certain amount of time (30 hours by default), the message will be returned to the sender as undelivered.



Image 47:How Email really works?
Reference:https://www.brainkart.com/article/Structure-and-Working-of-E-Mail_36840/

Structure of an Email message:

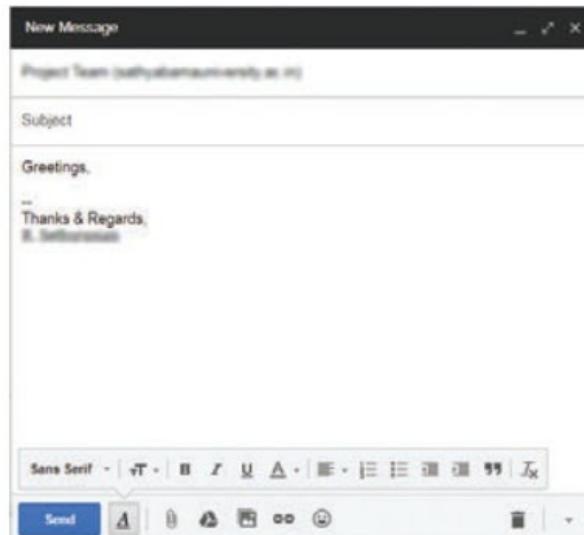


Image 48:Structure of an Email
Reference:https://www.brainkart.com/article/Structure-and-Working-of-E-Mail_36840/

To: This field consists of the address to whom the message has to be sent. This is mandatory.

CC: Short for carbon copy. This is optional. The people who were mailed copies of the message. The recipients of the message will know to whom all the copies have been sent.

BCC: It stands for Black Carbon Copy. It is used when we do not want one or more of the recipients to know that someone else was copied on the message. This is optional.

Subject : The Subject field indicates the purpose of the e-mail.

Attachment: Attachment contains files that you are sending, linked documents, pictures, etc. along with an e-mail.

Body: The email body is the main part of an email message. It contains the message's text, images and other data (such as attachments). The email's body is distinct from its header, which contains control information and data about the message (such as its sender, the recipient and the path an email took to reach its destination).

Signature: Name of the sender

Advantages and Disadvantages of Email:

Advantages:

- Reliable: Because it notifies the sender if not delivered.
- Speed: E-mail is very fast delivered in a fraction of seconds.
- Inexpensive: It's very cheap.
- Waste Reduction: Helps in paperless communication thus eco-friendly.

Disadvantages:

- Forgery: Anyone who hacks the password of the sender can send a message to anyone.

-
- Overload: Because it is cheap, loads and loads of messages keep coming.
 - Junk: Junk emails are not intended mails and are inappropriate also. Junk emails are sometimes referred to as spam.

Internet Applications

The Internet is a network of computers linking many different types of computers all over the world. It is a network of networks sharing a common mechanism for addressing(Identifying) computers and a common set of communication protocols for communications between two computers on the network.

Applications of Internet

1. Communication

Computer users around the world extensively use the email service on the internet to communicate with each other. Pictures, documents and other files are sent as email attachments. Emails can be cc-ed to multiple email addresses

Internet telephony is another common communications service made possible by the creation of the Internet. VoIP stands for Voice-over-Internet Protocol, referring to the protocol that underlies all Internet communication.

2. Job search

Nowadays, many people search for their jobs online as it is quicker and there is a larger variety of job vacancies present. People can publish resumes online for prospective jobs. Some of the web sites providing this service are naukri.com, monster.com, summerjob.com, recruitmentindia.com etc.

3. Online Shopping

The internet has also facilitated the introduction of a new market concept consisting of virtual shops. They provide information about products or services for sale through www servers. Using the internet services customers can submit specific product queries and request specific sales

quotes. For example amazon.com is a www based bookshop on the internet where information on all types of international books can be found and books can be ordered online.

4. Stock market updates

You can sell or buy shares while sitting on the computer through the internet. Several websites like ndtvprofit.com, moneypore.com, provide information regarding investment

5. Travel

One can use the internet to gather information about various tourist places . It can be used for booking Holiday tours , hotels, train,bus, flights and cabs. Some of the web sites providing this service are goibibo.com, makemytrip.com, olacabs.com.

6. Research

Research papers are present online which helps in the researcher doing a literature review

7. Video Conferencing:

It enables direct face-to-face communication across networks via web cameras, microphones, and other communication tools. Video conferencing can enable individuals in distant locations to participate in meetings on short notice, with time and money savings. The technology is also used for telecommuting, in which employees work from home. When video Conferencing is used in education, it is easier to have interactive communications between teacher to teacher, teacher to classroom, or classroom to classroom with students in different places.

8. E-Commerce

E-commerce (electronic commerce or EC) is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the Internet. These business transactions occur either business-to-business, business-to-consumer, consumer-to-consumer or consumer-to-business. Largest e-commerce companies in India are Flipkart, Snapdeal, Amazon India, Paytm.

9. On-line payments

The rising boom of online payments in India has given way to many new entrants in the industry such as Paytm, MobiKwik, Oxigen etc who are majorly wallet driven payment companies. this

growth has been driven by rapid adoption led by the increasing use of smartphones, tablets and speedy access to internet through broadband, 3G etc

10. Social networking

Social networking is the use of internet-based social media programs to make connections with friends, family, classmates, customers and clients. Social networking can be done for social purposes, business purposes or both. The programs show the associations between individuals and facilitate the acquisition of new contacts. Examples of social networking have included Facebook, LinkedIn, Classmates.com and Yelp.



Image 49:Internet Applications
Reference:<https://www.eucba.com/what-is-internet-application/>

Introduction to HTML

What is HTML?

HTML stands for Hyper Text Markup Language, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and web applications.

- HTML stands for Hyper Text Markup Language
- HTML elements are the building blocks of HTML pages
- HTML describes the structure of Web pages using HyperText markup language
- HTML elements are represented by tags
- When you test your page on browsers. Browsers do not display the HTML tags, but use them to render the content of the page.
- HTML page extension always will be .html (example.html)

"Hypertext" refers to the hyperlinks that an HTML page may contain. "Markup language" refers to the way tags are used to define the page layout and elements within the page

Structure of an HTML document

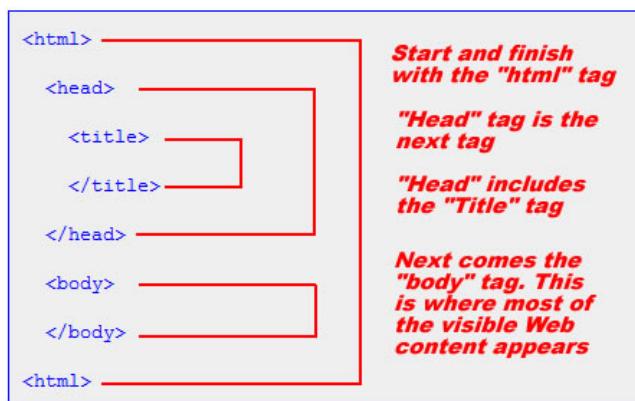


Image 50:Structure of an HTML
Reference:<https://openlab.citytech.cuny.edu/clarkeadv2450/html/>

<DOCTYPE! html>:

This tag is used to tell the HTML version. This currently tells that the version is HTML 5.

<html>:

This is called the HTML root element and used to wrap all the code.

<head>:

Head tag contains metadata, title, page CSS etc. All the HTML elements that can be used inside the <head> element are:

- <style>
- <title>
- <base>
- <noscript>
- <script>
- <meta>

<body>:

Body tag is used to enclose all the data which a web page has from texts to links. All of the content that you see rendered in the browser is contained within this element.

Example: HTML page can be created using any text editor (notepad). Then save that file using .htm or .html extension and open that file in the browser. It will get the HTML page response.

<meta>

The meta tag contains information for search engines, telling them what the page is about

<title>

The title tag is just that, the title that appears at the top of browser window and what first appears on search engine results

HTML Tags and Attributes

HTML Elements

Elements are the fundamentals of HyperText Markup Language (HTML). Each HTML document is made of elements that are specified using tags.

HTML elements and HTML tags are often confused. The tags are used to open and close the object, whereas the element includes both tags and its content. Let's consider an example with the `<h1>` tag: `<h1> Title of the document </h1>` - is an element, and `<h1>`, `</h1>` - are tags.

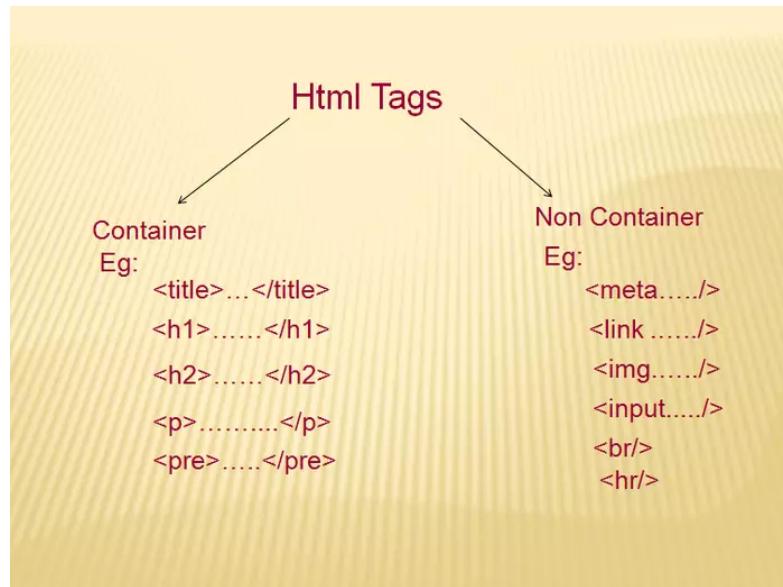


Image 51:HTML Elements
Reference:<https://www.phptpoint.com/html-tags/>

Types of HTML Elements

- empty elements
- Block-level elements
- Inline elements

The empty elements have no closing tags. In XHTML the empty elements are "closed" in the opening tag like this:
 .

The empty elements in HTML are: <area>, <base>,
, <col>, <embed>, <hr>, , <input>, <keygen>, <link>, <meta>, <param>, <source>, <track> and <wbr>.

Code

```
<!DOCTYPE html>
```

Source:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
  </head>
  <body>
    <h1>Title of the document</h1>
    <p>The first paragraph</p>
    <p>The second paragraph, <br/> where line break is inserted </p>
  </body>
</html>
```

Output:

Title of the document

The first paragraph

The second paragraph,
where line break is inserted

Block-level and Inline HTML Elements

For the purpose of styling, all HTML elements are divided into two categories: block-level elements and inline elements.

Block-level elements are those that structure the main part of your web page, by dividing your content into coherent blocks. They always start on a new line and take up the full width of a page, from left to right; they also can take up one line or multiple lines and have a line break before and after the element. Block-level elements can include both block-level & inline elements.

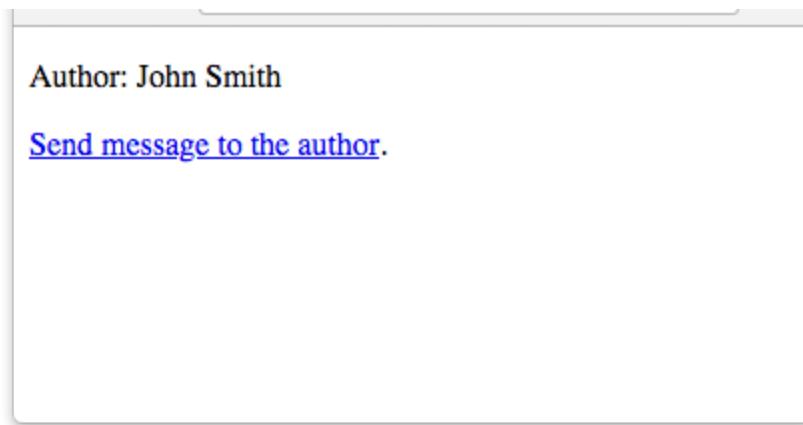
Block-level elements are `<address>`, `<article>`, `<aside>`, `<blockquote>`, `<canvas>`, `<dd>`, `<div>`, `<dl>`, `<dt>`, `<fieldset>`, `<figcaption>`, `<figure>`, `<footer>`, `<form>`, `<h1>-<h6>`, `<header>`, `<hr>`, ``, `<main>`, `<nav>`, `<noscript>`, ``, `<output>`, `<p>`, `<pre>`, `<section>`, `<table>`, `<tfoot>`, `` and `<video>`.

All block-level elements have opening and closing tags.

Source

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
  </head>
  <body>
    <footer>
      <p>Author: W3docs team</p>
      <p><a href="info@w3docs.com">Send message to the author</a>.</p>
    </footer>
  </body>
</html>
```

Output



Inline elements are used to distinguish part of a text, to give it a particular function or meaning. Inline elements usually include a single or a few words. They do not cause a line break and do not take up the full width of a page, only the space bounded by its opening and closing tag. The inline elements are usually used within other HTML elements.

Inline elements are: `<a>`, `<abbr>`, `<acronym>`, ``, `<bdo>`, `<big>`, `
`, `<button>`, `<cite>`, `<code>`, `<dfn>`, ``, `<i>`, ``, `<input>`, `<kbd>`, `<label>`, `<map>`, `<object>`, `<q>`, `<samp>`,

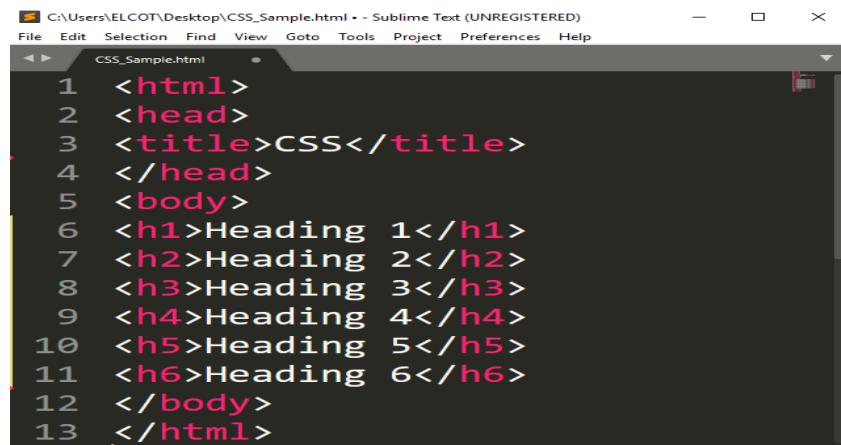
<script>, <select>, <small>, , , <sub>, <sup>, <textarea>, <time>, <tt> and <var>.

Headings

Headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

Source



```
C:\Users\ELCOT\Desktop\CMS_Sample.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
CSS_Sample.html
1 <html>
2 <head>
3 <title>CSS</title>
4 </head>
5 <body>
6 <h1>Heading 1</h1>
7 <h2>Heading 2</h2>
8 <h3>Heading 3</h3>
9 <h4>Heading 4</h4>
10 <h5>Heading 5</h5>
11 <h6>Heading 6</h6>
12 </body>
13 </html>
```

Output



Formatting

HTML also defines special elements for defining text with a special meaning.

The formatting tags are divided into two categories:

- physical tags, used for text styling (visual appearance of the text)
- logical or semantic tags used to add semantic value to the parts of the text.

Formatting includes,

- - Bold text
- - Important text
- <i> - Italic text
- - Emphasized text
- <mark> - Marked text
- <small> - Small text

-
- - Deleted text
 - <ins> - Inserted text
 - <sub> - Subscript text
 - <sup> - Superscript text

HTML uses elements like and <i> for formatting output, like bold or italic text. Formatting elements were designed to display special types of text:

 and Tags

The tag is a physical tag that stands for bold text, whereas the tag being a logical tag is used to emphasize the importance of the text.

This text is bold

This text is strong

<i> and Tags

The <i> and tags define italic text. The <i> tag is only responsible for visual appearance of the enclosed text, without any extra importance. The tag defines emphasized text, with added semantic importance.

<i>This text is italic</i>

This text is emphasized

<pre> Tag

The <pre> tag is used to define preformatted text. The browsers render the enclosed text with white spaces and line breaks.

<pre>Spaces

and line breaks

within this element

are shown as typed.

</pre>

<mark> Tag

The <mark> tag is used to present a part of text in one document as marked or highlighted for reference purposes.

<h2>HTML <mark>Marked</mark> Formatting</h2>

<small> Tag

The <small> tag decreases the text font size by one size smaller than a document's base font size (from medium to small, or from x-large to large).

The tag usually contains the items of secondary importance such as copyright notices, side comments, or legal notices.

<p>The interest rate is only 10%*</p>

<small>* per day</small>

 and <s> Tags

The tag specifies a part of the text that was deleted from the document. Browsers display this text as a strikethrough.

The <s> tag defines text that is no longer correct, or relevant. The content of both tags is displayed as strikethrough. However, despite the visual similarity, these tags cannot replace each other.

<ins> and <u> Tags

The <ins> tag defines the text that has been inserted (added) to the document. The content of the tag is displayed as underlined.

The <u> tag specifies text that is stylistically different from normal text, i.e. words or fragments of text that need to be presented differently. This could be misspelled words or proper nouns in Chinese.

<p> She likes violets snowdrops.</p>

<p>She likes violets <ins>snowdrops</ins>.</p>

<sub> and <sup> Tags

The <sub> defines subscript texts. Subscript text is under the baseline of other symbols of the line and has smaller font. The <sup> tag defines superscript, that is set slightly above the normal line of type and is relatively smaller than the rest of the text. The baseline passes through the upper or lower edge of the symbols.

<p>The formula of water is H₂O, and the formula of alcohol is C₂H₅OH </p>

<p>E = mc², where E — kinetic energy, m — mass, c — the speed of light. </p>

<p>,
 and <hr> Tags

The <p> tag defines the paragraph. Browsers automatically add 1em margin before and after each paragraph.

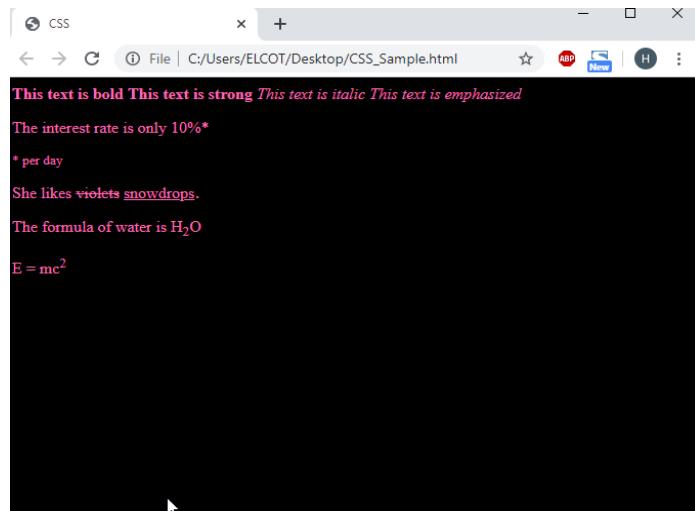
Source

C:\Users\ELCOT\Desktop\CSS_Sample.html - Sublime Text (UNREGISTERED)

```
File Edit Selection Find View Goto Tools Project Preferences Help
CSS_Sample.html x
1 <html>
2 <head>
3 <title>CSS</title>
4 </head>
5 <body bgcolor="lightyellow" text="blue">
6 <b>This text is bold</b>
7 <strong>This text is strong</strong>
8 <i>This text is italic</i>
9 <em>This text is emphasized</em>
10 <p>The interest rate is only 10%*</p>
11 <small>* per day</small>
12 <p>She likes <del>violets</del> <ins>snowdrops</ins>.</p>
13 The formula of water is H<sub>2</sub>O
14 <p>E = mc<sup>2</sup>
15
16 </body>
17 </html>
18
```

Line 7, Column 17 Tab Size: 4 HTML

Output



HTML Colors

There are three ways of how you can change the color of the text in HTML:

- Hex color codes
- HTML color names
- RGB values.

HTML Color Names

To color the text element using an HTML color name, put the name of the color (blue, for ex.) instead of Hex code from the previous step.

```
<p style="color:red;"> This is a text in green</p>
```

Hex Color Codes

A hex color code is a hex triplet, which represents three separate values defining the levels of the component colors. It is specified with a hexadecimal (hex) notation for a mixture of Red, Green, and Blue color values. The lowest value that can be given to one of the light sources is 0 (hex 00). The highest value is 255 (hex FF).

Hex values are written as six-digit numbers, starting with a # sign. Letters used in a hexadecimal digit may be uppercase or lowercase. For example, to specify white color you can write #FFFFFF or #ffffff.

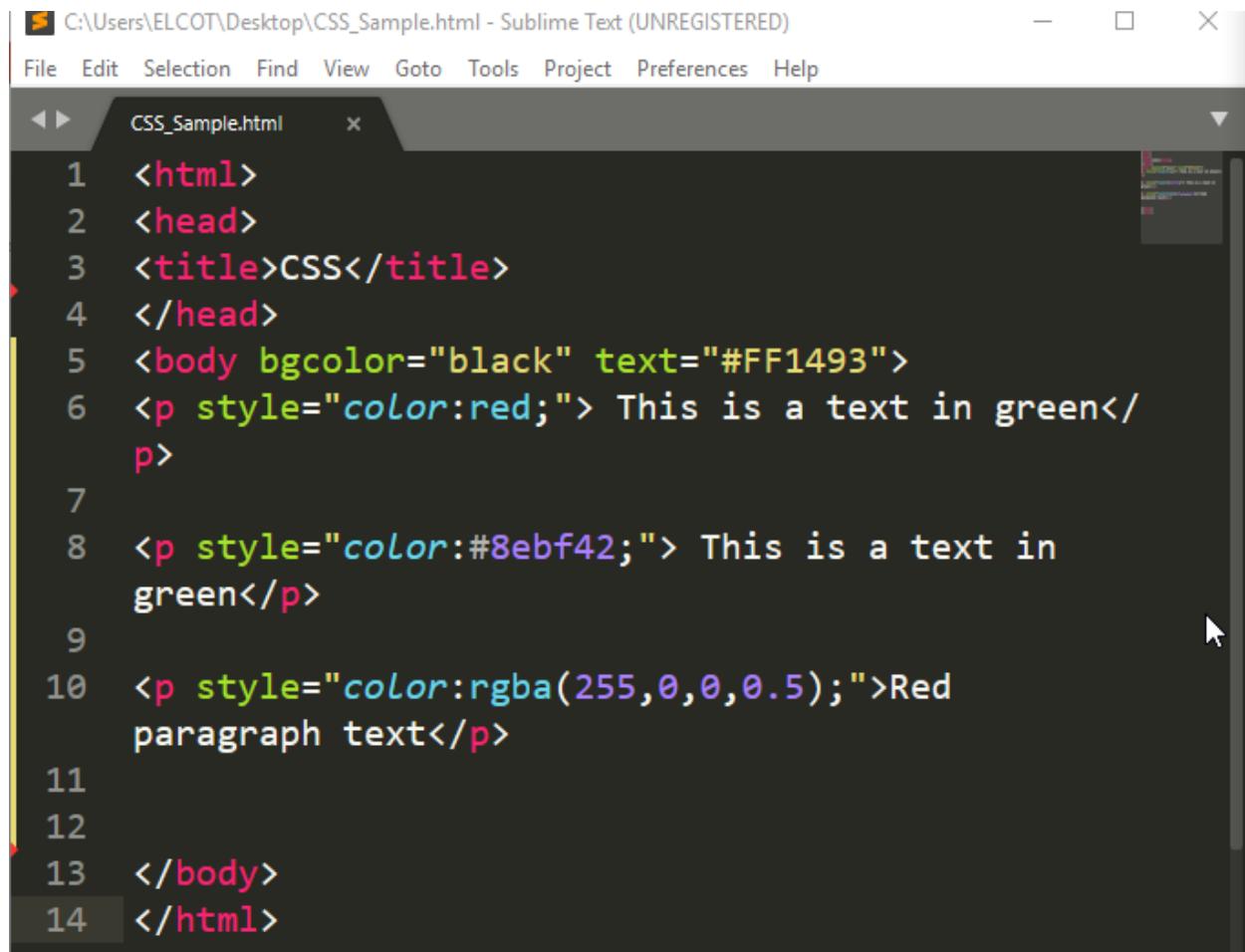
```
<p style="color:#8ebf42;"> This is a text in green</p>
```

RGB Color Values

To add a color to the text element, use the style attribute (where the color property is your RGB value)

```
<p style="color:rgba(255,0,0,0.5);">Red paragraph text</p>
```

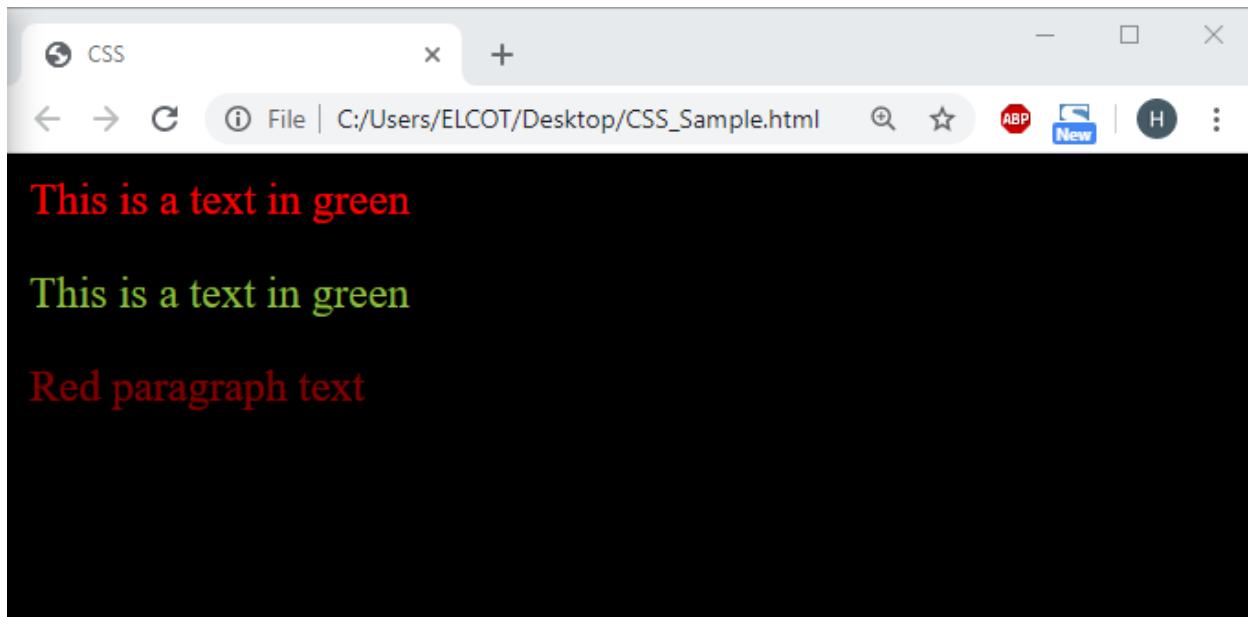
Source



The screenshot shows a Sublime Text window with the title bar "C:\Users\ELCOT\Desktop\CSS_Sample.html - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. A single tab labeled "CSS_Sample.html" is open. The code editor contains the following HTML and CSS:

```
1 <html>
2 <head>
3 <title>CSS</title>
4 </head>
5 <body bgcolor="black" text="#FF1493">
6 <p style="color:red;"> This is a text in green</p>
7
8 <p style="color:#8ebf42;"> This is a text in
green</p>
9
10 <p style="color:rgba(255,0,0,0.5);">Red
paragraph text</p>
11
12
13 </body>
14 </html>
```

Output:



HTML Image

- The `` tag is empty, it contains attributes only, and does not have a closing tag.
- The `src` attribute specifies the URL (web address) of the image:

```

```

Attributes of `` tag

Attributes	Description
vspace	Specifies the amount of space to the top and bottom of the image.
hspace	Specifies the amount of space to the left and right of the image.
alt	Specifies alternate text for an image when image is not found.

src	Indicate the source file.
border	Specifies the thickness of the border.
width and height	Specifies width and height of the image.
align	Use to set horizontal alignment of the image. (left, right)
valign	Used to set vertical alignment of the image. (top, bottom)

Images in Another Folder

If not specified, the browser expects to find the image in the same folder as the web page.

However, it is common to store images in a sub-folder. You must then include the folder name in the src attribute:

```

```

Images on Another Server

Some web sites store their images on image servers.

Actually, you can access images from any web address in the world:

```

```

Animated Images

HTML allows animated GIFs:

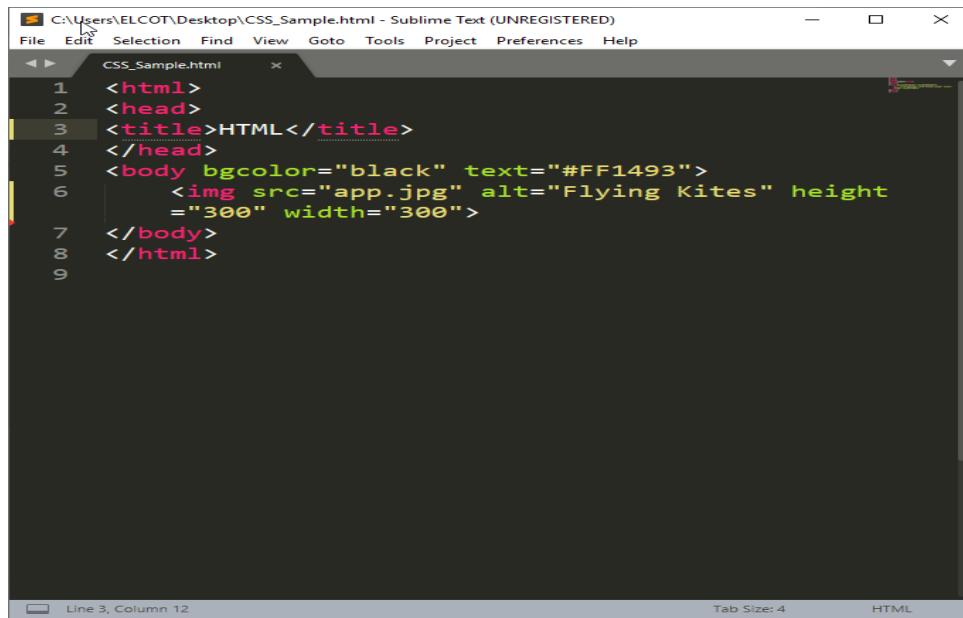
```

```

External Links (aka backlinks or inbound links)	Internal Links
Difficult to Control	Easy, fast and free to create
Pass SEO authority from other sites to your site, increasing your "domain authority"	Pass SEO authority between pages on your site, increasing the "page authority" of specific pages
Appear within the body text, in content	Appear in website navigation, as well as in the content

Image 52:External and internal links
Reference:<https://www.orbitmedia.com/blog/internal-linking/>

Source

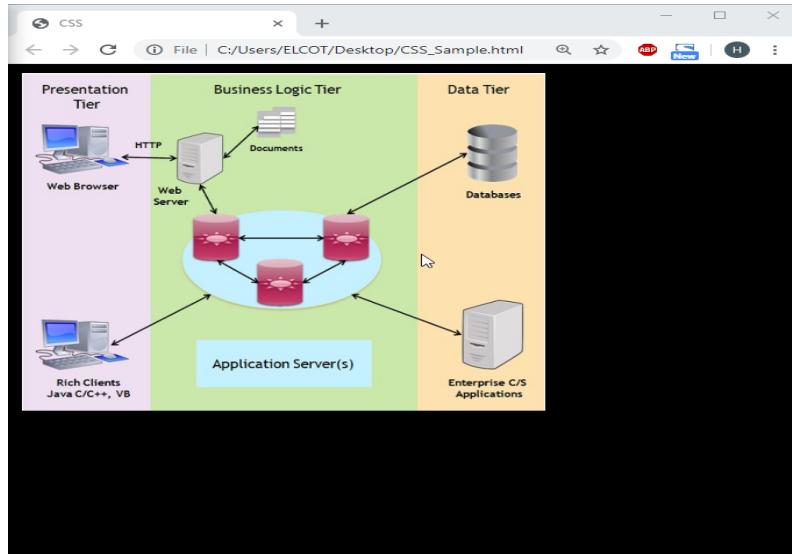


The screenshot shows the Sublime Text editor window with the file 'CSS_Sample.html' open. The code is as follows:

```
1 <html>
2 <head>
3 <title>HTML</title>
4 </head>
5 <body bgcolor="black" text="#FF1493">
6   
7 </body>
8 </html>
```

The status bar at the bottom indicates 'Line 3, Column 12' and 'Tab Size: 4'. The file path 'C:\Users\ELCOT\Desktop\CSS_Sample.html' is shown in the title bar.

Output



Linking

HTML links are hyperlinks.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

The href attribute specifies the destination address of the link.

```
<a href="https://www.w3schools.com/html/html_basic.asp">HTML tutorial</a>
```

ALINK: It indicates the colour of the active hyperlink. An active link is the one on which the mouse button is pressed. e. **VLINK:** It indicates the colour of the hyperlinks after the mouse is clicked on it.

HTML Links - The target Attribute

The target attribute specifies where to open the linked document.

The target attribute can have one of the following values:

-
- _blank - Opens the linked document in a new window or tab
 - _self - Opens the linked document in the same window/tab as it was clicked (default)
 - _parent - Opens the linked document in the parent frame
 - _top - Opens the linked document in the full body of the window

< a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!

Local Links

The example above used an absolute URL (a).

A local link (link to the same web site) is specified with a relative URL (without https://www....).

Absolute URL -full web address

< a href="https://www.w3schools.com/html/html_basic.asp">HTML tutorial

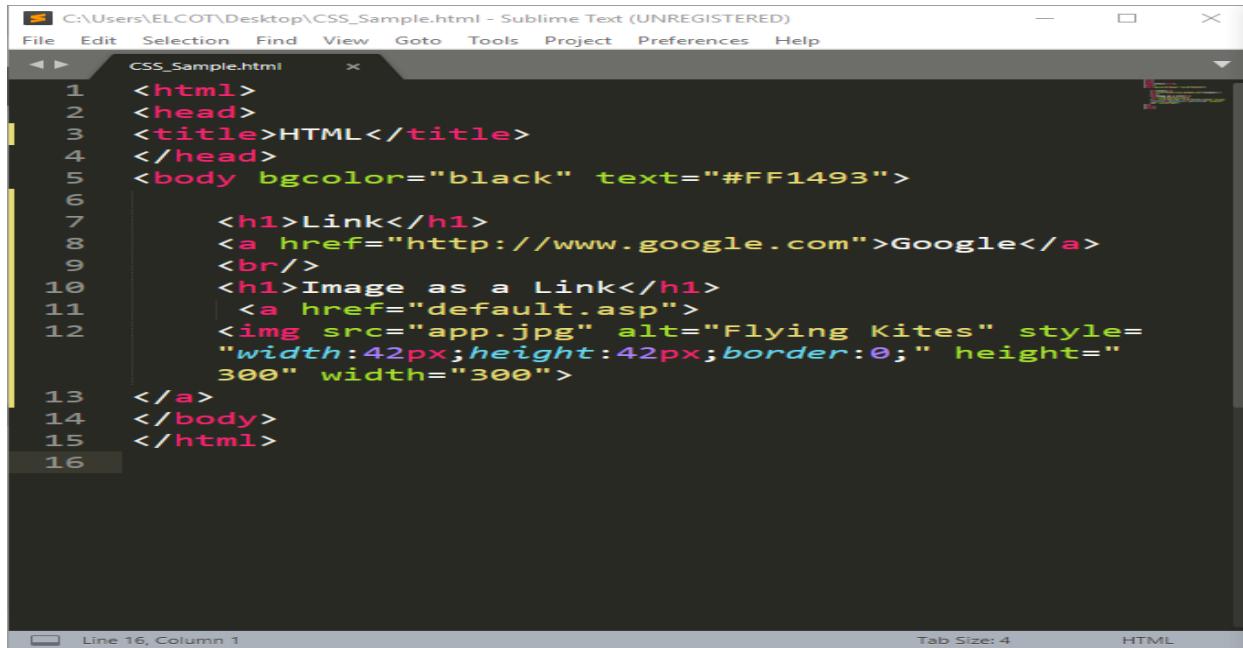
Relative URL (without <https://www....>).

< a href="html_images.asp">HTML Images

HTML Links - Image as a Link

```
<a href="default.asp">  
    
</a>
```

Source



The screenshot shows the Sublime Text editor window with the file 'CSS_Sample.html' open. The code is as follows:

```
1 <html>
2 <head>
3 <title>HTML</title>
4 </head>
5 <body bgcolor="black" text="#FF1493">
6
7     <h1>Link</h1>
8     <a href="http://www.google.com">Google</a>
9     <br/>
10    <h1>Image as a Link</h1>
11    <a href="default.asp">
12        
14    </a>
15 </body>
16 </html>
```

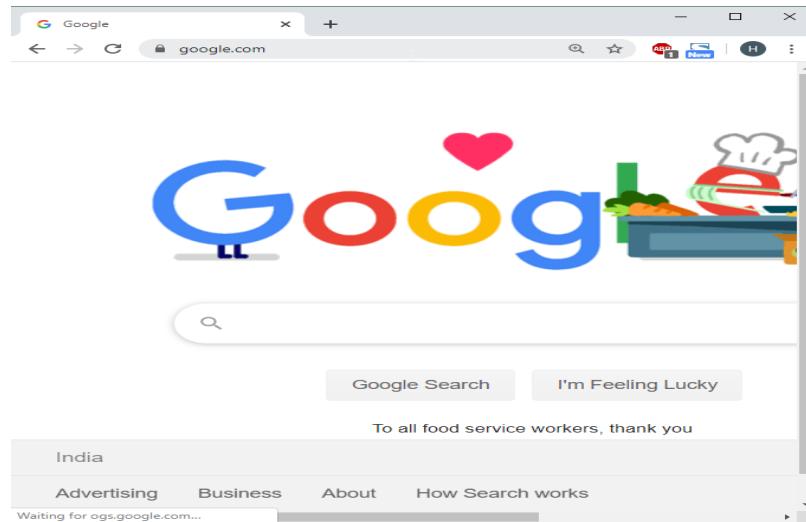
Line 16, Column 1

Tab Size: 4

HTML

Output





Tables

An HTML table is defined with the <table> tag.

Each table row is defined with the <tr> tag.

A table header is defined with the <th> tag.

By default, table headings are bold and centered.

A table data/cell is defined with the <td> tag.

Adding Captions to Tables

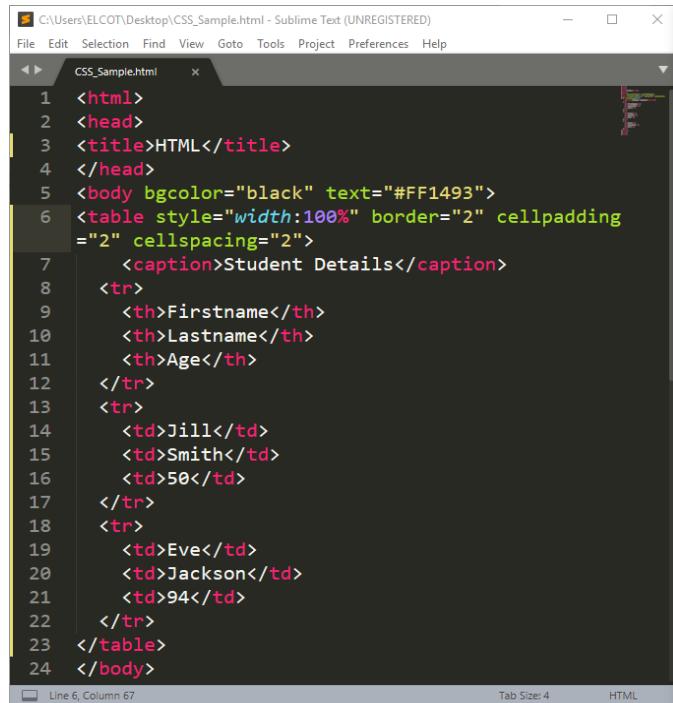
You can use the <caption> element to specify a caption for tables. It should be placed immediately after the opening <table>

```
<caption>Student Details</caption>
```

Cell Spacing is used to set space between different table cells.

CellPadding is used for the space between the edges of the cell and the content of the cell.

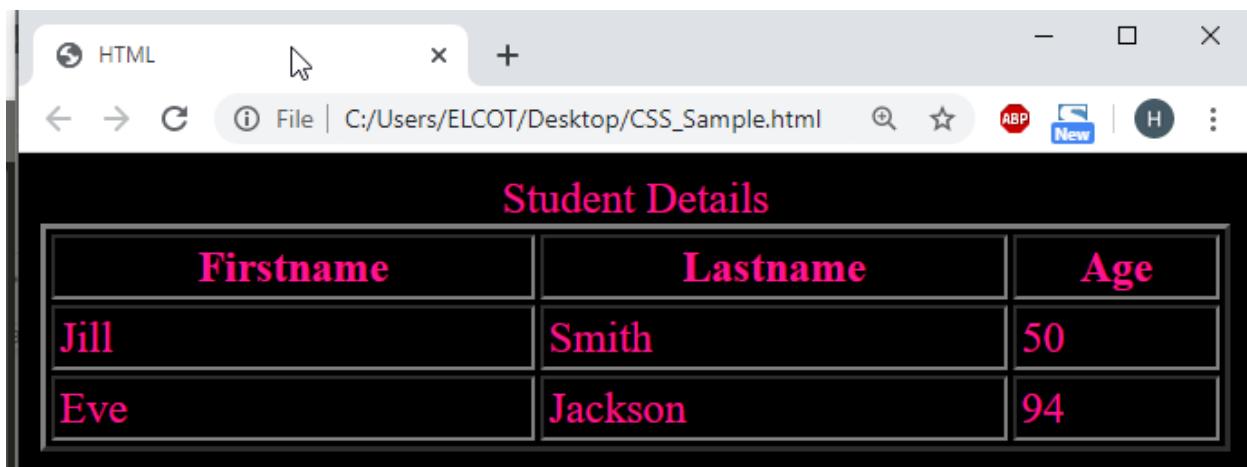
Source



The screenshot shows a Sublime Text window with the file 'CSS_Sample.html' open. The code is as follows:

```
1 <html>
2 <head>
3 <title>HTML</title>
4 </head>
5 <body bgcolor="black" text="#FF1493">
6 <table style="width:100%" border="2" cellpadding="2" cellspacing="2">
7   <caption>Student Details</caption>
8   <tr>
9     <th>Firstname</th>
10    <th>Lastname</th>
11    <th>Age</th>
12   </tr>
13   <tr>
14     <td>Jill</td>
15     <td>Smith</td>
16     <td>50</td>
17   </tr>
18   <tr>
19     <td>Eve</td>
20     <td>Jackson</td>
21     <td>94</td>
22   </tr>
23 </table>
24 </body>
```

Output



The screenshot shows a web browser window displaying the rendered HTML. The title bar says 'HTML'. The page content is as follows:

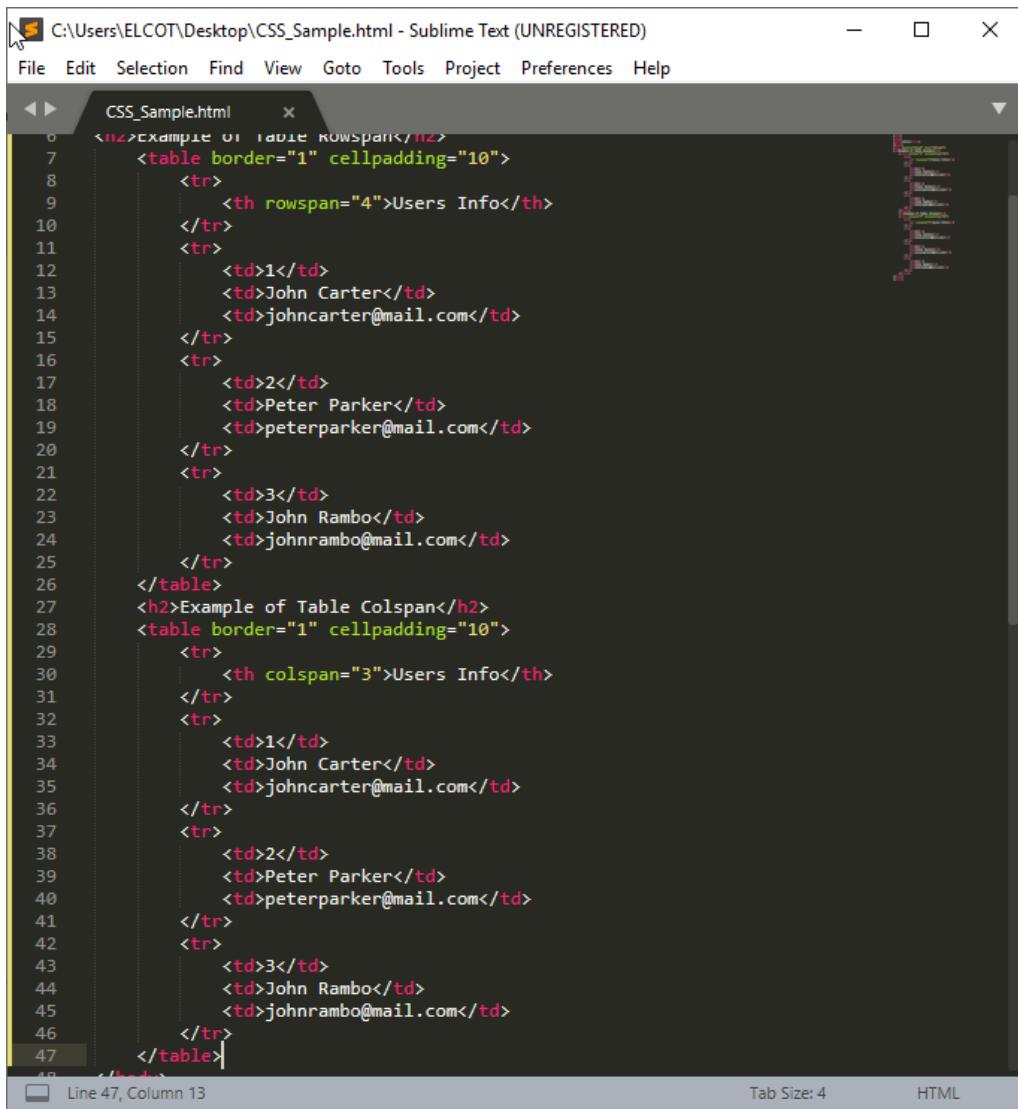
Student Details

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

Rowspan and Colspan

The rowspan and colspan are <td> tag attributes. These are used to specify the number of rows or columns a cell should span. The rowspan attribute is for rows as well as the colspan attribute is for columns.

Source



The screenshot shows a Sublime Text editor window displaying an HTML file named "CSS_Sample.html". The code contains two examples of tables demonstrating rowspan and colspan attributes.

```
<h2>Example of Table Rowspan</h2>
<table border="1" cellpadding="10">
<tr>
<th rowspan="4">Users Info</th>
<tr>
<td>1</td>
<td>John Carter</td>
<td>johncarter@mail.com</td>
</tr>
<tr>
<td>2</td>
<td>Peter Parker</td>
<td>peterparker@mail.com</td>
</tr>
<tr>
<td>3</td>
<td>John Rambo</td>
<td>johnrambo@mail.com</td>
</tr>
</table>
<h2>Example of Table Colspan</h2>
<table border="1" cellpadding="10">
<tr>
<th colspan="3">Users Info</th>
<tr>
<td>1</td>
<td>John Carter</td>
<td>johncarter@mail.com</td>
</tr>
<tr>
<td>2</td>
<td>Peter Parker</td>
<td>peterparker@mail.com</td>
</tr>
<tr>
<td>3</td>
<td>John Rambo</td>
<td>johnrambo@mail.com</td>
</tr>
</table>
```

The status bar at the bottom of the editor indicates "Line 47, Column 13" and "Tab Size: 4".

Output

The screenshot shows a web browser window with the title "HTML". The address bar indicates the file is located at "C:/Users/ELCOT/Desktop/CSS_Sample.html". The page content consists of two sections: "Example of Table Rowspan" and "Example of Table Colspan".

Example of Table Rowspan:

Users Info	1	John Carter	johncarter@mail.com
	2	Peter Parker	peterparker@mail.com
	3	John Rambo	johnrambo@mail.com

Example of Table Colspan:

Users Info		
1	John Carter	johncarter@mail.com
2	Peter Parker	peterparker@mail.com
3	John Rambo	johnrambo@mail.com

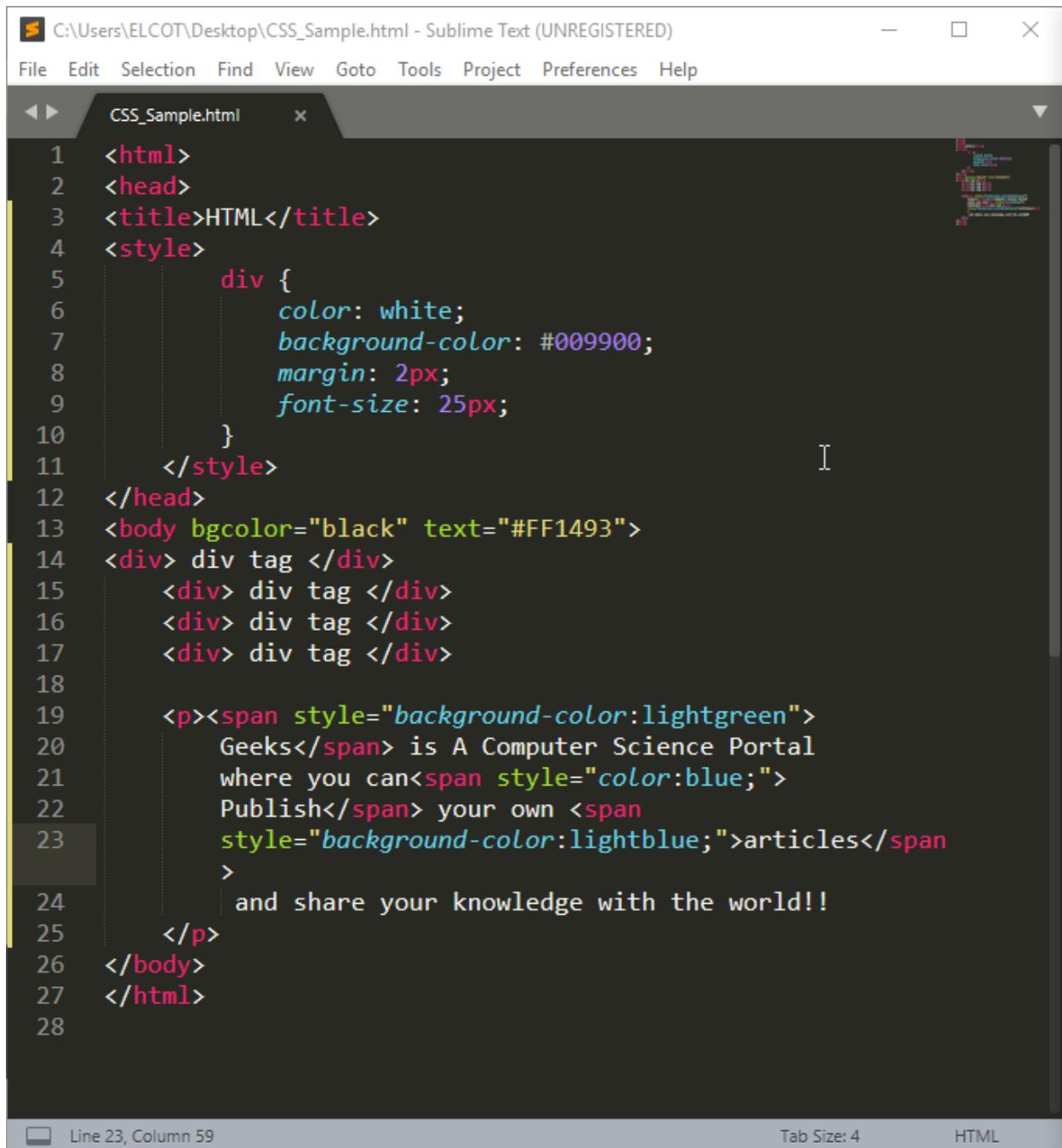
Div and Span

<div> tag is used as a block part of the webpage

 tag is used as an inline part of the webpage

The `div` should be used to wrap sections of a document, while use spans to wrap small portions of text, images, etc. The `<div>` element is used while creating CSS based layouts in html, whereas `` element is used to stylize texts.

Source

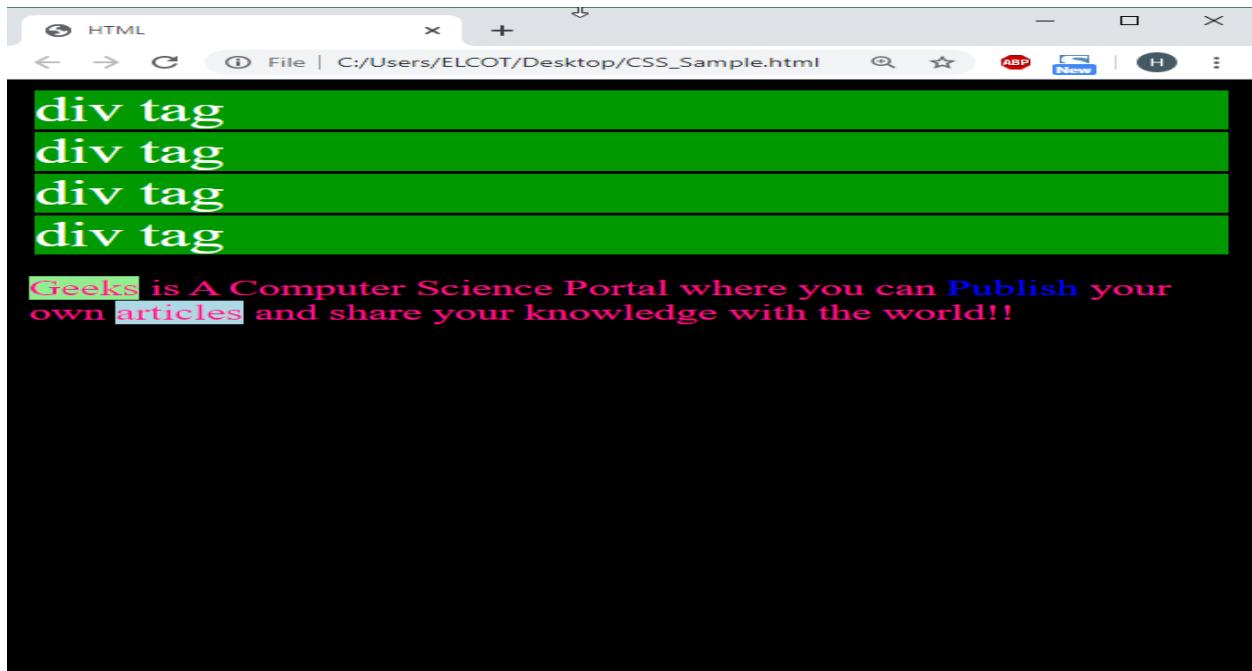


```
C:\Users\ELCOT\Desktop\CSS_Sample.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
CSS_Sample.html x

1 <html>
2 <head>
3 <title>HTML</title>
4 <style>
5     div {
6         color: white;
7         background-color: #009900;
8         margin: 2px;
9         font-size: 25px;
10    }
11   </style>
12 </head>
13 <body bgcolor="black" text="#FF1493">
14 <div> div tag </div>
15 <div> div tag </div>
16 <div> div tag </div>
17 <div> div tag </div>
18
19 <p><span style="background-color:lightgreen">
20     Geeks</span> is A Computer Science Portal
21     where you can<span style="color:blue;">
22     Publish</span> your own <span
23     style="background-color:lightblue;">articles</span
24     >
25     and share your knowledge with the world!!
26   </p>
27 </body>
28 </html>

Line 23, Column 59          Tab Size: 4          HTML
```

Output



<DIV>	
The <div> tag is a block level element.	The tag is an inline element.
It is best to attach it to a section of a web page.	It is best to attach a CSS to a small section of a line in a web page.
It accepts align attribute.	It does not accept align attribute.
This tag should be used to wrap a section, for highlighting that section.	This tag should be used to wrap any specific word that you want to highlight in your webpage.

Image 52:External and internal links
Reference:<https://www.geeksforgeeks.org/difference-between-div-and-span-tag-in-html/>

Lists

HTML List Tags are used to specify information in the form of list.

HTML Lists are very useful to group related information together. Often List items looks well-structured and they are easy to read for users. A list can contain one or more list elements.HTML

Type of Lists

- Unordered HTML List
- Ordered HTML List
- Description Lists
- Nested HTML Lists

Ordered lists

Ordered list is used to list related items in a numbered or other specific order. This is useful when you want to show counts of items in some way.

Ordered list is created using the HTML `` tag. Each item in the list start with the `` tag

Example of Ordered List

```
<ul>  
<li>Red</li>  
<li>Green</li>  
<li>Blue</li>  
</ul>
```

Above example will list colors items with numbers by default. There are different list style available for ordered lists such as numbers, letters etc.

Ordered List Style Type Attribute

There are two attributes can be used to customize ordered list, they are

- (1) Type - changing numbering style
- (2) Start - changing numbering order.

Type – is used to change the number style. The default number style is standard Arabic numerals (1,2,3,.....).

Start – is used to specify the number of letters with which start the list. The default starting point is 1. The value of the start attribute should be a decimal number, regardless of the numbering style being used

HTML Ordered List Style Type Attribute		
List Style Type	Description	Example and Syntax
Numbers	Starts a list using numbers (default)	<ol type="1">
Uppercase letters	Starts a list using uppercase letters	<ol type="A">
Lowercase letters	Starts a list using lowercase letters	<ol type="a">
Uppercase roman numbers	Starts a list using uppercase roman numbers	<ol type="I">
Lowercase roman numbers	Starts a list using lowercase roman numbers	<ol type="i">

List Style Type	Description	Example and Syntax
Numbers	Starts a list using numbers (default)	<ol type="1">
Uppercase letters	Starts a list using uppercase letters	<ol type="A">
Lowercase letters	Starts a list using lowercase letters	<ol type="a">
Uppercase roman numbers	Starts a list using uppercase roman numbers	<ol type="I">
Lowercase roman numbers	Starts a list using lowercase roman numbers	<ol type="i">

<ol type="1">

 Banana

 Apple

 Grapes

Unordered lists

Unordered lists are used to list sets of items when they have no special order or sequence. It is also called a bulleted list.

Unordered list is created using the HTML tag. Each item in the list start with the tag

Unordered List Style Type Attribute

Like an ordered list, type attribute is used to customize bullet style for the list of elements.

By default, a solid circle is used as bullets.

Type value : Numbering style

Disc : A solid circle

Square : A solid square

Circle : An unfilled circle

HTML List Types		
List Style Type	Description	Example & Syntax
disc	Starts a list using discs type bullets (default)	<ul type="disc">
circle	Starts a list using circle type bullets	<ul type="circle">
square	Starts a list using square type bullets	<ul type="square">
none	Starts a list without bullets	<ul type="type:none">

<ul type="disc">

```
<li>Apple</li>
<li>Banana</li>
<li>Mango</li>
</ul>
```

Definition Lists

Definition list is different from the other two types of list. No bullet or number is provided for the list items. In this list type, the list element has two parts.

- (1) A definition term
- (2) The definition description

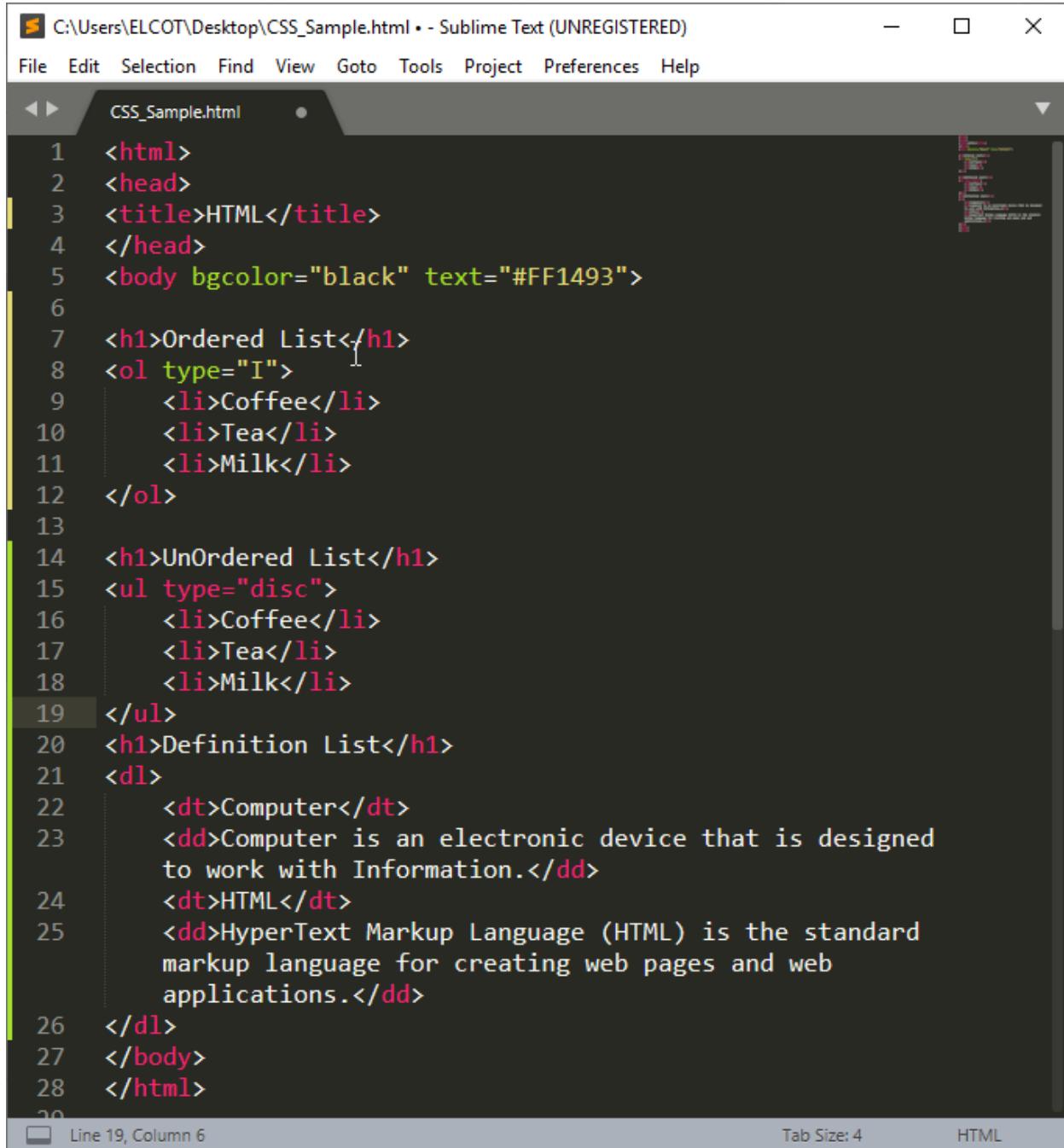
Definition list is surrounded within <DL> </DL> tags.

Definition term is presented in between <DT> </DT> tag and

Definition description should be surrounded within <DD> </DD> tag.

```
<dl>
<dt>Computer</dt>
<dd>Computer is an electronic device that is designed to work with Information.</dd>
</dl>
```

Source

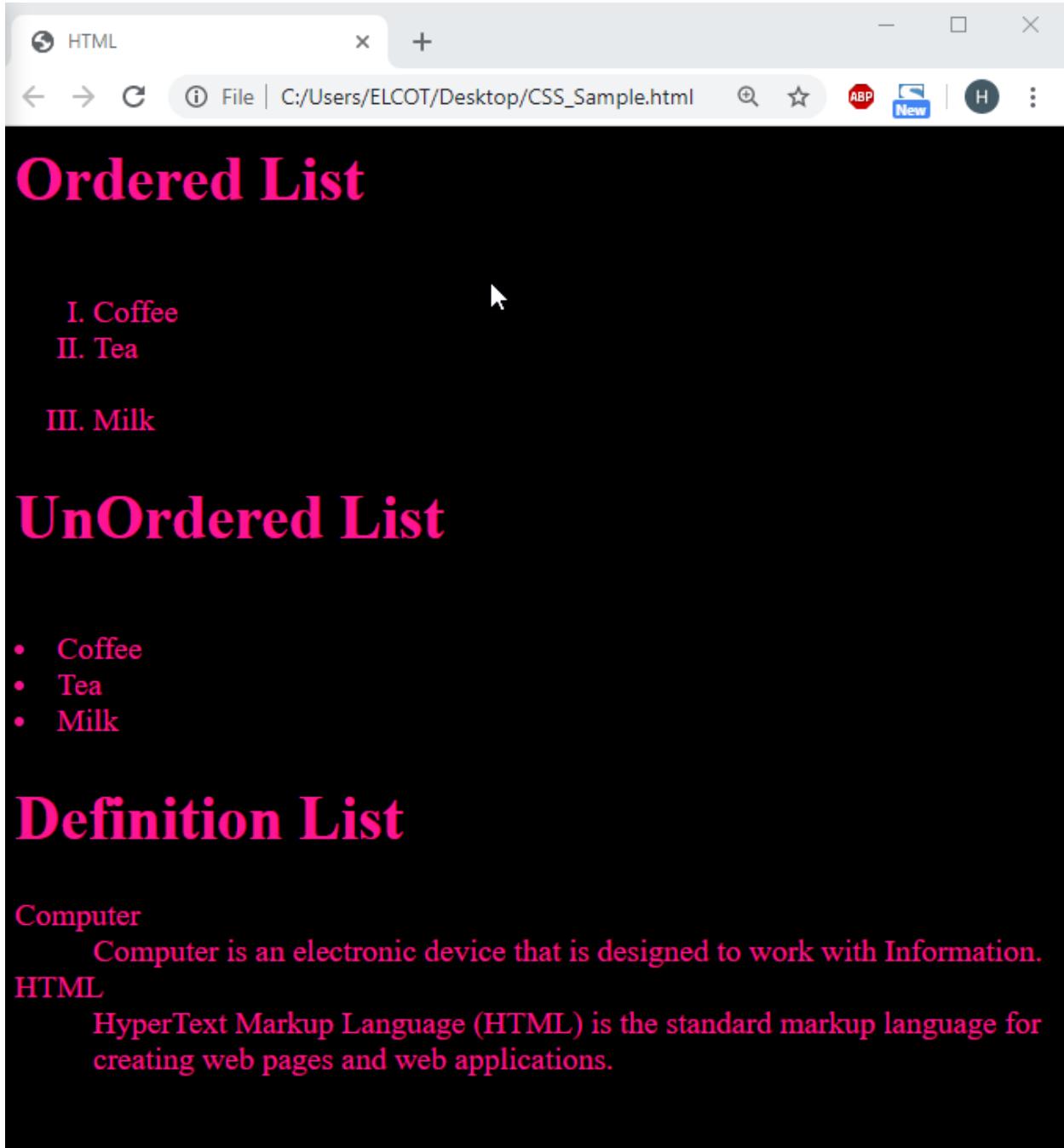


The screenshot shows a Sublime Text editor window with the following details:

- Title Bar:** C:\Users\ELCOT\Desktop\CSS_Sample.html - Sublime Text (UNREGISTERED)
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- File List:** Shows a single file named "CSS_Sample.html".
- Code Area:** Displays the following HTML code with syntax highlighting:

```
1 <html>
2 <head>
3 <title>HTML</title>
4 </head>
5 <body bgcolor="black" text="#FF1493">
6
7 <h1>Ordered List</h1>
8 <ol type="I">
9   <li>Coffee</li>
10  <li>Tea</li>
11  <li>Milk</li>
12 </ol>
13
14 <h1>UnOrdered List</h1>
15 <ul type="disc">
16   <li>Coffee</li>
17   <li>Tea</li>
18   <li>Milk</li>
19 </ul>
20 <h1>Definition List</h1>
21 <dl>
22   <dt>Computer</dt>
23   <dd>Computer is an electronic device that is designed
24     to work with Information.</dd>
25   <dt>HTML</dt>
26   <dd>HyperText Markup Language (HTML) is the standard
27     markup language for creating web pages and web
28     applications.</dd>
29 </dl>
30 </body>
31 </html>
```
- Status Bar:** Line 19, Column 6 | Tab Size: 4 | HTML

Output



The screenshot shows a web browser window with the title "HTML". The address bar indicates the file is located at "C:/Users/ELCOT/Desktop/CSS_Sample.html". The page content is as follows:

Ordered List

- I. Coffee
- II. Tea

III. Milk

UnOrdered List

- Coffee
- Tea
- Milk

Definition List

Computer
Computer is an electronic device that is designed to work with Information.

HTML
HyperText Markup Language (HTML) is the standard markup language for creating web pages and web applications.

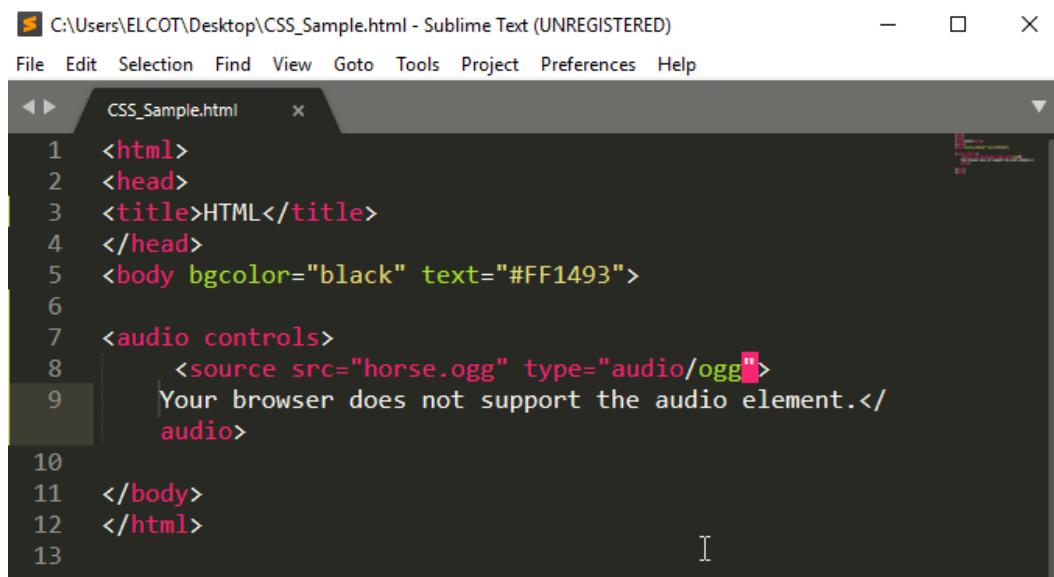
Audio and Videos

<audio> element specifies a standard way to embed audio in a web page.

HTML audio tag is used to define sounds such as music and other audio clips.

- mp3
- wav
- ogg

Source

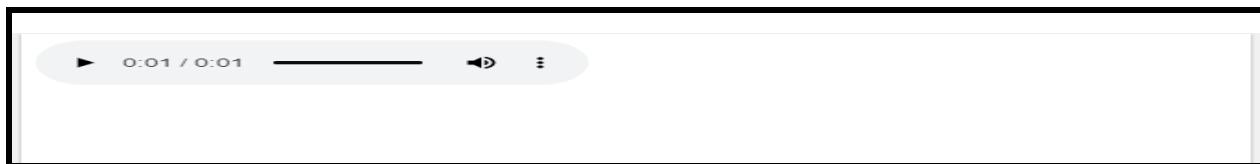


```
C:\Users\ELCOT\Desktop\CSS_Sample.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

CSS_Sample.html x

1 <html>
2 <head>
3 <title>HTML</title>
4 </head>
5 <body bgcolor="black" text="#FF1493">
6
7 <audio controls>
8   <source src="horse.ogg" type="audio/ogg">
9   Your browser does not support the audio element.</
10  audio>
11 </body>
12 </html>
13
```

Output



Video

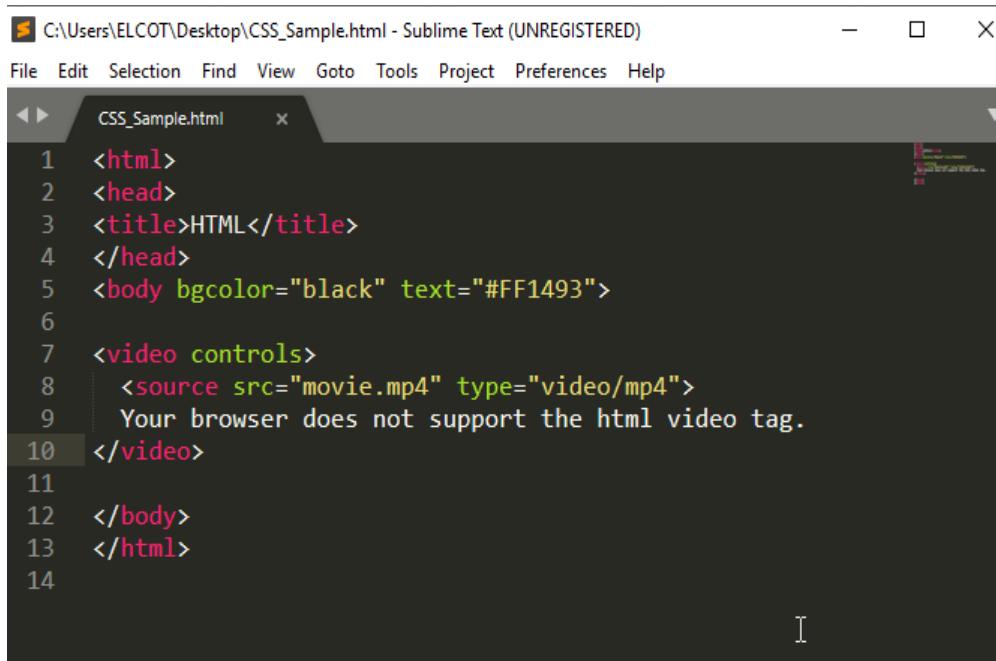
<video> element specifies a standard way to embed a video in a web page

The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page.

Currently, there are three video formats supported for HTML video tag:

- mp4
- webM
- ogg

Source



C:\Users\ELCOT\Desktop\CSS_Sample.html - Sublime Text (UNREGISTERED)

```
File Edit Selection Find View Goto Tools Project Preferences Help
CSS_Sample.html ×
1 <html>
2 <head>
3 <title>HTML</title>
4 </head>
5 <body bgcolor="black" text="#FF1493">
6
7 <video controls>
8   <source src="movie.mp4" type="video/mp4">
9   Your browser does not support the html video tag.
10 </video>
11
12 </body>
13 </html>
14
```

Output



HTML Forms and Input

An HTML form is a section of a document which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc.

HTML Form Tags

HTML <form> element

The HTML <form> element provide a document section to take input from user. It provides various interactive controls for submitting information to web server such as text field, text area, password field, etc.

```
<form>  
//Form elements  
</form>
```

^

Tag	Description
<form>	It defines an HTML form to enter inputs by the user side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.
<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

HTML <input> element

The HTML <input> element is fundamental form element. It is used to create form fields, to take input from user. We can apply different input filed to gather different information from user.

```
<body>  
<form>  
  Enter your name <br>  
  <input type="text" name="username">  
</form>  
</body>
```

Enter your name

Label Tag in Form

It is considered better to have label in form. As it makes the code parser/browser/user friendly.

If you click on the label tag, it will focus on the text control. To do so, you need to have for attribute in label tag that must be same as id attribute of input tag.

```
<form>
  <label for="firstname">First Name: </label> <br/>
    <input type="text" id="firstname" name="firstname"/> <br/>
  <label for="lastname">Last Name: </label>
    <input type="text" id="lastname" name="lastname"/> <br/>
</form>
```

First Name:

Last Name:

Radio Button Control

The radio button is used to select one option from multiple options. It is used for selection of gender, quiz questions etc.

If you use one name for all the radio buttons, only one radio button can be selected at a time.

Using radio buttons for multiple options, you can only choose a single option at a time.

```
<form>
  <label for="gender">Gender: </label>
  <input type="radio" id="gender" name="gender" value="male"/>Male
  <input type="radio" id="gender" name="gender" value="female"/>Female <br/>
</form>
```



Checkbox Control

The checkbox control is used to check multiple options from given checkboxes.

```
<form>
Hobby:<br>
  <input type="checkbox" id="cricket" name="cricket" value="cricket"/>
  <label for="cricket">Cricket</label> <br>
  <input type="checkbox" id="football" name="football" value="football"/>
  <label for="football">Football</label> <br>
  <input type="checkbox" id="hockey" name="hockey" value="hockey"/>
  <label for="hockey">Hockey</label>
</form>
```

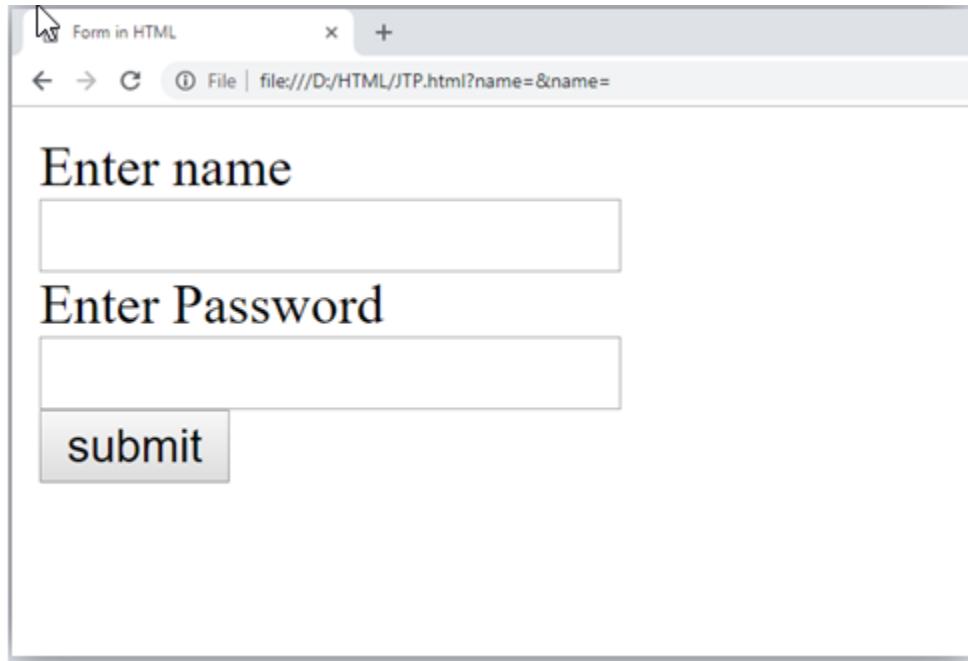
Hobby:

- Cricket
- Football
- Hockey

Submit button control

HTML <input type="submit"> are used to add a submit button on web page. When user clicks on submit button, then form get submit to the server

```
<form>
  <label for="name">Enter name</label><br>
  <input type="text" id="name" name="name"><br>
  <label for="pass">Enter Password</label><br>
  <input type="Password" id="pass" name="pass"><br>
  <input type="submit" value="submit">
</form>
```



HTML Form Example

```
<form>
  <fieldset>
    <legend>User personal information</legend>
    <label>Enter your full name</label><br>
    <input type="text" name="name"><br>
    <label>Enter your email</label><br>
    <input type="email" name="email"><br>
    <label>Enter your password</label><br>
    <input type="password" name="pass"><br>
    <label>confirm your password</label><br>
    <input type="password" name="pass"><br>
    <br><label>Enter your gender</label><br>
    <input type="radio" id="gender" name="gender" value="male"/>Male <br>
    <input type="radio" id="gender" name="gender" value="female"/>Female <br/>
    <input type="radio" id="gender" name="gender" value="others"/>others <br/>
    <br>Enter your Address:<br>
    <textarea></textarea><br>
    <input type="submit" value="sign-up">
  </fieldset>
</form>
```

Output

Registration form

User personal information

Enter your full name

Enter your email

Enter your password

confirm your password

Enter your gender
 Male
 Female
 others

Enter your Address:



Markup Validation Service

An HTML validator is a quality assurance program used to check Hypertext Markup Language (HTML) markup elements for syntax errors.

A validator can be a useful tool for an HTML user who receives data electronically from a variety of input sources.

Validating a web page is a process of ensuring that it conforms to the norms or standards defined by the World Wide Web Consortium (W3C) which is the authority to maintain HTML standards.

There are several specific reasons for validating a web page, some of them are:

It helps to create web pages that are cross-browser, cross-platform compatible. It is also likely to be compatible with the future version of web browsers and web standards.

Standards compliant web pages increase the search engine spiders and crawlers visibility, as a result your web pages will more likely appear in search results. It will reduce unexpected errors and make your web pages more accessible to the visitor.



Image 53:W3C Validation
Reference:<https://support.modernretail.com/hc/en-us/articles/201127998-W3C-Markup-Validation-Service>

HTML5

Introduction to HTML5

HTML5 is the latest standard for browsers to display and interact with web pages.

The latest versions of Apple Safari, Google Chrome, Mozilla Firefox, and Opera all support many HTML5 features and Internet Explorer 9.0 will also have support for some HTML5 functionality.

The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

Features

- New Semantic Elements – These are like <header>, <footer>, and <section>
- Forms 2.0 – Improvements to HTML web forms where new attributes have been introduced for <input> tag.
- Persistent Local Storage – To achieve without resorting to third-party plugins.
- WebSocket – A next-generation bidirectional communication technology for web applications.
- Server-Sent Events – HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- Canvas – This supports a two-dimensional drawing surface that you can program with JavaScript
- Audio & Video – You can embed audio or video on your webpages without resorting to third-party plugins.
- Geolocation – Now visitors can choose to share their physical location with your web application.
- Microdata – This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
- Drag and drop – Drag and drop the items from one location to another location on the same webpage.



Image 54:HTML

Reference:<https://www.markupbox.com/blog/wp-content/uploads/2017/02/html-benefits1.png>

Page Layout Semantic Elements

A semantic element clearly describes its meaning to both the browser and the developer.

The following is the list of semantic page elements with a brief definition of each. Be sure to find an HTML5 reference you can trust for a complete list of valid attributes and child elements for each semantic element.

Block elements:

- section - a generic page division used to help break up a larger block of content
- article - content that is originally from an outside source, sometimes added dynamically on page load by using an aggregator script
- header and footer -blocks that appear at the top and bottom of each page
- hgroup - a grouping of multiple related headers, such as a title and subtitle
- menu - a list of commands (see the command element), and attributes that specify the menu's behavior
- nav -identifies a block that's strictly for Web site navigation, typically some unordered list of links to other pages on the site

-
- address - includes contact information for the author of the content within a block, such as an article, section or entire page body
 - aside - indicates that the content should be treated as a sidebar

Inline elements:

- summary and details - toggle between a teaser/summary and full details for the same content
- figure and figcaption - elements used to apply common behavior to images, no matter what media elements (img, svg or canvas) are used to include them
- time - text representing a calendar date, clock time or both, formatted so that the browser can adjust for time zone differences if necessary
- command - a label and a behavior associated with that label when you use the keyboard or mouse to interact with it, typically used inside a menu block
- dfn - a term that's being defined within the content
- wbr - a tag indicating an acceptable place to break text within a word when it wraps across multiple lines

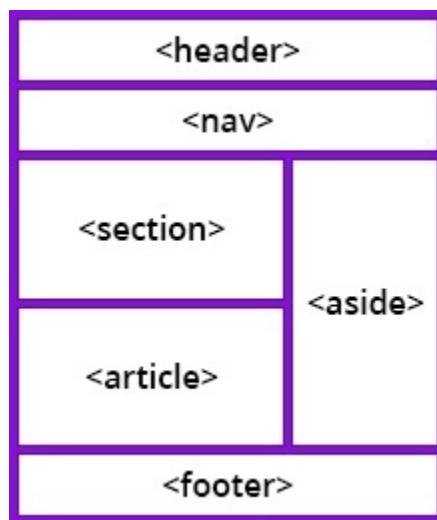


Image 55:Semantic Elements
Reference:<https://www.bitdegree.org/learn/html5-semantic-tags>

Source

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using the section tag</title>
  </head>
  <body>
    <section>
      <h1>Hypertext markup language HTML</h1>
      <p>HTML is the standard markup language for creating web pages and web applications. Browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.</p>
    </section>
    <section>
      <h1>CSS</h1>
      <p>A formal language that is used as a description zone, formatting the appearance of a web page written with the help of markup languages HTML and XHTML but it can be applied to any XML-document, for example, to SVG or XUL.</p>
    </section>
  </body>
</html>
```

Output

Hypertext markup language HTML

HTML is the standard markup language for creating web pages and web applications. Browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

CSS

A formal language that is used as a description zone, formatting the appearance of a web page written with the help of markup languages HTML and XHTML but it can be applied to any XML-document, for example, to SVG or XUL.

Page Layout

Page layout is the part of graphic design that deals with the arrangement of visual elements on a page.

Source

```
5   <body bgcolor="black" text="#FF1493">
6   <header>
7       <nav>
8           <ul>
9               <li>Your menu</li>
10          </ul>
11      </nav>
12  </header>
13
14  <section>
15
16      <article>
17          <header>
18              <h2>Article title</h2>
19              <p>Posted on <time datetime="2009-09-04T16:31:24+02:00">
20                  September 4th 2009</time> by <a href="#">Writer</a> - <a href="#"
21                  #comments">6 comments</a></p>
22          </header>
23          <p>Pellentesque habitant morbi tristique senectus et netus et
24              malesuada fames ac turpis egestas.</p>
25      </article>
26
27      <article>
28          <header>
29              <h2>Article title</h2>
30              <p>Posted on <time datetime="2009-09-04T16:31:24+02:00">
31                  September 4th 2009</time> by <a href="#">Writer</a> - <a href="#"
32                  #comments">6 comments</a></p>
33          </header>
34          <p>Pellentesque habitant morbi tristique senectus et netus et
35              malesuada fames ac turpis egestas.</p>
36      </article>
37
38  </section>
39
40  <aside>
41      <h2>About section</h2>
42      <p>Donec eu libero sit amet quam egestas semper. Aenean ultricies mi
43          vitae est. Mauris placerat eleifend leo.</p>
44  </aside>
45
46  <footer>
47      <p>Copyright 2009 Your name</p>
48  </footer>
```

Output

- Your menu

Article title

Posted on September 4th 2009 by [Writer](#) - [6 comments](#)

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.



Article title

Posted on September 4th 2009 by [Writer](#) - [6 comments](#)

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

About section

Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Copyright 2009 Your name

HTML5 Web Forms

The evolution of HTML5 is to make a better World Wide Web. Writing less code is good (especially javascript), at the end of the day, those repetitive works of web developers should be taken over by web browsers. After all, this is the main reason man invented the machine.

What you can see below is a list of new input attributes and input types, we shall go through each and every one.

New Input attributes <input type="text" "new input attribute"/>

Placeholder

Autofocus

Required

DataList

For years, We are only given a handful of "Type"s to be used in Input element, such as "text", "radio", "checkbox", "password", "file" and "submit", In HTML5, we will be having more fun with the new kids on the block.

New input types <input type="new input type"/>

Search

Email, URL and Phone

Range as Slider

Number as Spinner

Date and Times

Color picker

```
CSS_Sample.html
1 <form action="#" class="example">
2   <ul>
3     <li>
4       <label for="name">Name</label>
5       <input type="text" class="classed" required placeholder=" Name" id="name"/>
6     </li>
7     <li>
8       <label for="email">Email</label>
9       <input type="email" class="classed" multiple placeholder="email addresses"
10      required id="email"/>
11    </li>
12    <li>
13      <label for="dob">Birthday</label>
14      <input type="date" class="classed" required placeholder="YYYY-MM-DD" id="dob"/>
15    </li>
16    <li>
17      <label for="age">Age</label>
18      <input type="number" class="classed" required min="21" max="105" id="age"/>
19    </li>
20    <li>
21      <label for="phone">Phone</label>
22      <input type="tel" class="classed" pattern="\d{3}-\d{3}-\d{4}" placeholder="XXX-XXX-XXXX" id="phone"/>
23    </li>
24    <li>
25      <label for="range">Satisfaction</label>
26      <input type="range" id="range" min="0" max="10" step="1" list="rangelist" />
27    </li>
28    <li>
29      <label for="tweet">Twitter Handle</label>
30      <input type="text" class="classed" pattern="@\w{1,18}" placeholder="@twitter_handle" required id="tweet"/>
31    </li>
32    <li>
33      <input type="submit" value="submit">
34    </li>
35  </ul>
36  <datalist id="rangelist">
37    <option value=7>7</option>
38    <option value=8>8</option>
39    <option value=9>9</option>
40    <option value=10>10</option>
41  </datalist>
</form>
```

HTML5 Form

Name	<input type="text" value="Name"/>
Email	<input type="text" value="email addresses"/>
Birthday	<input type="text" value="dd-mm-yyyy"/>
Age	<input type="text"/>
Phone	<input type="text" value="XXX-XXX-XXXX"/>
Satisfaction	<input type="range" value="5"/>
Twitter	<input type="text" value="@twitter"/>
<input type="button" value="submit"/>	

SVG

SVG defines vector-based graphics in XML format.

SVG stands for Scalable Vector Graphics

Every element and every attribute in SVG files can be animated

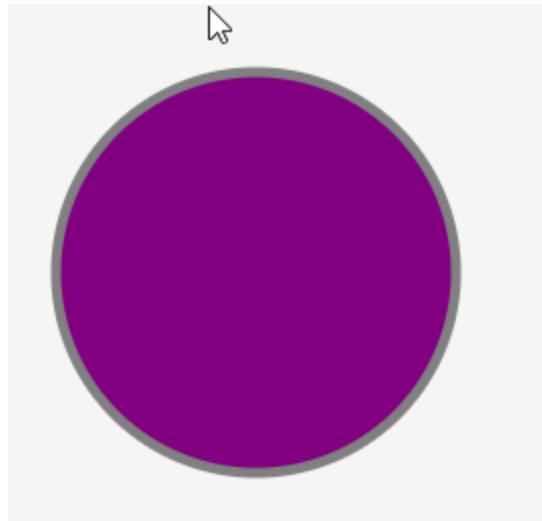
<circle> element

The SVG <circle> element creates circles, based on a center point and a radius. The coordinates of the circle's center are specified by the "cx" and "cy" attributes. And the radius of the circle is specified by the "r" attribute.

Source

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
  </head>
  <body>
    <svg height="300" width="300">
      <circle cx="150" cy="150" r="100" stroke="grey" stroke-width="5"
fill="purple" />
      Sorry, inline SVG isn't supported by your browser.
    </svg>
  </body>
</html>
```

Output



HTML5 Media (Video & Audio)

HTML Plug-ins

Plug-ins can be added to web pages with the `<object>` tag or the `<embed>` tag.

Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

```
<!DOCTYPE html>
<html>
<body>
<embed width="400" height="50" src="bookmark.swf">
</body>
</html>
```



Don't Forget to Bookmark This Site is Si

YouTube

The easiest way to play videos in HTML, is to use YouTube.

YouTube Video Id

YouTube will display an id (like tgbNymZ7vqY), when you save (or play) a video.

You can use this id, and refer to your video in the HTML code.

Playing a YouTube Video in HTML

To play your video on a web page, do the following:

- Upload the video to YouTube

- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and height attributes to specify the dimension of the player
- Add any other parameters to the URL (see below)

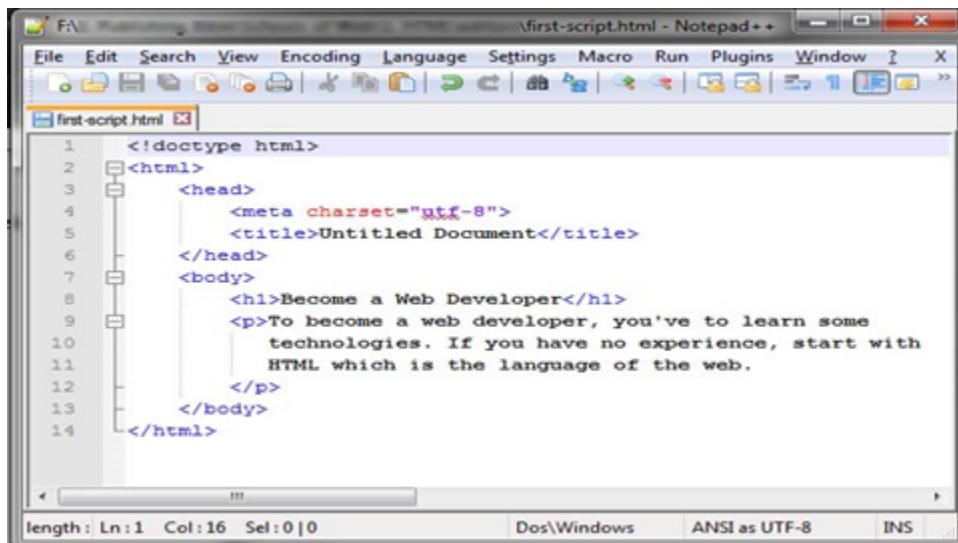
```
<!DOCTYPE html>
<html>
<body>
<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
</body>
</html>
```



Different editors used for Web Page Developing

HTML Editors

HTML text editors are used to create and modify web pages. HTML codes can be written in any text editors including the notepad. One just needs to write HTML in any text editor and save the file with an extension ".html".



The screenshot shows the Notepad++ application window with the title bar 'first-script.html - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Plugins, Window, and Help. The toolbar has various icons for file operations like Open, Save, Find, and Print. The main editor area displays the following HTML code:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Untitled Document</title>
  </head>
  <body>
    <h1>Become a Web Developer</h1>
    <p>To become a web developer, you've to learn some technologies. If you have no experience, start with HTML which is the language of the web.</p>
  </body>
</html>
```

At the bottom of the editor, status bars show 'length: Ln:1 Col:16 Sel:0|0', 'Dos\Windows', 'ANSI as UTF-8', and 'INS'.

Common features of HTML Code Editors

Text editors commonly used for HTML typically include either built-in functions or integration with external tools for such tasks as version control, link-checking and validation, code cleanup and formatting, spell-checking, uploading by FTP or WebDAV, and structuring as a project.

- Auto-completion.
- Adding a library for HTML entities.

-
- With the help of Site Explorer, you can view the files in a hierarchical pattern.
 - Some editors have built-in FTP to upload the files faster.
 - Advanced HTML editors provide support for other languages like CSS and JavaScript.
 - highlighting syntax errors

Key Features of an HTML Editor
Interactive Text, HTML & Source Code Editor
Cleaving of Messy Code
Word to HTML Conversion
Find and Replace Tools for Texts Replacements
Table to DIV Conversions

Image 56:Key Features
Reference:<https://www.goodfirms.co/blog/best-free-open-source-html-editors-software>

Different editors used for Web Page Developing

Some of the Popular Html Editors

- Phase 5 HTML Editor
- Notepad ++
- Sublime Text
- jEdit HTML Editor
- AdobeBrackets
- SynWrite Editor

- Visualcode Editor

Phase 5 HTML Editor

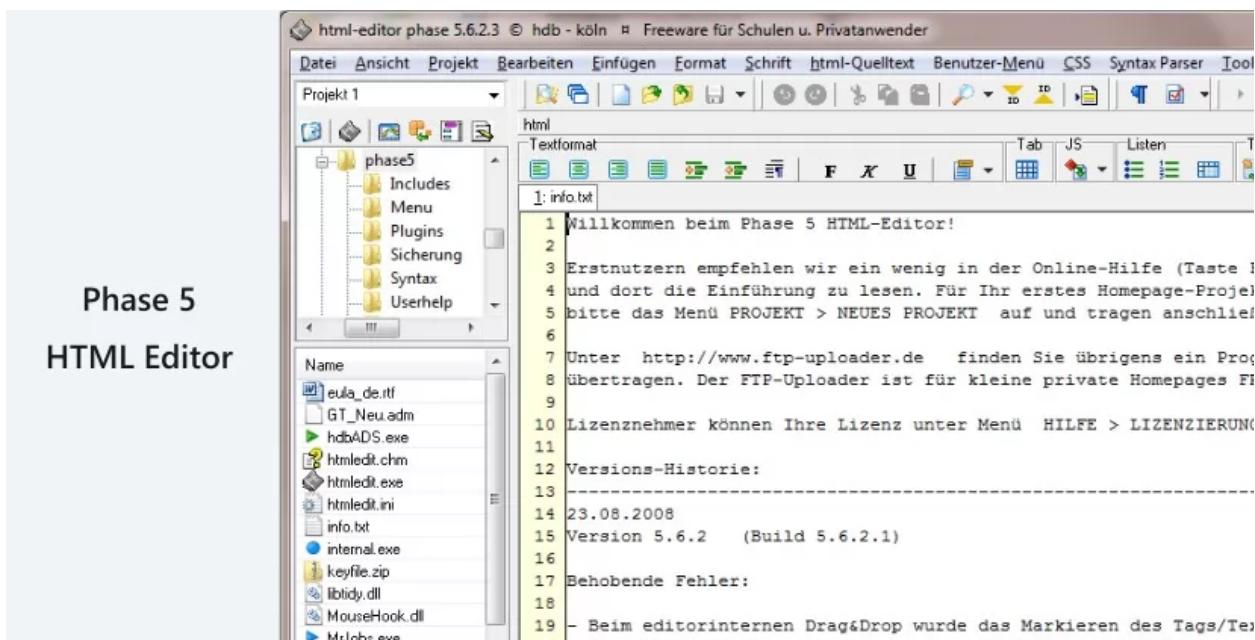


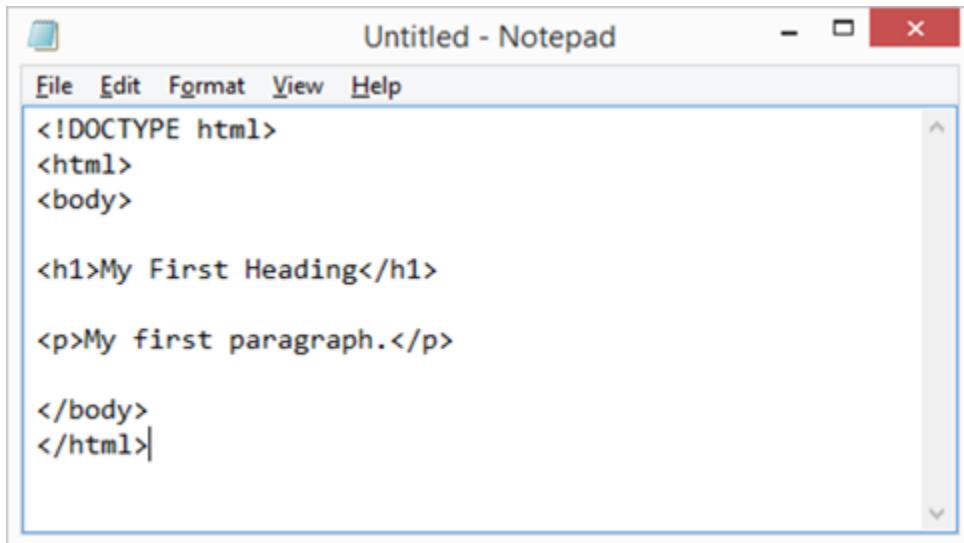
Image 57:Phase5 Editor
Reference:<https://blog.templatetoaster.com/best-free-html-editors/>

Features

- Phase 5 is an impressive German HTML editor.
- It is freeware but only for Schools and Home users. If you run a big organization or a Company then you are required to buy the license key to run the program.
- Phase 5 is compatible with Windows only.
- Phase 5 HTML editors support different languages such as HTML, PHP, Java, JavaScript, Pearl, and VBScript.
- It has a crisp and clear Menu arrangement.
- Integrated file management makes the switching between different documents easy.
- Phase 5 has a tidy interface to work with.

Notepad:

Notepad is a simple text editor. It is an inbuilt desktop application available in Windows OS.

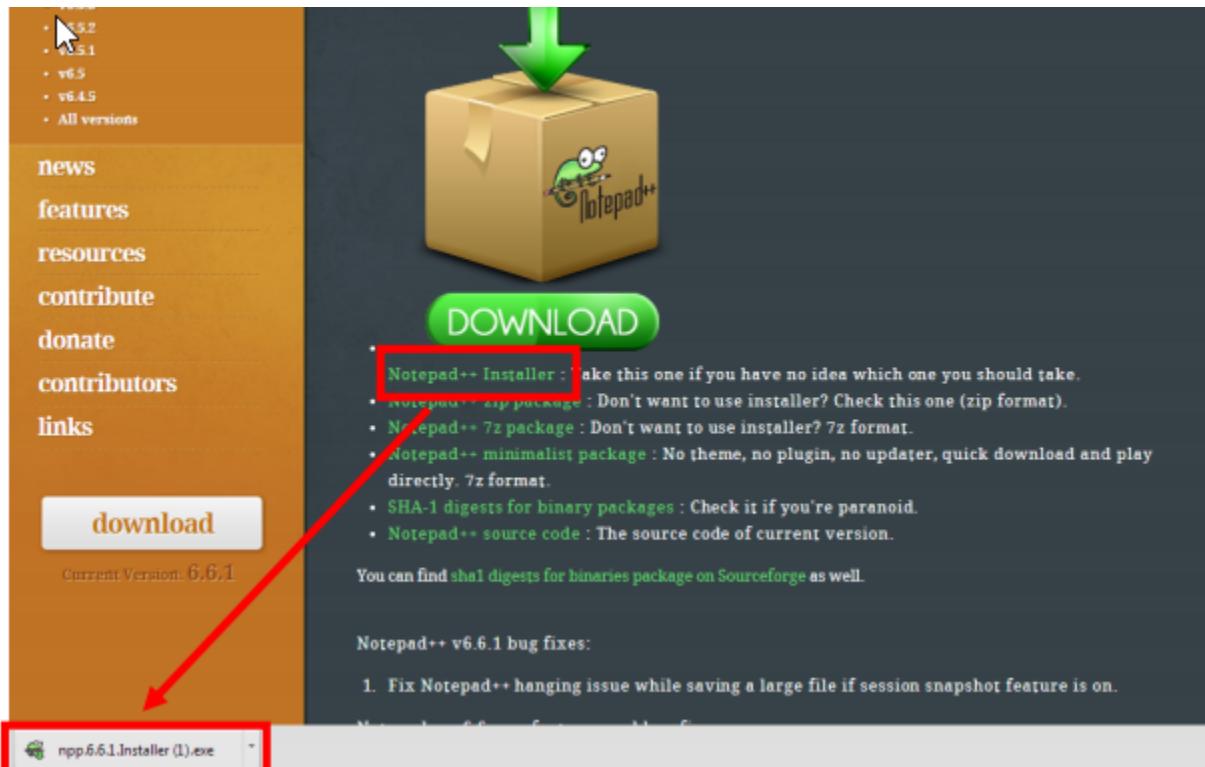


Notepad++

Installation

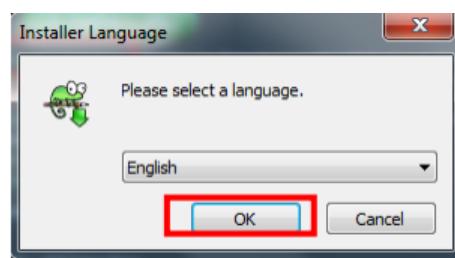
Step 1:- Go to the following website: - <http://notepad-plus-plus.org/download/v6.6.1.html>

Step 2:- Click on 'Notepad++ Installer'.



Step 3:- A ‘account user control’ window opens. Click ‘yes’.

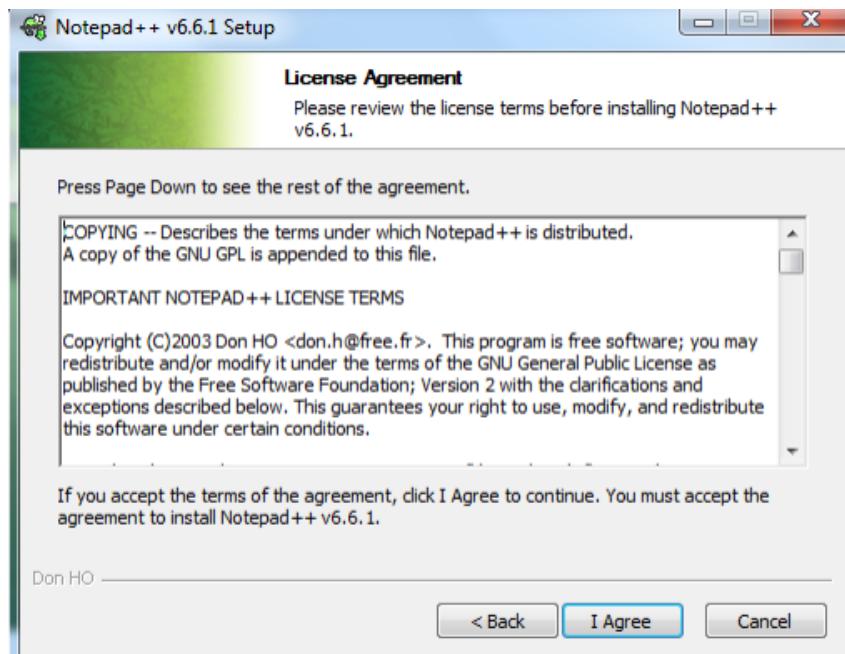
Step 4:- Click ‘Ok’



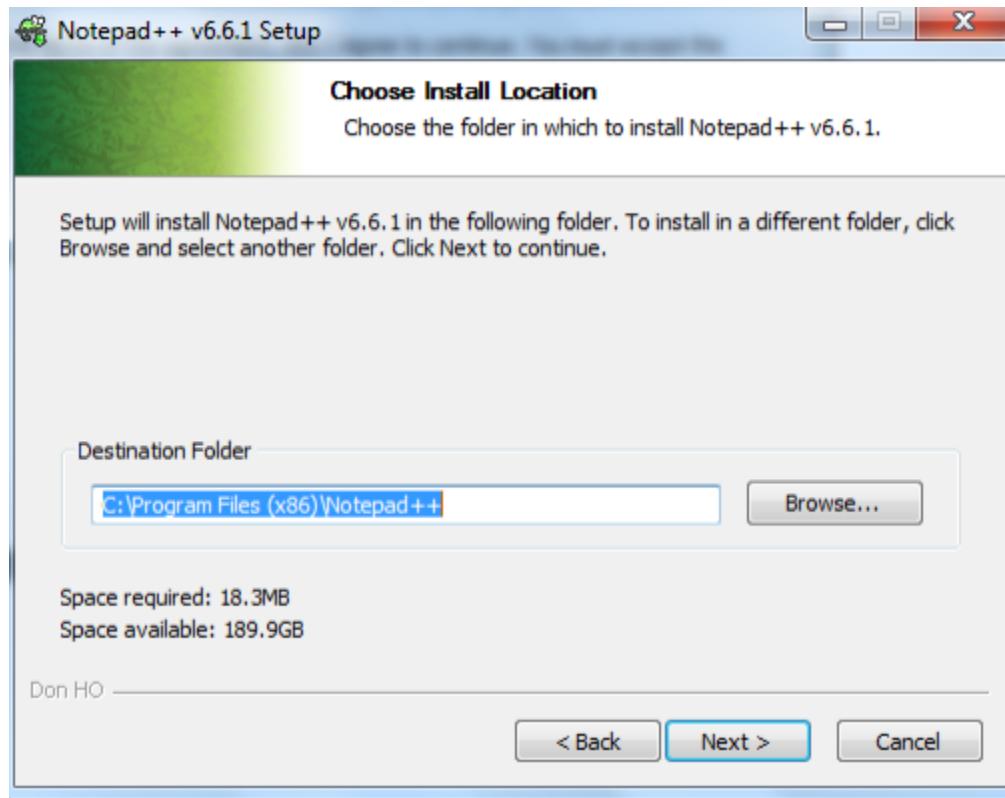
Step 5:- Click ‘Next’.



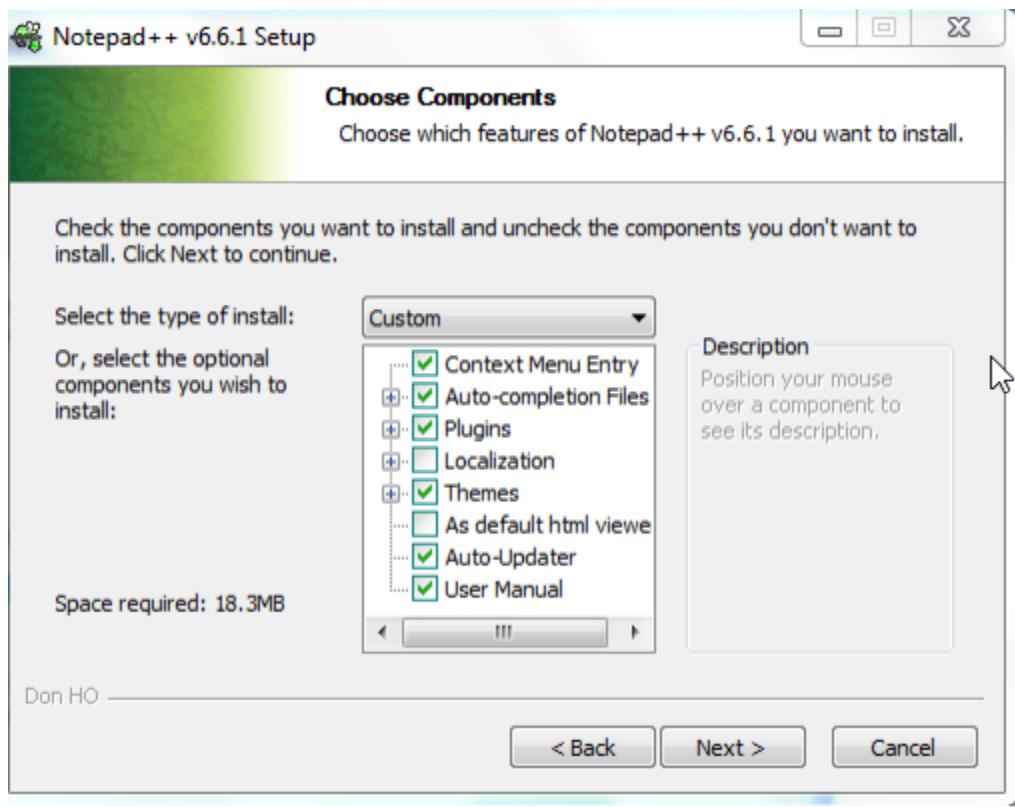
Step 6:- Click 'I Agree'.



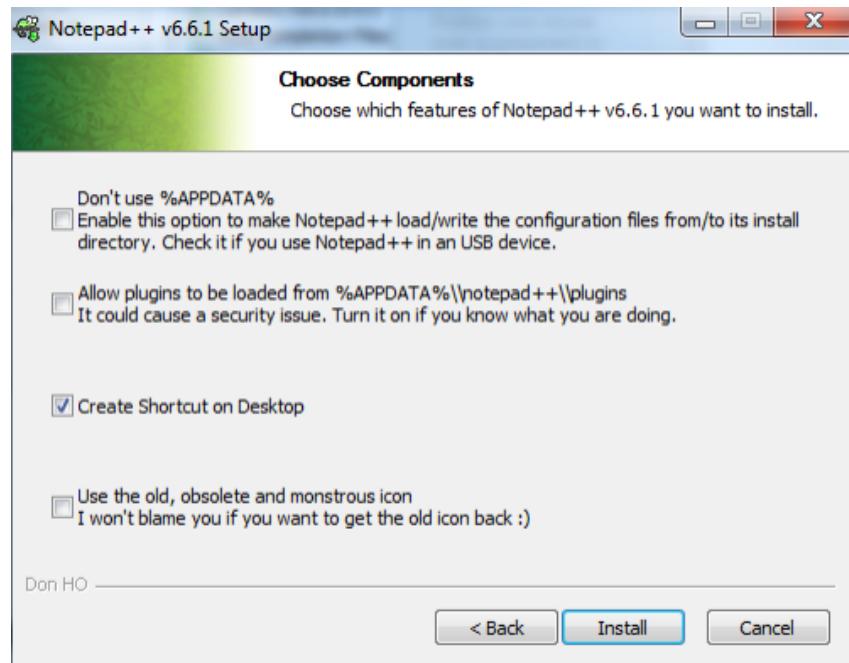
Step 7:-Click 'Next'.



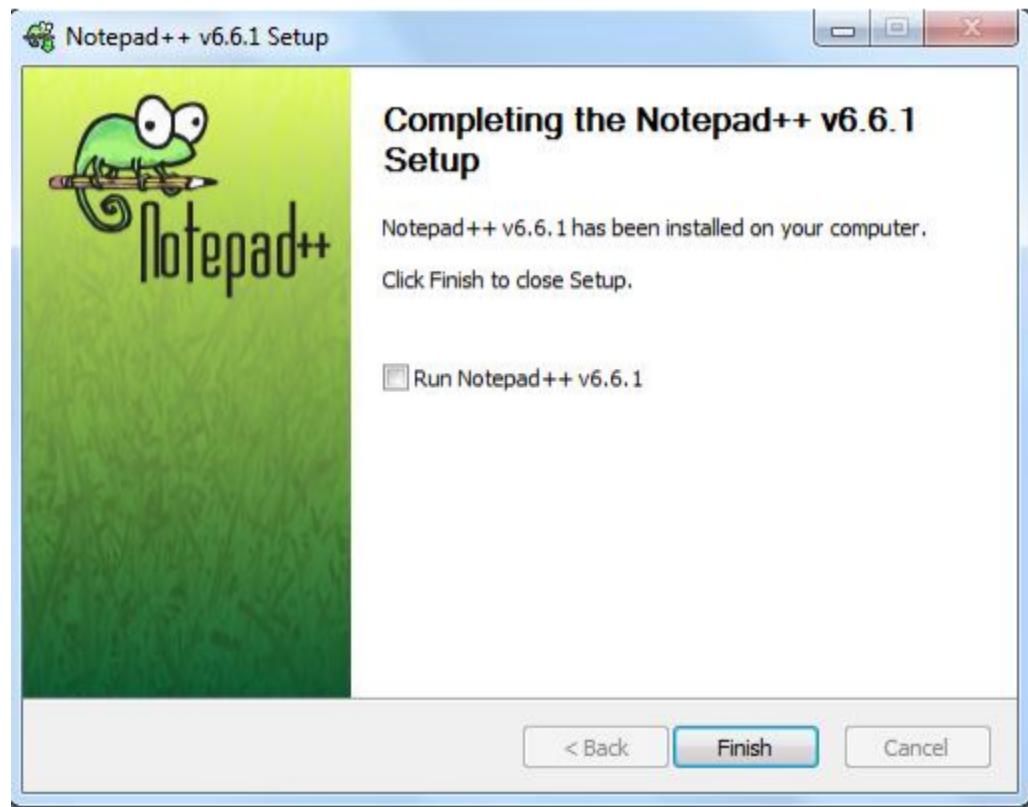
Step 8:- Click 'Next'.



Step 9: - Click 'Install'.



Step 10:- Click 'Finish'.



Features

- Autosave.
- Finding and replacing strings of text with regular expressions.
- Guided indentation.
- Line bookmarking.
- Macros.
- Simultaneous editing.
- Split screen editing and synchronized scrolling.

Pros and cons

Pros

- Light and fast
- Portable
- Free

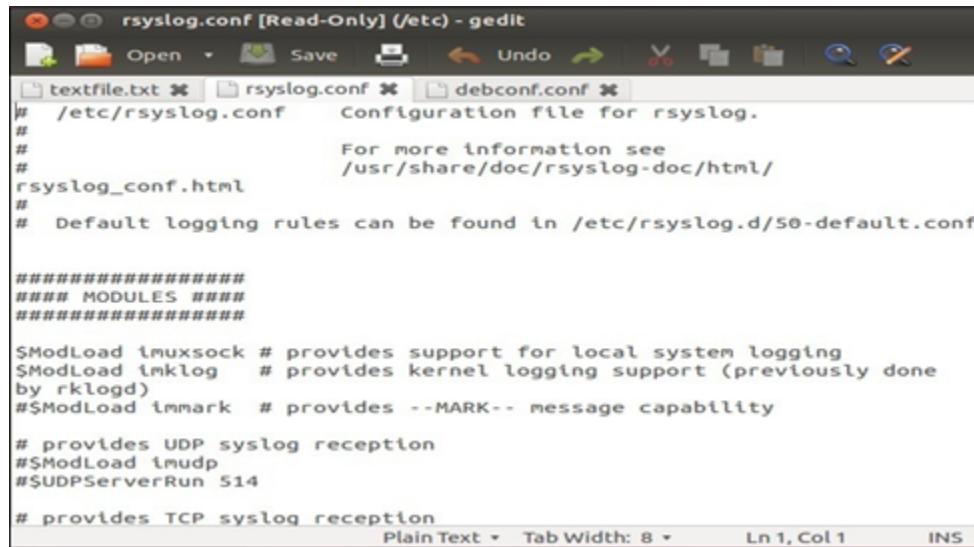
-
- Collaborative editing

Cons

- Single platform support – Despite the software being as good as it can to be offered for free, it has limited support.
- Notepad++ is only available on Windows leaving out macOS and Linux.

geditor

geditor provides a simple interface from which you have access to a full text editor with programming functions and is compatible with most languages



The screenshot shows the gedit text editor window with the file `/etc/rsyslog.conf` open. The window title is "rsyslog.conf [Read-Only] (/etc) - gedit". The text in the editor is a configuration file for rsyslog, containing various directives like \$ModLoad, #Provides, and #Default logging rules. The code is as follows:

```
# /etc/rsyslog.conf      Configuration file for rsyslog.
#
#           For more information see
#           /usr/share/doc/rsyslog-doc/html/
#           rsyslog_conf.html
#
# Default logging rules can be found in /etc/rsyslog.d/50-default.conf

#####
#### MODULES ####
#####

$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog  # provides kernel logging support (previously done
by rklogd)
#$ModLoad immark # provides --MARK-- message capability

# provides UDP syslog reception
#$ModLoad imudp
#$UDPServerRun 514

# provides TCP syslog reception
```

Image 58:gEdit

Reference:<https://geek-university.com/linux/gedit-text-editor>

Features

Some of the text-editing platform's features include

- Display line-number feature: This makes the software ideal for managing large source codes and it complements the 'Go to Line' function.

-
- The platform supports internationalized texts and handles Arabic characters without any problem.
 - It comes with an integrated syntax highlighting feature.
 - The integrated highlighting syntax feature makes the software ideal for editing scripts and configuring files.
 - It comes with an integrated spell checker: This allows users to autosave the document they are working on and access its copy in case it is damaged.
 - Supports GNOME: The software supports different types of document formats and allows users to open any ASCII file when using GNOME

Pros of gedit

- Some of the benefits of using this software are
- It comes with a plugin for the document statistics: This makes the software able to show statistics on selections with information such as the number of lines, characters, words, and bytes included.
- It is a light text-editing platform that is easy to use and highly efficient.
- The software ensures that users get access to a beautiful and user-friendly interface.
- It allows software developers to modify the code and analyze existing codes easily.
- The software is compatible with remote editing and a variety of languages.

Cons of gedit

- Some of the limitations of the platform are
- It does not come with the integrated find and replace feature.
- Users might be concerned that the platform does not include a code-folding feature.

Sublime Text 3:

Sublime is a cross platform code editor tool. It supports all markup languages

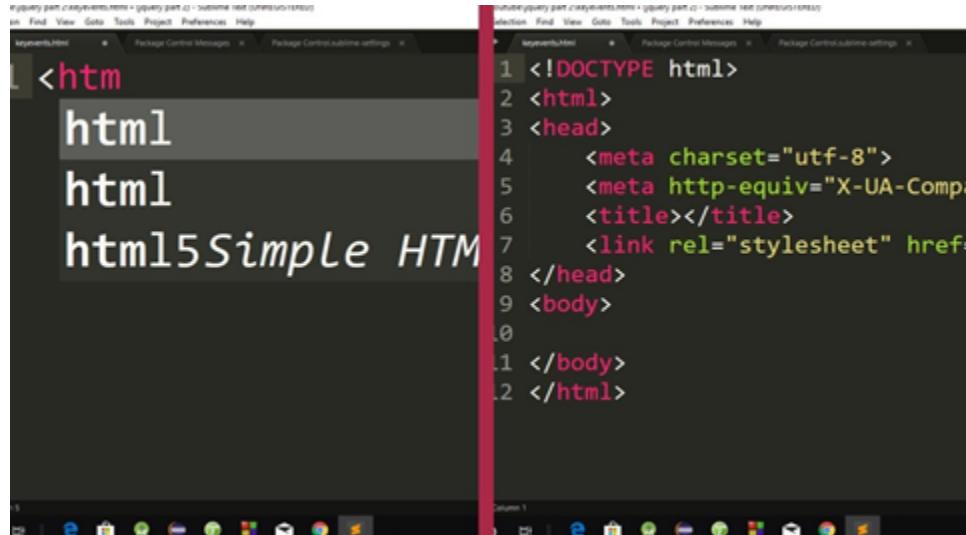


Image 59:Sublime

Reference:<https://www.thapatechnical.com/2018/09/html-css-autocomplete-plugin-in-sublime-Text-3.html>

Step 1 – Download the .exe package from the official website as shown below

<https://www.sublimetext.com/3>

Step 2 – Now, run the executable file. This defines the environment variables. When you run the executable file, you can observe the following window on your screen. Click Next.

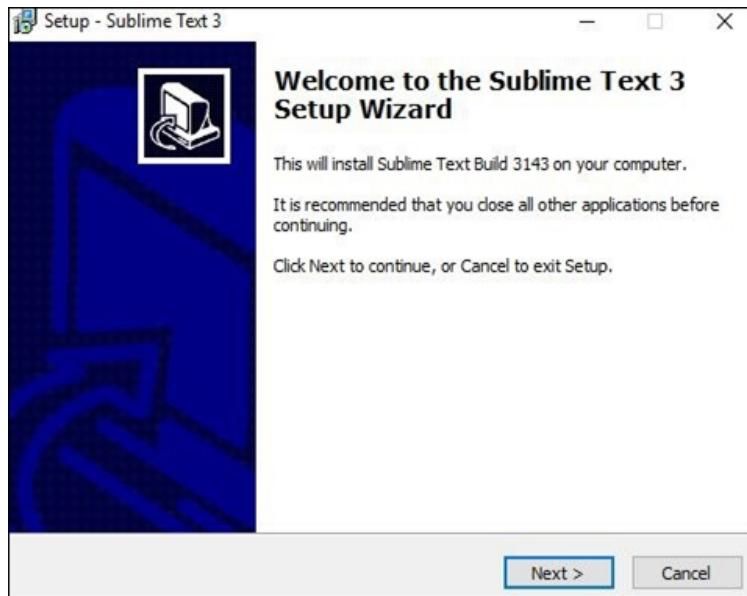
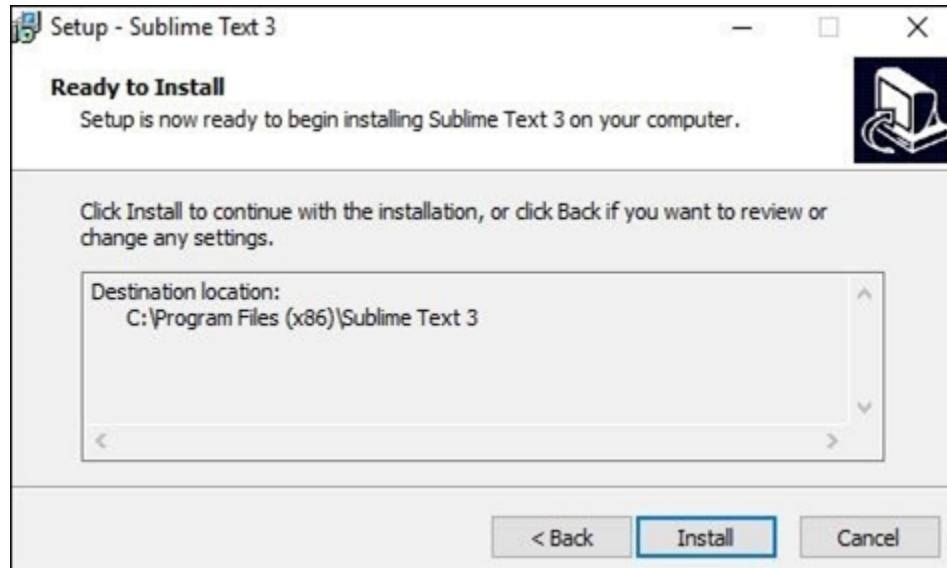


Image 59:Sublime

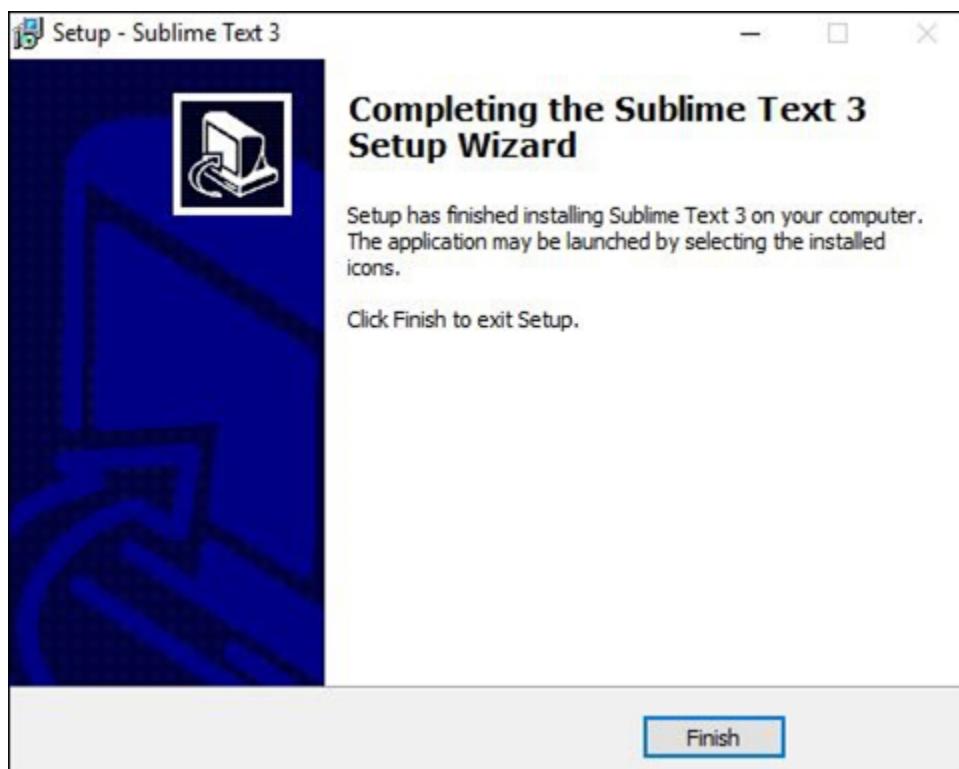
Reference:<https://www.thapatechnical.com/2018/09/html-css-autocomplete-plugin-in-sublime-Text-3.html>

Step 3 – Now, choose a destination location to install Sublime Text3 and click Next.

Step 4 – Verify the destination folder and click Install.



Step 5 – Now, click Finish to complete the installation.



Step 6 – Upon a successful installation, your editor will appear as shown below

Features

- Syntax Highlight
- Auto Indentation
- File Type Recognition
- Sidebar with files of mentioned directory
- Macros
- Plug-in and Packages

Pros

- Performance: all the operations like opening, closing, searching is fast
- Package: lots of packages and themes
- Customize: looks nice, also has many themes to choose from, and is able to configure using JSON file.
- Reliable: no need to do anything else once everything is installed and set up.

Cons

- Cost: free download, but require \$70 license fee, otherwise frequent “buy license” pop-up.
- Package: package control is not installed ahead, so some searching needed; powerful only with plugins.
- Suggestions: only based on snippet, no context or language based suggestions

Bracket:

Bracket is an open-source software primarily used for Web development. It provides live HTML, CSS, JavaScript editing functionality.

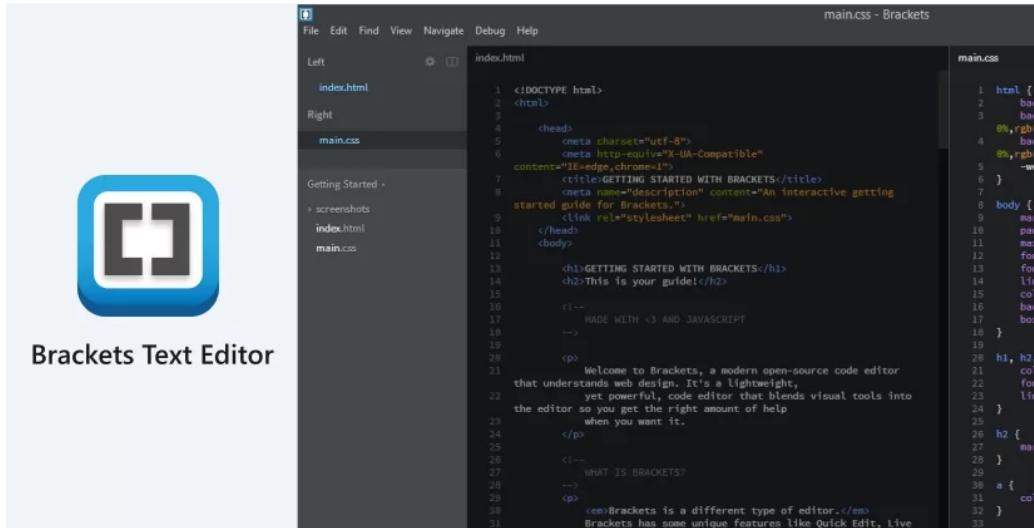


Image 60:Bracket

Reference:<https://blog.templatetoaster.com/best-free-html-editors/>

Features

- Brackets is a top-notch Text editor for mac.
- Open source and free to download a text editor.
- Brackets support multiple platforms like macOS, Windows, Linux.
- The major feature which makes Brackets different from the rest of the HTML editors is the “Extract” feature.
- Extract feature lets you extract information straight from PSD(s) like fonts, colors, and measurements with a clear CSS without contextual code references.
- Simple to implement JavaScript, HTML, and CSS.
- Easy to customize. It is a free HTML Editor.

Pros

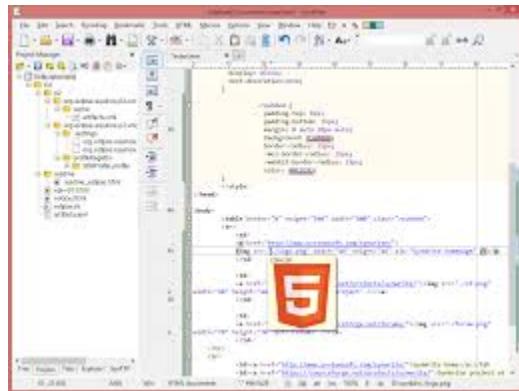
- Free to use for everyone
- The open source nature of the software allows you to add your own extensions
- Includes standard features that are must-haves for web developers and code editors
- Unique features, such as Live Preview and Quick Edit, make code editing even easier
- Built using CSS, HTML, and Javascript, making it easy to hack and extend
- Gives you the ability to quickly download user-created extensions

Cons

- Mostly made for front-end developers who use HTML, Javascript, or CSS, with little functionality for those using server side coding languages
- The extension registry does not have a filtering option making it hard to find what you are looking for

SynWrite Editor

SynWrite combines great ideas from many well known editors into a single, freely available product. It's a complete environment for Web workers, coders and writers.

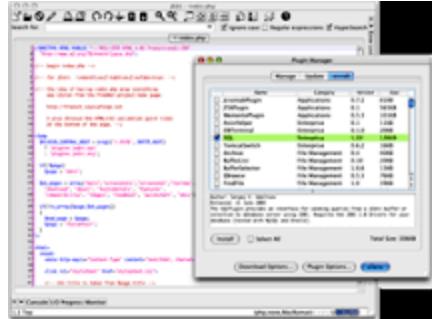


Features

- macro recording
- code highlighting
- code folding,
- multi-caret editing
- regular expressions.
- PlainEdit.NET
- Portable PlainEdit.NET is an intuitive word processor with advanced settings and support for syntax highlighting.

jEdit HTML Editor

jEdit is a mature programmer's text editor with hundreds (counting the time developing plugins) of person-years of development behind it.



Features

- Written in Java, so it runs on Mac OS X, OS/2, Unix, VMS and Windows.
- Built-in macro language; extensible plugin architecture. ...
- Plugins can be downloaded and installed from within jEdit using the "plugin manager" feature.
- Auto indent, and syntax highlighting for more than 200 languages.

Visual Studio Code

Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications.

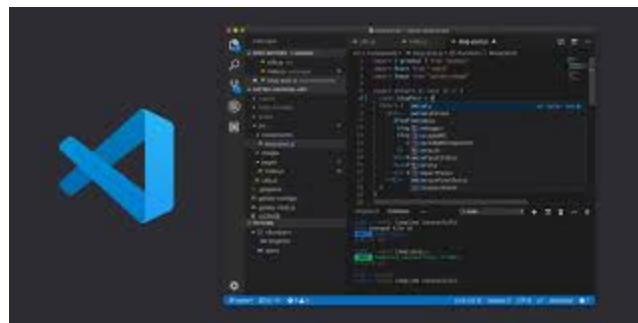


Image 61:Visual Studio
Reference:<https://code.visualstudio.com/download>

Installation

Step 1- Download

Head over to the Visual Studio Code downloads page.

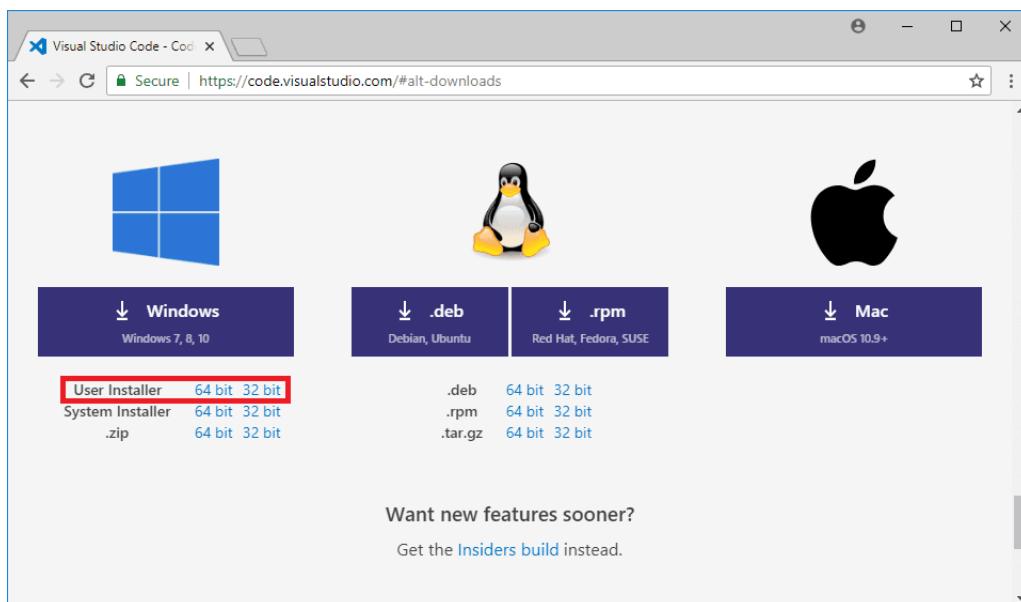
There are three Windows installer versions available:

User installer: installs in your User folder and does not need Administrator privileges.

System installer: installs for all users on the system and needs Administrator privileges.

.zip installer: a portable version.

Check your windows bit version and click on the corresponding link. visual studio code download user installer Wait for the download to complete.

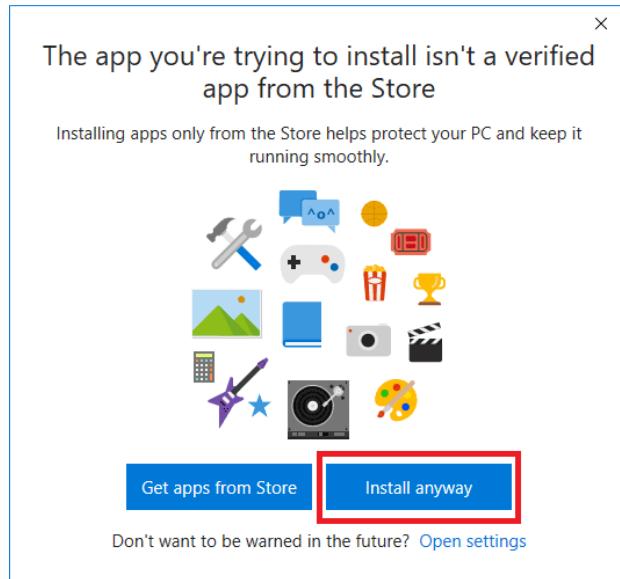


Step 2- Install

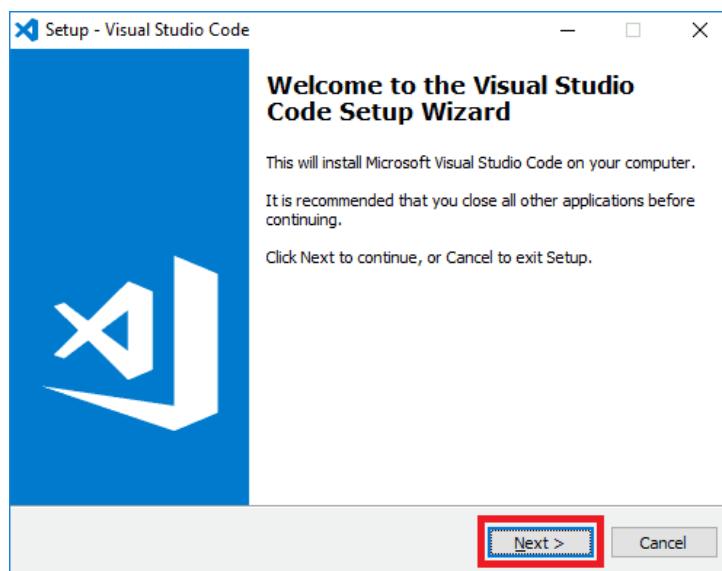
Open the location of the downloaded executable. visual studio code downloaded installer

Double-click it to run the installer.

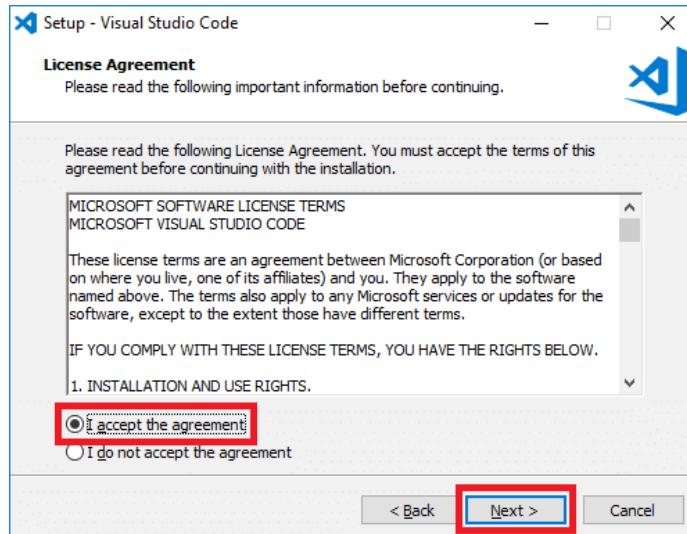
On Windows 10 a pop-up window will appear: The app you're trying to install isn't a verified app from the StoreClick on Install anyway.



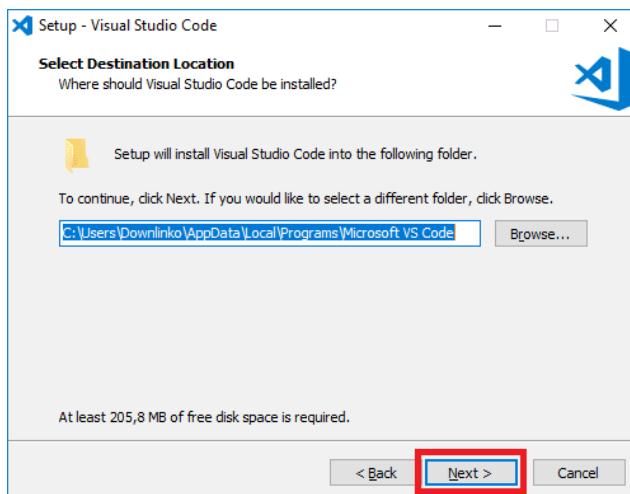
Step3-The installer setup wizard will open.Click Next.



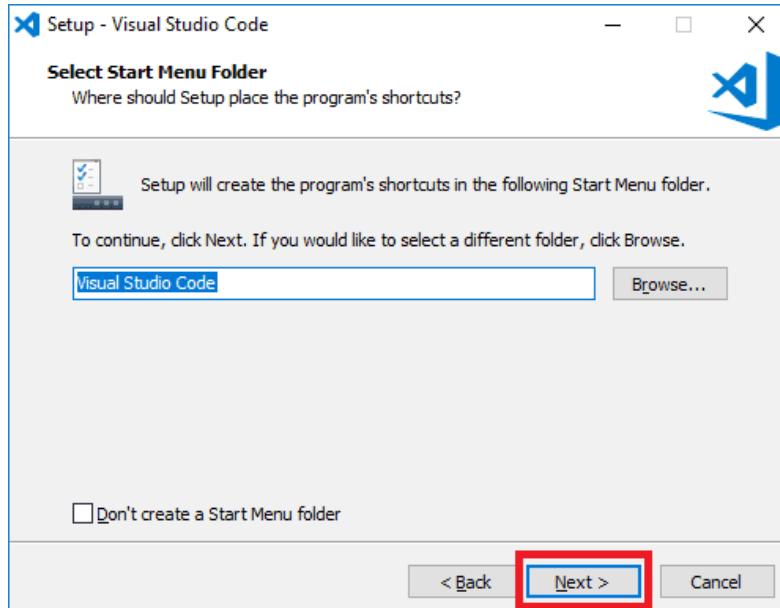
Step4- Click on the radio button next to I accept the agreement.Click Next.



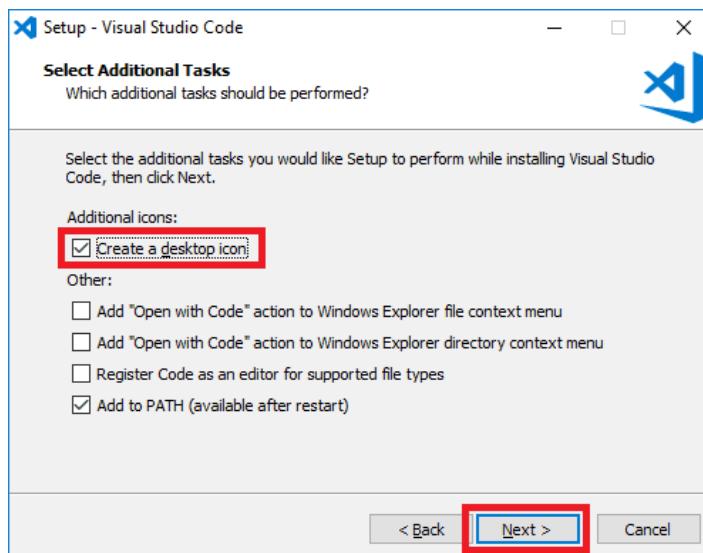
Step5-You can change the installation location by clicking on the Browse... button.In this example, we keep the default install location.Click Next.



Step6- Keep the default Start Menu Folder.Click Next.

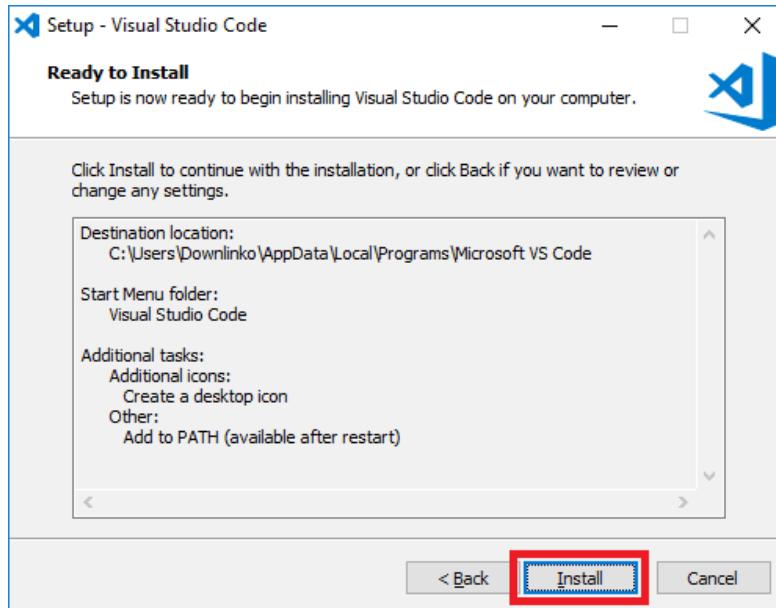


Step7- Select the Create a desktop icon checkbox.Click Next.



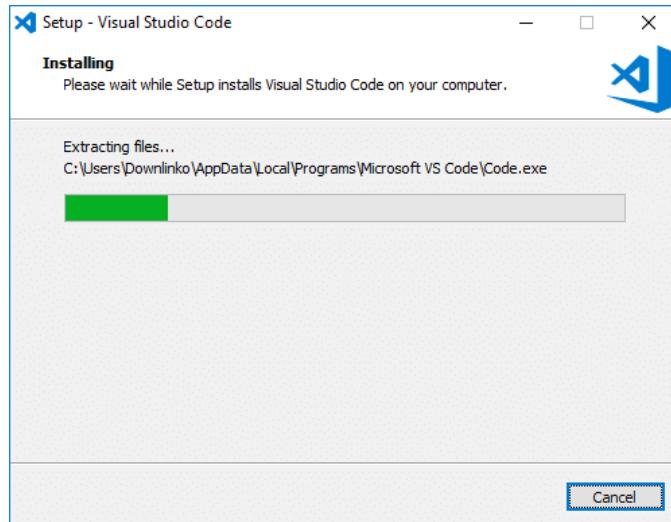
Step8-An overview of the selected installation settings is shown.

Click Install to start the installation.

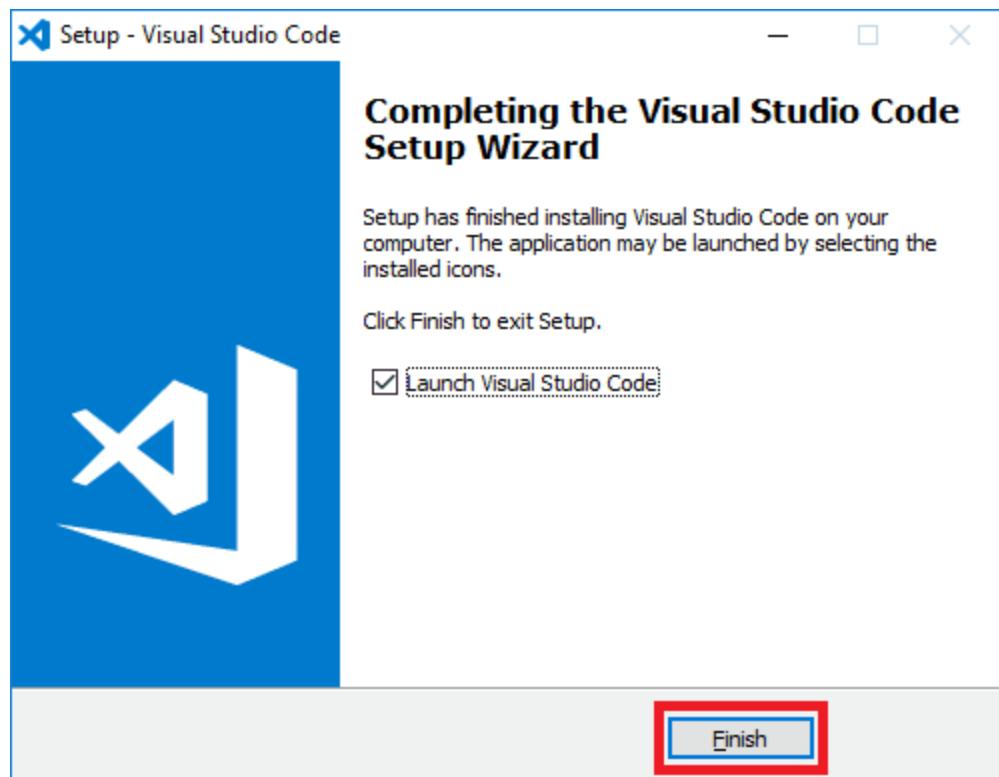


Step9-The Visual Studio Code installation will now start.

A progress bar shows the various steps that are executed.

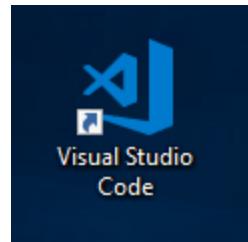


Step10-Once the installation is complete, click Finish.



Step 11: Run

To start Visual Studio Code double-click on the desktop shortcut.



Features

- Supports the use of multiple split windows and horizontal/vertical orientation
- Includes VS Code IntelliSense that supports many different languages
- Command Palette that provides instant access to popular VS Code components

Pros

- User friendly layout and visually stunning interface
- Easily customizable
- Thousands of plugins/extensions available through the VS Code Marketplace

Cons

- Lacks most features of a full IDE suite
- Slow File Search
- Longer launch time than most of its competitors

Comparison



The Top 5 Free HTML Editors

	CoffeeCup HTML Editor	Komodo Edit'	NetBeans	Notepad++	Visual Studio Code
Free	✓	✓	✓	✓	✓
Open Source		✓	✓		
Number of Users	1 user	1 user	1 user	1 user	1 user
Upgrade Cost	\$29 once	\$7 monthly	Free	Free	Not available
Support	Online	Online	Online	Forum	Online
Text Editor	✓	✓	✓	✓	✓
WYSIWYG Editor	✓		With plugin		
Training	Documentation	Documentation	Documentation	Documentation	Documentation
Deployment	Installed (Windows)	Installed (Mac & Windows)	Cloud, SaaS, Web	Installed (Windows)	Installed (Mac & Windows)

Capterra

Image 62:Comparison of Editors
Reference:<https://blog.capterra.com/best-free-html-editors/>

Applications of HTML

How is its industry using HTML?

The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards.

If you are learning about Web technology, you may wish to start with the introduction below,

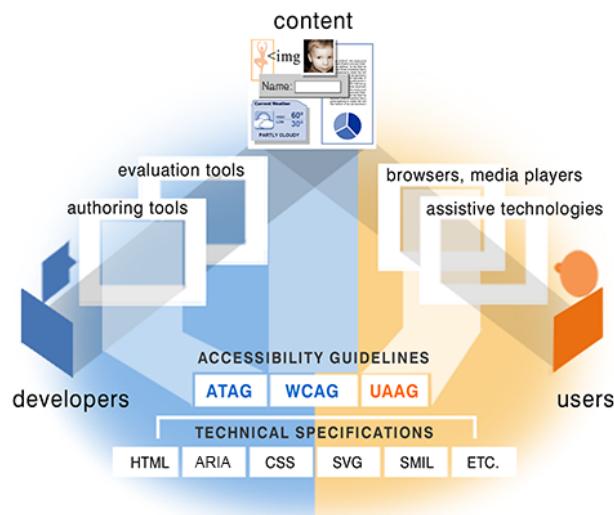


Image 63: Application in Industry
Reference:<https://www.w3.org/WAI/standards-guidelines/>

What is a Static Website?

Website

Website is a collection of related web pages that may contain text, images, audio and video. The first page of a website is called a home page. Each website has a specific internet address (URL) that you need to enter in your browser to access a website.

Website is hosted on one or more servers and can be accessed by visiting its homepage using a computer network. A website is managed by its owner that can be an individual, company or an organization.



Image 64:Website -www
Reference:<https://www.javatpoint.com/website-static-vs-dynamic>

A website can be of two types:

- Static Website
- Dynamic Website

Static website

Static website is the basic type of website that is easy to create. You don't need the knowledge of web programming and database design to create a static website. Its web pages are coded in HTML.

The codes are fixed for each page so the information contained in the page does not change and it looks like a printed page.

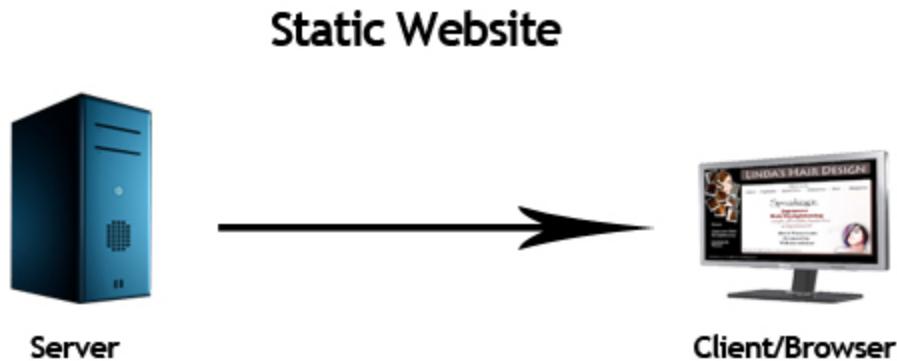


Image 65:Static Website
Reference:<https://www.javatpoint.com/website-static-vs-dynamic>

What is a Dynamic Website?

Dynamic website is a collection of dynamic web pages whose content changes dynamically. It accesses content from a database or Content Management System (CMS). Therefore, when you alter or update the content of the database, the content of the website is also altered or updated.

Dynamic website uses client-side scripting or server-side scripting, or both to generate dynamic content.

Client side scripting generates content at the client computer on the basis of user input. The web browser downloads the web page from the server and processes the code within the page to render information to the user.

In server side scripting, the software runs on the server and processing is completed on the server then plain pages are sent to the user.

Dynamic Website

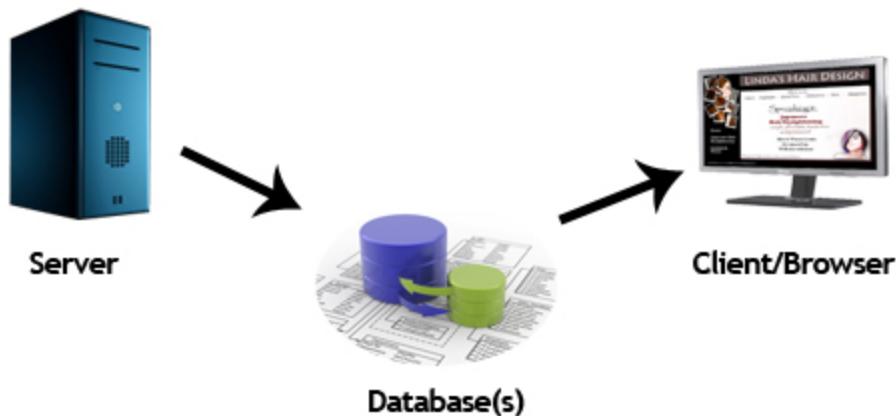


Image 66:Dynamic Website
Reference:<https://www.javatpoint.com/website-static-vs-dynamic>

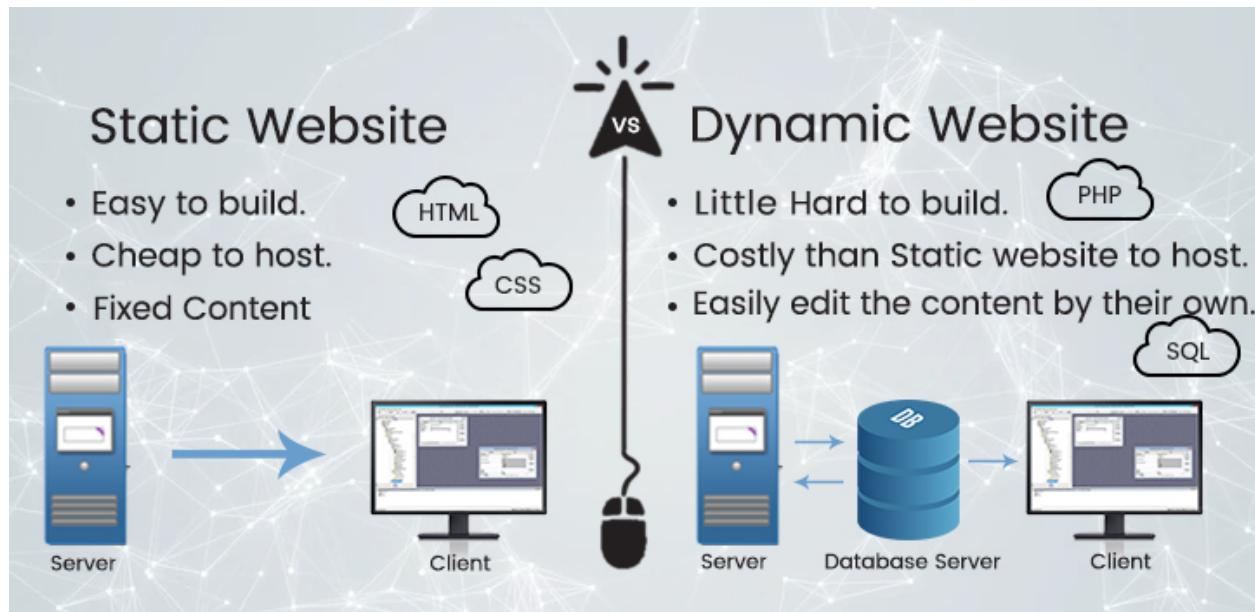
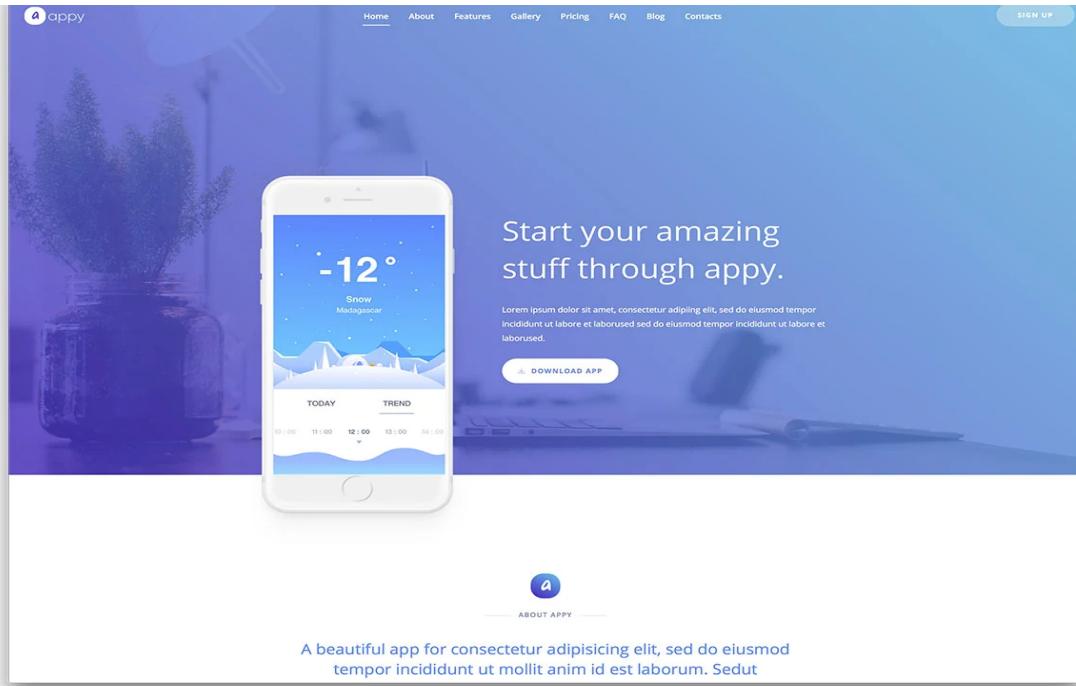


Image 67:Difference between static and dynamic
Reference:<https://www.jeewangarg.com/blog/difference-between-static-and-dynamic-website>

Practical Application



Source Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="style.css" />
<title>LandscapeTitles| florida web design</title>
```

```
</head>

<body>

<div id="container">

    <div id="header">

        <h1><span class="off">Landscape</span>Titles</h1>

        <div id="links">

            <a href="#">Home</a>

            <a href="#">About</a>

            <a href="#">Products</a>

            <a href="#">Services</a>

            <a href="#">Portfolio</a>

            <a href="#">Contact</a>

        </div>

    </div>

    <div id="content">

        <div class="contenttext">

            <h2>Template Information</h2>

            <p>You may use this template in any way you would like, please just leave the link at the bottom to our site in place. In order to add your own pictures, simply insert an image as usual, and just add class="picture" to each image. The shadow is automatically added to the images. Make sure your image is exactly 750px wide for best results.</p>

        </div>

    </div>

</div>
```

```
</div>



<div class="contenttext">

<h2>Template Information</h2>

<p>You may use this template in any way you would like, please just leave the link at the bottom to our site in place. In order to add your own pictures, simply insert an image as usual, and just add class="picture" to each image. The shadow is automatically added to the images. Make sure your image is exactly 750px wide for best results.</p>

</div>



<div class="contenttext">

<h2>Template Information</h2>

<p>You may use this template in any way you would like, please just leave the link at the bottom to our site in place. In order to add your own pictures, simply insert an image as usual, and just add class="picture" to each image. The shadow is automatically added to the images. Make sure your image is exactly 750px wide for best results.</p>

</div>

</div>

<div id="footer"><a href="http://www.google.com">web design florida</a></div>

</div>

</body>

</html>
```

```
body {  
    margin: 0;  
    padding: 0;  
    text-align: left;  
    font: Arial, Helvetica, sans-serif;  
    font-size: 13px;  
    color: #000000;  
    background: #1364A6;  
    background-image:url(images/background.png);  
    background-repeat:repeat-x;  
}  
  
*  
  
{  
    margin: 0 auto 0 auto;  
    text-align:left;}  
  
#container  
{  
    display: block;  
    height:auto;  
    position: relative;  
    width: 750px;
```

```
background-image:url(images/background.jpg);  
background-repeat:repeat-y;  
padding-left:17px;  
padding-right:17px;  
overflow:hidden;  
}
```

```
#links  
{  
display:block;  
float:right;  
margin-top:25px;  
margin-right:10px;  
}  
  
#links a, #links a:visited  
{  
font-size:16px;  
font-weight:bold;  
margin-left:4px;  
margin-right:4px;  
color: #1652B4;
```

```
text-decoration:none;  
}  
  
#links a:hover  
{  
border-bottom:dotted 1px #1652B4;  
}  
  
#header  
{  
height:60px;  
}  
  
#header h1  
{  
margin-top:15px;  
margin-left:10px;  
font-size:33px;  
color:#0A2645;  
float:left;  
}  
  
.off  
{  
color:#1652B4;
```

}

.picture

{

background-image:url(images/imagebackground.jpg);

background-repeat:repeat-x;

background-position:bottom;

padding-bottom:8px;

margin-top:22px;

}

.contenttext h2

{

color:#123F83;

font-size:24px;

margin-bottom:10px;

}

.contenttext

{

padding-left:10px;

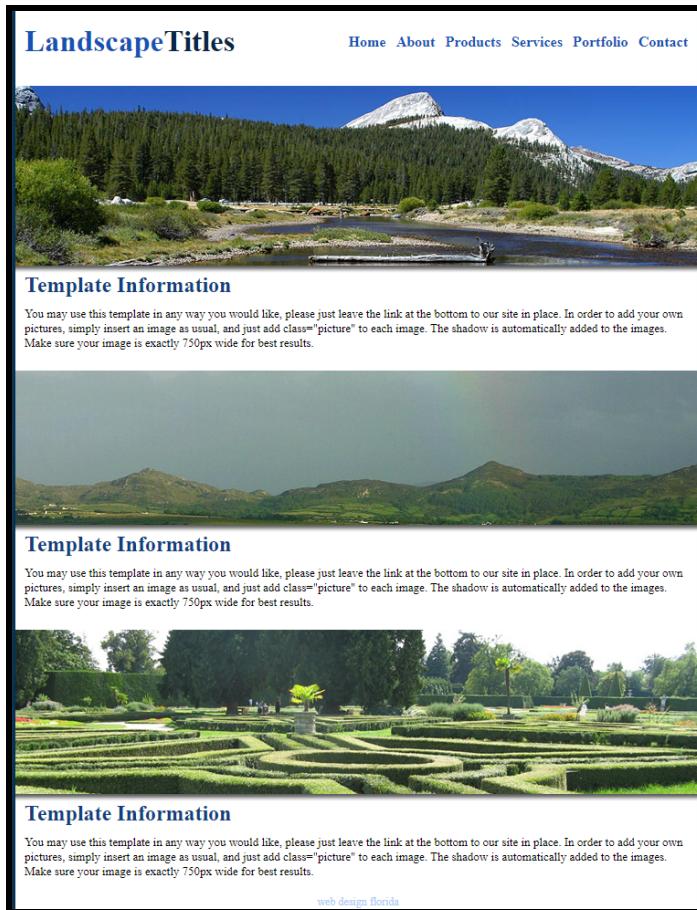
padding-right:10px;

}

#footer

```
{  
font-size:12px;  
color:#519FEE;  
margin-top:18px;  
width:750px;  
text-align:center;  
}  
  
#footer a, #footer a:visited  
{  
text-align:center;  
color:#ADCCF1;  
text-decoration:none;  
}  
  
html, body {  
text-align: center;  
}  
  
p {text-align: left;}
```

Output



Top 10 Uses of HTML

Web pages development:

HTML is heavily used for creating pages that are displayed on the world wide web. Every page contains a set of HTML tags including hyperlinks which are used for connecting to other pages. Every page that we witness, on the world wide web, is written using a version of HTML code.

Web document creation:

Document creation on the internet is dominated by HTML and its basic concept via tag and DOM i.e. document object model. HTML tags are inserted before and afterward or phrases to locate their format and location on the page. A web document consists of three sections: title, head, and body. Head includes the information to identify the document, including title and any

other important keyword. A title can be seen on the browser's bar and the body section is the main portion of the website visible to the viewer. All three segments are designed and created by the use of HTML tags. Every section has its own specific set of tags, which are dedicatedly rendered keeping the head, title and body concepts in a loop.

Internet navigation:

This is one of the most important uses of HTML which is revolutionary. This navigation is possible by utilizing the concept of Hypertext. It is basically a text which refers to other web pages or text and when the user clicks on it, would navigate to the referenced text or page. HTML is heavily used to embed the hyperlink within the web pages. A user can easily navigate within the web pages and between websites as well, which are located on different servers.

Cutting edge feature:

HTML5 with its set of standards and API is being used to introduce some of the latest trends in website creation business. Like polyfill libraries, which are supported by old browsers equally well. Browsers like Google Chrome are the perfect choice when it comes to implementing an HTML5 latest set of standards and APIs. There is a JavaScript library available called Modernizr, which can detect features that let the developer dynamically load polyfill libraries as required.

Responsive images on web pages:

At the elementary level in applications of HTML, queries can be set to utilize the images, which are responsive in nature. With the srcset attribute of img element in HTML, and combining it with picture elements, a developer can fully control how the user will render an image. Now different types of an image with size variation can be loaded by using the img element. Rules can be easily set with the picture element, we can declare an img element with default source and then for every case, a source can be provided.

Client-side storage:

Earlier, a user could not save the user's browser data that would persist across sessions. To meet this requirement, server-side infrastructure has to be built or user's cookies can be used. But with HTML5, client-side storage is feasible using localStorage and IndexDB. These two strategies have their own standards and features. localStorage basically gives string-based hash-table storage. Its API is very simple and provides the developer with setItem, getItem, and removeItem

methods. IndexDB, on the other hand, is a larger and better client-side data store. IndexDB database can be expanded with the user's permission.

Offline capabilities usage:

Once data can be stored in the browser, the developer can think of a strategy to make the application work, when a user is disconnected. HTML5 has its application cache mechanism which would define how the browser manages the offline situation. Application cache, responsible for offline ability actually comprises different components, which includes API methods that create an update, read manifest file and events. By using the certain property in HTML5, a developer can check if the application is online or not. A developer can also specify in a website's application cache manifest file the information like, what browser manages resources for offline use. In the manifest file, resources which are available offline can also be specified.

Data Entry support with HTML:

HTML5 standard and set of APIs can be used to support data entry level of work. As browsers implement new HTML5 standards, developers can simply add the attributes to the tag which indicate required fields, text, data format etc. HTML5 has come up with several new attributes to drive on-screen keyboards, validation, and other data-entry experiences so that end-users can have a better data-entry.

Game development usage:

Before the advent of HTML5, game development was an exclusive domain of Flash and Silverlight. Since browsers support new specifications for HTML5 including CSS3 and light-fast JavaScript engine to drive a new rich experience, HTML5 can bring the reality of game development possible, which was earlier the forte of Flash and Silverlight. Every single feature of APIs need not be implemented, but most appropriate ones can be utilized while eliminating the rest of the features.

Native APIs usage to enrich website:

HTML5 adds so many new abilities and tools, which was just an imagination in the past. A large set of new APIs regarding file system, Geolocation, drag, and drop, event handling, client-storage etc. are the capabilities which make usage of HTML5 more easier than ever before. Application experience can be enhanced with other APIs like Fullscreen, Visibility and

Media Capture. A modern web application has asynchronous nature which can be fostered using Websockets and Web workers like APIs.

Able to create Styles of web pages using CSS

In this section, we will read about:

- Introduction to CSS
- Limitations of CSS
- Advantages of CSS
- Three ways to integrate CSS.
- Merits and Demerits of external Style Sheets, Embedded Style Sheets

Introduction to CSS.

What is CSS?

Cascading Style Sheets which is in a better way known as CSS, is a very simple designed process that is used for making the web pages a lot more presentable. CSS allows you to put styles to customize your web pages. The best part about making use of this styling feature is that the CSS is independent of the HTML way of creating web pages.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. CSS is combined with the markup languages HTML or XHTML.

Designing the HTML Page

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to describe the content of a web page, like:

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

When tags like , and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. The development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS is the style sheet language for describing the presentation and design of web pages including colors, fonts, and layouts. It is mainly designed to enable the distinction between presentation and content, including colors, layouts, and fonts. It can be used in different types of devices, like large or small screens and printers. It is independent of HTML and can be used with any XML-based markup language. The CSS specifications are mainly maintained by the World Wide Web Consortium. CSS rule-set consists of a selector and a declaration block.

Some of the rules are described below:

Example: `h1 { color: white; font-style: italic }`. Here,

- Selector: `h1` (It indicates the HTML element which needs to be styled)
- Property: “color and font-style” (It defines the aspects of elements that need to be changed)
- Declaration Block: “color: white; font-style: italic” (It describes one or more declarations separated by semicolons)
- Values: “white and italic” (It indicates the parameters of the selected properties.)

CSS Saves a Lot of Work!

The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file.

Why is CSS required

- Compatibility: While using CSS, a user can be stress-free with its older language versions as it is well compatible even with the older versions. Because of this, even if the applications of CSS are developed in some older version of a programming language and

if the developers integrate the same with new enhancements then CSS can be easily incorporated and implemented with those changes easily. So, the changes associated with the predecessor of CSS, need not be removed and a developer can update the existing code successfully.

- **Website Maintenance:** Another most common reason, why the uses of CSS are important? CSS plays a very important role while performing website maintenance. It makes website maintenance much easier. The CSS file makes the website look and feels more flexible and it can be altered in a more convenient manner. Also, it makes the HTML formatting and the corresponding data elements modification easier. Because of this, the website maintenance becomes more convenient from the development perspective.
- **Social Media Impact:** Uses of CSS in HTML frameworks are also incorporated with social media website developments as well. Facebook applications can directly correlate with the corresponding frameworks. The HTML client library files can be implemented based on CSS stylings to develop the application and can work on different extensions. These social media-based platforms can be inter-related with the frameworks and can be used to develop a few enhancements from the end user's perspective. Thus, the CSS styling and updating the user interphase becomes easier and this creates a direct impact on the social media platforms.
- **Easy Accessibility:** From the accessibility point of view, applications of CSS provide much better solutions that allow users to update the user interface to suit the business requirements. It also allows web pages to be easily rendered by different devices like speaking browsers, PDAs, etc. This actually provides a much deeper impact while considering the look and feel modification of a web page from the end-user and business perspective.
- **Image File Handling:** In the case of image handling, the use of CSS provides the styling Library, and this helps to output images along with the XML to the browser. Initially, it was a bit difficult to update and style the existing image. But now using the CSS files, output images can be received in different formats like jpeg, png, and gif and this can be modified to the specified styling formats as per the requirement. This feature will also

allow editing different image types and is also used to create thumbnails, watermarks, image cropping, etc.

- Handling Dynamic Website Templates: Dynamic Website Templates can be created and handled by uses of CSS in the HTML framework and it basically helps while adding and editing web pages and elements to the web server and sites. By adding CSS extension to the concerned HTML pages and incorporating the same with the server-side templates, it makes it easier to handle the dynamically allocated elements. So, using these templates the handling of dynamic elements can be more organized and properly implemented by CSS frameworks. These elements can also be used to style the pages in a dynamic pattern while using the CSS templates.
- Handling Flash Animation and Effects: With the help of CSS Flash files, flash elements can be directly placed and handled on the website. There is a presence of inbuilt frameworks and styling sheets in CSS which can be used and to handle the situation. The animations can be directly created, and effects can be updated using the CSS frameworks. The required flash files can be processed and implemented to create movies and animation in the web pages with the help of CSS properties.
- Web-based Online Community and UI Approach: Based on the current standard, to develop any web-based application or any online community, the application of CSS can be implemented in different ways. There are different stylesheet frameworks present in CSS and those can be implemented easily over here. Using CSS, the styling of the own online community can be developed. There are also different add-ins present and those can be incorporated by the uses of CSS frameworks to develop the look and feel of the web-based community.
- End-User and Server-side Representation: CSS files can directly interact with the user end server-side response and can be used for web server-side interphase styling purposes. This allows producing better web representation from the end user's perspective.

CSS with HTML or XHTML

CSS can be applied to HTML or XHTML using three methods: linked, embedded, and inline. In the linked method, the CSS is stored in a separate file, instead of directly in the HTML page. In

the embedded method, CSS is stored as part of the HTML page, in the header section. In the inline method, CSS is stored directly in the style attributes of the HTML tags, as

```
<div style="font-weight:bold">Bold Font</div>.
```

The neatest method is probably the linked one, but the other ones are convenient and quick in the phases of prototyping a web page. The embedded and inline methods do not require having a separate file. The inline method saves you the trouble of considering what CSS classes your document should have. For a larger site, in which many web pages share the same styling, and in which the styling should be customizable by the user, the linked method is the only viable option.

Introduction to CSS3

Cascading Style Sheets (CSS) is a language that is used to illustrate the look, style, and format of a document written in any markup language. In simple words, it is used to style and organize the layout of Web pages. CSS3 is the latest version of an earlier CSS version, CSS2.

A significant change in CSS3 in comparison to CSS2 is the introduction of modules. The benefit of this functionality is that it allows the specification to be finalized and accepted faster, as segments are finalized and accepted in portions. Also, this allows the browser to support segments of the specification.

Some of the key modules of CSS3 are:

- Box model
- Image values and replaced content
- Text effects
- Selectors
- Backgrounds and borders
- Animations

-
- User interface (UI)
 - Multiple column layouts
 - 2D/3D transformations



Image 1: CSS3

Reference- https://commons.wikimedia.org/wiki/File:CSS3_logo_and_wordmark.svg

The features of the CSS3 are as follows:

- Selectors: Selectors allow the designer to select on more precise levels of the web page. They are structural pseudo-classes that perform partial matches to help match attribute and attribute values. New selectors target a pseudo-class to style the elements targeted in

the URL. Selectors also include a checked pseudo-class to style checked elements such as checkboxes and radio buttons.

- Text Effects and Layout: With CSS3, we can change the justification of text, whitespace adjustment of the document and also style the hyphenation of words.
- First-Letter and First-Line Pseudo-Classes: CSS 3 includes properties that help with kerning (adjusting the spacing between characters to achieve a visually pleasing effect) and with the positioning of drop-caps (large decorative capital letter at the starting of a paragraph).
- Paged Media and Generated Content: CSS 3 has additional choices in Paged Media, such as page numbers and running headers and footers. There are additional properties for printing Generated Content as well, like properties for cross-references and footnotes.
- Multi-Column Layout: This feature includes properties to allow designers to present their content in multiple columns with options like the column-count, column-gap, and column-width.

Important Properties of CSS3

- border-radius : The border-radius property defines the radius of the element's corners. This property allows you to add rounded corners to elements. This property can have from one to four values.
 1. Four values - border-radius: 15px 50px 30px 5px; (first value applies to top-left corner, second value applies to top-right corner, third value applies to bottom-right corner, and fourth value applies to bottom-left corner).
 2. Three values - border-radius: 15px 50px 30px; (first value applies to top-left corner, second value applies to top-right and bottom-left corners, and third value applies to bottom-right corner).
 3. Two values - border-radius: 15px 50px; (first value applies to top-left and bottom-right corners, and the second value applies to top-right and bottom-left corners).
 4. One value - border-radius: 15px; (the value applies to all four corners, which are rounded equally).

-
- **box-shadow:** The box-shadow property attaches one or more shadows to an element. To attach more than one shadow to an element, add a comma-separated list of shadows.
 - **text-stroke:** The text-stroke property is an experimental property providing decoration options for a text. It is a shorthand for the following properties:
 - **text-stroke-width**
 - **text-stroke-color**

There is the text-fill-color property, which overrides the color property, allowing for a graceful fallback to a different text color in browsers that do not support the text-stroke property.

- **text-overflow:** The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. It can be clipped, display an ellipsis (...), or display a custom string.

Both of the following properties are required for text-overflow:

- **white-space: nowrap;**
- **overflow: hidden;**
- **resize:** The resize property defines if (and how) an element is resizable by the user.

Note: The resize property does not apply to inline elements or to block elements where overflow="visible". So, make sure that overflow is set to "scroll", "auto", or "hidden".

- **transition:** The transition property is a shorthand property for:
 - **transition-property:** The transition-property property specifies the name of the CSS property the transition effect is for (the transition effect will start when the specified CSS property changes).

A transition effect could typically occur when a user hovers over an element.

Note: Always specify the transition-duration property, otherwise the duration is 0, and the transition will have no effect.

- transition-duration: The transition-duration property specifies how many seconds (s) or milliseconds (ms) a transition effect takes to complete.
- transition-timing-function: The transition-timing-function property specifies the speed curve of the transition effect.

This property allows a transition effect to change speed over its duration.

- transition-delay: The transition-delay property specifies when the transition effect will start. The transition-delay value is defined in seconds (s) or milliseconds (ms).
- background-size: The background-size property specifies the size of the background images.

There are four different syntaxes you can use with this property: the keyword syntax ("auto", "cover" and "contain"), the one-value syntax (sets the width of the image (height becomes "auto")), the two-value syntax (first value: width of the image, second value: height), and the multiple background syntax (separated with comma).

- padding: An element's padding is the space between its content and its border.

The padding property is a shorthand property for:

- padding-top
- padding-right
- padding-bottom
- padding-left

Note: Padding creates extra space within an element, while margin creates extra space around an element.

This property can have from one to four values.

If the padding property has four values:

- padding:10px 5px 15px 20px;
 - top padding is 10px
 - right padding is 5px
 - bottom padding is 15px
 - left padding is 20px

If the padding property has three values:

- padding:10px 5px 15px;
 - top padding is 10px
 - right and left padding are 5px
 - bottom padding is 15px

If the padding property has two values:

- padding:10px 5px;
 - top and bottom padding are 10px
 - right and left padding are 5px

If the padding property has one value:

- padding:10px;
 - all four paddings are 10px

Note: Negative values are not allowed.

- margin : The margin property sets the margins for an element, and is a shorthand property for the following properties:
 - margin-top

-
- margin-right
 - margin-bottom
 - margin-left

If the margin property has four values:

- margin: 10px 5px 15px 20px;
 - top margin is 10px
 - right margin is 5px
 - bottom margin is 15px
 - left margin is 20px

If the margin property has three values:

- margin: 10px 5px 15px;
 - top margin is 10px
 - right and left margins are 5px
 - bottom margin is 15px

If the margin property has two values:

- margin: 10px 5px;
 - top and bottom margins are 10px
 - right and left margins are 5px

If the margin property has one value:

- margin: 10px;
 - all four margins are 10px

Note: Negative values are allowed.

Limitations of CSS

Limitation of CSS

CSS has various limitations as a programming language that's are as follows:

- CSS cannot perform any logical operations like if/else or for/while or +/-.
- We can not read any files using CSS.
- It can not interact with databases.
- CSS can not request a web page.

Overcome the limitations of CSS

There is a lack of media types in CSS1. In other words, CSS1 is primarily a screen-device language, intended to be used to put content onto a computer monitor. There is some thought toward paged media, like printouts, but not much. (Despite this, CSS1 is not a pixel-perfect control mechanism.) In an effort to overcome this limitation, CSS2 introduces media types, which makes it possible to create separate style sheets that are applied to a document depending on its display media. CSS2 also introduces properties and behavior specifically aimed at paged media and aural media.

CSS2 Introduced Media Types

The `@media` rule, introduced in CSS2, made it possible to define different style rules for different media types.

Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.

Unfortunately, these media types never got a lot of support from devices, other than the print media type.

CSS3 Introduced Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries is a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Advantages of CSS

Advantages of CSS

Advantages of CSS are as follow:

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, etc.

The following are the advantages of CSS –

- CSS saves time - You can write CSS once and then reuse the same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many web pages as you want.
- Pages load faster - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
- Easy maintenance - To make a global change, simply change the style, and all the elements in all the web pages will be updated automatically.
- Superior styles to HTML - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- Multiple Device Compatibility - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cellphones or for printing.
- Global web standards – Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.

Web application using CSS

Inside the <body> element of our website we will use our "Layout Picture" and create:

- A navigation bar
- A slide show
- A header
- Some sections and articles

- A footer

Semantic Elements

HTML5 introduced several new semantic elements. Semantic elements are important to use because they define the structure of web pages and help screen readers and search engines to read the page correctly.

These are some of the most common semantic HTML elements:

- The <section> element can be used to define a part of a website with related content.
- The <article> element can be used to define an individual piece of content.
- The <header> element can be used to define a header (in a document, a section, or an article).
- The <footer> element can be used to define a footer (in a document, a section, or an article).
- The <nav> element can be used to define a container of navigation links.

The Navigation Bar: On our "Layout Draft" we have a "Navigation bar". We can use a <nav> or <div> element as a container for the navigation links.

- The w3-bar class is a container for navigation links.
- The w3-black class defines the color of the navigation bar.
- The w3-bar-item and w3-button class styles the navigation links inside the bar.

Slide Show: On the "Layout Draft" we have a "Slide show". For the slide show we can use a <section> or <div> element as a picture container.

Sections and Articles: Looking at the "Layout Draft" we can see that the next step is to create articles. First we will create a <section> or <div> element containing band information:

- The w3-container class takes care of standard padding.
- The w3-center class centers on the content.
- The w3-wide class provides a wider heading.

-
- The w3-opacity class provides text transparency.
 - The max-width style sets a maximum width of the band description section

Footer : Finally we will use a <footer> or <div> to create a footer.

Three ways to Integrate CSS

There are three ways of inserting a style sheet:

- Internal or embedded— add <style> tag in the <head> section of HTML document.
- External— link the HTML sheet to a separate. CSS file.
- Inline— apply CSS rules for specific elements.

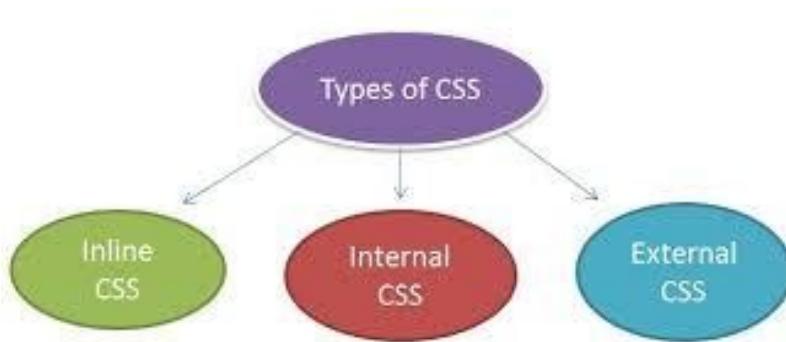


Image: Types of CSS

Reference: <https://www.c-sharpcorner.com/UploadFile/e6a884/types-of-css/>

Inline CSS

An inline style may be used to apply a unique style for a single element.

The styling is done inside the element itself.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Inline CSS is used to style a specific HTML element. For this CSS style, you'll only need to add the style attribute to each HTML tag, without using selectors.

We can set multiple properties inside it and all properties are separated by (;)semi-colon.

Here is the generic syntax:

```
<element style="....style rules....">
```

Internal CSS

The internal or embedded CSS is used to add a unique style for a single document. It is defined in the <head> section of the HTML page inside the <style> tag.

This CSS style is an effective method of styling a single page. However, using this style for multiple pages is time-consuming as you need to put CSS rules to every page of your website.

Here's how you can use internal CSS:

- Open your HTML page and locate <head> opening tag.
- Put the following code right after the <head> tag

Rules defined using this syntax will be applied to all the elements available in the document.

Here is the generic syntax:

```
<head>
<style type="text/css" media="...">
Style Rules
.....
</style>
</head>
```

External CSS

The element can be used to include an external stylesheet file in your HTML document.

External CSS contains a separate CSS file that contains only style property with the help of tag attributes (For example class, id, heading, ... etc). CSS property is written in a separate file with .css extension and should be linked to the HTML document using the link tag. This means that for each element, style can be set only once and that will be applied across web pages.

Here is the generic syntax of including external CSS file:

```
<head>
<link type="text/css" href="..." media="..." />
</head>
```

Class and ID as a selector

- The CSS ID Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Note: An id name cannot start with a number!

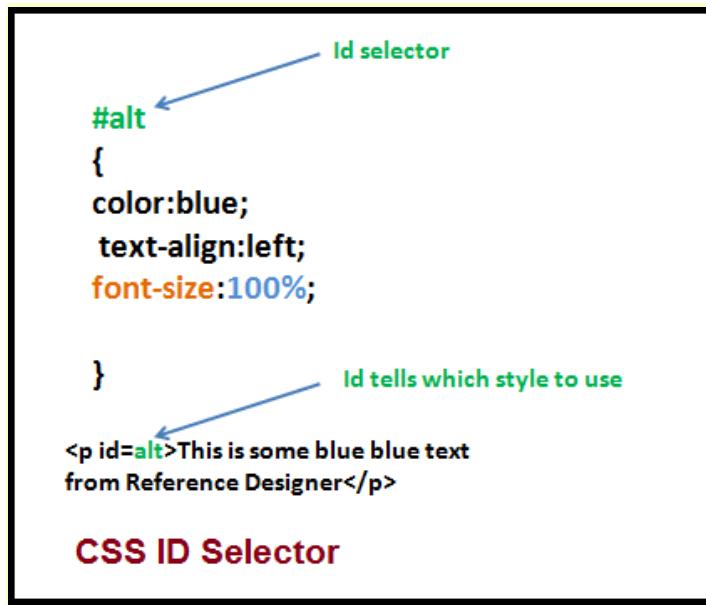


Image : CSS ID Selector

Reference: http://referencedesigner.com/tutorials/css/css_tutorial_03.php

- The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

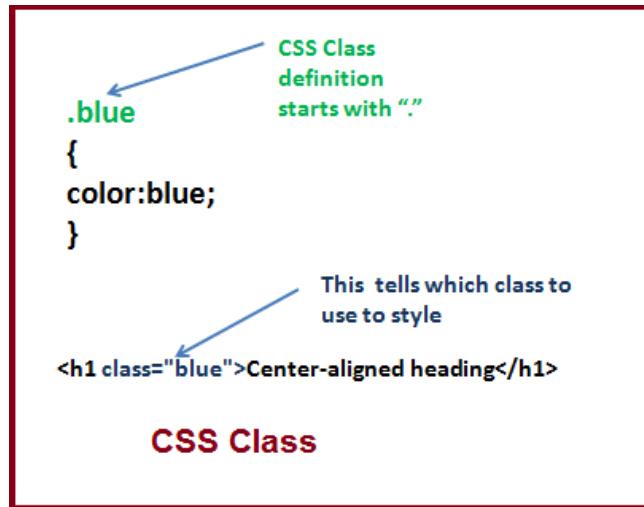


Image: class selector

Reference: http://www.referencedesigner.com/tutorials/css/css_tutorial_04.php

- The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

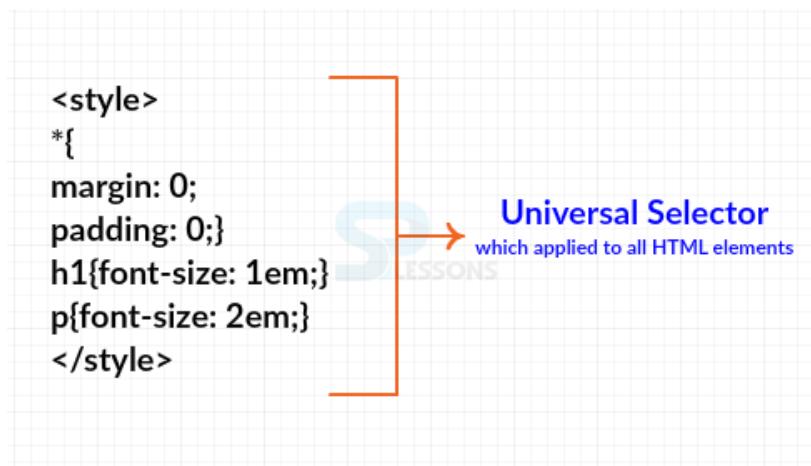


Image: universal selector

Reference: <https://www.splessons.com/lesson/css-class-and-id/>

-
- The CSS element Selector

The element selector selects HTML elements based on the element name.

```
h1 {  
    color: blue;  
    background-color: yellow;  
}  
  
p {  
    color: red;  
}
```

Image: element selector

Reference: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors

CSS Combinator

A Combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

- Descendant Selector

The descendant selector matches all elements that are descendants of a specified element. They do not need to be direct children to match.

Descendant Selector

```
<style>
  p a {
    font-family: sans-serif;
    font-size: 15pt;
    line-height: 150%;
  }
</style>
<p>  Lorem ipsum dolor sit amet, consectetur adipiscing elit..
  Nam pulvinar nunc ac magna aliquam quis sodales dui nunc
  sit elementum. <a href="page1.html">Donec eu nisi turpis,</a>
  sit amet rutrum leo.
</p>
Click <a href="page2.html">here</a>
```

12

Image: descendant selector

Reference: <https://www.slideshare.net/NosheenQamar/web-engineering-introduction-to-css-53957593>

- Child Selector

The child combinator is a greater-than symbol (>). The child selector selects all elements that are the children of a specified element.

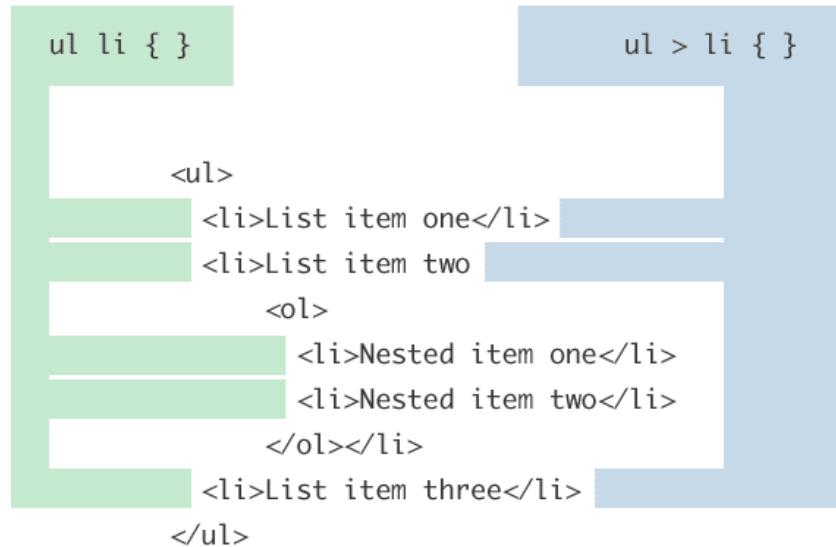


Image : child selector

Reference : <https://css-tricks.com/child-and-sibling-selectors/>

- Adjacent Sibling Selector

The adjacent sibling combinator (+) separates two selectors and matches the second element only if it immediately follows the first element, and both are children of the same parent

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

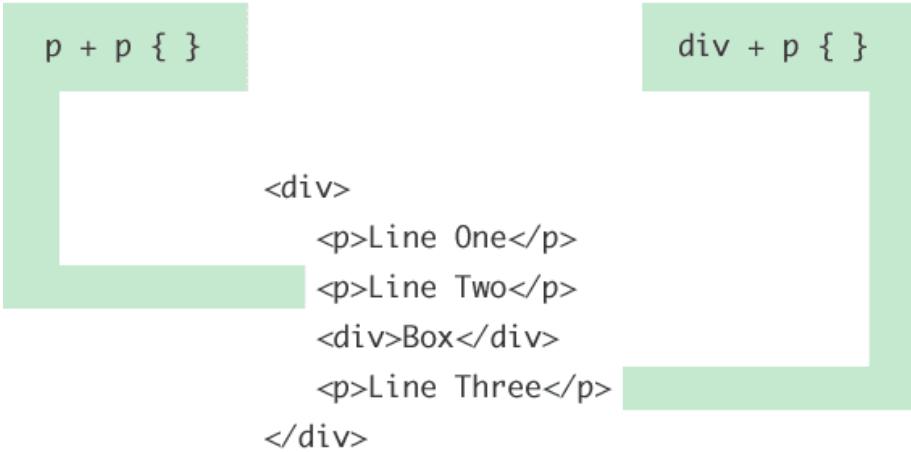


Image : Adjacent sibling selector

Reference : <https://css-tricks.com/child-and-sibling-selectors/>

- General Sibling Selector

The general sibling combinator (\sim) separates two selectors and matches the second element only if it follows the first element (though not necessarily immediately), and both are children of the same parent element.

The general sibling selector selects all elements that are siblings of a specified element.

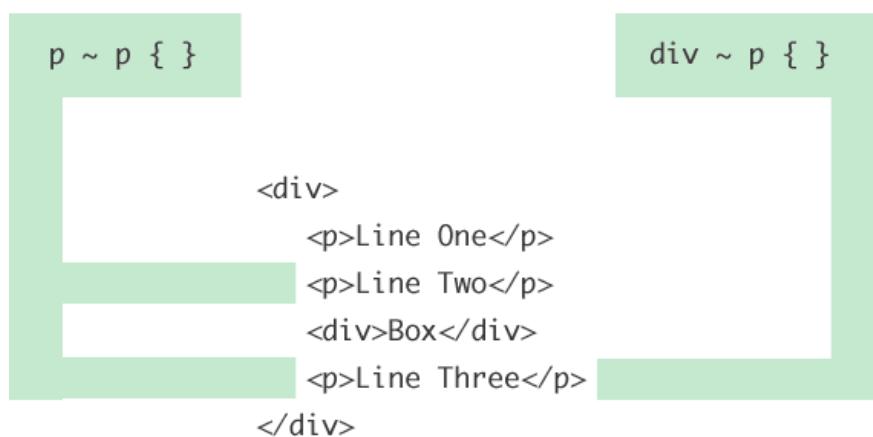


Image : General sibling selector

Reference : <https://css-tricks.com/child-and-sibling-selectors/>

CSS Pseudo-classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

The syntax of pseudo-classes:

```
selector:pseudo-class {  
    property: value;  
}
```

- Anchor Pseudo-classes

Using CSS pseudo-class selectors, namely, :active, :hover, :link and :visited we can style different states of a link/anchor. For proper functionality, the order of these selectors should be –

- :link
- :visited
- :hover
- :active

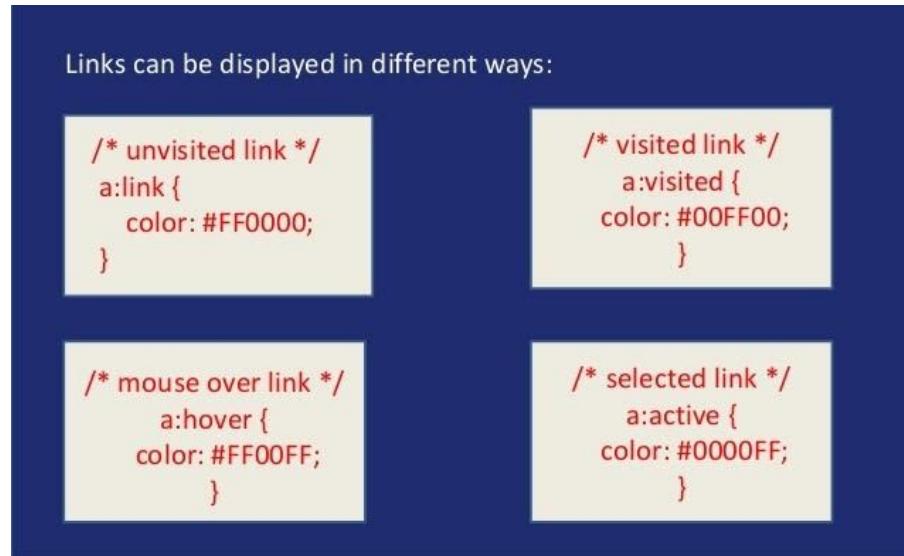


Image : Anchor Pseudo-Class

Reference: <https://www.slideshare.net/webtechlearningchd/css-pseudoclasses>

- The `:first-child` pseudo-class

The `:first-child` pseudo-class matches a specified element that is the first child of another element and adds a special style to that element that is the first child of some other element.

```
p:first-child {  
    border:1px solid red;  
}
```

```
✓ <p>1</p>  
<div>  
✓ <p>1.1</p>  
<p>1.2</p>  
<div>  
✓ <p>1.1.1</p>  
</div>  
</div>
```

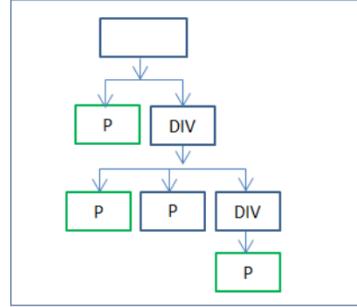


Image : :first-child Pseudo-class

Reference : <https://www.pinterest.com/pin/855332154209291153/>

- The :lang Pseudo-class

The language pseudo-class :lang, allows constructing selectors based on the language setting for specific tags.

This class is useful in documents that must appeal to multiple languages that have different conventions for certain language constructs. For example, the French language typically uses angle brackets (< and >) for quoting purposes, while the English language uses quote marks (' and ').

In a document that needs to address this difference, you can use the :lang pseudo-class to change the quote marks appropriately.

```
:lang(en) > q { quotes: '\201C' '\201D' '\2018' '\2019'; }  
:lang(fr) > q { quotes: '« ' '»'; }  
:lang(de) > q { quotes: '„ „ „ „'; }
```

CSS Pseudo-elements

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

The syntax of pseudo-elements:

```
selector::pseudo-element {  
    property:value;  
}
```

- The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

The ::first-line pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration

-
- vertical-align
 - text-transform
 - line-height
 - clear
-
- The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

The ::first-letter pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-letter pseudo-element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

- The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.

- The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.

- The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to ::selection: color, background, cursor, and outline.

CSS Attribute Selectors

- CSS [attribute] Selector

The [attribute] selector is used to select elements with a specified attribute.

- CSS [attribute="value"] Selector

The [attribute="value"] selector is used to select elements with a specified attribute and value.

-
- CSS [attribute \sim "value"] Selector

The [attribute \sim "value"] selector is used to select elements with an attribute value containing a specified word.

- CSS [attribute $=$ "value"] Selector

The [attribute $=$ "value"] selector is used to select elements with the specified attribute starting with the specified value.

- CSS [attribute \wedge "value"] Selector

The [attribute \wedge "value"] selector is used to select elements whose attribute value begins with a specified value.

- CSS [attribute\$="value"] Selector

The [attribute\$="value"] selector is used to select elements whose attribute value ends with a specified value.

- CSS [attribute*="value"] Selector

The [attribute*="value"] selector is used to select elements whose attribute value contains a specified value.

Merits and Demerits of all kinds of CSS

Inline CSS

- Merits of Inline CSS
 - Testing: Many web designers use Inline CSS when they begin working on new projects, this is because it's easier to scroll up in the source, rather than change the source file. Some also use it to debug their pages, if they encounter a problem that is not so easily fixed. This can be done in combination with the Important rule of CSS.
 - Quick-fixes: There are times where you would just apply a direct fix in your HTML source, using the style attribute, but you would usually move the fix to the relevant files when you are either able or got the time.
 - Smaller Websites: The website such as Blogs where there are only a limited number of pages, using Inline CSS helps users and service providers.
 - Lower the HTTP Requests: The major benefit of using Inline CSS is lower HTTP Requests which means the website loads faster than External CSS.

- Demerits of Inline CSS
 - Overriding: Because they are the most specific in the cascade, they can override things you didn't intend them to.
 - Every Element: Inline styles must be applied to every element you want them on. So if you want all your paragraphs to have the font family "Arial" you have to add an inline style to each `<p>` tag in your document. This adds both maintenance work for the designer and download time for the reader.
 - Pseudo-elements: It's impossible to style pseudo-elements and classes with inline styles. For example, with external and internal style sheets, you can style the visited, hover, active, and link color of an anchor tag. But with an inline style all you can style is the link itself, because that's what the style attribute is attached to.

Internal CSS

- Merits of Internal CSS
 - Cache Problem: Internal styles will be read by all browsers unless they are hacked to hide from certain ones. This removes the ability to use media=all or @import to hide styles from old, crotchety browsers like IE4 and NN4.
 - Pseudo-elements: It's impossible to style pseudo-elements and classes with inline styles. With Internal style sheets, you can style the visited, hover, active, and link color of an anchor tag.
 - One style of the same element: Internal styles need not be applied to every element. So if you want all your paragraphs to have the font family "Arial" you have to add an Inline style <p> tag in the Internal Style document.
 - No additional downloads: No additional downloads necessary to receive style information or we have less HTTP Request.

- Demerits of Internal CSS
 - Multiple Documents: This method can't be used, if you want to use it on multiple web pages.
 - Slow Page Loading: As there is less HTTP Request but by using the Internal CSS the page load slow as compared to Inline and External CSS.
 - Large File Size: While using the Internal CSS the page size increases but it helps only to Designers while working offline but when the website goes online it consumes much time as compared to offline.

External CSS

- Merits of External CSS
 - Since the CSS code is in a separate document, your HTML files will have a cleaner structure and are smaller in size.
 - You can use the same .css file for multiple pages.

- Disadvantages of External CSS:
 - Your pages may not be rendered correctly until the external CSS is loaded.
 - Uploading or linking to multiple CSS files can increase your site's download time.

Introduction to Bootstrap

What is Bootstrap ?

It is an open-source and free framework of CSS which helps in directing a responsive device friendly mobile-first front-end web page development tool. Bootstrap includes the CSS (Cascading Style Sheets), and an optional JavaScript supported design template (plug-ins) that deals with typography, implementation of buttons, forms and various other components for the user interface. This framework helps in faster web development and supports developers in creating responsive web pages at a quicker pace.

Twitter Blueprint was the first name of Bootstrap and was developed at Twitter by Mr. Mark Otto and Jacob Thornton. It got released as an open-source product in August 2011 on GitHub. The framework was mainly designed for encouraging reliability and uniformity of web pages across internal tools. Before the existence of Bootstrap, for making responsive sites and interface

development, various external libraries were used that not only brought inconsistency but also gave rise to the heavy maintenance burden.

Why Bootstrap ?

- Browser supportive: Every browser supports this Bootstrap Framework.
 - Mobile-first approach: In the Bootstrap 3 framework, there is a preexisting mobile-first style all through the library and not as separate files.
 - Simple and easy to start: If you know HTML and CSS, you can quickly start working with Bootstrap, and its documentation is provided on the official site.
 - Responsive design and looks: Web pages designed using the Bootstrap framework have responsive CSS that can adjust to the screen size of large desktops, notebooks, tablets, and mobiles.
 - Easy customization: It provides some built-in components and functionalities that are easy for customizing.
 - Clean interface or Developers: Bootstrap framework provides a new and consistent result for building user interfaces in web pages.
 - It is an open-source framework with web-based customization.
-

Disadvantages of Bootstrap

- There will be a requirement of lots of style overrides or rewriting files that can thus lead to a lot of time spent on designing and coding the website if the design tends to deviate from the customary design used in Bootstrap.
- You would have to go the extra mile while creating a design otherwise all the websites will look the same if you don't do heavy customization.
- Styles are verbose and can lead to lots of output in HTML which is not needed.

-
- JavaScript is tied to jQuery and is one of the commonest libraries which thus leaves most of the plugins unused.
 - Non-compliant HTML.

Benefits of Bootstrap Framework

- It produces less cross-browser bugs.
- It is a consistent framework supported by all the browsers plus CSS based compatibility fixes.
- It is lightweight and hence a widely used framework for creating responsive sites.
- Looks, structure, and styles can be customized as per requirement.
- A simple and effective grid system.

What Bootstrap Package Includes?

- Scaffolding: Bootstrap provides a basic structure with Grid System, link styles, background. This is covered in detail in the section Bootstrap Basic Structure
- CSS: Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system. This is covered in detail in the section Bootstrap with CSS.
- Components: Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, popovers, and much more. This is covered in detail in the section Layout Components.
- JavaScript Plugins: Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one. This is covered in details in the section Bootstrap Plugins.

-
- **Customize:** You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

Ways of using Bootstrap

There are two ways of using Bootstrap

- one by using Bootstrap CDN where the Bootstrap code is hosted somewhere else, and you can use it by simply taking the link and include them in your project. So the actual Bootstrap functionalities and code is hosted somewhere in the CDN, and so you are simply using its link to connect your project to implement the responsive features provided by it.

Bootstrap CDN

The folks over at MaxCDN graciously provide CDN support for Bootstrap's CSS and JavaScript. Just use these Bootstrap CDN links.

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRakycuHARg320mUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduH9SVrLvRYooPp2bwYgmgJQIXw1/Sp" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfulWxZxUPnPJA712mCWNIpG9mGCD8wGNicPD7Txa" crossorigin="anonymous"></script>
```

Image : bootstrap CDN

Reference: <https://getbootstrap.com/docs/3.3/getting-started/>

-
- In a second way, you need to download the Bootstrap package in your local machine, and with the use of this downloaded Bootstrap files, you can implement the features of it in your site. For this approach the internet connectivity during your project run is not required because the bootstrap files are available in your local machine.

Download

Bootstrap (currently v3.3.7) has a few easy ways to quickly get started, each one appealing to a different skill level and use case. Read through to see what suits your particular needs.

Bootstrap

Compiled and minified CSS, JavaScript, and fonts. No docs or original source files are included.

[Download Bootstrap](#)

Source code

Source Less, JavaScript, and font files, along with our docs. **Requires a Less compiler and some setup.**

[Download source](#)

Sass

Bootstrap ported from Less to Sass for easy inclusion in Rails, Compass, or Sass-only projects.

[Download Sass](#)

Image: bootstrap Download

Reference: <https://getbootstrap.com/docs/3.3/getting-started/>

Bootstrap Grid System

- What is the Grid?

In graphic design, a grid is a structure (usually two-dimensional) made up of a series of intersecting straight (vertical, horizontal) lines used to structure content. It is widely used to design layout and content structure in print design. In web design, it is a very effective method to create a consistent layout rapidly and effectively using HTML and CSS.

To put it simple words grids in web design organize and structure content, makes websites easy to scan and reduces the cognitive load on users.

- What is Bootstrap Grid System?

Bootstrap includes a responsive, mobile-first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

Let us understand the above statement. Bootstrap 3 is mobile-first in the sense that the code for Bootstrap now starts by targeting smaller screens like mobile devices, tablets, and then “expands” components and grids for larger screens such as laptops, desktops.

- Mobile-first strategy

- Content
 - Determine what is most important.
- Layout
 - Design to smaller widths first.
 - Base CSS address mobile device first; media queries address for tablet, desktops.
- Progressive Enhancement
 - Add elements as screen size increases.

- Working of Bootstrap Grid System

Grid systems are used for creating page layouts through a series of rows and columns that house your content.

Here's how the Bootstrap grid system works:

- Rows must be placed within a .container class for proper alignment and padding.
- Use rows to create horizontal groups of columns.

-
- Content should be placed within columns, and only columns may be immediate children of rows.
 - Predefined grid classes like .row and .col-xs-4 are available for quickly making grid layouts. LESS mixins can also be used for more semantic layouts.
 - Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and last column via negative margin on .rows.
 - Grid columns are created by specifying the number of twelve available columns you wish to span. For example, three equal columns would use three .col-xs-4.
-
- Grid options

The following table summarizes aspects of how Bootstrap grid system works across multiple devices:

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints
Max container width	None (auto)	750px	970px	1170px
Class prefix # of Columns	.col-xs-12	.col-sm-12	.col-md-12	.col-lg-12
Max column width	Auto 30px	60px 30px	78px 30px	95px 30px

Gutter width	(15px on each side of a column)			
Nestable	Yes	Yes	Yes	Yes
Offsets	Yes	Yes	Yes	Yes
Column Ordering	Yes	Yes	Yes	Yes

- Basic Grid Structure

Following is basic structure of Bootstrap grid:

```
<div class="container">
<div class="row">
<div class="col-*-*"></div>
<div class="col-*-*"></div>
</div>
<div class="row">...</div>
</div>
<div class="container">....
```

Able To Create Own Account in Cloud and Hosting

In this section, we will read about:

- Introduction cloud computing
- Amazon web server (AWS)
- Industrial Faculty-webinar, How cloud and Webinar works

Introduction to Cloud Computing

Cloud Overview

Cloud computing is the on-demand delivery of computing power, database storage, applications, and other IT resources through a cloud services platform via the Internet with pay-as-you-go pricing. Whether you are running applications that share photos to millions of mobile users or you're supporting the critical operations of your business, a cloud services platform provides rapid access to flexible and low-cost IT resources. With cloud computing, you don't need to make large upfront investments in hardware and spend a lot of time on the heavy lifting of managing that hardware. Instead, you can provision exactly the right type and size of computing resources you need to power your newest bright idea or operate your IT department. You can access as many resources as you need, almost instantly, and only pay for what you use.



Image 1: Cloud Computing Scenario
Reference: <https://p0.pikrepo.com/preview/21/464/cloud-computing-saas-software-as-a-service.jpg>

Cloud computing provides a simple way to access servers, storage, databases and a broad set of application services over the Internet. A cloud services platform, such as Amazon Web Services,

owns and maintains the network-connected hardware required for these application services, while you provision and use what you need via a web application.

Cloud computing is a term used to describe both a platform and type of application. A Cloud computing platform dynamically provisions, configures, reconfigures, and de-provision servers as needed. Servers in the Cloud can be physical machines or virtual machines. Advanced Clouds typically include other computing resources such as storage area networks (SANs), network equipment, firewall, and other security devices.

Cloud computing also describes applications that are extended to be accessible through the Internet. These Cloud applications use large data centers and powerful servers that host Web applications and Web services. Anyone with a suitable Internet connection and a standard browser can access a Cloud application.

Essential Characteristics

- On-demand Self-Service
- Broad Network Access
- Resource Pooling
- Rapid Elasticity

Six Advantages of Cloud Computing

- ***Trade capital expense for variable expense*** - Instead of having to invest heavily in data centers and servers before you know how you're going to use them, you can pay only when you consume computing resources, and pay only for how much you consume.
- ***Benefit from massive economies of scale*** - By using cloud computing, you can achieve a lower variable cost than you can get on your own. Because usage from hundreds of thousands of customers is aggregated in the cloud, providers such as AWS can achieve higher economies of scale, which translates into lower pay-as-you-go prices.
- ***Stop guessing capacity*** - Eliminate guessing on your infrastructure capacity needs. When you make a capacity decision prior to deploying an application, you often end up either sitting on expensive idle resources or dealing with limited capacity. With cloud computing, these problems go away. You can access as much or as little capacity as you need, and scale up and down as required with only a few minutes' notices.

- **Increase speed and agility** - In a cloud computing environment, new IT resources are only a click away, which means that you reduce the time to make those resources available to your developers from weeks to just minutes. This results in a dramatic increase in agility for the organization since the cost and time it takes to experiment and develop is significantly lower.
- **Stop spending money running and maintaining data centers** - Focus on projects that differentiate your business, not the infrastructure. Cloud computing lets you focus on your own customers, rather than on the heavy lifting of racking, stacking, and powering servers.
- **Go global in minutes** - Easily deploy your application in multiple regions around the world with just a few clicks. This means you can provide lower latency and a better experience for your customers at a minimal cost.

Cloud Computing Service Delivery Models

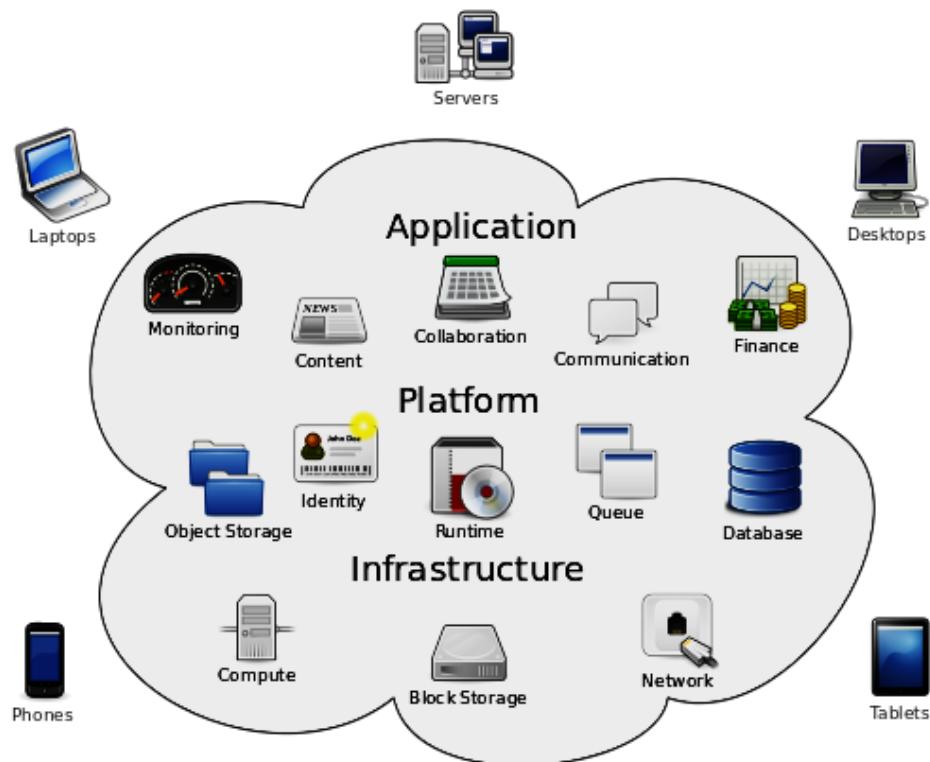


Image 2: Cloud Computing Service Models-1

Reference: https://upload.wikimedia.org/wikipedia/commons/thumb/b/b5/Cloud_computing.svg/530px-Cloud_computing.svg.png

- **Cloud Infrastructure as a Service (IaaS)** - Infrastructure as a Service (IaaS) contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.
- **Cloud Platform as a Service (PaaS)** - Platform as a Service (PaaS) removes the need for your organization to manage the underlying infrastructure (usually hardware and operating systems) and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.
- **Cloud Software as a Service (SaaS)** - Software as a Service (SaaS) provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering, you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is a web-based email which you can use to send and receive email without having to manage feature additions to the email product or maintain the servers and operating systems that the email program is running on.

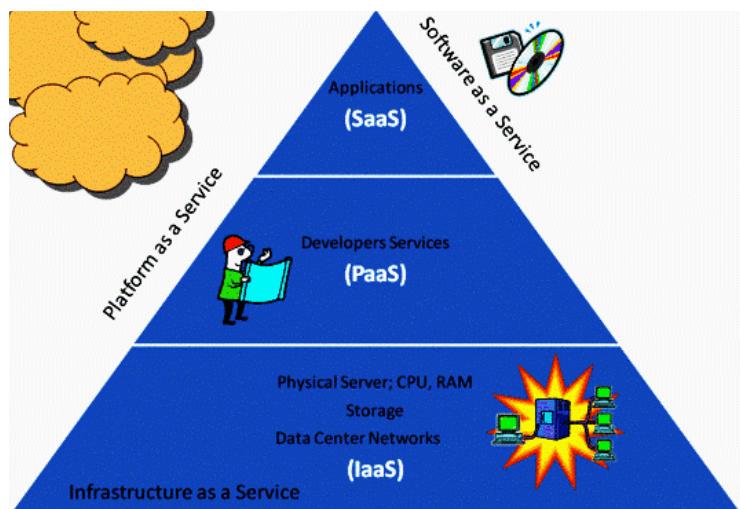


Image 3: Cloud Service Models-2

Reference: https://upload.wikimedia.org/wikipedia/commons/thumb/6/60/Cloud_Services.gif/640px-Cloud_Services.gif

Cloud Computing Deployment Models

- **All-in Cloud** - A cloud-based application is fully deployed in the cloud and all parts of the application run in the cloud. Applications in the cloud have either been created in the cloud or have been migrated from an existing infrastructure to take advantage of the benefits of cloud computing. Cloud-based applications can be built on low-level infrastructure pieces or can use higher-level services that provide abstraction from the management, architecting, and scaling requirements of core infrastructure.
- **Hybrid Cloud** - A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud. The most common method of hybrid deployment is between the cloud and existing on-premises infrastructure to extend, and grow, an organization's infrastructure into the cloud while connecting cloud resources to the internal system. For more information on how AWS can help you with your hybrid deployment, please visit our hybrid page.
- **On-premises** - The deployment of resources on-premises, using virtualization and resource management tools, is sometimes called the “private cloud.” On-premises deployment doesn’t provide many of the benefits of cloud computing but is sometimes sought for its ability to provide dedicated resources. In most cases, this deployment model is the same as legacy IT infrastructure while using application management and virtualization technologies to try and increase resource utilization.

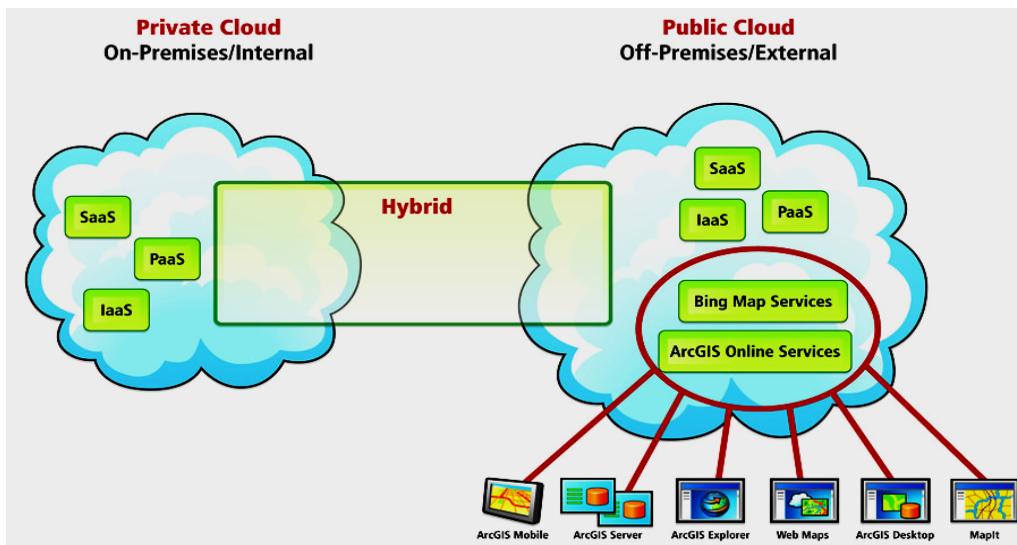


Image 4: Cloud Deployment Models

Reference: https://upload.wikimedia.org/wikipedia/commons/2/23/Hybrid_cloud.jpg

Cloud Computing Uses

- Application Hosting
- Backup and Storage
- Content Delivery
- Websites
- Enterprise IT
- Databases

Important Cloud Terminology

- ***High Availability*** – Accessible whenever you need it
- ***Fault Tolerance*** – Ability to withstand a certain amount of failure and still remain functional
- ***Scalability*** – Ability to easily grow in size, capacity and/ or scope when required. Growth is usually based on demand.
- ***Elasticity*** – Ability to grow or scale when required and reduce in size when resources are no longer needed.

Amazon Web Services (AWS)

AWS is a secure cloud web service provider with more than 100 different services that include a solution for Compute power, Storage, Databases, Networking, Security, analytics, IoT, Mobile service, Developer Tools, etc. A Web Service is a piece of software that makes itself available over the internet and uses standard formats such as JSON or XML for the request and response of API interaction.

Access to AWS Services

Access to AWS services can be made through the following interfaces:

1. **AWS Management Console** - Access and manage Amazon Web Services through the AWS Management Console, a simple and intuitive user interface. You can also use the AWS Console Mobile Application to quickly view resources on the go.



Image 5: AWS Management Console
Reference:

https://upload.wikimedia.org/wikipedia/commons/thumb/5/5b/AWS_Simple_Icons_Non-Service_Specific_AWS_Management_Console.svg/1024px-AWS_Simple_Icons_Non-Service_Specific_AWS_Management_Console.svg.png

2. **AWS Command Line Interface** - The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.



Image 6: AWS Command Line Interface

Reference: <https://d2908q01vomqb2.cloudfront.net/9e6a55b6b4563e652a23be9d623ca5055c356940/2019/09/12/EC2-Instances-Blog.jpg>

3. **Software Development Kits** - Our Software Development Kits (SDKs) simplify using AWS services in your applications with an Application Program Interface (API) tailored to your programming language or platform.

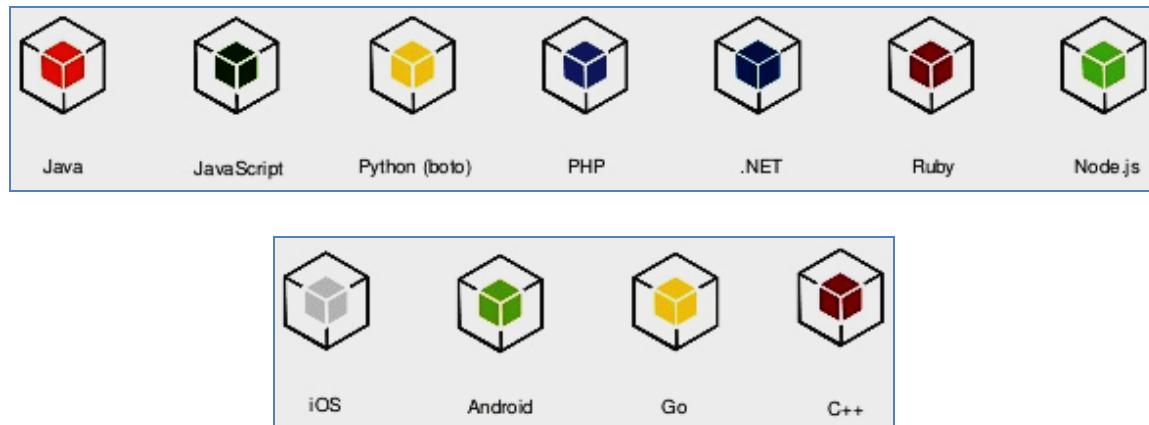


Image 7: AWS SDKs

Reference:

<https://image.slidesharecdn.com/2016-04-28-buildmobileappsusingawssdksandmobilehub-160503090605/95/build-mobile-apps-using-aws-sdks-and-aws-mobile-hub-4-638.jpg?cb=1462266888>

AWS Global Infrastructure

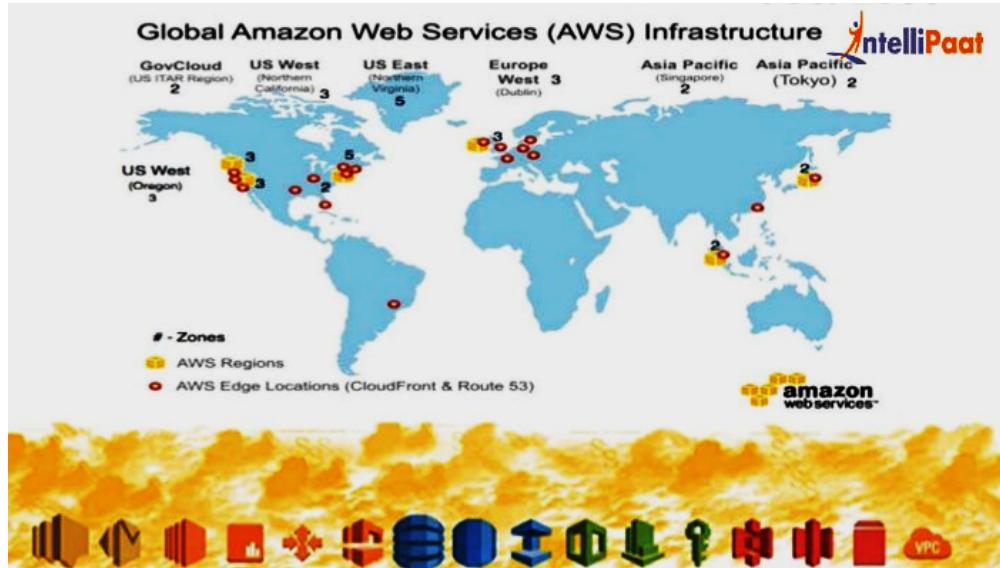


Image 8: AWS Global Infrastructure
Reference: <https://intellipaat.com/mediaFiles/2016/05/AWS-Global-Infrastructure.jpg>

The AWS Global Cloud Infrastructure is the most secure, extensive, and reliable Cloud Computing environment anywhere, on and off the planet. Whether you need to deploy your application workloads across the globe in a single click, or you want to build and deploy specific applications closer to your end-users with single-digit millisecond latency, AWS provides you the cloud infrastructure where and when you need it.

AWS now spans 70 Availability Zones within 22 geographic regions around the world and has announced plans for fifteen more Availability Zones and five more AWS Regions in Indonesia, Italy, Japan, South Africa, and Spain.

Data Centers

A data center is a building, dedicated space within a building, or a group of buildings used to house computer systems and associated components, such as telecommunications and storage systems. Since IT operations are crucial for business continuity, it generally includes redundant or backup components and infrastructure for power supply, data communications connections,

environmental controls (e.g. air conditioning, fire suppression) and various security devices. A large data center is an industrial-scale operation using as much electricity as a small town.



Image 9: AWS Data Center

Reference: <https://static.techspot.com/images2/news/bigimage/2018/10/2018-10-12-image-3.jpg>

The datacenter is a location where actual physical servers, storage media, and other infrastructure elements reside. A typical data center has 50 to 80 thousand physical servers. All data centers are securely designed and are always online. A data center has a strong network among servers, storage racks and is equipped with secure firewalls and IDSs.

Availability Zones

Each Availability Zone (AZ) is made up of one or more data centers, designed for fault isolation, interconnected with other availability zones using high speed and low latency private links. AWS recommends replication of data across more than one availability zone for resiliency.

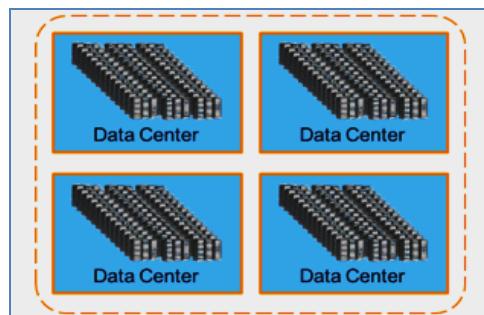


Image 10: AWS Availability Zone
Reference: <https://aws.amazon.com/about-aws/global-infrastructure/>

AWS Regions

An AWS Region is a geographical area made up of two or more Availability Zones. Currently, AWS has 22 regions and declared 5 more announced regions to be active soon. We can enable and control data replication across regions. Communication among regions uses AWS Backbone Network connections infrastructure.

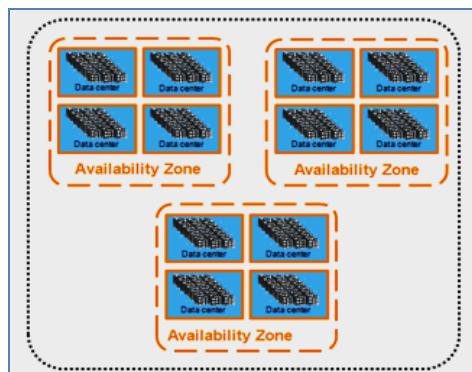


Image 10: AWS Availability Zone
Reference: <https://aws.amazon.com/about-aws/global-infrastructure/>

Edge Locations

An Edge Location is where the user accesses AWS services. It is a global network of 216 points of presence, i.e. 205 edge locations and 11 regional edge caches across 24 countries. It is specifically used with Amazon CloudFront, a global content delivery network (CDN) to deliver content to end-user with reduced latency. Regional edge caches are used for content with infrequent access.

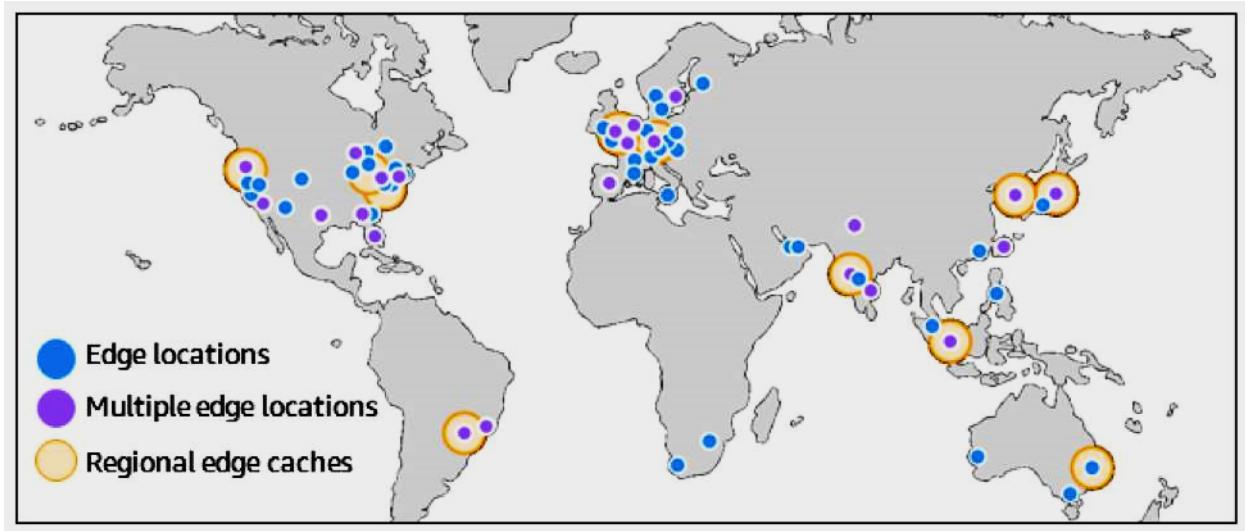


Image 12: AWS Global Edge Locations Network

Reference:

https://res.cloudinary.com/hy4kyit2a/f_auto,f_l_lossy,q_70/learn/modules/aws-cloud/explore-the-aws-global-infrastructure/images/a71a1d65836a142d9435cc46f0997fce_ck-2-kwgcty-003-c-0-z-9-w-8-mye-8-ry-0.png

AWS Infrastructure Features

- Elastic and Scalable:
 - Elastic Infrastructure; dynamic adaption of capacity
 - Scalable Infrastructure; adapts to accommodate growth
- Fault-Tolerant:
 - Continues operating properly in the presence of failure
 - Built-in resiliency of components
- High Availability:
 - High level of operational performance
 - Minimized downtime
 - No human intervention

AWS Core Services

Compute

1. **Amazon EC2:** Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make

web-scale computing easier for developers. Amazon Elastic Compute Cloud (EC2) offers virtual computing environments, known as instances that can be managed by a few mouse clicks or few lines of code. These are virtual server machines that can be acquired on-demand as per the requirement. Amazon EC2 instance types are optimized for different use cases and workload requirements.



Image 13: Amazon EC2

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

The Amazon EC2 simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances (called Amazon EC2 instances) to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers and system administrators the tools to build failure resilient applications and isolate themselves from common failure scenarios.

They come in multiple sizes and can be created as application servers, web servers, database server, game server, proxy server, bastion host, mail server, media server, file server or for computing purpose only.

When choosing instances, consider core count, memory, storage, network performance, CPU technologies. There are four ways to pay for Amazon EC2- On-Demand, Spot Instances, Reserved Instance and Dedicated Host.

2. **AWS Lambda:** AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume—there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service—all with zero administration. Just upload your code, and Lambda takes care of everything required to run and scale your code with high availability. You

can set up your code to automatically trigger from other AWS services, or you can call it directly from any web or mobile app.



Image 14: AWS Lambda

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

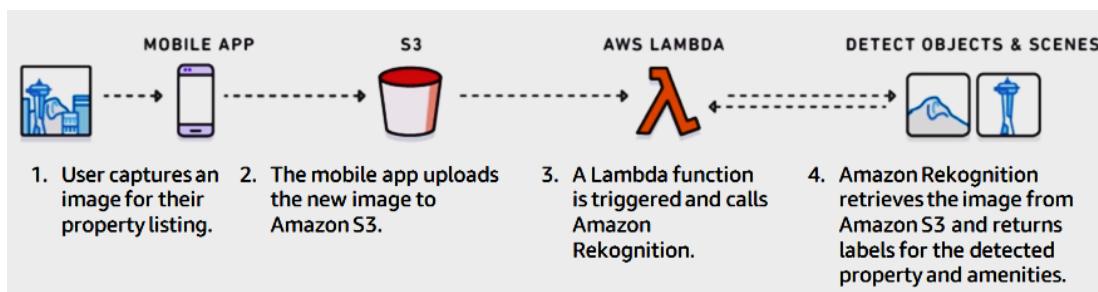


Image 15: AWS Lambda Use Case- Image Recognition Application

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

3. **AWS Elastic Beanstalk:** AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and Internet Information Services (IIS).



Image 16: AWS Elastic Beanstalk

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

You can simply upload your code, and AWS Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, and auto scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.

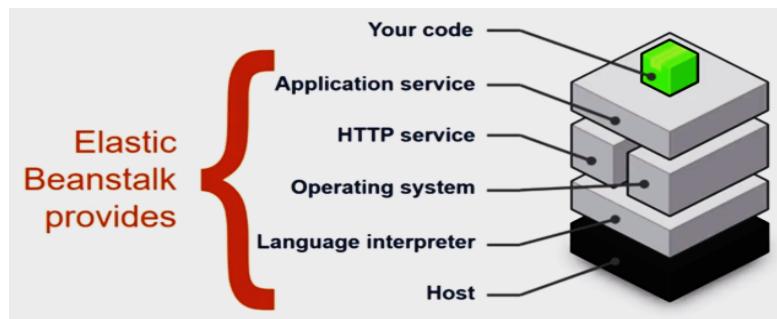


Image 17: AWS Elastic Beanstalk Working

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

4. **Automatic Scaling:** Amazon EC2 Auto Scaling helps you maintain application availability and allows you to automatically add or remove EC2 instances according to conditions you define. You can use the fleet management features of Amazon EC2 Auto Scaling to maintain the health and availability of your fleet. You can also use the dynamic and predictive scaling features of Amazon EC2 Auto Scaling to add or remove EC2 instances. Dynamic scaling responds to changing demand and predictive scaling automatically schedules the right number of EC2 instances based on predicted demand. Dynamic scaling and predictive scaling can be used together to scale faster.

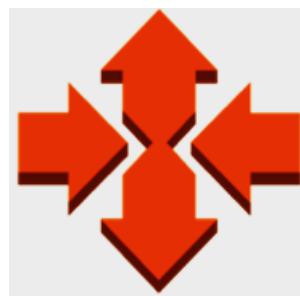


Image 18: Auto Scaling

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

5. **Amazon Lightsail:** Amazon Lightsail is a virtual server to manage simple web servers and application servers. It has all resources needed to jump-start a project. It can launch a site or app powered by AWS in minutes. It provides high reliability and performance.

Amazon Lightsail is designed to be the easiest way to launch and manage a virtual private server with AWS. Lightsail plans include everything you need to jumpstart your project – a virtual machine, SSD- based storage, data transfer, DNS management, and a static IP address – for a low, predictable price.



Image 19: Amazon Lightsail

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

Storage

Amazon Simple Storage Service (Amazon S3) provides developers and IT teams secure, durable, highly-scalable object storage at a very low cost. You can store and retrieve any amount of data, at any time, from anywhere on the web through a simple web service interface. You can write, read, and delete objects containing from zero to 5 TB of data. Amazon S3 is highly scalable, allowing the concurrent read or write access to data by many separate clients or application threads.

1. **Amazon Elastic Block Store (EBS):** Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with Amazon EC2 instances in the AWS Cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads. With Amazon EBS, you can scale your usage up or down within minutes—all while paying a low price for only what you provision.



Image 20: Amazon Elastic Block Store

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

2. **Amazon Simple Storage Service (S3):** Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements. Amazon S3 is designed for 99.99999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world. Amazon S3 offers a range of storage classes designed for different use cases including the following:
 - a. Amazon S3 Standard, for general-purpose storage of frequently accessed data
 - b. Amazon S3 Standard-Infrequent Access (Standard-IA), for long-lived, but less frequently accessed data
 - c. Amazon Glacier, for low-cost archival data



Image 21: Amazon Simple Storage Service

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

3. **Amazon Elastic File System (EFS):** Amazon Elastic File System (Amazon EFS) provides a simple, scalable, elastic file system for Linux-based workloads for use with AWS Cloud services and on-premises resources. It is built to scale on-demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, so your applications have the storage they need – when they need it. It is designed to provide massively parallel shared access to thousands of Amazon EC2 instances, enabling your applications to achieve high levels of aggregate throughput and IOPS with consistent low latencies. Amazon EFS is a fully managed service that requires no changes to your existing applications and tools, providing access through a standard file system interface for seamless integration. Amazon EFS is a regional service storing

data within and across multiple Availability Zones (AZs) for high availability and durability. You can access your file systems across AZs and regions and share files between thousands of Amazon EC2 instances and on-premises servers via AWS Direct Connect or AWS VPN.



Image 22: Amazon Elastic File System

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

4. **Amazon Glacier:** Amazon S3 Glacier is a secure, durable, and extremely low-cost storage service for data archiving and long-term backup. It is designed to deliver 99.99999999% durability and provides comprehensive security and compliance capabilities that can help meet even the most stringent regulatory requirements. Amazon S3 Glacier provides query-in-place functionality, allowing you to run powerful analytics directly on your archive data at rest. You can store data for as little as \$0.004 per gigabyte per month, significant savings compared to on-premises solutions. To keep costs low yet suitable for varying retrieval needs, Amazon S3 Glacier provides three options for access to archives, from a few minutes to several hours.



Image 23: Amazon Glacier

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

Networking

1. **Amazon VPC:** Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications.



Image 24: Amazon Virtual Private Cloud

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

You can easily customize the network configuration for your VPC. For example, you can create a public-facing subnet for your web servers that has access to the Internet, and place your backend systems, such as databases or application servers, in a private-facing subnet with no Internet access. You can leverage multiple layers of security (including security groups and network access control lists) to help control access to EC2 instances in each subnet.

2. **Elastic Load Balancer (ELB):** Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.

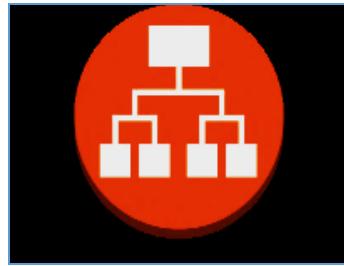


Image 25: Elastic Load Balancer

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

- a. **Application Load Balancer** is best suited for load balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers. Operating at the individual request level (Layer 7), Application Load Balancer routes traffic to targets within Amazon Virtual Private Cloud (Amazon VPC) based on the content of the request.
 - b. **Network Load Balancer** is best suited for load balancing of TCP traffic where extreme performance is required. Operating at the connection level (Layer 4), Network Load Balancer routes traffic to targets within Amazon Virtual Private Cloud (Amazon VPC) and is capable of handling millions of requests per second while maintaining ultra-low latencies. Network Load Balancer is also optimized to handle sudden and volatile traffic patterns.
 - c. **Classic Load Balancer** provides basic load balancing across multiple Amazon EC2 instances and operates at both the request level and connection level. Classic Load Balancer is intended for applications that were built within the EC2-Classic network.
3. **Amazon CloudFront:** Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment. CloudFront is integrated with AWS – both physical locations that are directly connected to the AWS global infrastructure, as well as other AWS services. CloudFront works seamlessly with services including AWS Shield for DDoS mitigation, Amazon S3, Elastic Load Balancing or Amazon EC2 as origins for your applications, and Lambda@Edge to run custom code closer to customers' users and to customize the user experience.

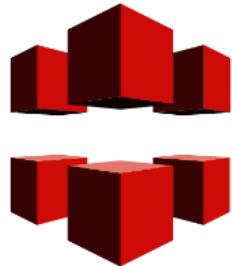


Image 26: Amazon CloudFront

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

4. **Amazon Route 53:** Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications by translating human-readable names, such as www.example.com, into the numeric IP addresses, such as 192.0.2.1, that computers use to connect to each other. Amazon Route 53 is fully compliant with IPv6 as well.



Image 27: Amazon Route 53

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

Amazon Route 53 effectively connects user requests to infrastructure running in AWS—such as EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets—and can also be used to route users to infrastructure outside of AWS. You can use Amazon Route 53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of your application and its endpoints. Amazon Route 53 traffic flow makes it easy for you to manage traffic globally through a variety of routing types, including latency-based routing. Amazon Route 53 also offers Domain Name Registration—you can purchase and manage domain names such as example.com and Amazon Route 53 will automatically configure DNS settings for your domains.

5. **AWS Direct Connect:** AWS Direct Connect makes it easy to establish a dedicated network connection from your premises to AWS. Using AWS Direct Connect, you can establish private connectivity between AWS and your data center, office, or co-location environment, which in many cases can reduce your network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections.



Image 28: AWS Direct Connect

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

Database

1. **Amazon Aurora:** Amazon Aurora is a MySQL and PostgreSQL compatible relational database engine that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. Amazon Aurora is up to five times faster than standard MySQL databases and three times faster than standard PostgreSQL databases. It provides the security, availability, and reliability of commercial databases at 1/10th of the cost. Amazon Aurora is fully managed by Amazon Relational Database Service (RDS), which automates time-consuming administration tasks like hardware provisioning, database setup, patching, and backups.



Image 29: Amazon Aurora

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

2. **Amazon RDS:** Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need. Amazon RDS is available on several database instance types like Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server.



Image 30: Amazon Relational Database Service

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

3. **Amazon DynamoDB:** Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multi-region, multi-master database with built-in security, backup and restores, and in-memory caching for internet-scale applications.



Image 31: Amazon DynamoDB

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

DynamoDB can handle more than 10 trillion requests per day and support peaks of more than 20 million requests per second. Many of the world's fastest-growing businesses such as Lyft, Airbnb, and Redfin as well as enterprises such as Samsung, Toyota, and Capital One depend on the scale and performance of DynamoDB to support their mission-critical workloads.

4. **Amazon DocumentDB:** Amazon DocumentDB (with MongoDB compatibility) is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads. Amazon DocumentDB is designed from the ground-up to give you the performance, scalability, and availability you need when operating mission-critical MongoDB workloads at scale. Amazon DocumentDB implements the Apache 2.0 open source MongoDB 3.6 API by emulating the responses that a MongoDB client expects from a MongoDB server, allowing you to use your existing MongoDB drivers and tools with Amazon DocumentDB.
5. **Amazon ElastiCache:** Amazon ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.



Image 32: Amazon ElastiCache

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

6. **Amazon Redshift:** Amazon Redshift is a fast, scalable data warehouse that makes it simple and cost-effective to analyze all your data across your data warehouse and data lake. Redshift delivers ten times faster performance than other data warehouses by using machine learning, massively parallel query execution, and columnar storage on high-performance disk. You can set up and deploy a new data warehouse in minutes, and run queries across petabytes of data in your Redshift data warehouse, and exabytes of data in your data lake built on Amazon S3.



Image 33: Amazon Redshift

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

Security

The AWS infrastructure has been architected to be one of the most flexible and secure cloud computing environments available today. It is designed to provide an extremely scalable, highly reliable platform that enables customers to deploy applications and data quickly and securely.

This infrastructure is built and managed not only according to security best practices and standards but also with the unique needs of the cloud in mind. AWS uses redundant and layered controls, continuous validation and testing, and a substantial amount of automation to ensure that the underlying infrastructure is monitored and protected 24x7. AWS ensures that these controls are replicated in every new data center or service.

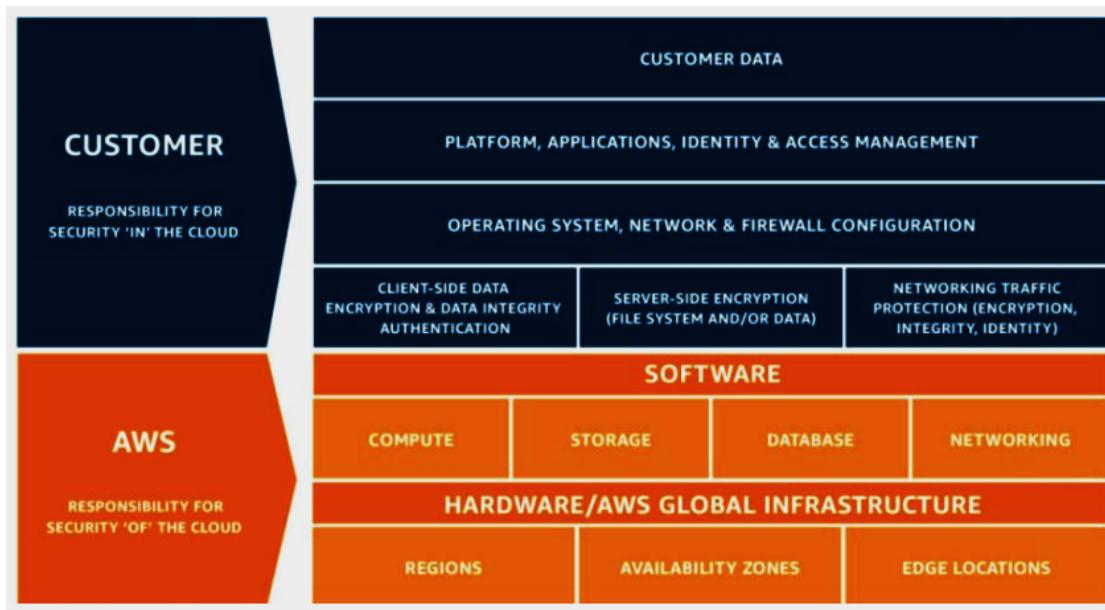


Image 34: AWS Shared Responsibility Model

Reference: https://d1.awsstatic.com/whitepapers/Security/Intro_to_AWS_Security.pdf?did=wp_card&trk=wp_card

All AWS customers benefit from a data center and network architecture built to satisfy the requirements of our most security-sensitive customers. This means that you get a resilient infrastructure, designed for high security, without the capital outlay and operational overhead of a traditional data center.

AWS operates under a shared security responsibility model, where AWS is responsible for the security of the underlying cloud infrastructure and you are responsible for securing workloads you deploy in AWS (Figure 1). This gives you the flexibility and agility you need to implement the most applicable security controls for your business functions in the AWS environment. You can tightly restrict access to environments that process sensitive data, or deploy less stringent controls for information you want to make public.

Important security-related services include:

1. **AWS Identity and Access Management:** AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. IAM allows you to do the following:

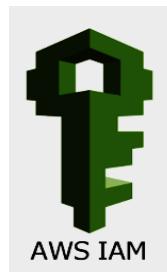


Image 35: AWS Identity and Access Management

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

- Manage IAM users and their access: You can create users in IAM, assign them individual security credentials (access keys, passwords, and multi-factor authentication devices), or request temporary security credentials to provide users access to AWS services and resources. You can manage permissions in order to control which operations a user can perform.
- Manage IAM roles and their permissions: You can create roles in IAM and manage permissions to control which operations can be performed by the entity, or AWS service, that assumes the role. You can also define which entity is allowed to assume the role.
- Manage federated users and their permissions: You can enable identity federation to allow existing identities (users, groups, and roles) in your enterprise to access the AWS Management Console, call AWS APIs, and access resources, without the need to create an IAM user for each identity.

2. **Amazon GuardDuty:** Amazon GuardDuty is a threat detection service that continuously monitors for malicious or unauthorized behavior to help you protect your AWS accounts and workloads. It monitors for activity such as unusual API calls or potentially unauthorized deployments that indicate a possible account compromise. GuardDuty also detects potentially compromised instances.



Image 36: Amazon GuardDuty

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

3. **Amazon Inspector:** Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for exposure, vulnerabilities, and deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed list of security findings prioritized by level of severity.



Image 37: Amazon Inspector

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

4. **AWS Key Management Service:** AWS Key Management Service (KMS) makes it easy for you to create and manage keys and control the use of encryption across a wide range of AWS services and in your applications. AWS KMS is a secure and resilient service that uses FIPS 140-2 validated hardware security modules to protect your keys.



AWS KMS

Image 38: AWS Key Management Service

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

5. **AWS Shield:** AWS Shield is a managed Distributed Denial of Service (DDoS) protection service that safeguards web applications running on AWS. AWS Shield provides always-on detection and automatic inline mitigations that minimize application downtime and latency, so there is no need to engage AWS Support to benefit from DDoS protection. There are two tiers of AWS Shield: Standard and Advanced.



Image 39: AWS Shield

Reference: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

Industrial Faculty-Webinar, How Cloud and Webinar Works

Industrial Faculty-Webinar

Live online webinars are important video collaboration tools in business and education because they make real-time communication with prospective clients, employees, business partners, and students easy, fast, affordable and efficient. In addition, live webinars allow seamless presenter-participant interaction, which allows the attendees to ask questions and get answers immediately. In other words, live webinar meaning is that these sessions are hosted in real-time.



Image 40: Live Webinar
Reference: <https://pxhere.com/en/photo/1585323>

Simply put, live webinars are online seminars, lectures, or classes presented online and in real-time and to a live audience. These are engaging online events in which one or more speakers, present information to a large and live audience. The attendees receive the information and they can interact with the presenters by asking questions and participating in polls or surveys while the presenter can evaluate how well the webinar is working towards achieving the intended goals. In the academic world, webinars can allow students to remotely attend the lectures from the convenience of their hostels or homes, and this helps both business people and students to save on commuting to business premises or lecture halls on campus.

Live webinars differ from meeting tools and traditional media such as video streaming that doesn't offer the audience room to interact with the presenter. The live webinars also differ from pre-recorded or on-demand webinars, which aren't presented in real-time.

How Cloud and Webinar Works

There are numerous webinar platforms, but not all can win most of the popularity. For example, WordPress Live Webinar runs well with various features, but WordPress Live Webinar plugin is too complicated to install for a newbie. EzTalks Webinar stands from the rest as it better facilitates the hosting of live webinars easily and flexibly. But how does a live webinar work? Well, this outline specifically presents the process of running a live webinar.

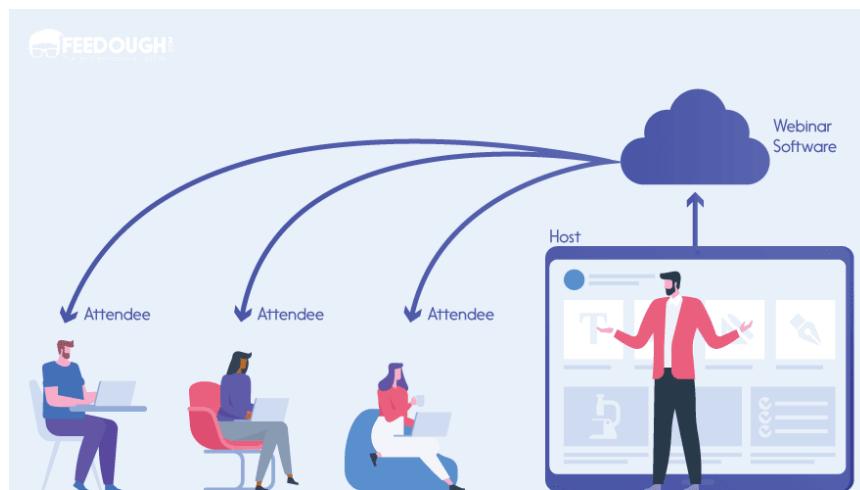


Image 41: Live Webinar Working Scenario

Reference: <https://i2.wp.com/www.feedough.com/wp-content/uploads/2020/02/how-webinar-works.png?resize=883%2C493&ssl=1>

Webinar marketing is a vital strategy for B2B businesses and a lot of consumer brands are also turning to it for their own B2C marketing efforts. Webinars give you the chance to build a more personal relationship with your audience, delve deeper into the topics that concern them and build your brand as a place people can come to for important info. Some of famous cloud based webinar platforms are: EzTalks, WebinarNinja, WebEx, GoToWebinar, Zoom, HangoutMeet, etc.

Major issues related to cloud based webinar hosting includes:

- Video Encoding
- Audio Encoding
- Multipoint Conferencing and Scaling

-
- Interoperability
 - Collaboration and Content Sharing
 - Security and Network Management
 - User Interface

Significance of Cloud in Webinars

Cloud videoconferencing technology is best understood, in contrast to “traditional” videoconferencing. Cloud technology's most distinguishing feature is that client conferencing software installed on local computing devices accesses videoconferencing software on servers managing communication. Servers take advantage of the camera and audio resources that are built into or added onto the client devices, such as a laptop's built-in camera and microphone or desktop's external USB camera and microphone.

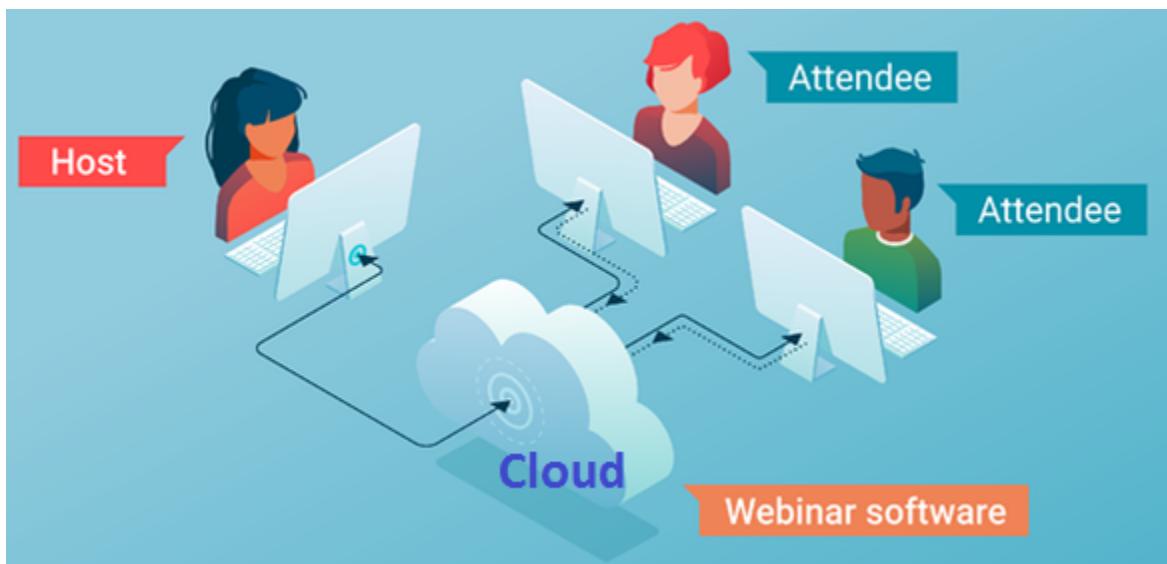


Image 42: Live Webinar Working Scenario

Reference:

<https://learn.g2.com/hs-fs/hubfs/Pillar%20Pages/webinars/what%20is%20a%20webinar/what-is-a-webinar.png?width=632&name=what-is-a-webinar.png>

Traditional videoconferencing is accomplished through the use of appliances usually permanently installed in a room or placed on moveable carts. A common communication standard (H.323) ensures that end-point appliances interoperate to exchange audio and video at transfer rates from 128 kilobits/s (Kbps) to 4 megabits/s (Mbps), with the latter providing 1920×1080 pixels of high-definition progressive video incorporating the H.264 video standard.

Typical appliances include pan, tilt, and zoom (PTZ) cameras that can be locally or remotely controlled to show the entire room or persons within it and omnidirectional microphones that can detect audio from long distances, with built-in echo cancellation so that sounds picked up from the speakers do not produce feedback.

Terms associated with cloud conferencing are Web real-time communication (WebRTC), unified communication, and video as a service. WebRTC refers to open-source research and development efforts aimed at incorporating videoconferencing directly into browsers. Because browsers share the video across the Internet, WebRTC involves cloud computing, but only by incorporating video into browser architecture. The term unified communications refers to integrating real-time communication (telephony and videoconferencing) with other network data resources, such as interactive whiteboards, and non-real-time communication, such as e-mail and voicemail, so that voicemail might be accessed as e-mail, or vice versa. Cloud conferencing technologies attempt to integrate videoconferencing with other applications, at least those that are real-time, and subscribe to the idea of unifying communication to some extent. Video as a service is a term describing accessing network videoconferencing services located in the cloud, usually paid for by subscription.

Able to configure embedded database with different web pages using mongodb

In this section, we will read about:

- Purpose of database systems
- Data abstraction
- Database Users
- Data Independence (Logical & Physical)
- Instance & Schemes
- Three layered Architecture of DBMS
- Different Levels of Abstraction.
- Data Modeling
- E-R Modeling
- Logical Model: Object & Record based – Object oriented model - Entity relationship models
- Entity sets & relationships sets
- Concept of attributes and relationships
- Introduction to mapping constraints.
- Basic Concepts of ER Model in DBMS
- Introduction to DBMS
- Structure of DBMS
- Relational Models
- Introduction to Hierarchical Model and Network Model
- Introduction to RDBMS and Relational Models
- Introduction to relational algebra and relational calculus
- Understanding database technologies
- Relational Data Structure
- Keys and Relational Data Manipulation
- Relational Algebra
- Relational Algebraic Operations
- Set Operations
- Fundamental Operations

-
- Relational Calculus
 - Data Definition Language
 - Operators: select, project, join, rename etc
 - Introduction to MongoDB
 - Advantages of MongoDB over RDBMS

Purpose of database systems

What is Database?

A **database** is an organized collection of data, so that it can be easily accessed and managed.

You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

Database handlers create a database in such a way that only one set of software program provides access of data to all the users.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many **dynamic websites** on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.



Image1: Database

Reference: <https://static.javatpoint.com/sqlpages/images/database.png>

What is DBMS?

Database Management System (DBMS) refers to the technology solution used to optimize and manage the storage and retrieval of data from databases. DBMS offers a systematic approach to manage databases via an interface for users as well as workloads accessing the databases via apps. The management responsibilities for DBMS encompass information within the databases, the processes applied to databases (such as access and modification), and the database's logic structure. DBMS also facilitates additional administrative operations such as change management, disaster recovery, compliance, and performance monitoring, among others.

In order to facilitate these functions, DBMS has the following key components:

Software. DBMS is primarily a software system that can be considered as a management console or an interface to interact with and manage databases. The interfacing also spreads across real-world physical systems that contribute data to the backend databases. The OS, networking software, and the hardware infrastructure is involved in creating, accessing, managing, and processing the databases.

Data. DBMS contains operational data, access to database records and metadata as a resource to perform the necessary functionality. The data may include files with such as index files, administrative information, and data dictionaries used to represent data flows, ownership, structure, and relationships to other records or objects.

Procedures. While not a part of the DBMS software, procedures can be considered as instructions on using DBMS. The documented guidelines assist users in designing, modifying, managing, and processing databases.

Database languages. These are components of the DBMS used to access, modify, store, and retrieve data items from databases; specify database schema;

control user access; and perform other associated database management operations. Types of DBMS languages include Data Definition Language (DDL), Data Manipulation Language (DML), Database Access Language (DAL) and Data Control Language (DCL).

Query processor. As a fundamental component of the DBMS, the query processor acts as an intermediary between users and the DBMS data engine in order to communicate query requests. When users enter an instruction in SQL language, the command is executed from the high-level language instruction to a low-level language that the underlying machine can understand and process to perform the appropriate DBMS functionality. In addition to instruction parsing and translation, the query processor also optimizes queries to ensure fast processing and accurate results.

Runtime database manager. A centralized management component of DBMS that handles functionality associated with runtime data, which is commonly used for context-based database access. This component checks for user authorization to request the query; processes the approved queries; devises an optimal strategy for query execution; supports concurrency so that multiple users can simultaneously work on same databases; and ensures integrity of data recorded into the databases.

Database manager. Unlike the runtime database manager that handles queries and data at runtime, the database manager performs DBMS functionality associated with the data within databases. Database manager allows a set of commands to perform different DBMS operations that include creating, deleting, backup, restoring, cloning, and other database maintenance tasks. The database manager may also be used to update the database with patches from vendors.

Database engine. This is the core software component within the DBMS solution that performs the core functions associated with data storage and retrieval. A database engine is also accessible via APIs that allow users or apps to create, read, write, and delete records in databases.

Reporting. The report generator extracts useful information from DBMS files and displays it in structured format based on defined specifications. This

information may be used for further analysis, decision making, or business intelligence.

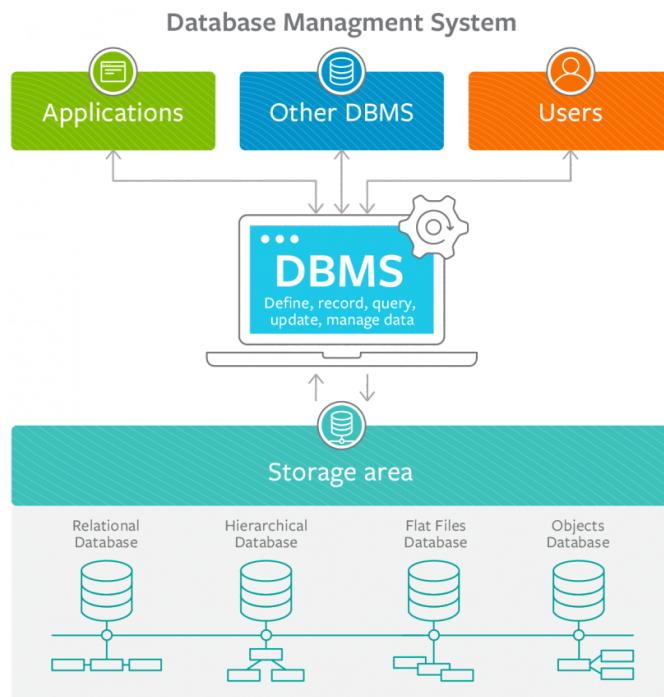


Image2: DBMS

Reference: <https://blogs.bmc.com/wp-content/uploads/2018/08/dbms-database-management-systems-810x898.png>

What are the types of DBMS?

Depending upon the usage requirements, there are following types of databases available in the market:

- Centralised database.
- Distributed database.
- Personal database.

- End-user database.
- Commercial database.
- NoSQL database.
- Operational database.
- Relational database.
- Cloud database.
- Object-oriented database.
- Graph database.

Centralised Database

The information(data) is stored at a centralized location and the users from different locations can access this data. This type of database contains application procedures that help the users to access the data even from a remote location.

Various kinds of authentication procedures are applied for the verification and validation of end users, likewise, a registration number is provided by the application procedures which keeps a track and record of data usage. The local area office handles this thing.

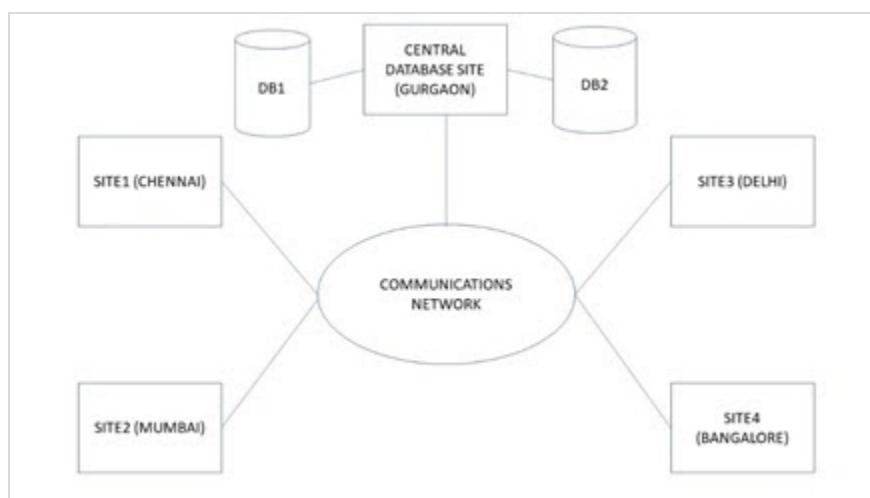


Image3: Centralised Database

Reference: <https://www.tutorialspoint.com/assets/questions/images/109302-1532341956.jpg>

Distributed Database

Just opposite of the centralized database concept, the distributed database has contributions from the common database as well as the information captured by local computers also. The data is not at one place and is distributed at various sites of an organization. These sites are connected to each other with the help of communication links which helps them to access the distributed data easily.

You can imagine a distributed database as a one in which various portions of a database are stored in multiple different locations(physical) along with the application procedures which are replicated and distributed among various points in a network.

There are two kinds of distributed database, viz. homogenous and heterogeneous. The databases which have same underlying hardware and run over same operating systems and application procedures are known as homogeneous DDB, for eg. All physical locations in a DDB. Whereas, the operating systems, underlying hardware as well as application procedures can be different at various sites of a DDB which is known as heterogeneous DDB.

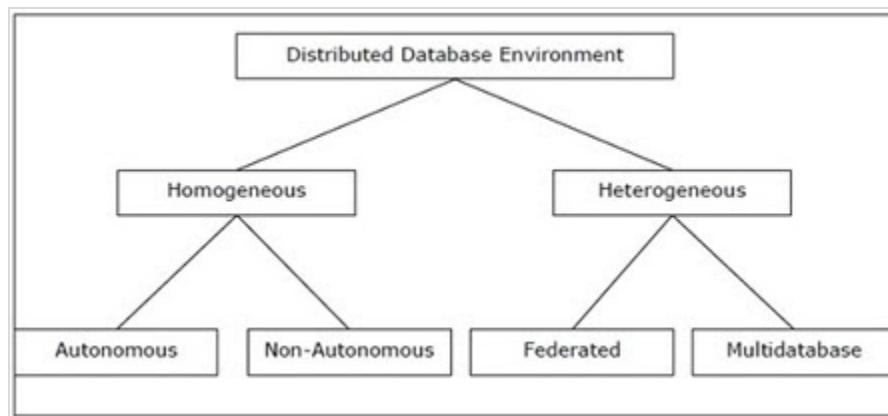


Image4: Distributed Database

Reference: <https://www.tutorialspoint.com/assets/questions/images/104606-1532341968.jpg>

Personal Database

Data is collected and stored on personal computers which is small and easily manageable. The data is generally used by the same department of an organization and is accessed by a small group of people.

End User Database

The end user is usually not concerned about the transaction or operations done at various levels and is only aware of the product which may be a software or an application. Therefore, this is a shared database which is specifically designed for the end user, just like different levels' managers. Summary of whole information is collected in this database.

Commercial Database

These are the paid versions of the huge databases designed uniquely for the users who want to access the information for help. These databases are subject specific, and one cannot afford to maintain such a huge information. Access to such databases is provided through commercial links.

NoSQL Database

These are used for large sets of distributed data. There are some big data performance issues which are effectively handled by relational databases, such kind of issues are easily managed by NoSQL databases. There are very efficient in analyzing large size unstructured data that may be stored at multiple virtual servers of the cloud.

Operational Database

Information related to operations of an enterprise is stored inside this database. Functional lines like marketing, employee relations, customer service etc. require such kind of databases.

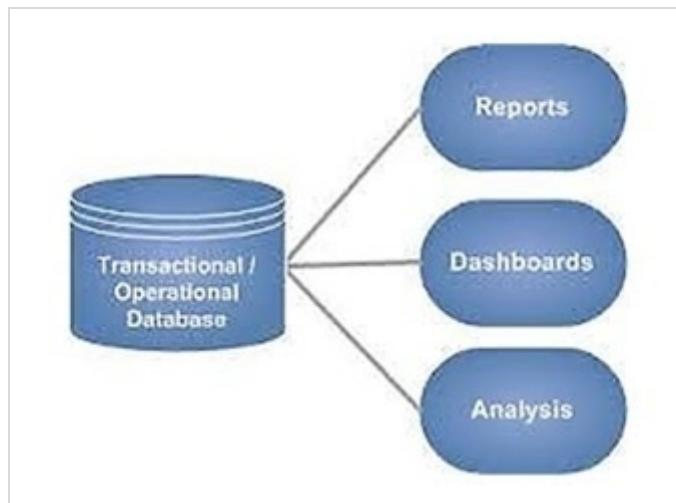


Image5: Operational Database

Reference: <https://www.tutorialspoint.com/assets/questions/images/104532-1532341980.jpg>

Relational Databases

These databases are categorized by a set of tables where data gets fit into a pre-defined category. The table consists of rows and columns where the column has an entry for data for a specific category and rows contains instance for that data defined according to the category. The Structured Query Language (SQL) is the standard user and application program interface for a relational database.

There are various simple operations that can be applied over the table which makes these databases easier to extend, join two databases with a common relation and modify all existing applications.

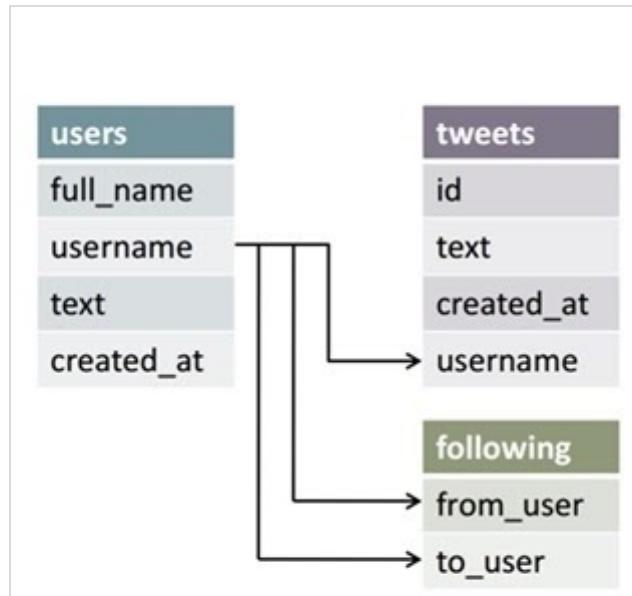


Image6: Relational Database

Reference: <https://www.tutorialspoint.com/assets/questions/images/112548-1532342000.jpg>

Cloud Databases

Now a day, data has been specifically getting stored over clouds also known as a virtual environment, either in a hybrid cloud, public or private cloud. A cloud database is a database that has been optimized or built for such a virtualized environment. There are various benefits of a cloud database, some of which are the ability to pay for storage capacity and bandwidth on a per-user basis, and they provide scalability on demand, along with high availability.

A cloud database also gives enterprises the opportunity to support business applications in a software-as-a-service deployment.

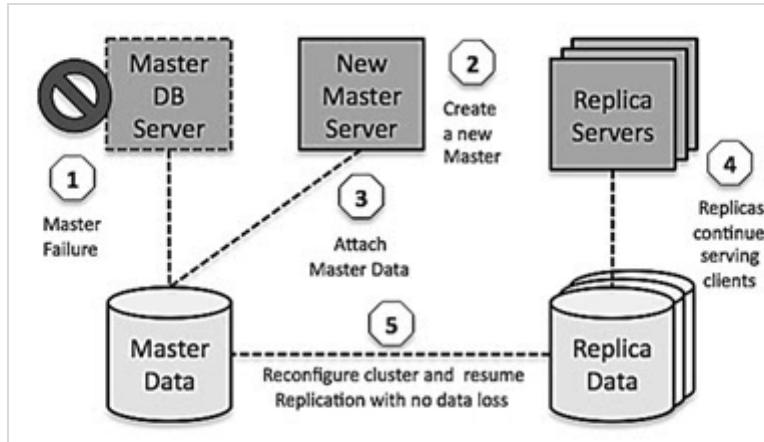


Image7: Cloud Database

Reference: <https://www.tutorialspoint.com/assets/questions/images/112415-1532342015.jpg>

Object-Oriented Databases

An object-oriented database is a collection of object-oriented programming and relational database. There are various items which are created using object-oriented programming languages like C++, Java which can be stored in relational databases, but object-oriented databases are well-suited for those items.

An object-oriented database is organized around objects rather than actions, and data rather than logic. For example, a multimedia record in a relational database can be a definable data object, as opposed to an alphanumeric value.

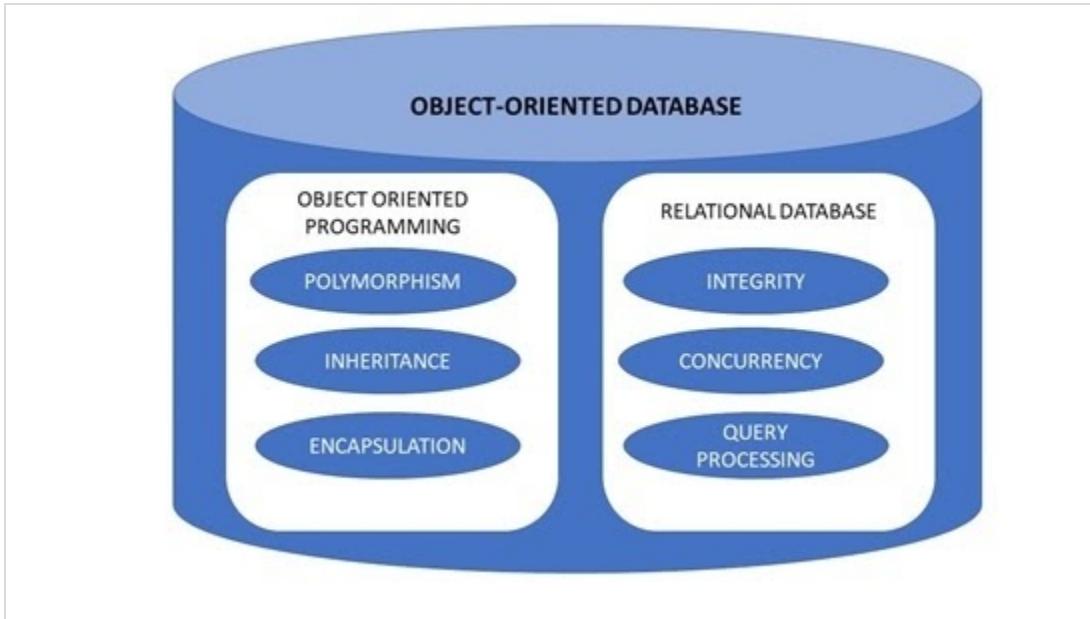


Image8: Object OrientedDatabase

Reference: <https://www.tutorialspoint.com/assets/questions/images/113587-1532342027.jpg>

Graph Databases

The graph is a collection of nodes and edges where each node is used to represent an entity and each edge describes the relationship between entities. A graph-oriented database, or graph database, is a type of NoSQL database that uses graph theory to store, map and query relationships.

Graph databases are basically used for analyzing interconnections. For example, companies might use a graph database to mine data about customers from social media.

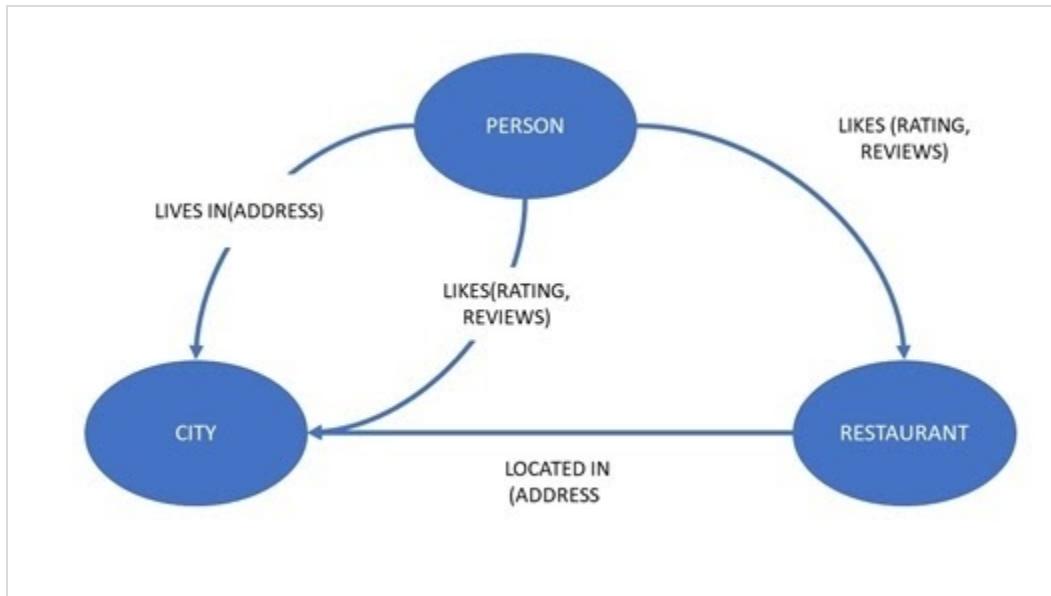


Image9: Graph Database

Reference: <https://www.tutorialspoint.com/assets/questions/images/114315-1532342037.jpg>

DBMS vs. File System

There are following differences between DBMS and File system:

DBMS	File System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.
DBMS takes care of Concurrent access of data using some form of locking.	In the File system, concurrent access has many problems like redirecting the file while deleting some information or updating some information.

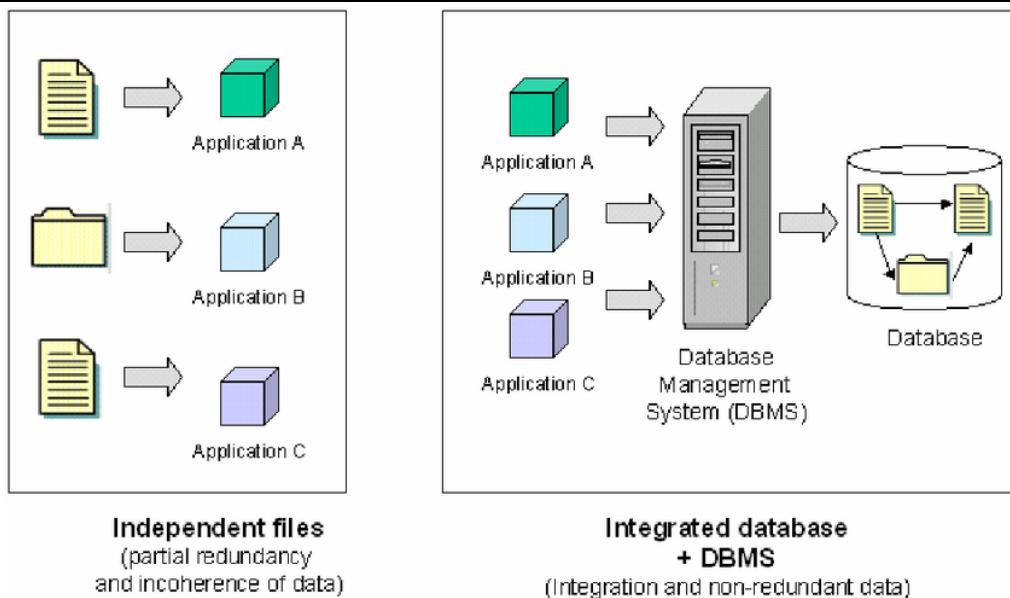


Image10: DBMS vs File System

Reference:

https://www.researchgate.net/profile/Yvan_Bedard/publication/11041878/figure/fig1/AS:277384914849800@1443145125825/independent-files-Ys-integrated-database-DBMS.png

Data Abstraction

It is one of the main and important characteristics of database approach. Data abstraction is the concept of hiding the details like data definition, data organization and storage of data from the end users and showing them only the essential things as per their requirement.

For example, an end user may be naïve user, application programmer, or an expert in DBMS. Their requirement in viewing data is different for different user. The users may not be interested in everything about a database. Since most of the end users of any database may not have enough idea about the implementation, the database developers have hidden many of the unwanted (for end users) information through several levels of data abstraction.

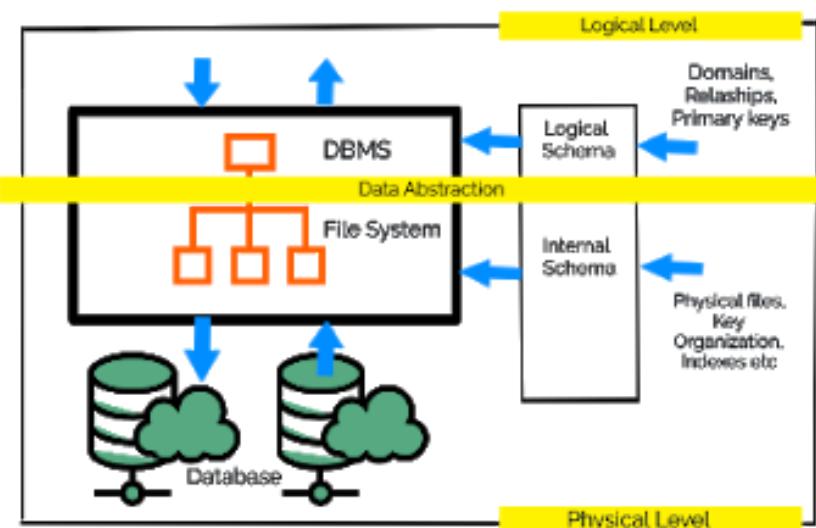


Image11: Data Abstraction

Reference:

https://lh3.googleusercontent.com/proxy/SPb-TFbsIwUqDqf3AzqIpABk6nb_6eIt5QB3howu17kzRsiYJQ29r3JOoJMYGRtrHkZsRrmhe9osOTmHv77fBwl1knff7cExBkZHBSI7bi1c10wW5vv5NYA

Database Users

Database Users are the one who interacts with the system. There can be four **different types of users** according to the way they interact with the system and for all the different users, different kind of user interfaces are designed as well. The four **types of users** are:

1. Naive users
2. Application Programmers
3. Sophisticated Users
4. DBA

Let us have a look over all the four users and their interfaces:

Naive Users

Naive users are also termed as unsophisticated users and they interact with the system by calling anyone application program that has been written previously.

For e.g. – To transfer the program from one account to another, there is a need for an application program called transfer.

The user interface that is required for the naïve users is a forms interface, in which the user can fill the required fields.

Naive users can also easily read the reports that are generated from the database.

Application programmers

Application programmers are the one who is responsible to write the application programs. They develop user interfaces through different tools. To construct the forms and the reports such that there is no need to write the program, there is a tool named Rapid Application Development (RAD).

Some special type of programming languages is also available such that includes vital control structures such as for loops, while loops and many others with the data manipulation language's

statements. These special programming languages are termed as fourth-generation languages and they include the special features to provide the ability for the generation of the forms and to display the data on the screen.

In today's world, a large variety of commercial database systems includes these fourth generation languages.

Sophisticated users

Sophisticated users aren't interested in writing programs and they interact with the system without writing any programs. Contrary, they use database query languages to interact with the system.

Sophisticated Users submit their queries to a query processor. Query Processor provides the facility to break the DML statements into instructions that can be understood by the storage manager.

Analysts are one among the sophisticated users. They use the tools to perform their task such as:

1. **Online analytical processing (OLAP)** - It helps the analysts to view them the summaries of the data in different ways.
2. **Data Mining Tools** – It helps the analysts find a certain kind of pattern in the given data.

DBA [Database Administrators]

- In any organization where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources.
- In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software.
- Administering these resources is the responsibility of the database administrator (DBA).
- The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.
- The DBA is accountable for problems such as security breaches and poor system response time. In large organizations, the DBA is assisted by a staff that carries out these functions.

Data Independence (Logical & Physical)

- o Data independence can be explained using the three-schema architecture.
- o Data independence refers to the characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

Types of Data Independence

In DBMS there are two types of data independence

1. Physical data independence
2. Logical data independence.

Levels of Database

Before we learn Data Independence, a refresher on Database Levels is important. The database has 3 levels as shown in the diagram below

1. Physical/Internal
2. Conceptual
3. External

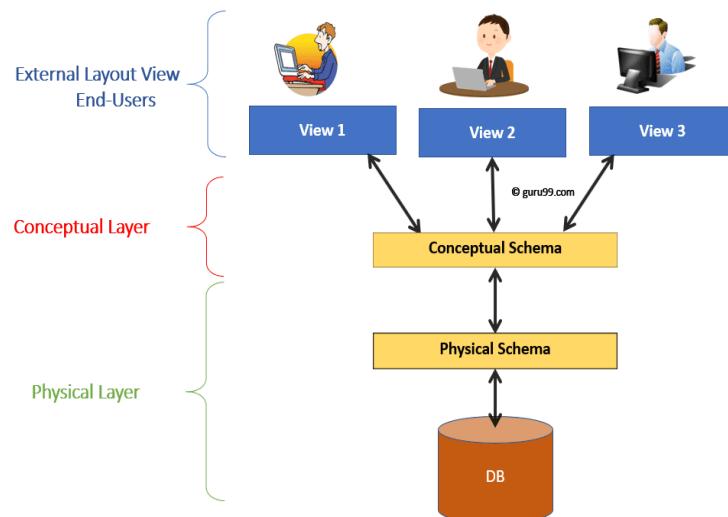


Image12: Levels of DBMS Architecture Diagram

Reference: https://www.guru99.com/images/1/042919_0417_DataIndependence1.png

Consider an Example of a University Database. At the different levels this is how the implementation will look like:

Type of Schema	Implementation
External Schema	View 1: Course info(cid:int,cname:string) View 2: studeninfo(id:int, name:string)
Conceptual Shema	Students(id: int, name: string, login: string, age: integer) Courses(id: int, cname:string, credits:integer) Enrolled(id: int, grade:string)
Physical Schema	<ul style="list-style-type: none"> • Relations stored as unordered files. • Index on the first column of Students.

Physical Data Independence

Physical data independence helps you to separate conceptual levels from the internal/physical levels. It allows you to provide a logical description of the database without the need to specify physical structures. Compared to Logical Independence, it is easy to achieve physical data independence.

With Physical independence, you can easily change the physical storage structures or devices with an effect on the conceptual schema. Any change done would be absorbed by the mapping between the conceptual and internal levels. Physical data independence is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

Examples of changes under Physical Data Independence

Due to Physical independence, any of the below changes will not affect the conceptual layer.

- Using a new storage device like Hard Drive or Magnetic Tapes

-
- Modifying the file organization technique in the Database
 - Switching to different data structures.
 - Changing the access method.
 - Modifying indexes.
 - Changes to compression techniques or hashing algorithms.
 - Change of Location of Database from say C drive to D Drive

Logical Data Independence

Logical Data Independence is the ability to change the conceptual scheme without changing

1. External views
2. External API or programs

Any change made will be absorbed by the mapping between external and conceptual levels.

When compared to Physical Data independence, it is challenging to achieve logical data independence.

Examples of changes under Logical Data Independence

Due to Logical independence, any of the below change will not affect the external layer.

1. Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
2. Merging two records into one
3. Breaking an existing record into two or more records

Difference between Physical and Logical Data Independence

Logical Data Independence	Physical Data Independence
Logical Data Independence is mainly concerned with the structure or changing the data definition.	Mainly concerned with the storage of the data.
It is difficult as the retrieving of data is mainly dependent on the logical structure of data.	It is easy to retrieve.

Compared to Logic Physical independence it is difficult to achieve logical data independence.	Compared to Logical Independence it is easy to achieve physical data independence.
You need to make changes in the Application program if new fields are added or deleted from the database.	A change in the physical level usually does not need change at the Application program level.
Modification at the logical levels is significant whenever the logical structures of the database are changed.	Modifications made at the internal levels may or may not be needed to improve the performance of the structure.
Concerned with conceptual schema	Concerned with internal schema
Example: Add/Modify/Delete a new attribute	Example: change in compression techniques, hashing algorithms, storage devices, etc

Importance of Data Independence

- Helps you to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- You don't need to alter data structure in application programs
- Permit developers to focus on the general structure of the Database rather than worrying about the internal implementation
- It allows you to improve state which is undamaged or undivided
- Database incongruity is vastly reduced.
- Easily making modifications in the physical level is needed to improve the performance of the system.

Instance & Schemes

- o The data which is stored in the database at a particular moment of time is called an instance of the database.
- o The overall design of a database is called schema.
- o A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- o A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- o A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
- o A database schema is designed by the database designers to help programmers whose software will interact with the database. The process of database creation is called data modeling.

A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram. For example, the given figure neither shows the data type of each data item nor the relationship among various files.

In the database, actual data changes quite frequently. For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database.

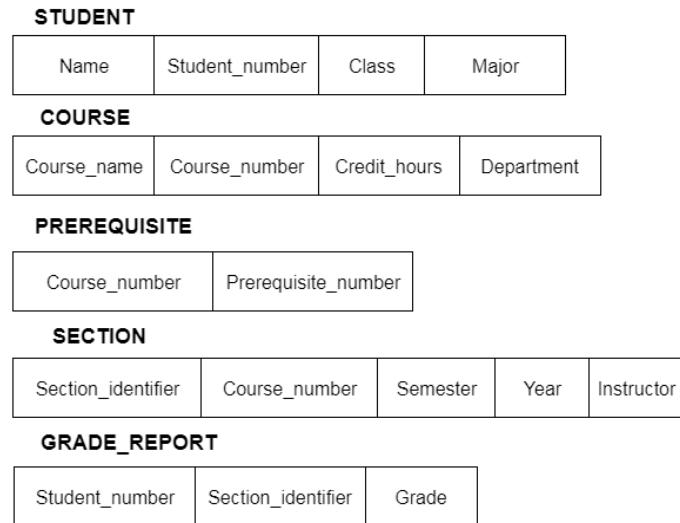


Image13: Data Model Schema & Instance

Reference: <https://static.javatpoint.com/dbms/images/dbms-data-model-schema-and-instance.png>

Differences Between Schema and Instance

1. A schema is the design representation of a database whereas instance is the snapshot of a database at a particular moment.
2. Instance changes very frequently, whenever data is removed or added in the database. As against, the changes in schema occurs rarely.
3. For example, schema and instance can be easily perceived by analogy to a program. At the time of writing a program in a programming language, the variables of that program is declared at first, this is analogous to the schema definition. Additionally, each variable in a program must have some values associated at a particular time; this is similar to an instance.

Sub-Schema

A subschema lets the user have access to different areas of applications in which the user designed. The areas that are included in an application are set, types, record types, data items, and data aggregates. Schemas may have many different subschemas' that are all very different. There are several different reasons for subschema's. One reason for a subschema is to let a user or programmer see different views without having to see all the data contained in the database. Another reason would be to improve security so that someone who is not authorized can change or add to the data.

Data definition language describes the database in which there are possibly many programs that have been writing in different program languages. The description is in the form of names and characteristics of data items, data aggregates, records, areas, and sets included in the database, and the relationships that exist and that have to be maintained between occurrences of elements within the database.

A data item is represented by a value in a database, which is an occurrence of the smallest unit of the named data.

A Data aggregate is the occurrence of named collections of data items that are inside the record. There are two kinds, the first is a vector that is a one-dimensional sequence of data items. These all have identical characteristics. The other is a repeating group and is a collection of data that occurs several times within a record occurrence.

Record is an occurrence of a named collection of zero, one, or more items. Keys are a way to uniquely identify a record.

Set is an occurrence of a named collection of records. For every record in the schema there must be one occurrence.

Area is a named collection of records but it does not have to be linked between each record.

Database contains all the records within the schema; however each database must contain a separate schema.

Three Layered of Architecture in DBMS

- o The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- o The client/server architecture consists of many PCs and a workstation which are connected via the network.
- o DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture

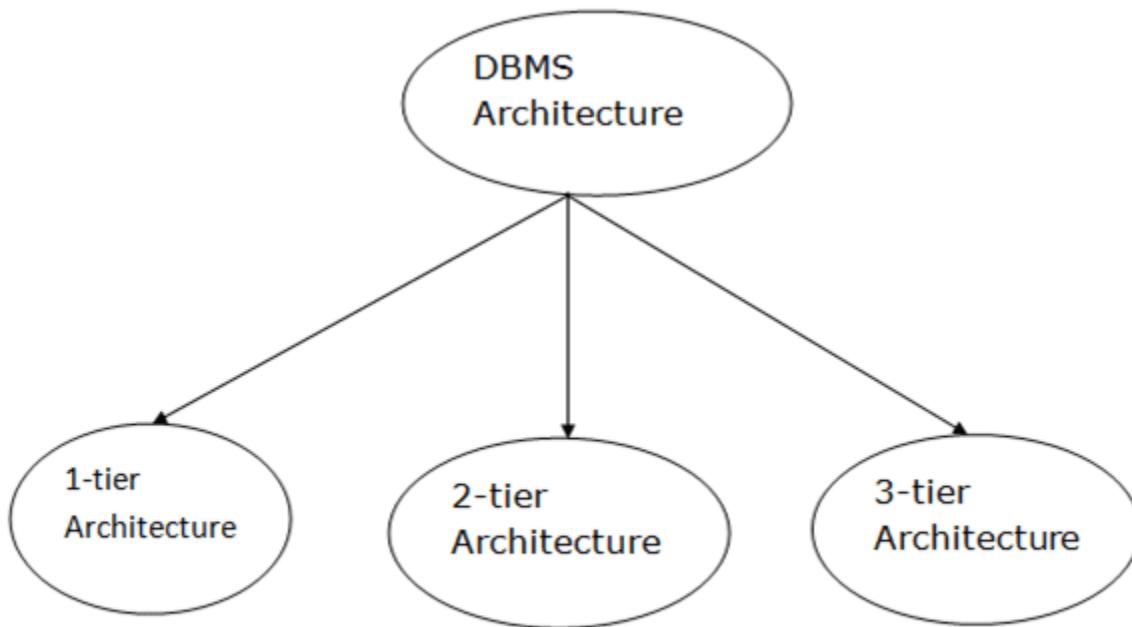


Image14: DBMS Architecture

Reference: <https://static.javatpoint.com/dbms/images/dbms-architecture.png>

Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

1-Tier Architecture

- o In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- o Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- o The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- o The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- o The user interfaces and application programs are run on the client-side.
- o The server side is responsible to provide the functionalities like: query processing and transaction management.
- o To communicate with the DBMS, client-side application establishes a connection with the server side.

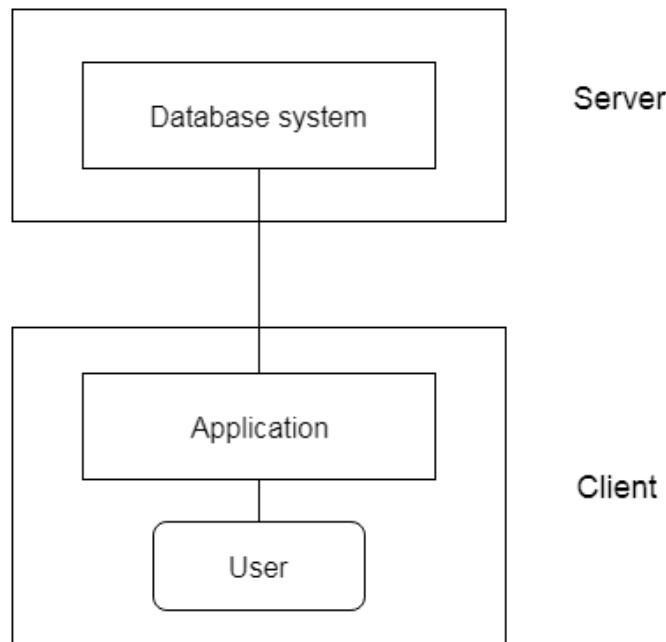


Image15: 2-Tier Architecture

Reference: <https://static.javatpoint.com/dbms/images/dbms-2-tier-architecture.png>

3-Tier Architecture

- o The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- o The application on the client-end interacts with an application server which further communicates with the database system.
- o End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- o The 3-Tier architecture is used in case of large web application.

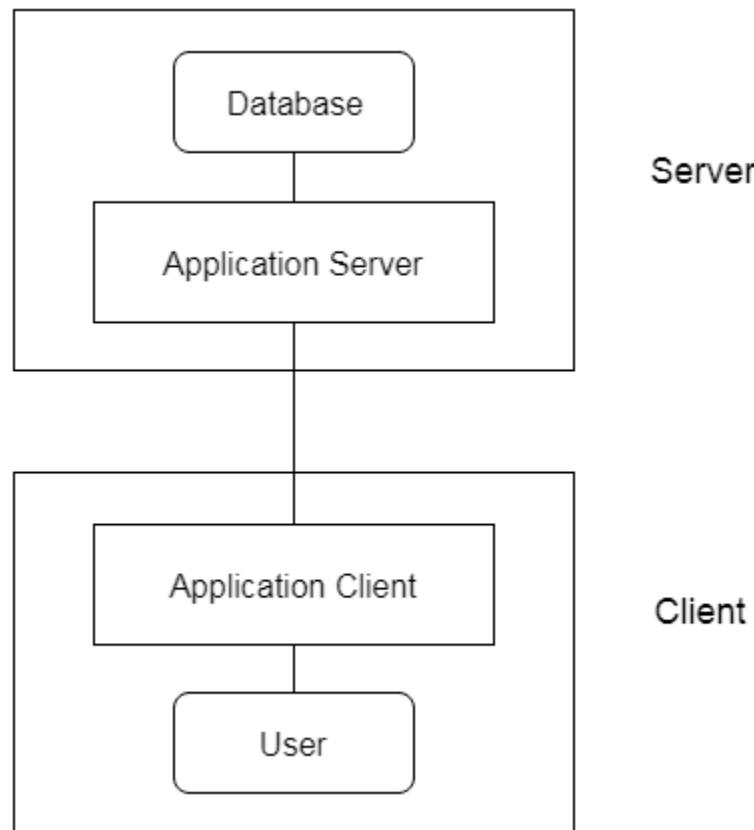


Image16: 3-Tier Architecture

Reference: <https://static.javatpoint.com/dbms/images/dbms-3-tier-architecture.png>

Different Levels of Abstraction

- o The three schema architecture is also called ANSI/SPARC architecture or different-level of Abstraction.
- o This framework is used to describe the structure of a specific database system.
- o The three schema architecture is also used to separate the user applications and physical database.
- o The three schema architecture contains three-levels. It breaks the database down into three different categories.

The level of Abstraction is as follows

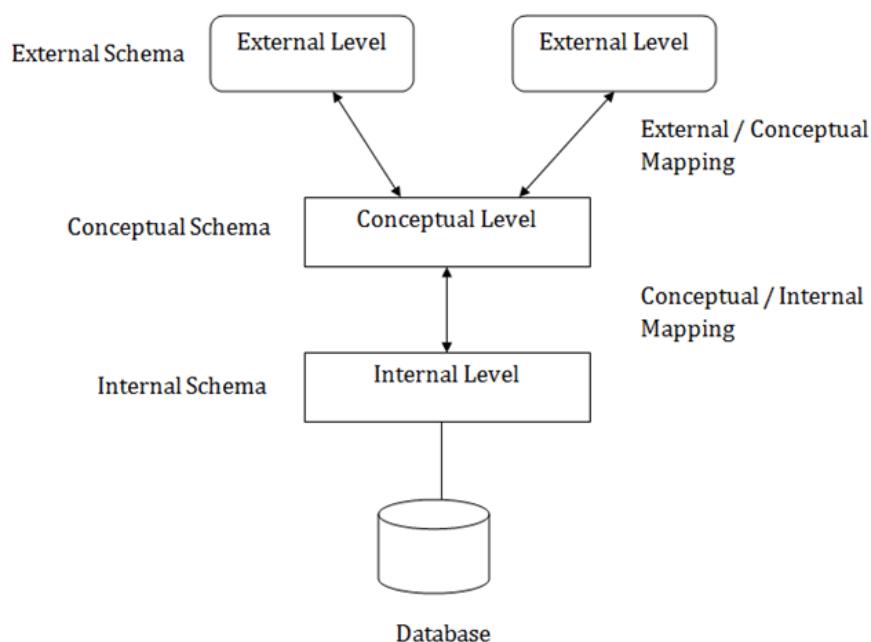


Image17: Level of Abstraction

Reference: <https://static.javatpoint.com/dbms/images/dbms-three-schema-architecture.png>

In the above diagram:

- o It shows the DBMS architecture.
- o Mapping is used to transform the request and response between various database levels of architecture.
- o Mapping is not good for small DBMS because it takes more time.
- o In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- o In Conceptual / Internal mapping, DBMS transforms the request from the conceptual to internal level.

Physical level or Internal Level

- o The internal level has an internal schema which describes the physical storage structure of the database.
- o The internal schema is also known as a physical schema.
- o It uses the physical data model. It is used to define that how the data will be stored in a block.
- o The physical level is used to describe complex low-level data structures in detail.

Logical Level or Conceptual Level

- o The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- o The conceptual schema describes the structure of the whole database.
- o The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- o In the conceptual level, internal details such as an implementation of the data structure are hidden.
- o Programmers and database administrators work at this level.

View Level or External Level

- o At the external level, a database contains several schemas that are sometimes called as subschema. The subschema is used to describe the different view of the database.
- o An external schema is also known as view schema.
- o Each view schema describes the database part that a particular user group is interested in and hides the remaining database from that user group.
- o The view schema describes the end user interaction with database systems.

Data Modelling

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:

Relational Data Model

This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

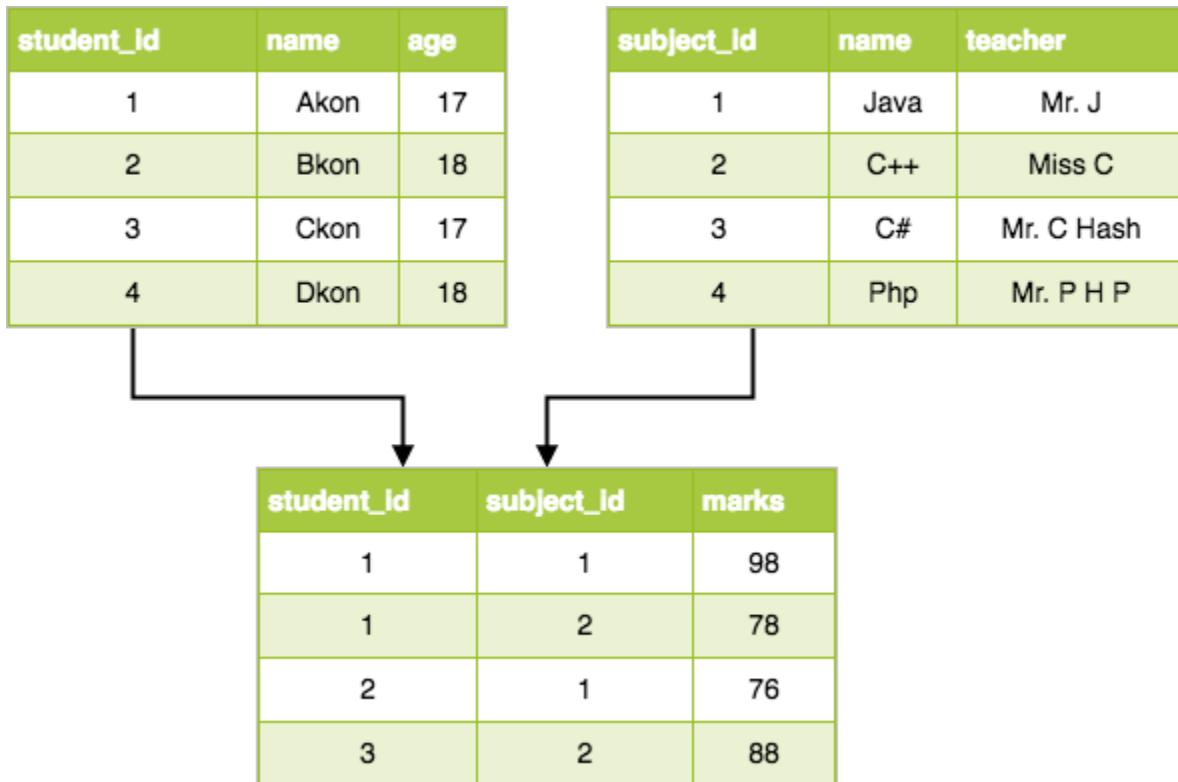


Image18: Relational Data Models

Reference: <https://www.studytonight.com/dbms/images/relational-dbms-model.png>

Entity-Relationship Data Model

An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student_name, student_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

Object-based Data Model

An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

Semistructured Data Model

This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets. The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data. Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

Hierarchical Model

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like indexes of a book, recipes etc.

In hierarchical models, data is organised into a tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

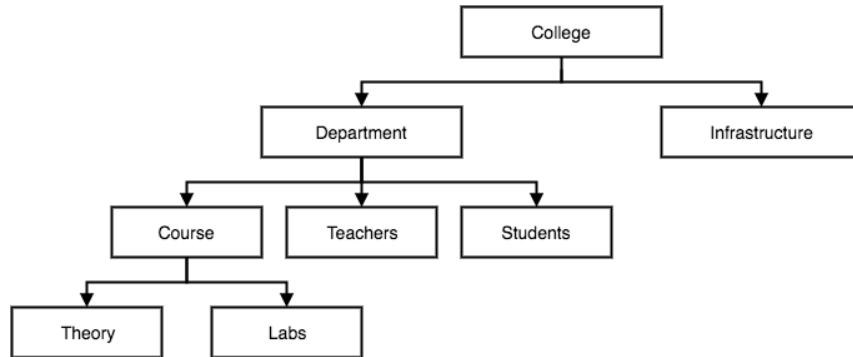


Image19: Hierarchical Model

Reference: <https://www.studytonight.com/dbms/images/hierarchical-dbms-model.png>

Network Model

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and faster. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before the Relational Model was introduced.

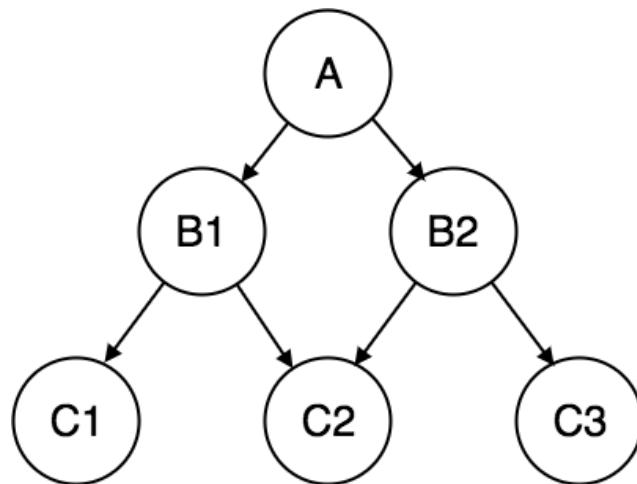


Image20: Network Model

Reference: <https://www.studytonight.com/dbms/images/network-dbms-model.png>

Flat Data Model

Flat data model is the first and foremost introduced model and in this all the data used is kept in the same plane. Since it was used earlier this model was not so scientific.

Roll No	Name	Course
5482	Mark	Web Designing
5486	Steve	Java
5496	Smith	Oracle

Image21: Flat Data Model

Reference: <https://whatisdbms.com/wp-content/uploads/2016/06/Flat-Data-Model-in-DBMS.jpg>

Record base Data Model

Record base model is used to specify the overall structure of the database and in this there are many record types. Each record type has fixed no. of fields having the fixed length.

ER modeling

- o ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- o It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- o In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.

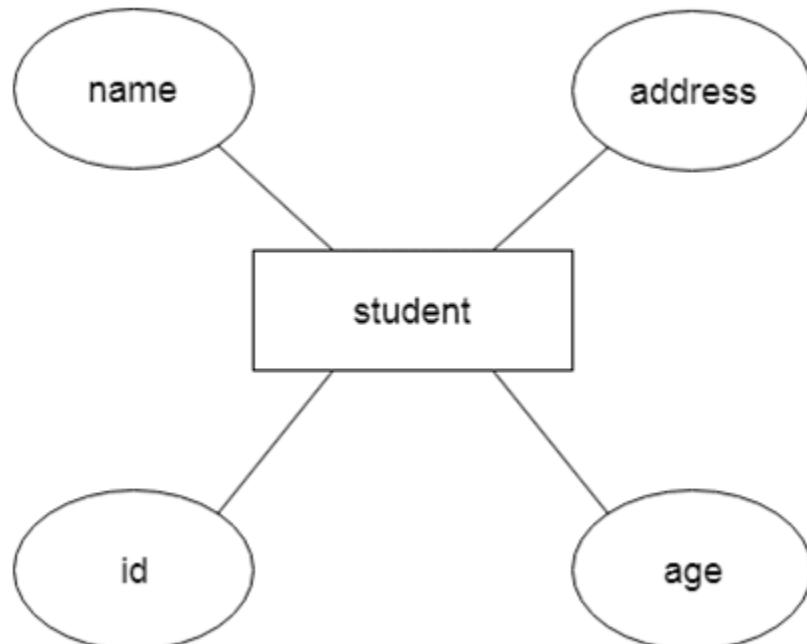


Image22: ER Model

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept.png>

Component of ER Diagram

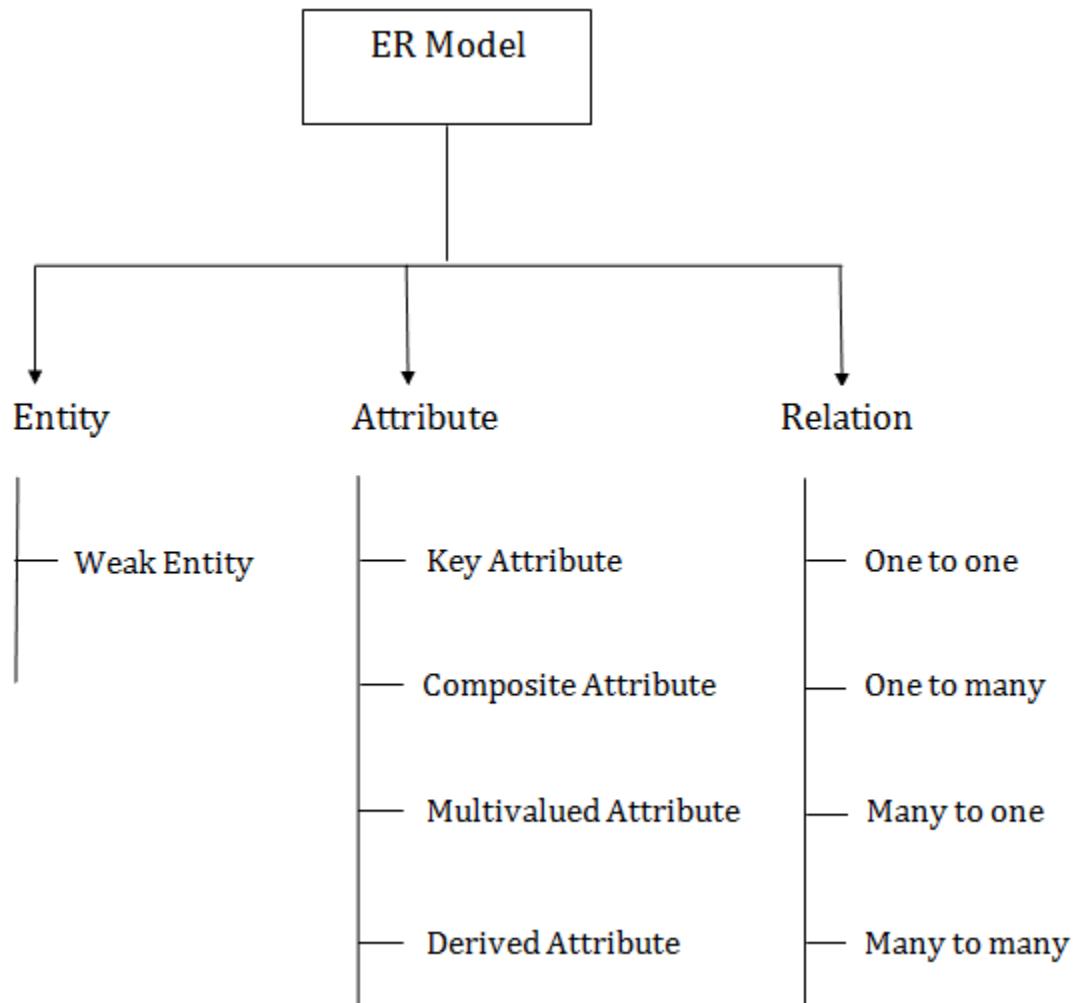


Image23: ER Model Concept Diagram

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept-diagram.png>

Logical Data Model

The logical data model is the one used most in designing BI applications. It builds upon the requirements provided by the business group. It includes a further level of detail, supporting both the business system-related and data requirements.

The business rules are appropriated into the logical data model, where they form relationships between the various data objects and entities. As opposed to a conceptual data model, which may have very general terms, the logical data model is the first step in designing and building out the architecture of the applications.

Like the conceptual data model, the logical data model is independent of specific database and data storage structures. It uses indexes and foreign keys to represent data relationships, but these are defined in a generic database context independent of any specific DBMS product.

The characteristics of the logical data model include:

- Features independent of specific database and data storage structures.
- Specific entities and attributes to be implemented.
- Identification of the business rules and relationships between those entities and attributes.
- Definitions of the primary keys, foreign keys, alternate keys, and inversion entities.

The logical model is used as a bridge from the application designer's view to the database design and the developer's specifications. This model should be used to validate whether the resulting applications that are built fulfill business and data requirements.

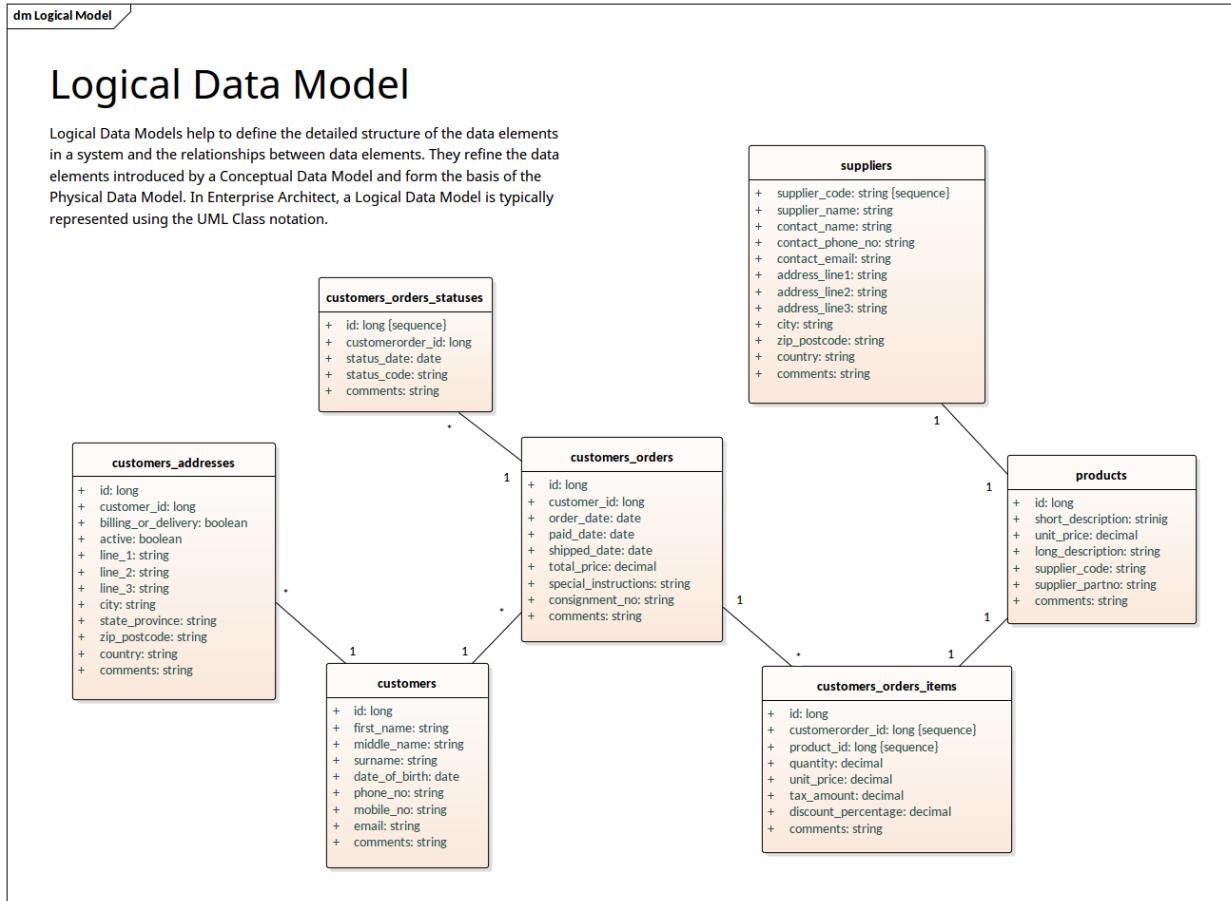


Image24: Logical Data Model Diagram

Reference: <https://sparxsystems.com/resources/gallery/diagrams/images/logical-data-model-uml-notation.png>

Entity Sets

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

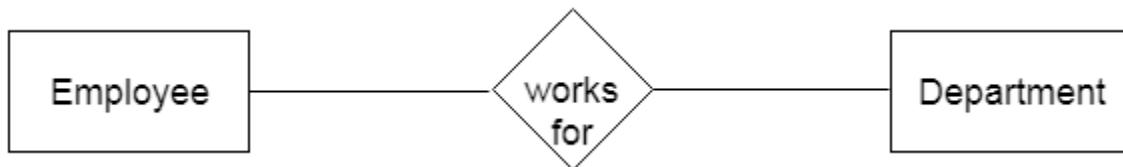


Image25: ER Model for Entity

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept2.png>

Definition of Strong Entity

The **Strong Entity** is the one whose existence does not depend on the existence of any other entity in a schema. It is denoted by a **single rectangle**. A strong entity always has the **primary key** in the set of attributes that describes the strong entity. It indicates that each entity in a strong entity set can be uniquely identified.

Set of similar types of strong entities together forms the **Strong Entity Set**. A strong entity holds the relationship with the weak entity via an **Identifying Relationship**, which is denoted by double diamond in the ER diagram. On the other hand, the relationship between two strong entities is denoted by a single diamond and it is simply called a **relationship**.

Let us understand this concept with the help of an example; a customer borrows a loan. Here we have two entities first a customer entity, and second a loan entity.

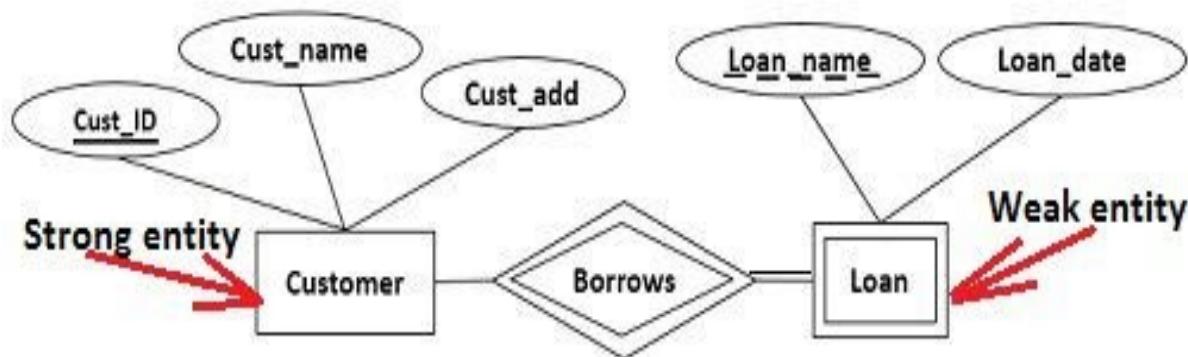


Image26: Strong & Weak Entity Diagram

Reference: <https://techdifferences.com/wp-content/uploads/2016/12/Strong-entity-and-weak-entity.jpg>

Observing the ER-diagram above, for each loan, there should be at least one borrower otherwise that loan would not be listed in the Loan entity set. But even if a customer does not borrow any loan it would be listed in the Customer entity set. So we can conclude that a customer entity does not depend on a loan entity.

Cust_ID	Cust_name	Cust_add
101	John	Xyzzy
103	Ruby	Pqr
109	John	Uvfw

Customer Entity Set

Image27: Customer EntitySet Diagram

Reference: https://techdifferences.com/wp-content/uploads/2016/12/Strong_Entity-set.jpg

The second thing you can observe is that the Customer entity has a primary key Cust_ID which uniquely identifies each entity in Customer Entity set. This makes the Customer entity a strong entity on which a loan entity depends.

Definition of Weak Entity

A **Weak entity** is the one that depends on its owner entity i.e. a strong entity for its existence. A weak entity is denoted by the **double rectangle**. Weak entities do **not** have the **primary key** instead it has a **partial key** that uniquely discriminates against the weak entities. The **primary key of a weak entity** is a composite key formed from the **primary key of the strong entity** and **partial key of the weak entity**.

The collection of similar weak entities is called **Weak Entity Set**. The relationship between a weak entity and a strong entity is always denoted with an **Identifying Relationship** i.e. **double diamond**.

For further illustration let us discuss the above example, this time from a weak entity's point of view. We have Loan as our weak entity, and as I said above for each loan there must be at least one borrower. You can observe in the loan entity set, no customer has borrowed a car loan and hence, it has totally vanished from the loan entity set. For the presence of a car loan in the loan entity set, it must have been borrowed by a customer. In this way, the weak Loan entity is dependent on the strong Customer entity.

Loan_name	Loan_date	Amount
Home	20/11/2015	20000
Education	5/10/2015	10000
Home	20/11/2015	20000

Loan Entity Set

Image28: Loan EntitySet Diagram

Reference: https://techdifferences.com/wp-content/uploads/2016/12/Weak_Entity_set.jpg

The second thing, we know is a weak entity does not have a primary key. So here Loan_name, the partial key of the weak entity and Cust_ID primary key of the customer entity makes the primary key of the loan entity.

In the Loan entity set, we have two exactly same entities i.e. a **Home loan on date 20/11/2015 with amount 20000**. Now how to identify who had borrowed them this can be done with the help of the primary key of the weak entity (Loan_name + Cust_ID). So, it will be determined that one home loan is borrowed by Customer 101 Jhon and other by Customer 103 Ruby. This is how the composite primary key of weak entities identifies each entity in the weak entity set.

Difference between Strong and Weak Entity

S.NO	STRONG ENTITY	WEAK ENTITY
1.	Strong entity always has a primary key.	While a weak entity has a partial discriminator key.

2.	Strong entity is not dependent on any other entity.	Weak entities depend on strong entities.
3.	Strong entity is represented by a single rectangle.	Weak entity is represented by a double rectangle.
4.	Two strong entity's relationships are represented by a single diamond.	While the relation between one strong and one weak entity is represented by a double diamond.
5.	Strong entities have either total participation or not.	While a weak entity always has total participation.

Relationship Sets

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



Image29: Relationship

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept9.png>

Types of relationship are as follows:

One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.



Image30: One-to-One Relationship

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept10.png>

One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, scientists can invent many inventions, but the invention is done by only specific scientist.

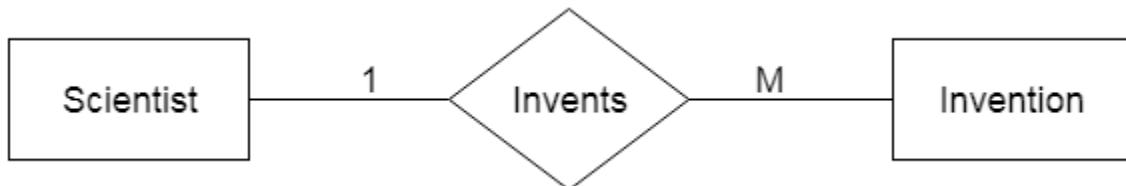


Image31: One-to-Many Relationship

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept11.png>

Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.

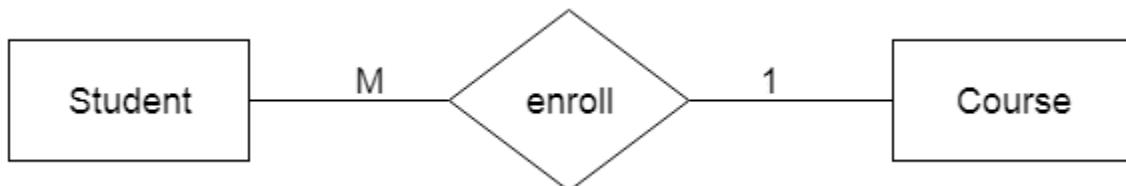


Image32: Many-to-one Relationship

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept12.png>

Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, employees can be assigned to many projects and projects can have many employees.



Image33: Many-to-Many Relationship

Reference: <https://static.javatpoint.com/dbms/images/dbms-er-model-concept13.png>

Concept of Attributes

Introduction

Attributes are the descriptive properties which are owned by each entity of an Entity Set.

There exists a specific domain or set of values for each attribute from where the attribute can take its values.

It is a single-valued property of either an entity-type or a relationship-type.

For example, a lecture might have attributes: time, date, duration, place, etc.

An attribute is represented by an Ellipse

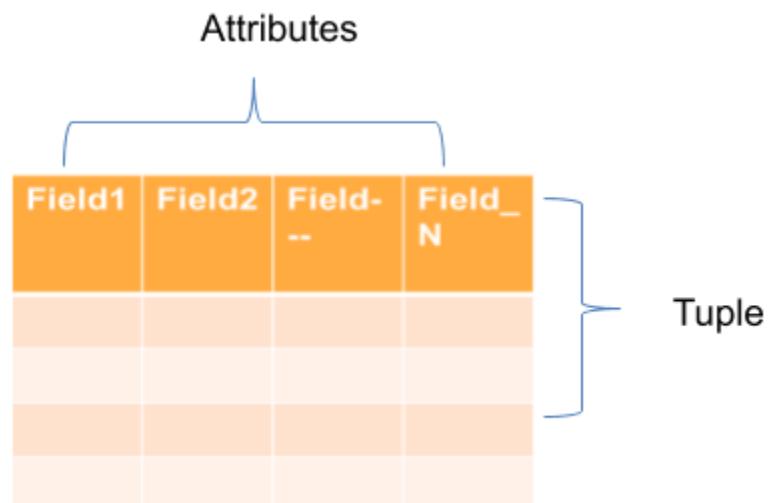
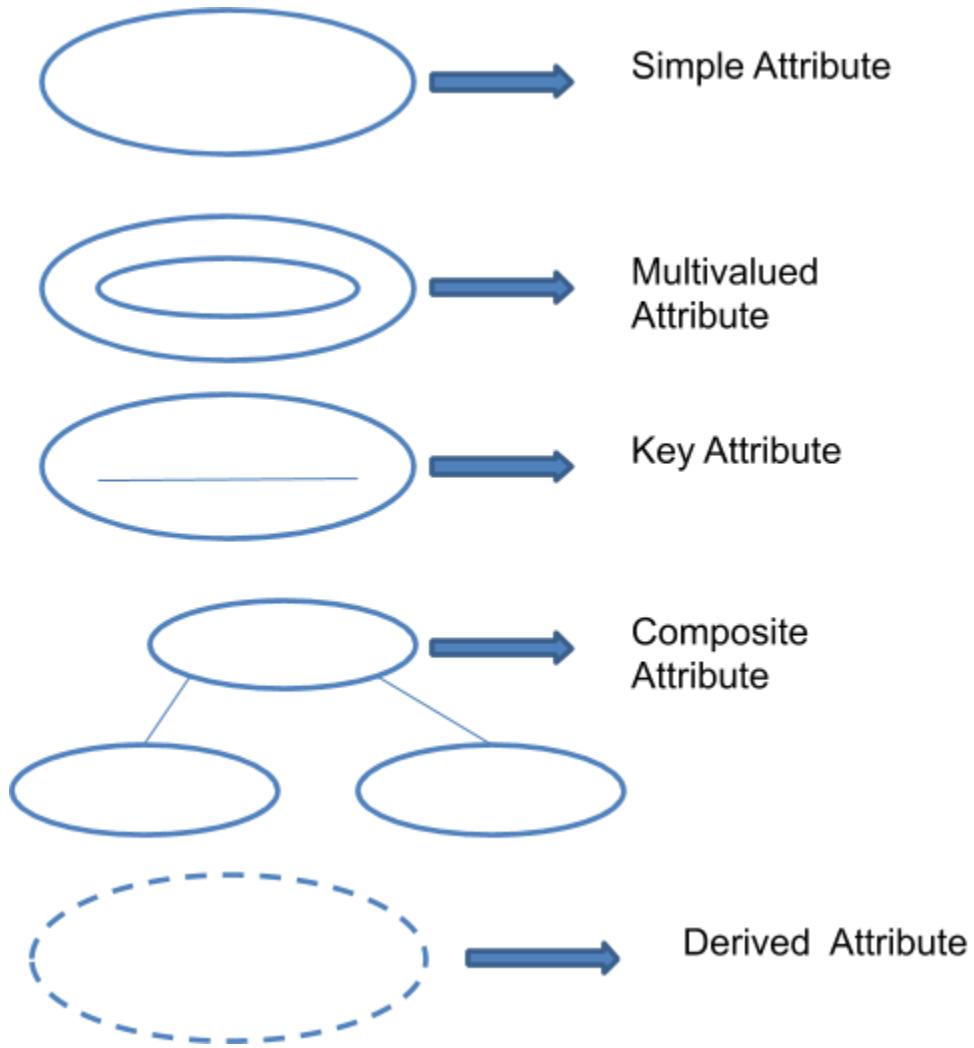


Image 1 – Attributes identification

Symbols used to represent attributes



Types of Attributes

- Simple Attributes
- Composite Attributes
- Single Valued Attributes
- Key Attributes
- Derived Attributes

- Multivalued Attributes

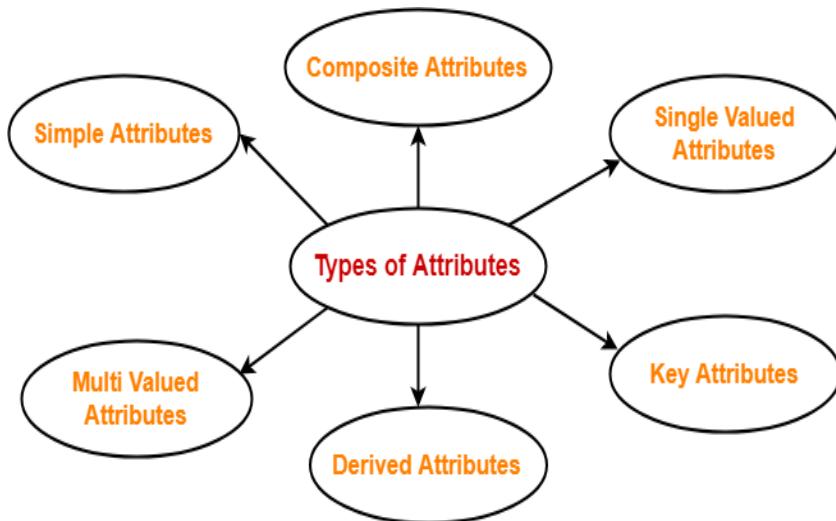


Image 2 – Type of Attributes

Reference - <https://www.gatevidyalay.com/wp-content/uploads/2018/06/Attributes-in-DBMS-Types.png>

Simple Attributes

A simple attribute is identify entity from an entity set.

Simple attribute represent oval symbol with its value.

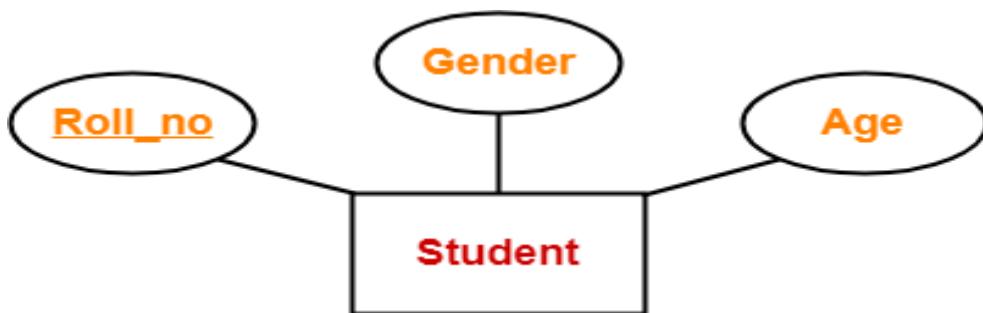


Image 3 – Simple Attribute

Reference - <https://www.gatevidyalay.com/wp-content/uploads/2018/06/Key-Attributes-Example.png>

Here, all the attributes are simple attributes as they cannot be divided further.

Single Valued Attributes

Single valued attributes are those attributes which can take only one value for a given entity from an entity set.

Key Attributes

A key attribute is uniquely identify entity from an entity set.

Key attribute represent oval symbol same as like other with underline.

Derived Attributes

Derived attributes are those attributes which can be derived from other attribute(s).

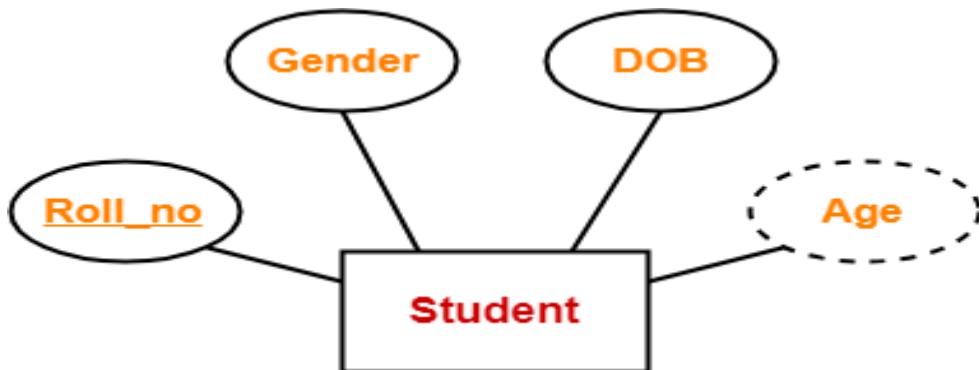


Image 4 – Key Attribute

Reference - <https://www.gatevidyalay.com/wp-content/uploads/2018/06/Derived-Attributes-Example.png>

Here, the attribute “Age” is a derived attribute as it can be derived from the attribute “DOB”.

Multivalued Attributes

Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.

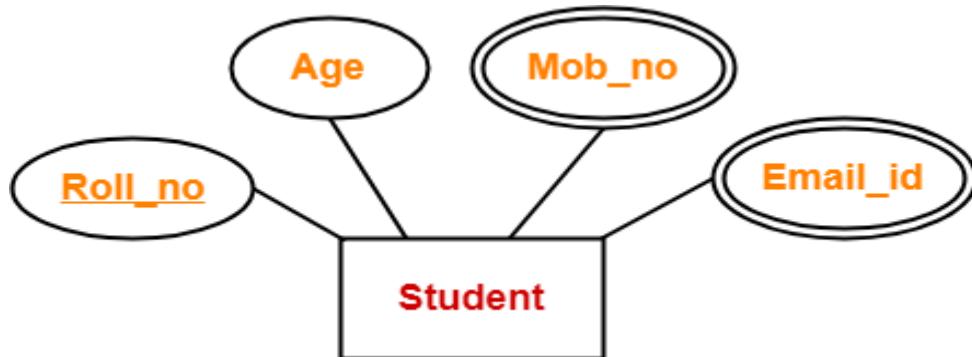


Image 5 – Multivalued Attribute

Reference - <https://www.gatevidyalay.com/wp-content/uploads/2018/06/Multi-Valued-Attributes-Example.png>

Here, the attributes “Mob_no” and “Email_id” are multi valued attributes as they can take more than one values for a given entity.

Example 1 – Student Book Relation

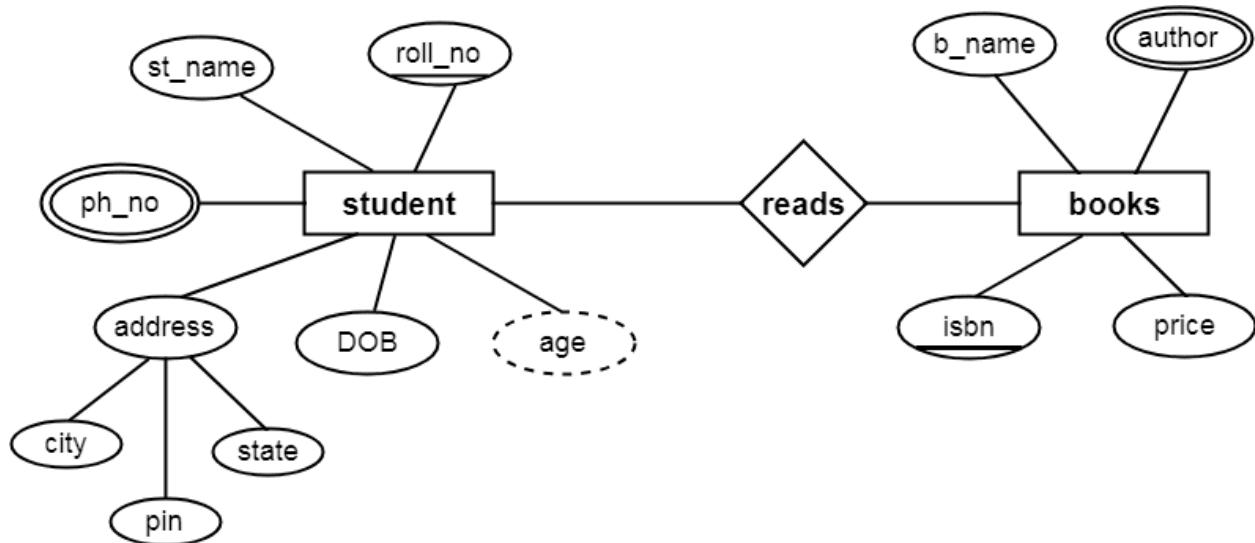


Image 6 – Student Book Relation

Reference - <https://www.csetutor.com/wp-content/uploads/2018/09/ER-Diagram-in-DBMS-Example.png>

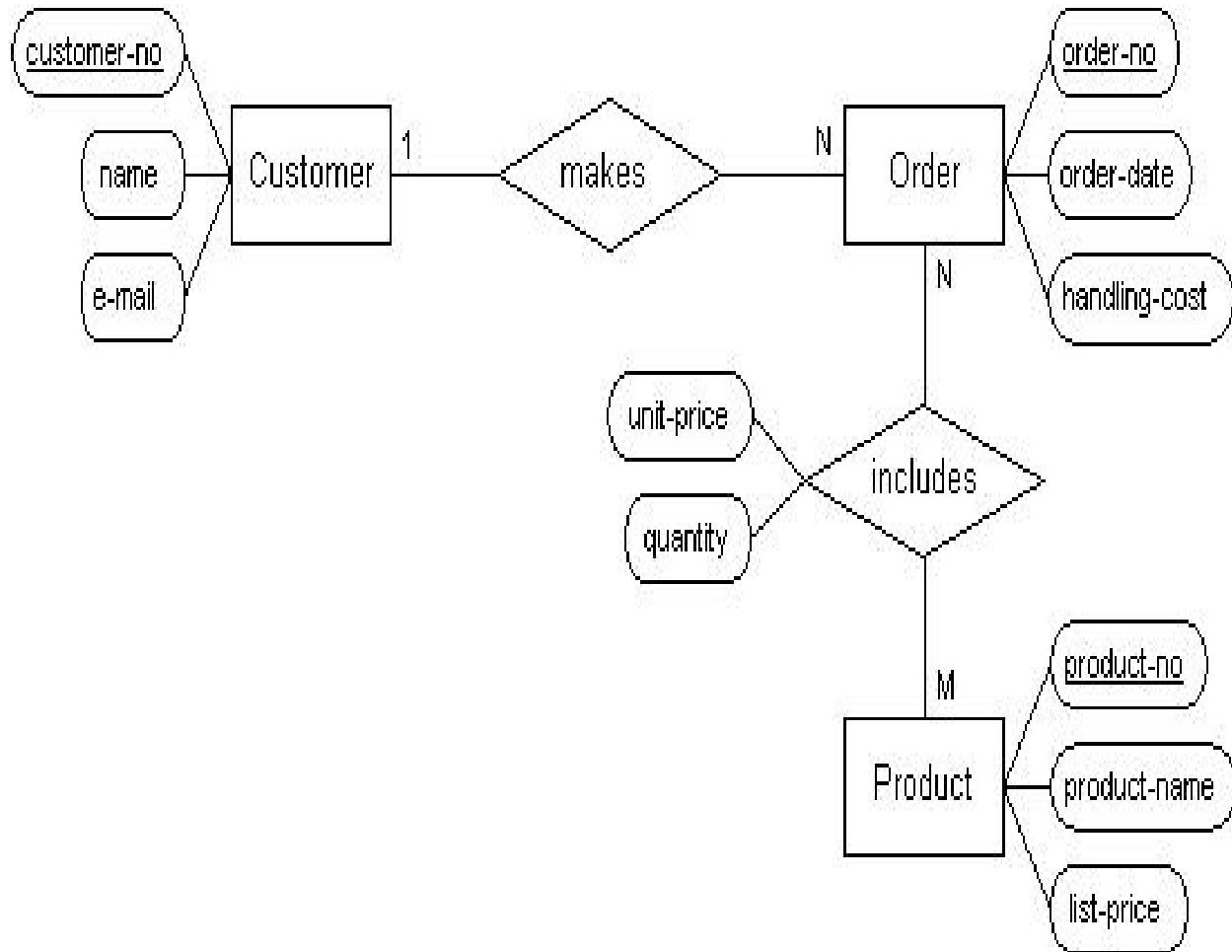
Example 2 – Product Order System

Image 7 – Product Order System

Reference - <https://codeandwork.github.io/courses/cs/media/erdiagram2.jpg>

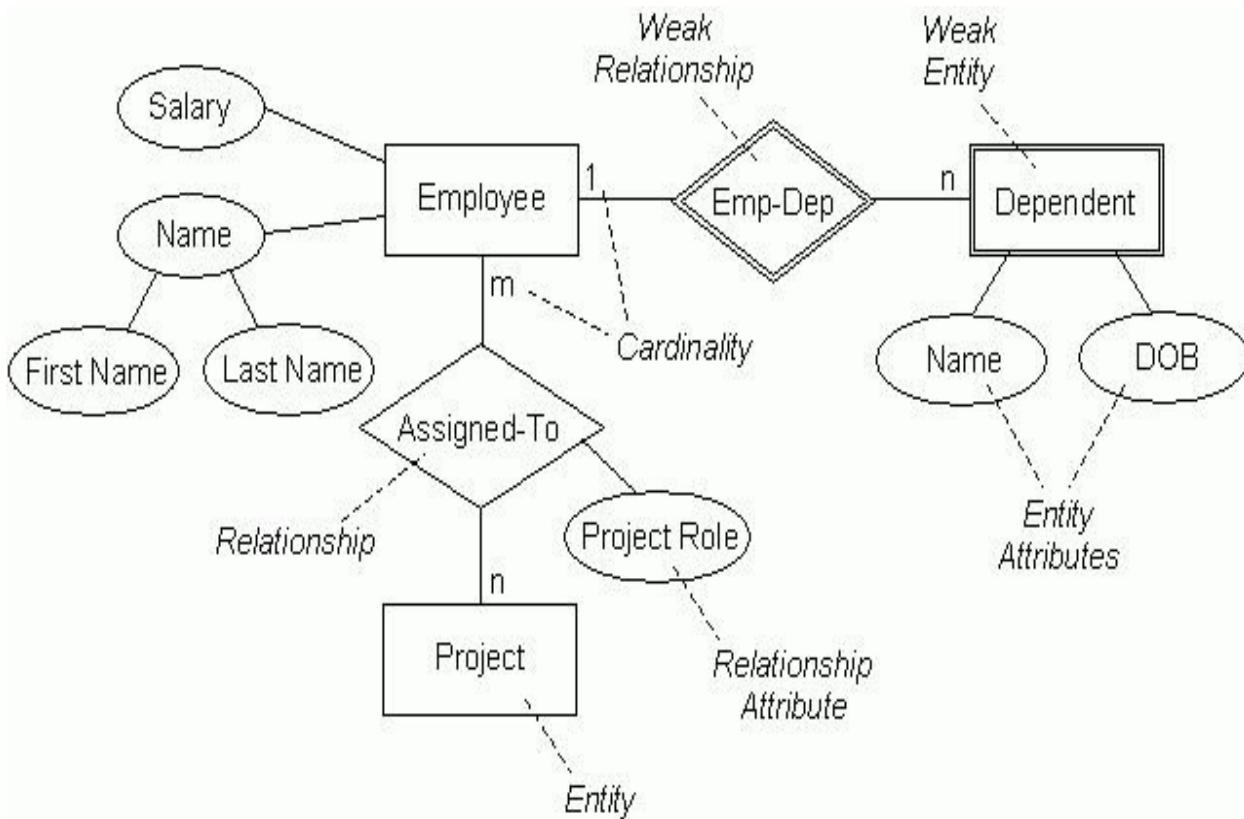
Example 3 –Employee Management

Image 8 – Employee Management

Reference - <https://codeandwork.github.io/courses/cs/media/erd-employee.jpg>

Concept of Relationship

Introduction

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.

Relationship is nothing but an association among two or more entities.

Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

Symbol



Example

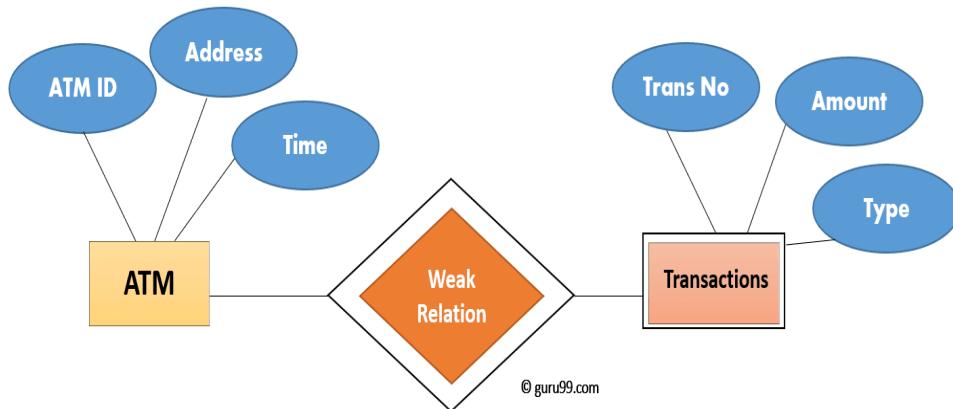


Image 9 – Transaction

Reference - https://www.guru99.com/images/1/100518_0621_ERDiagramTu5.png

Types of Keys

Keys

Keys play an important role in the relational database.

It is used to uniquely identify any record or row of data from the table.

It is also used to establish and identify relationships between tables.

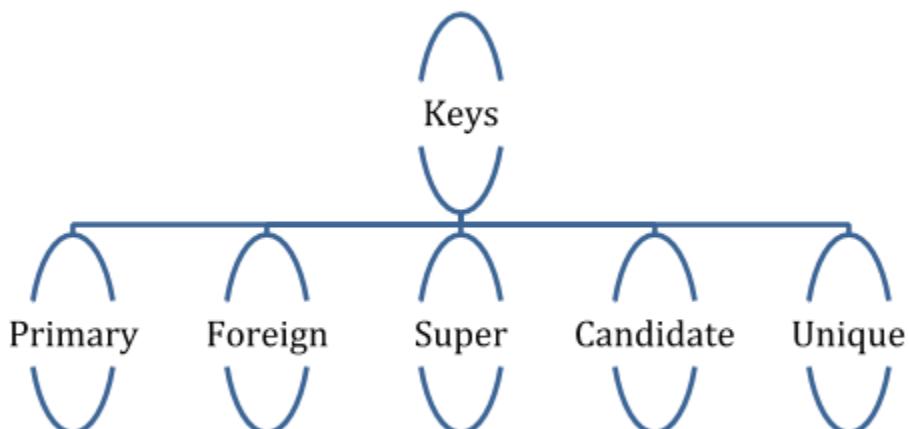


Image 10 – Transaction

Primary Key

PRIMARY KEY It is the first key which is used to identify one and only one instance of an entity uniquely.

Rules Defining Primary Key –

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

There can be more than one candidate key in relation out of which one can be chosen as the primary key. For Example, STUD_NO, as well as STUD_PHONE both, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys).

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNT_RY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajasthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Image 11 – Student table

Reference - <https://media.geeksforgeeks.org/wp-content/uploads/image7.png>

Foreign Key

FOREIGN KEY is a column that creates a relationship between two tables.

The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.

Rules defining foreign key –

- Foreign key columns must use their referenced column's type.
- Each column cannot belong to more than 1 Foreign Key constraint.
- Cannot be a computed column.
- Foreign key columns must be indexed.

Based of Image 9 f an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers. The relation which is being referenced is called referenced relation and the corresponding attribute is called referenced attribute and the relation which refers to the referenced relation is called referencing relation and the corresponding attribute is called referencing attribute. The referenced attribute of the referenced relation should be the primary key for it. For Example, STUD_NO in STUDENT_COURSE is a foreign key to STUD_NO in STUDENT relation.

It may be worth noting that unlike, Primary Key of any given relation, Foreign Key can be NULL as well as may contain duplicate tuples i.e. it need not follow uniqueness constraint.

For Example, STUD_NO in STUDENT_COURSE relation is not unique. It has been repeated for the first and third tuple. However, the STUD_NO in STUDENT relation is a primary key and it needs to be always unique and it cannot be null.

Candidate Key

A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.

The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.

Rules defining candidate key –

- It must contain unique values
- Candidate key may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

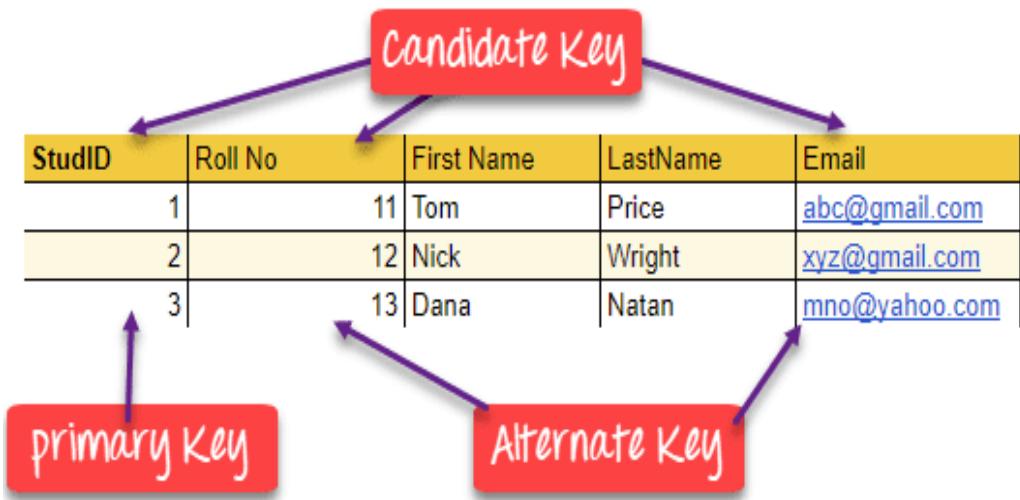


Image 12 – Candidate Key

Reference - https://www.guru99.com/images/1/100518_0517_DBMSKeysPri1.png

Super Key

A superkey is a group of single or multiple keys which identifies rows in a table.

A Super key may have additional attributes that are not needed for unique identification.

In the above-given example, EmpNo and Emp_Name are superkeys.

The set of attributes which can uniquely identify a tuple is known as Super Key. For Example, STUD_NO, (STUD_NO, STUD_NAME) etc.

Adding zero or more attributes to candidate key generates super key.

A candidate key is a super key but vice versa is not true.

Compound Key

COMPOUND KEY has two or more attributes that allow you to uniquely recognize a specific record.

It is possible that each column may not be unique by itself within the database

Order ID and Product ID could be used as it uniquely identified each record.

In database design, a composite key is a candidate key that consists of two or more attributes (table columns) that together uniquely identify an entity occurrence (table row). A compound key is a composite key for which each attribute that makes up the key is a simple (foreign) key in its own right.

Alternate Key

ALTERNATE KEYS is a column or group of columns in a table that uniquely identify every row in that table.

A table can have multiple choices for a primary key but only one can be set as the primary key.

All the keys which are not primary key are called an Alternate Key.

The candidate key other than the primary key is called an alternate key. For Example, STUD_NO, as well as STUD_PHONE both, are candidate keys for relation STUDENT but STUD_PHONE will be alternate key (only one out of many candidate keys).

Unique Key

A unique key is a set of one or more than one fields/columns of a table that uniquely identify a record in a database table.

In database relational modeling and implementation, a unique key (also known as a candidate key or just a key) is a set of attributes (columns) within a relational database table (also called a relation), such that:

the table does not have two distinct rows or records with the same values for these columns;

this set of columns is minimal; i.e., removing any column from the key would result in duplicate values in the resulting subset.

When a column or set of columns is defined as unique to the database management system, the system verifies that each set of values is unique before assigning the constraint. After the columns are defined as unique, an error will occur if an insertion is attempted with values that already exist. Some systems will not allow key values to be updated, all systems will not allow duplicates. This ensures that uniqueness is maintained in both the primary table and any relations that are later bound to it.

Mapping Constraints

Introduction

A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.

Mapping constraints defines how many entities can be related to another entity to a relationship.

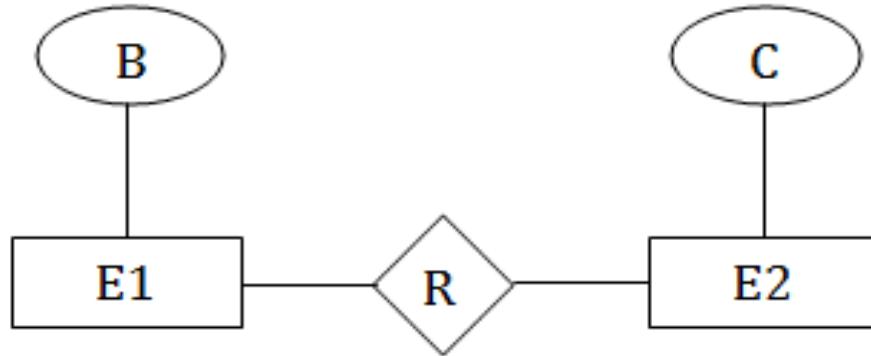


Image 13 – Mapping Constraint

Reference - <https://static.javatpoint.com/dbms/images/dbms-mapping-constraints4.png>

It is most useful in describing the relationship sets that involve more than two entity sets.

For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:

Types of Mapping Constraints

- One to one (1:1)
- One to many (1:M)
- Many to one (M:1)
- Many to many (M:M)

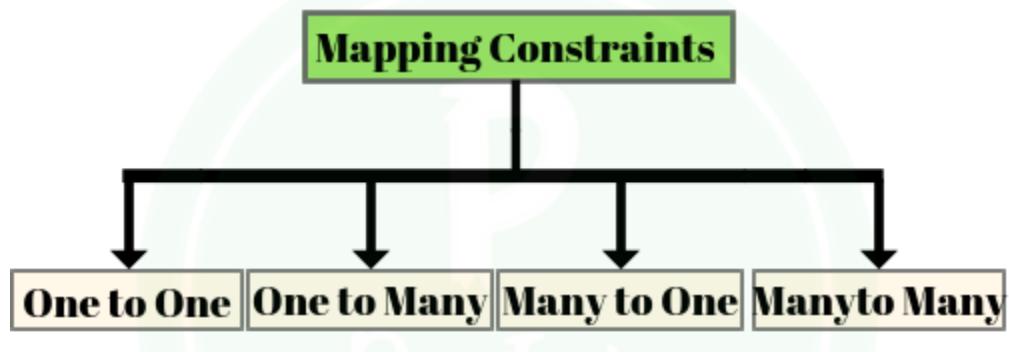


Image 14 – Mapping Constraint

Reference - <https://prepinsta.com/wp-content/uploads/2019/07/30-300x300.png>

One-to-One

In one-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1.

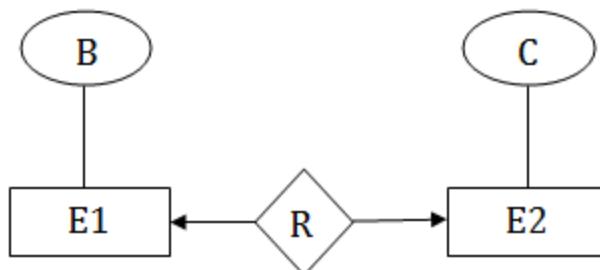


Image 15 – One to One

Reference - <https://static.javatpoint.com/dbms/images/dbms-mapping-constraints.png>

One-to-many

In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.

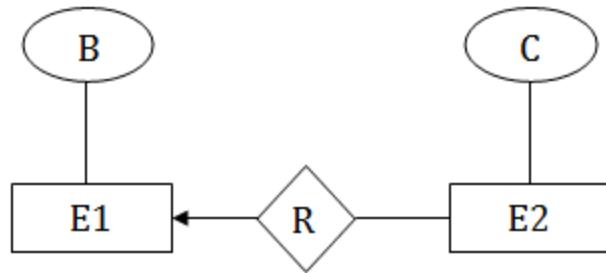


Image 16 – One to Many

Reference - <https://static.javatpoint.com/dbms/images/dbms-mapping-constraints2.png>

Many-to-One

In one-to-many mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.

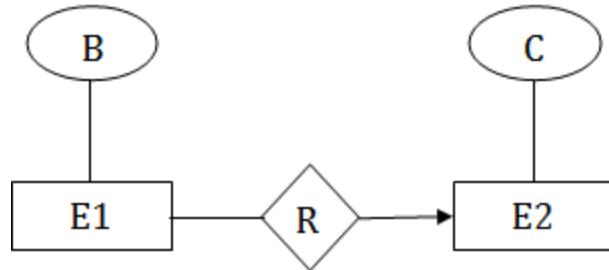


Image 17 – One to Many

Reference - <https://static.javatpoint.com/dbms/images/dbms-mapping-constraints2.png>

Many-to-Many

In many-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1.

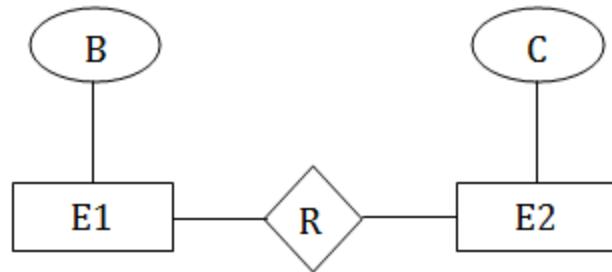


Image 18 – One to Many

Reference - <https://static.javatpoint.com/dbms/images/dbms-mapping-constraints4.png>

Example of Employee Management System

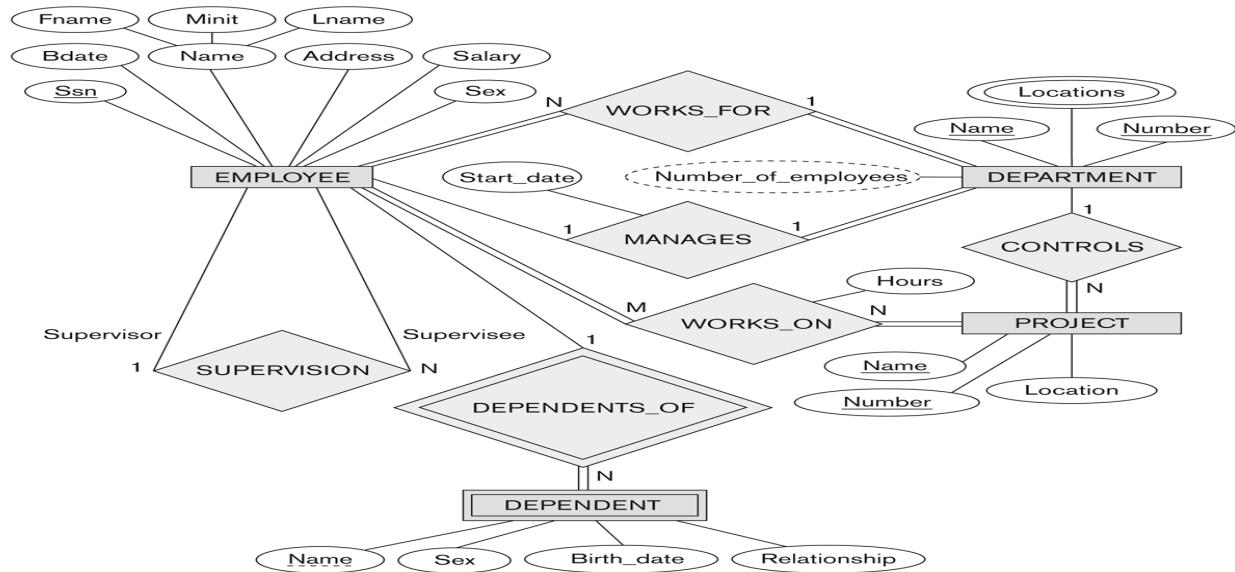


Image 19 – Employee Management

Reference - <http://pld.cs.luc.edu/database/images/fig7.2.png>

Example of Movie Ticket Management

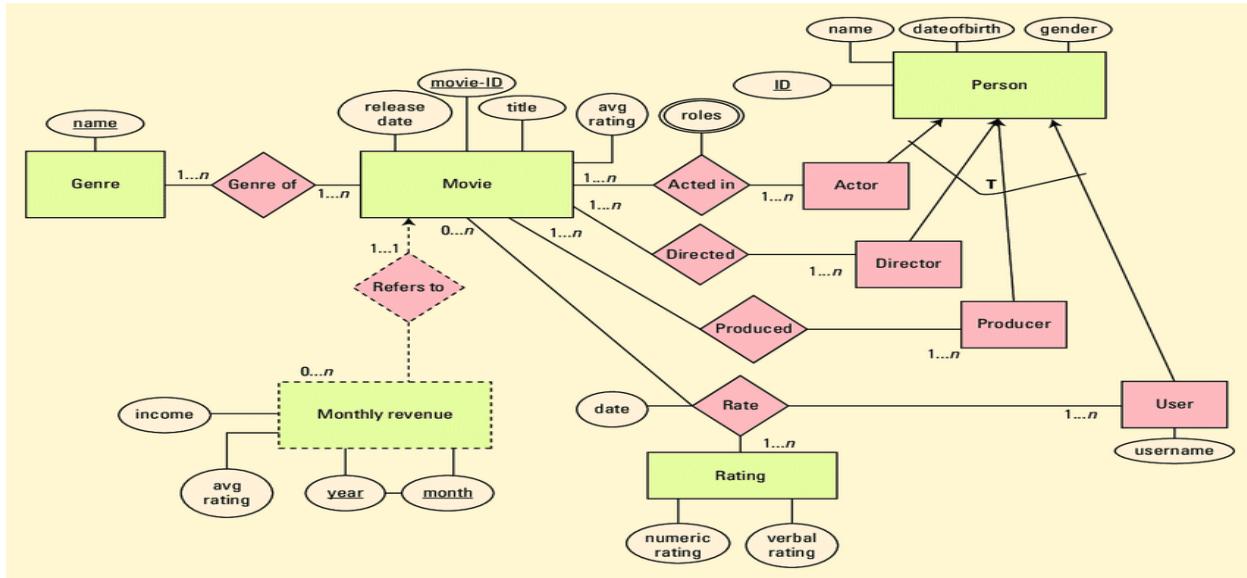


Image 20 – Movie Ticket Management

Reference -

https://www.researchgate.net/profile/Peretz_Shoval/publication/321352935/figure/fig1/AS:571569942142976@1513284301301/The-entity-relationship-diagram-for-the-movie-recommendation-system-Subtypes-are-not.png

Entity Relationship Diagram

Introduction

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

Components of ER Diagram

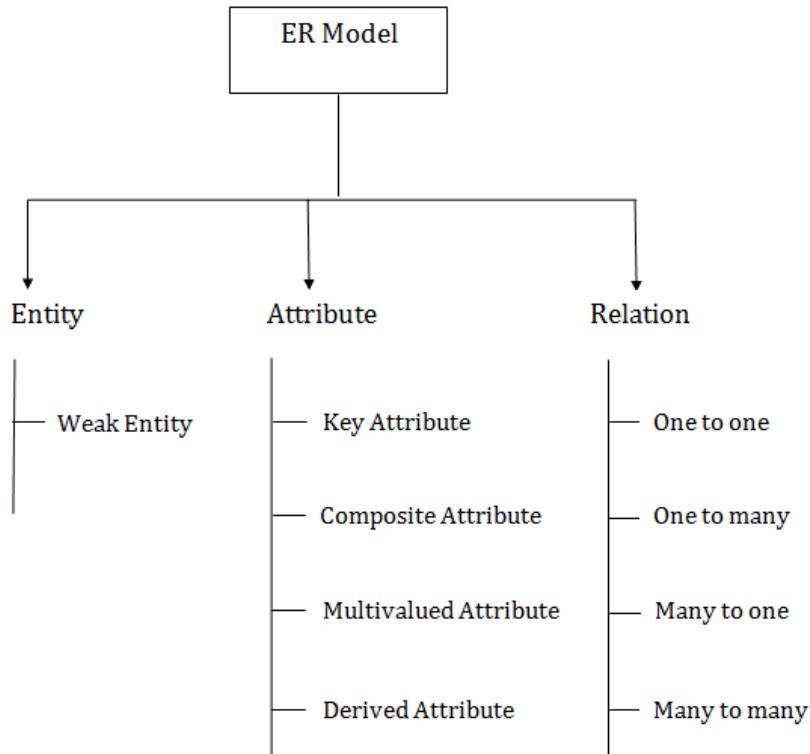


Image 21 – Components of ERD

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept-diagram.png>

Entity

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

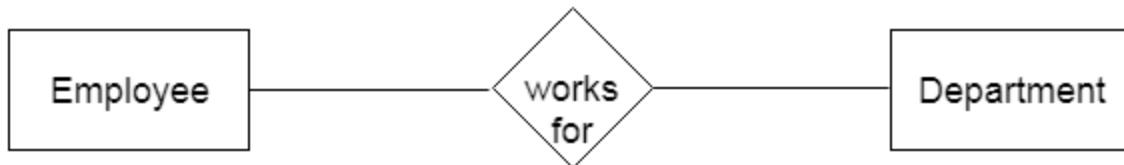


Image 22 – Entity

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept2.png>

a. Weak Entity

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

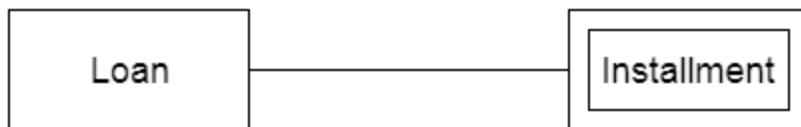


Image 23 – Weak Entity

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept3.png>

Examples of entities:

Person: Employee, Student, Patient

Place: Store, Building

Object: Machine, product, and Car

Event: Sale, Registration, Renewal

Concept: Account, Course

Difference between Strong and Weak Entity

Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

Attribute

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.

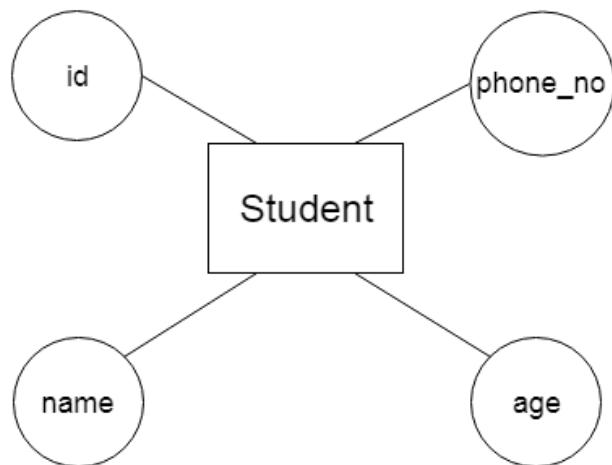


Image 23 – Attribute

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept4.png>

a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

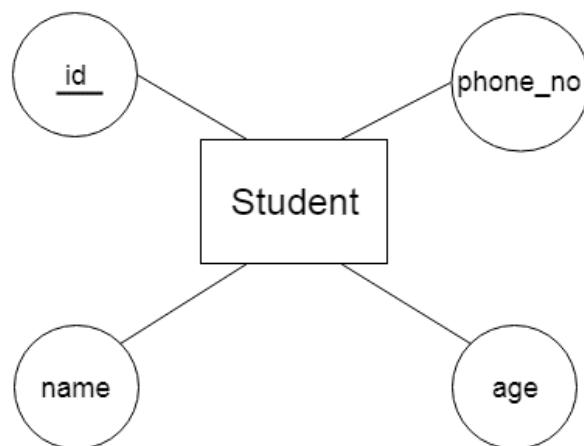


Image 24 – Key Attribute

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept5.png>

b. Composite Attribute

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

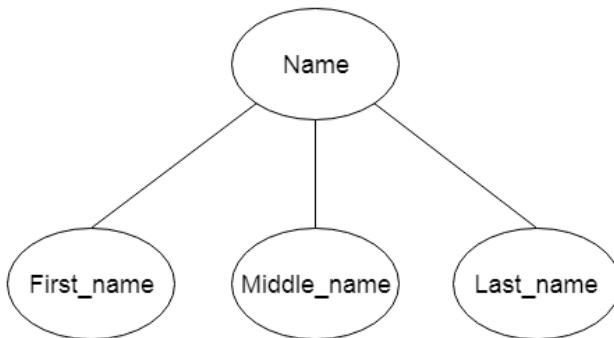


Image 25 – Composite Attribute

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept6.png>

c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.

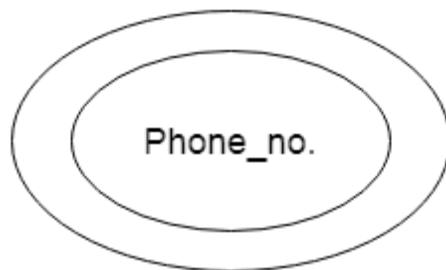


Image 26 – Multivalued Attribute

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept7.png>

d. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.

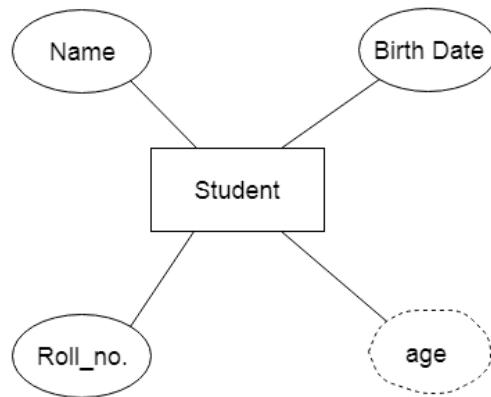


Image 27 – Derived Attribute

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept8.png>

3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



Image 28 – Relationship Attribute

Reference - <https://static.javatpoint.com/dbms/images/dbms-er-model-concept9.png>

Weak Relationship

Transaction ID is weak relation between bank and customer account money withdrawal

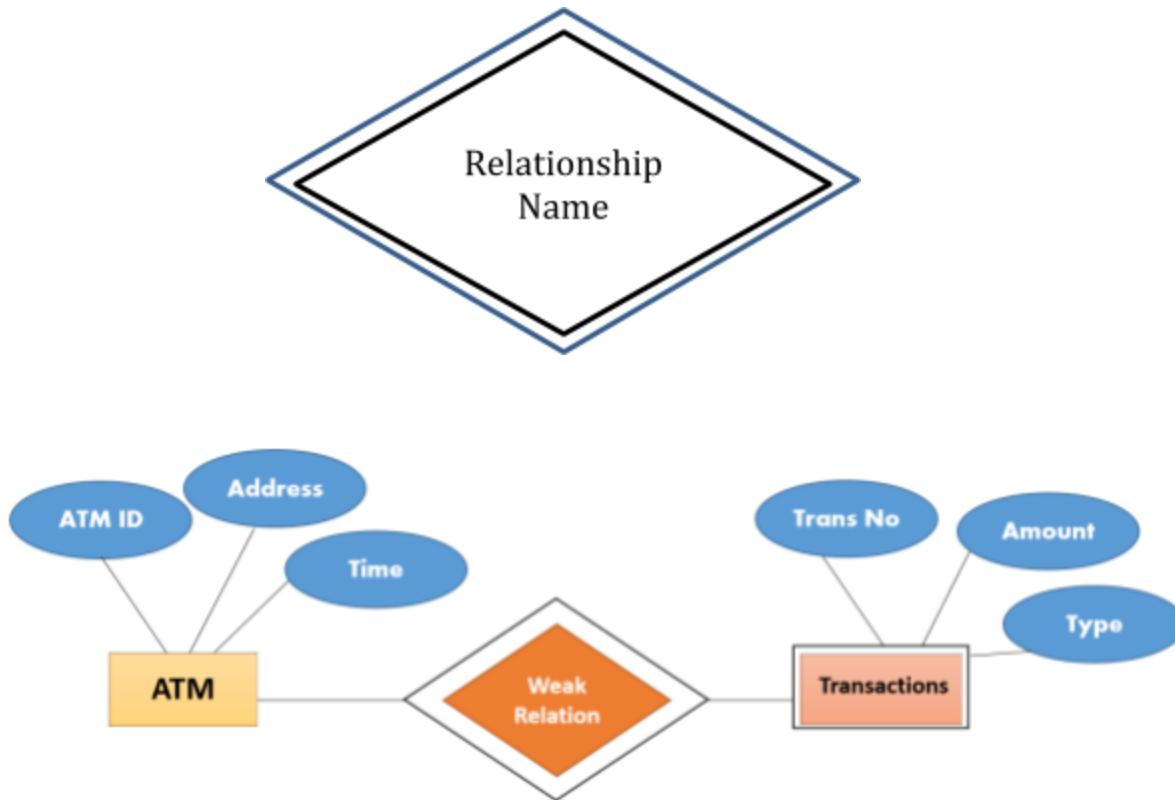


Image 29 – Relationship Attribute

Reference - https://www.guru99.com/images/1/100518_0621_ERDiagramTu5.png

Example : Library Book Management

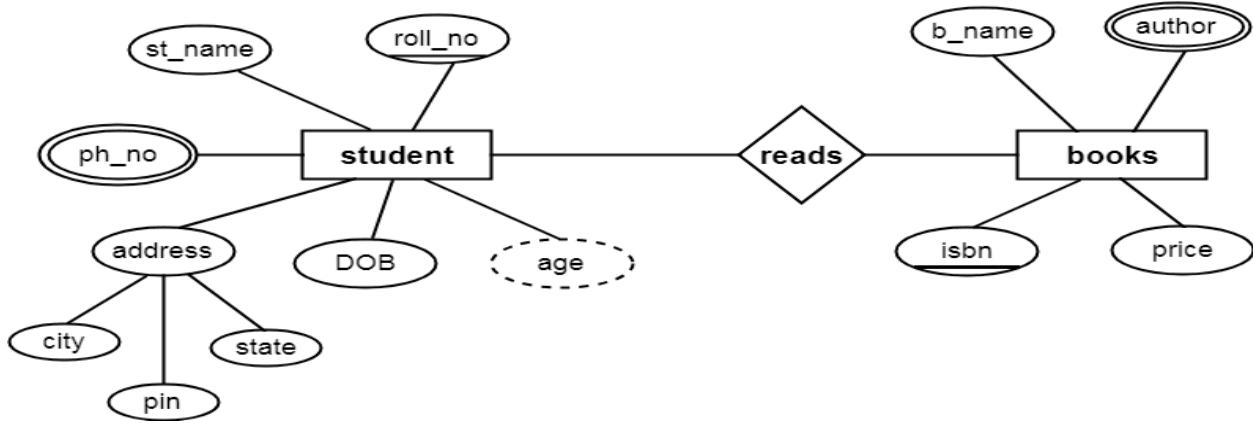


Image 30 – Library Book Management

Reference - <https://www.csetutor.com/wp-content/uploads/2018/09/ER-Diagram-in-DBMS-Example.png>

Example: Employee Management System

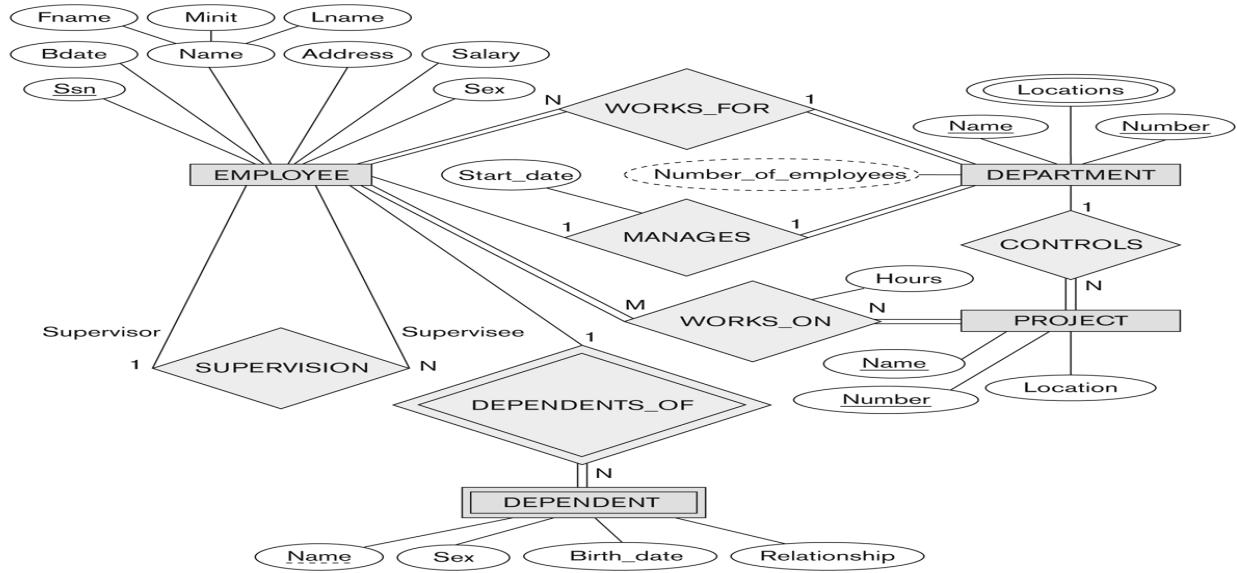


Image 31 – Employee Management

Reference - <http://pld.cs.luc.edu/database/images/fig7.2.png>

Example: Banking Organization

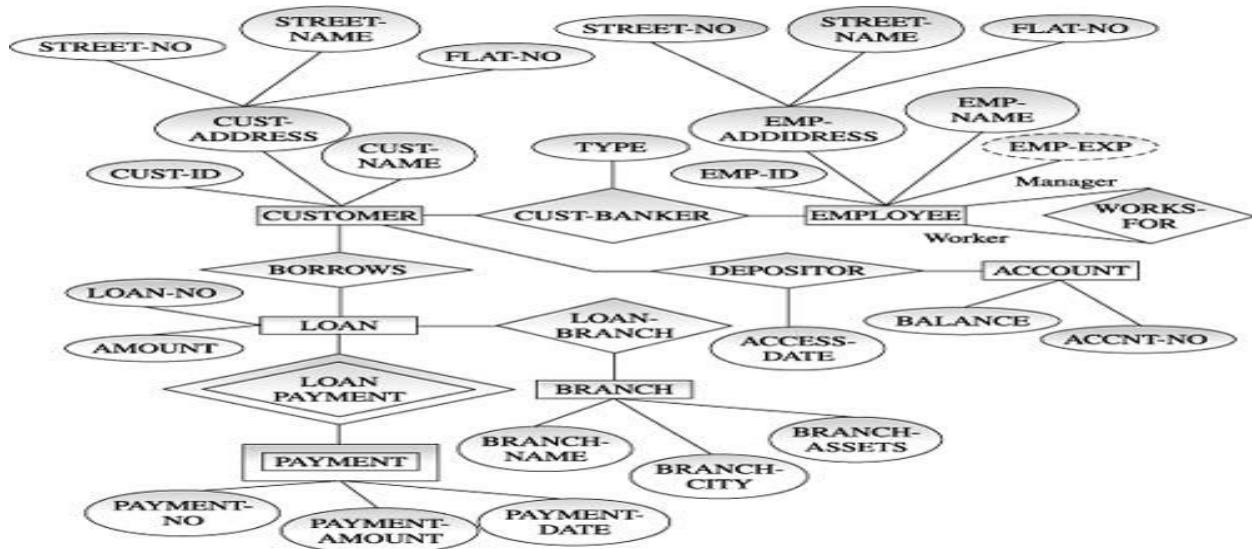


Image 32 – Banking Management

Reference - https://www.oreilly.com/library/view/database-systems-concepts/9788177585674/9788177585674_ch06lev1sec5_image01.jpeg

Example: Hospital Management

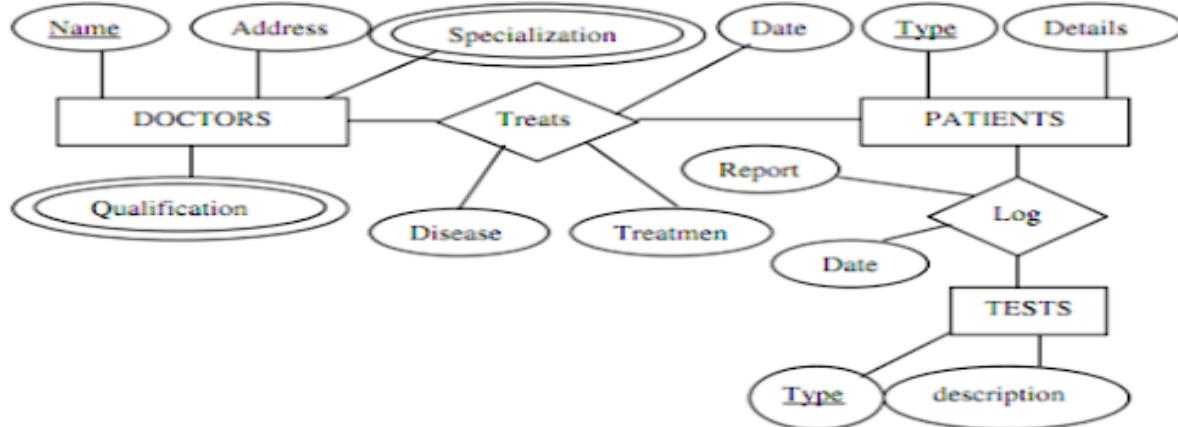


Image 33 – Banking Management

Reference -

https://lh3.googleusercontent.com/proxy/ApY5GXZPpliyg9sRl18JwNH8N4Dyd3wAZQNwwtlIEwk4Q89tllJq_P4RwtlKUvVEQw0zaS4VydBz1soUEARVNFnHv9YLu5y17jQ9EdTrGGD1zgWnJYvx09MMc9C7Y70

Relational Model

Introduction

Relational Model was proposed by E.F. Codd to model data in the form of relations or tables. After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDMBS languages like Oracle SQL, MySQL etc. So we will see what Relational Model is.

What is Relational Model?

Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables). Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE and AGE shown in Table 1.

STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156253132	18

IMPORTANT TERMINOLOGIES

Attribute: Attributes are the properties that define a relation. e.g.; ROLL_NO, NAME

Relation Schema: A relation schema represents name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE and AGE) is relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.

Tuple: Each row in the relation is known as tuple. The above relation contains 4 tuples, one of which is shown as:

1 RAM DELHI 9455123451 18

Relation Instance: The set of tuples of a relation at a particular instance of time is called as relation instance. Table 1 shows the relation instance of STUDENT at a particular time. It can change whenever there is insertion, deletion or updation in the database.

Degree: The number of attributes in the relation is known as degree of the relation. The STUDENT relation defined above has degree 5.

Cardinality: The number of tuples in a relation is known as cardinality. The STUDENT relation defined above has cardinality 4.

Column: Column represents the set of values for a particular attribute. The column ROLL_NO is extracted from relation STUDENT.

ROLL_NO

1

2

3

4

NULL Values: The value which is not known or unavailable is called NULL value. It is represented by blank space. e.g.; PHONE of STUDENT having ROLL_NO 4 is NULL.

Constraints in Relational Model

While designing Relational Model, we define some conditions which must hold for data present in database are called Constraints. These constraints are checked before performing any operation (insertion, deletion and updation) in database. If there is a violation in any of constraints, operation will fail.

Domain Constraints: These are attribute level constraints. An attribute can only take values which lie inside the domain range. e.g.,; If a constraint AGE>0 is applied on STUDENT relation, inserting negative value of AGE will result in failure.

Key Integrity: Every relation in the database should have atleast one set of attributes which defines a tuple uniquely. Those set of attributes is called key. e.g.; ROLL_NO in STUDENT is a key. No two students can have same roll number. So a key has two properties:

It should be unique for all tuples.

It can't have NULL values.

Referential Integrity: When one attribute of a relation can only take values from other attribute of same relation or any other relation, it is called referential integrity. Let us suppose we have 2 relations

STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE	BRANCH_CODE
1	RAM	DELHI	9455123451	18	CS
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI		18	IT

BRANCH

BRANCH_CODE	BRANCH_NAME
CS	COMPUTER SCIENCE
IT	INFORMATION TECHNOLOGY
ECE	ELECTRONICS AND COMMUNICATION ENGINEERING
CV	CIVIL ENGINEERING

BRANCH_CODE of STUDENT can only take the values which are present in BRANCH_CODE of BRANCH which is called referential integrity constraint. The relation which is referencing to other relation is called REFERENCING RELATION (STUDENT in this

case) and the relation to which other relations refer is called REFERENCED RELATION (BRANCH in this case).

An anomaly is an irregularity, or something which deviates from the expected or normal state. When designing databases, we identify three types of anomalies: Insert, Update and Delete.

Insertion Anomaly in Referencing Relation:

We can't insert a row in REFERENCING RELATION if referencing attribute's value is not present in referenced attribute value. e.g.; Insertion of a student with BRANCH_CODE 'ME' in STUDENT relation will result in error because 'ME' is not present in BRANCH_CODE of BRANCH.

Deletion/ Updation Anomaly in Referenced Relation:

We can't delete or update a row from REFERENCED RELATION if value of REFERENCED ATTRIBUTE is used in value of REFERENCING ATTRIBUTE. e.g; if we try to delete tuple from BRANCH having BRANCH_CODE 'CS', it will result in error because 'CS' is referenced by BRANCH_CODE of STUDENT, but if we try to delete the row from BRANCH with BRANCH_CODE CV, it will be deleted as the value is not been used by referencing relation. It can be handled by following method:

ON DELETE CASCADE: It will delete the tuples from REFERENCING RELATION if value used by REFERENCING ATTRIBUTE is deleted from REFERENCED RELATION. e.g; if we delete a row from BRANCH with BRANCH_CODE 'CS', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be deleted.

ON UPDATE CASCADE: It will update the REFERENCING ATTRIBUTE in REFERENCING RELATION if attribute value used by REFERENCING ATTRIBUTE is updated in REFERENCED RELATION. e.g; if we update a row from BRANCH with BRANCH_CODE 'CS' to 'CSE', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be updated with BRANCH_CODE 'CSE'.

SUPER KEYS:

Any set of attributes that allows us to identify unique rows (tuples) in a given relation are known as super keys. Out of these super keys we can always choose a proper subset among these which

can be used as a primary key. Such keys are known as Candidate keys. If there is a combination of two or more attributes which is being used as the primary key then we call it as a Composite key.

Attribute - Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.

Tables – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

Tuple – It is nothing but a single row of a table, which contains a single record.

Relation Schema: A relation schema represents the name of the relation with its attributes.

Degree - The total number of attributes which in the relation is called the degree of the relation.

Cardinality - Total number of rows present in the Table.

Column - The column represents the set of values for a specific attribute.

Relation instance – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.

Relation Key - Every row has one, two or multiple attributes, which is called relation key.

Attribute Domain – Every attribute has some pre-defined value and scope which is known as attribute domain.

Properties of Relation

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- Tuple has no duplicate value
- Order of tuple can have a different sequence

Example 1

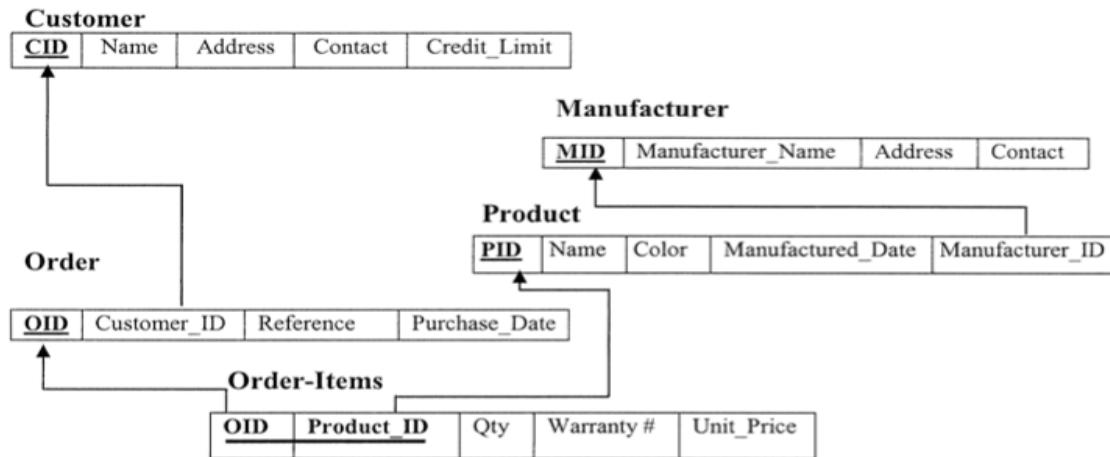


Image 34 – Production Management

Reference - <https://d1whlypfis84e.cloudfront.net/guides/wp-content/uploads/2019/01/01111318/Relational-databse.png>

Example 2

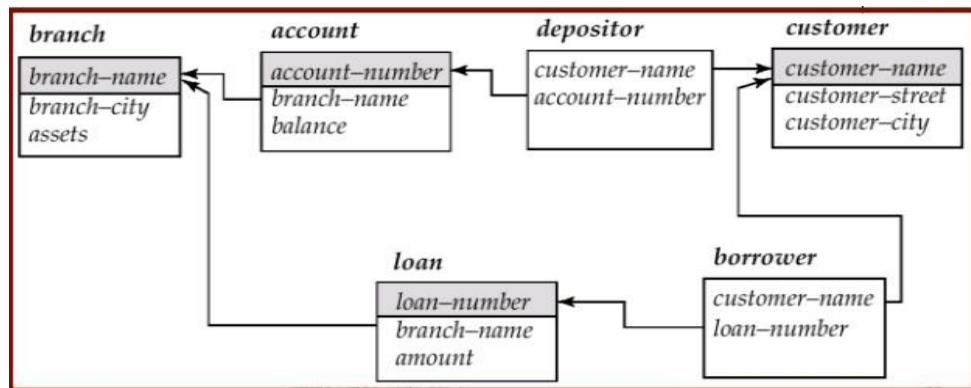


Image 35 – Production Management

Reference -

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.slideshare.net%2FProsantaGhosh%2Fdbms-ii-mcach4relational-model2013&sig=AQvVaw1UFOBY4OLPKvlkomOjJh9&ust=1587013292216000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCJC5_K_T6egCFQAAA
AAdAAAAABAD

Network Model

Introduction

Charles Bachman was the original inventor of the network model. In 1969, the Conference on Data Systems Languages (CODASYL) Consortium developed the network model into a standard specification. A second publication was introduced in 1971, which later turned into the basis for virtually all implementations.

The benefits of the network model include:

Simple Concept: Similar to the hierarchical model, this model is simple and the implementation is effortless.

Ability to Manage More Relationship Types: The network model has the ability to manage one-to-one (1:1) as well as many-to-many (N: N) relationships.

Easy Access to Data: Accessing the data is simpler when compared to the hierarchical model.

Data Integrity: In a network model, there's always a connection between the parent and the child segments because it depends on the parent-child relationship.

Data Independence: Data independence is better in network models as opposed to the hierarchical models.

The drawbacks of the network model include:

System Complexity: Each and every record has to be maintained with the help of pointers, which makes the database structure more complex.

Functional Flaws: Because a great number of pointers is essential, insertion, updates, and deletion become more complex.

Lack of Structural Independence: A change in structure demands a change in the application as well, which leads to lack of structural independence.

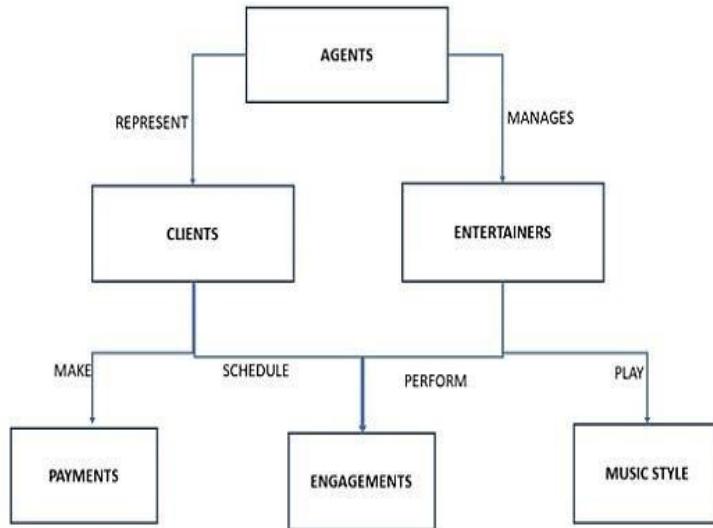


Image 36 – Network Model

Reference - <https://www.tutorialspoint.com/assets/questions/images/120543-1532343127.jpg>

The network model is the extension of the hierarchical structure because it allows many-to-many relationships to be managed in a tree-like structure that allows multiple parents.

It can represent redundancy in data more efficiently than that in the hierarchical model.

There can be more than one path from a previous node to successor node/s.

The operations of the network model are maintained by indexing structure of linked list (circular) where a program maintains a current position and navigates from one record to another by following the relationships in which the record participates.

Records can also be located by supplying key values.

Hierarchical Model

Introduction

A hierarchical model represents the data in a tree-like structure in which there is a single parent for each record. To maintain order there is a sort field which keeps sibling nodes into a recorded manner. These types of models are designed basically for the early mainframe database management systems, like the Information Management System (IMS) by IBM.

This model structure allows the one-to-one and a one-to-many relationship between two/ various types of data. This structure is very helpful in describing many relationships in the real world; table of contents, any nested and sorted information.

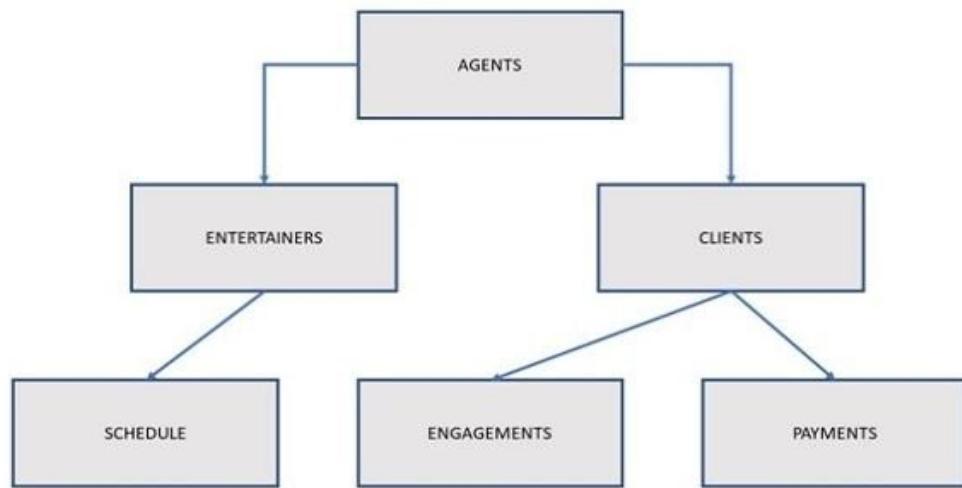


Image 37 – Hierarchical Model

Reference - <https://www.tutorialspoint.com/assets/questions/images/154411-1532346635.jpg>

Advantages

A user can retrieve data very quickly due to the presence of explicit links between the table structures.

The referential integrity is built in and automatically enforced due to which a record in a child table must be linked to an existing record in a parent table, along with that if a record deleted in the parent table then that will cause all associated records in the child table to be deleted as well.

Disadvantages

When a user needs to store a record in a child table that is currently unrelated to any record in a parent table, it gets difficulty in recording and user must record an additional entry in the parent table.

This type of database cannot support complex relationships, and there is also a problem of redundancy, which can result in producing inaccurate information due to the inconsistent recording of data at various sites.

Example

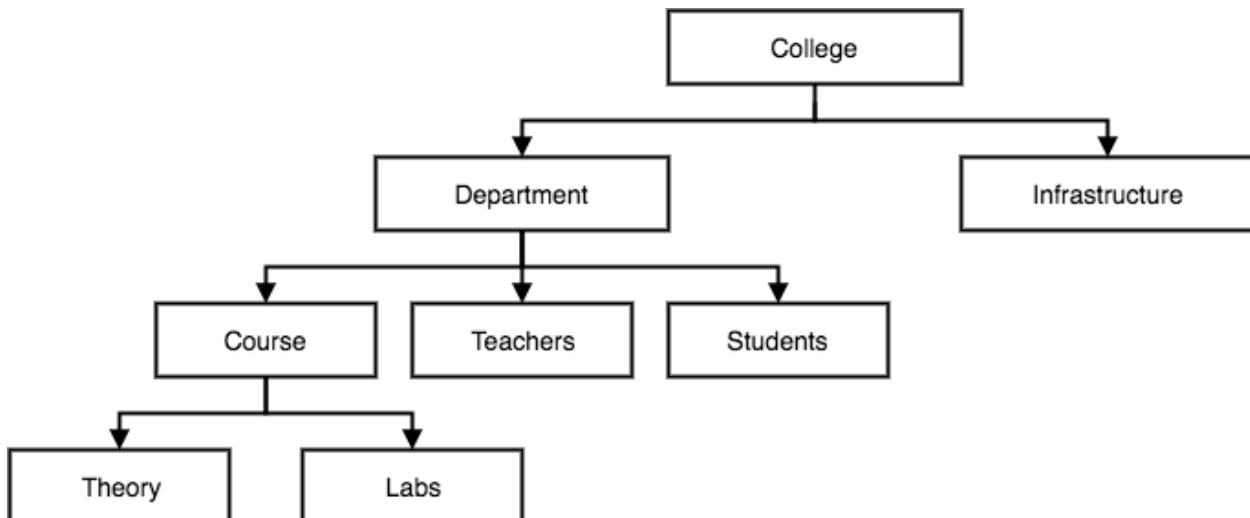


Image 38 – Hierarchical Model

Reference - <https://www.studytonight.com/dbms/images/hierarchical-dbms-model.png>

Relational Database Management System

Introduction

RDBMS stands for Relational Database Management Systems.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.

It is called Relational Data Base Management System (RDBMS) because it is based on relational model introduced by E.F. Codd.

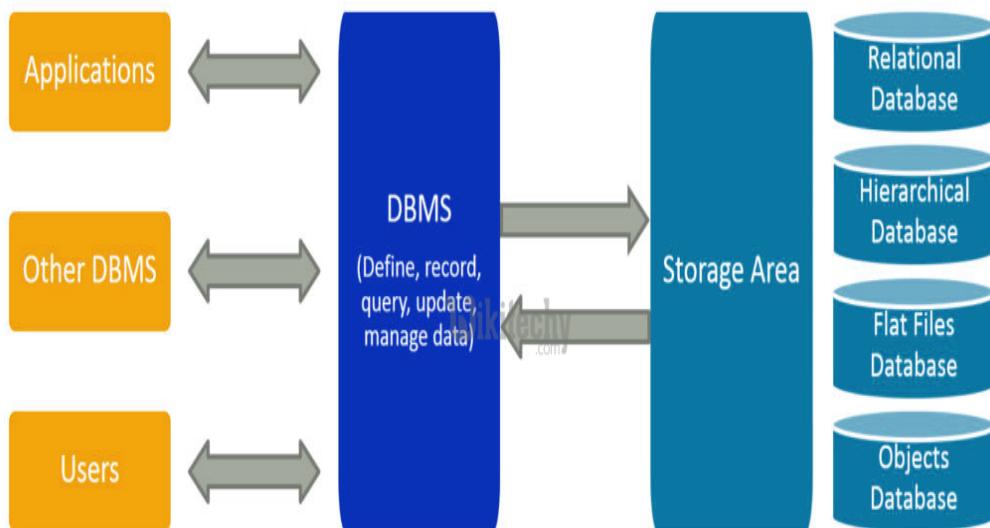


Image 39 – RDBMS Model

Reference - <https://d5ngkkf53wl41.cloudfront.net/interview-questions/dbms/what-is-dbms.png>

It is the process of making a description of the execution of the database on secondary storage, which describes the base relations, file organizations as well as indexes used to gain efficient access to the data and any associated integrity constraints and security measures.

Characteristics of physical database design

It typically illustrates data requirements for a single project or application. Sometimes even a part of an application

May be incorporated into other physical data models by means of a repository of shared entities

It typically includes 10-1000 tables; although these numbers are highly variable, depending on the scope of the data model

It has the relationships between tables that address cardinality and null ability (optionality) of the relationships

Designed and developed to be reliant on a specific version of a DBMS, storage location of data or technology

Database columns will have data types with accurate precisions and lengths assigned to them. Columns will have null ability (optional) assigned

Tables and columns will have specific definitions

Steps required to implement physical database

The steps of the physical database design methodology are as follows:

Transform the logical data model for target DBMS

- Design base relations
- Design representation of derived data
- Design general constraints
- Design file organizations and indexes
- Analyze transactions
- Choose file organizations
- Choose indexes
- Estimate disk space requirements
- Design user views

-
- Design security mechanisms
 - Consider the introduction of controlled redundancy
 - Monitor and tune the operational system

Database Engine

A database engine (or storage engine) is the underlying software component that a database management system (DBMS) uses to create, read, update and delete (CRUD) data from a database. Most database management systems include their own application programming interface (API) that allows the user to interact with their underlying engine without going through the user interface of the DBMS.

The term "database engine" is frequently used interchangeably with "database server" or "database management system". A 'database instance' refers to the processes and memory structures of the running database engine.

Database Schema

The database schema of a database is its structure described in a formal language supported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). The formal definition of a database schema is a set of formulas (sentences) called integrity constraints imposed on a database.[citation needed] These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modeled in the database.

"A database schema specifies, based on the database administrator's knowledge of possible applications, the facts that can enter the database, or those of interest to the possible end-users."

The notion of a database schema plays the same role as the notion of theory in predicate calculus. A model of this "theory" closely corresponds to a database, which can be seen at any instant of time as a mathematical object. Thus a schema can contain formulas representing integrity constraints specifically for an application and the constraints specifically for a type of database, all expressed in the same database language.

In a relational database, the schema defines the tables, fields, relationships, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views, synonyms, database links, directories, XML schemas, and other elements.

A database generally stores its schema in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure. In other words, schema is the structure of the database that defines the objects in the database.

Advantages of relational database management system

The use of an RDBMS can be beneficial to most organizations; the systematic view of raw data helps companies better understand and execute the information while enhancing the decision-making process. The use of tables to store data also improves the security of information stored in the databases. Users are able to customize access and set barriers to limit the content that is made available. This feature makes the RDBMS particularly useful to companies in which the manager decides what data is provided to employees and customers.

Furthermore, RDBMSes make it easy to add new data to the system or alter existing tables while ensuring consistency with the previously available content.

Other advantages of the RDBMS include:

Flexibility -- updating data is more efficient since the changes only need to be made in one place.

Maintenance -- database administrators can easily maintain, control and update data in the database. Backups also become easier since automation tools included in the RDBMS automate these tasks.

Data structure -- the table format used in RDBMSes is easy to understand and provides an organized and structural manner through which entries are matched by firing queries.

Disadvantages of RDBMS

Software is expensive.

Complex software refers to expensive hardware and hence increases overall cost to avail the RDBMS service.

It requires skilled human resources to implement.

Certain applications are slow in processing.

It is difficult to recover the lost data.

Applications of RDBMS

Sector	Use of DBMS
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges and grades.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	Use for storing customer, product & sales information.

Manufacturing

It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.

HR Management

For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

Structure of DBMS

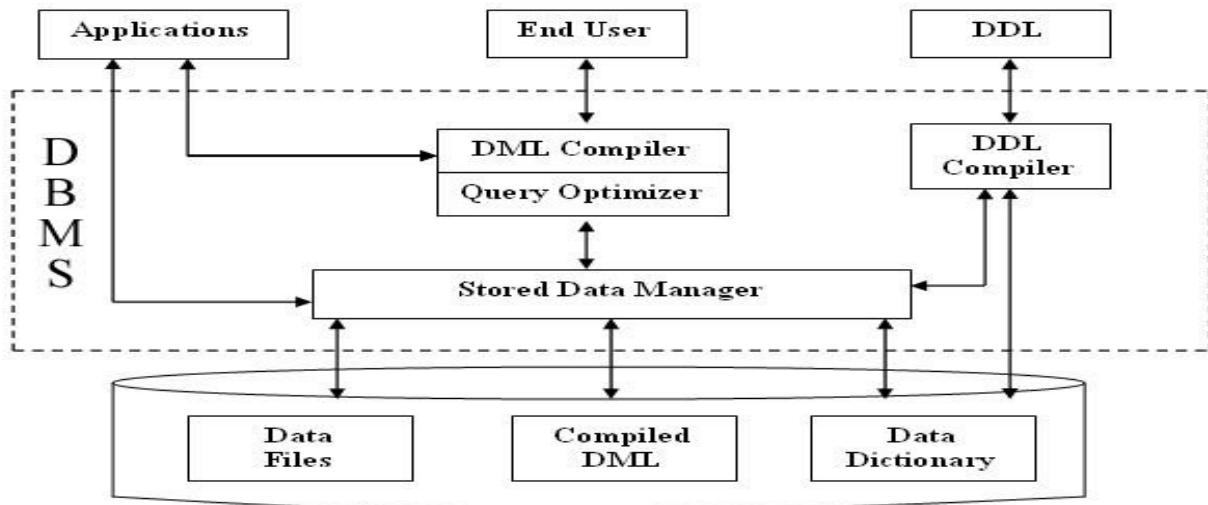


Image 40 – Structure of DBMS

Reference -

http://1.bp.blogspot.com/-GmlmEbglhWk/Vc12mfuuTYI/AAAAAAAABak/jYO7eiIdprw/s1600/Structure_of_DBMS_DBMSbasics.blogspot.com.jpg

At very high level, a database is considered as shown in below diagram. Let us see them in detail below.

Applications: – It can be considered as a user friendly web page where the user enters the requests. Here he simply enters the details that he needs and presses buttons to get the data.

End User: – They are the real users of the database. They can be developers, designers, administrator or the actual users of the database.

DDL: – Data Definition Language (DDL) is a query fired to create database, schema, tables, mappings etc in the database. These are the commands used to create the objects like tables, indexes in the database for the first time. In other words, they create structure of the database.

DDL Compiler: – This part of database is responsible for processing the DDL commands. That means these compiler actually breaks down the command into machine understandable codes. It is also responsible for storing the metadata information like table name, space used by it, number of columns in it, mapping information etc.

DML Compiler: – When the user inserts, deletes, updates or retrieves the record from the database, he will be sending request which he understands by pressing some buttons. But for the database to work/understand the request, it should be broken down to object code. This is done by this compiler. One can imagine this as when a person is asked some question, how this is broken down into waves to reach the brain!

Query Optimizer: – When user fires some request, he is least bothered how it will be fired on the database. He is not all aware of database or its way of performance. But whatever be the request, it should be efficient enough to fetch, insert, update or delete the data from the database. The query optimizer decides the best way to execute the user request which is received from the DML compiler. It is similar to selecting the best nerve to carry the waves to brain!

Stored Data Manager: – This is also known as Database Control System. It is one the main central system of the database. It is responsible for various tasks

It converts the requests received from query optimizer to machine understandable form. It makes actual request inside the database. It is like fetching the exact part of the brain to answer.

It helps to maintain consistency and integrity by applying the constraints. That means, it does not allow inserting / updating / deleting any data if it has child entry. Similarly it does not allow entering any duplicate value into database tables.

It controls concurrent access. If there is multiple users accessing the database at the same time, it makes sure, all of them see correct data. It guarantees that there is no data loss or data mismatch happens between the transactions of multiple users.

It helps to backup the database and recover data whenever required. Since it is a huge database and when there is any unexpected exploit of transaction, and reverting the changes are not easy. It maintains the backup of all data, so that it can be recovered.

Data Files: – It has the real data stored in it. It can be stored as magnetic tapes, magnetic disks or optical disks.

Compiled DML: – Some of the processed DML statements (insert, update, delete) are stored in it so that if there is similar requests, it will be re-used.

Data Dictionary: – It contains all the information about the database. As the name suggests, it is the dictionary of all the data items. It contains description of all the tables, view, materialized views, constraints, indexes, triggers etc.

DBMS VS File System

DBMS	File System
In DBMS, the user is not required to write the procedures.	In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.

DBMS takes care of Concurrent access of data using some form of locking.	In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information.
--	---

DBMS VS RDBMS

No.	DBMS	RDBMS
1)	DBMS applications store data as file.	RDBMS applications store data in a tabular form.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	Normalization is not present in DBMS.	Normalization is present in RDBMS.
4)	DBMS does not apply any security with regards to data manipulation.	RDBMS defines the integrity constraint for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
5)	DBMS uses file system to store data, so there will be no relation between the tables.	in RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
6)	DBMS does not support distributed database.	RDBMS supports distributed database.
7)	DBMS is meant to be for small organization and deal with small data. it supports single user.	RDBMS is designed to handle large amount of data. it supports multiple users.
8)	Examples of DBMS are file systems, xml etc.	Example of RDBMS are mysql, postgresql, sql server, oracle etc.

1-Tier Architecture

- o In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- o Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.

- o The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- o The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- o The user interfaces and application programs are run on the client-side.
- o The server side is responsible to provide the functionalities like: query processing and transaction management.
- o To communicate with the DBMS, client-side application establishes a connection with the server side.

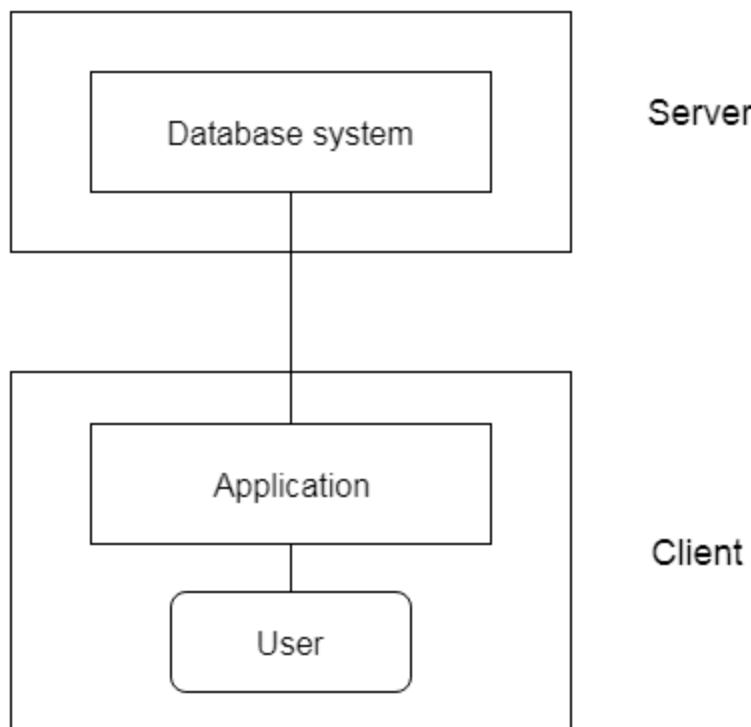


Image 41 – 2 tier Architecture

Reference - <https://static.javatpoint.com/dbms/images/dbms-2-tier-architecture.png>

3-Tier Architecture

- o The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- o The application on the client-end interacts with an application server which further communicates with the database system.
- o End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- o The 3-Tier architecture is used in case of large web application.

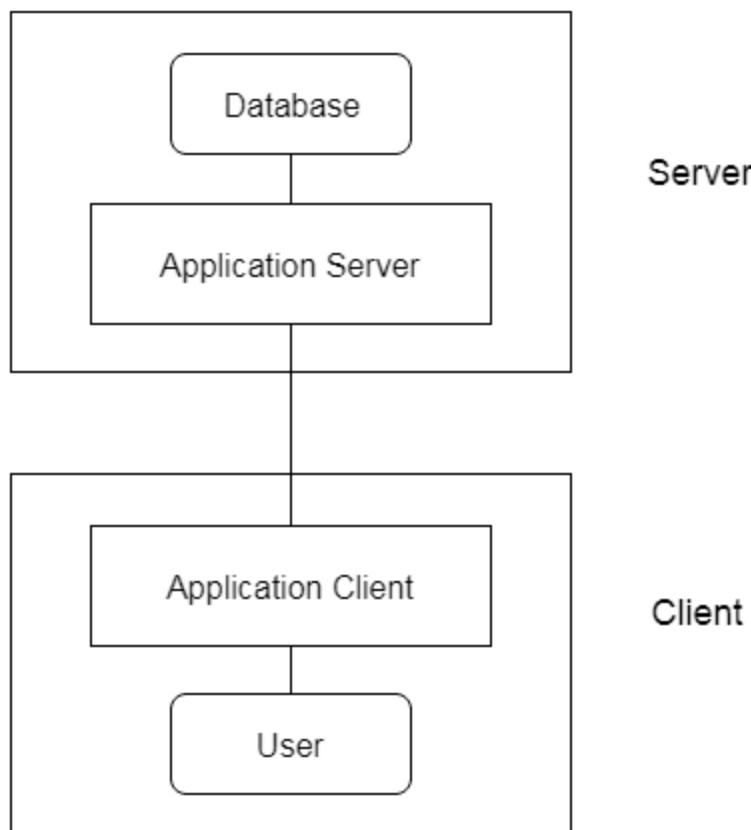


Image 42 – 3 tier Architecture

Reference - <https://static.javatpoint.com/dbms/images/dbms-3-tier-architecture.png>

Three Schema Architecture

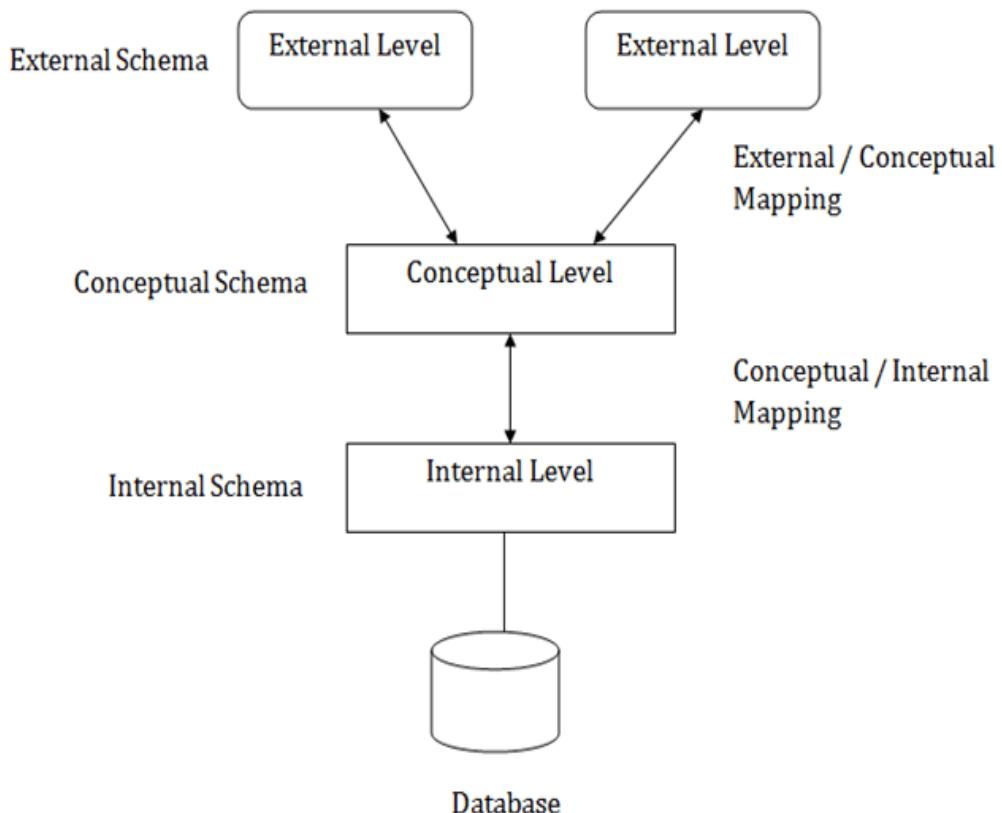


Image 43 – 3 Schema Architecture

Reference - <https://static.javatpoint.com/dbms/images/dbms-three-schema-architecture.png>

This framework is used to describe the structure of a specific database system.

The three schema architecture is also used to separate the user applications and physical database.

The three schema architecture contains three-levels. It breaks the database down into three different categories

Internal Level

The internal level has an internal schema which describes the physical storage structure of the database.

The internal schema is also known as a physical schema.

It uses the physical data model.

It is used to define that how the data will be stored in a block.

The physical level is used to describe complex low-level data structures in detail.

Conceptual Level

The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.

The conceptual schema describes the structure of the whole database.

The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.

Programmers and database administrators work at this level.

External Level

At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.

An external schema is also known as view schema.

The view schema describes the end user interaction with database systems.

Relational Algebra

Relational Algebra is procedural query language, which takes Relation as input and generate relation as output.

Relational algebra operations are performed recursively on a relation.

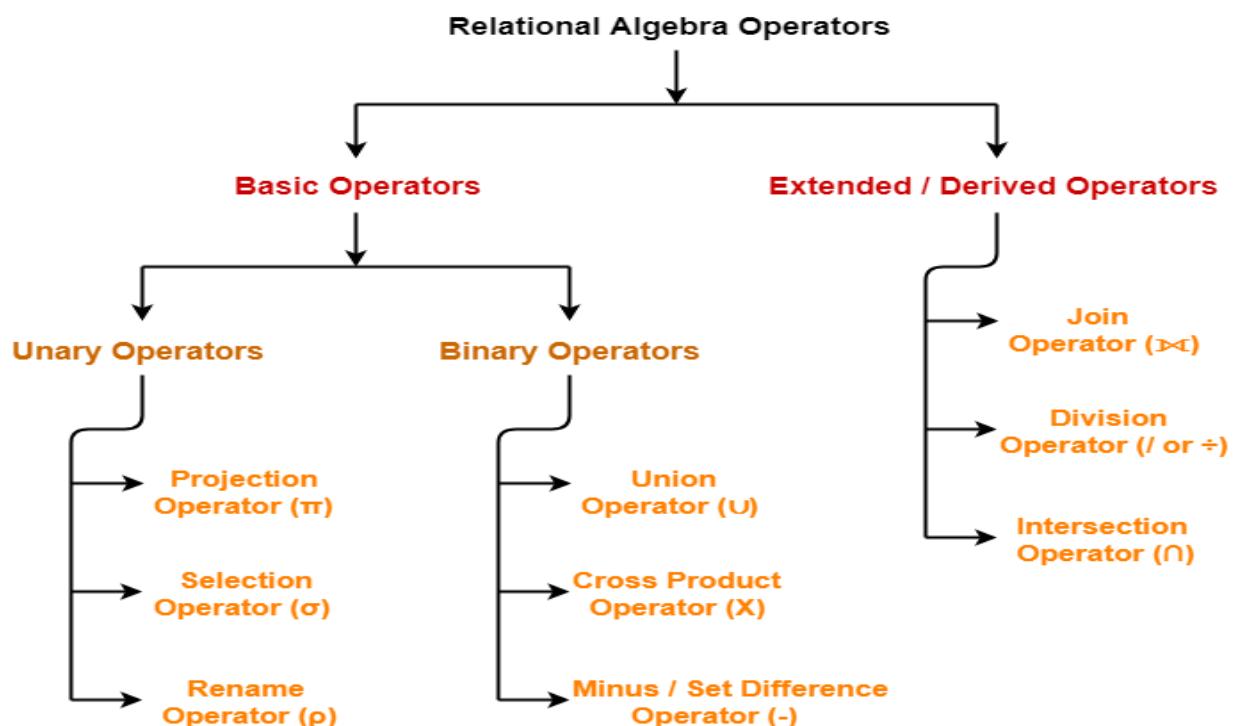


Image 44 – Relational Algebra

Reference -

https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcRt8Z79xBi_fs1wYROuo33pDaGmXFbTyRBWowkH_UGkPdrf2Ft&usqp=CAU

Select Operation

The select operation selects tuples that satisfy a given predicate.

It is denoted by sigma (σ).

Notation: $\sigma p(r)$

Where:

σ is used for selection prediction

r is used for relation

p is used as a propositional logic formula which may use connectors like: AND OR and NOT.
These relational can use as relational operators like $=, \neq, \geq, <, >, \leq$.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

$\sigma \text{BRANCH_NAME} = \text{"perryride"} (\text{LOAN})$

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

Project Operation

This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.

It is denoted by Π .

Notation: $\Pi A_1, A_2, A_n (r)$

Where

A_1, A_2, A_3 is used as an attribute name of relation r .

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

$\Pi \text{NAME, CITY } (\text{CUSTOMER})$

Output:

NAME	CITY
Jones	Harrison
Smith	Rye

Hays Harrison

Union Operation

Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.

It eliminates the duplicate tuples. It is denoted by \cup .

Notation: $R \cup S$

A union operation must hold the following condition:

R and S must have the attribute of the same number.

Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
---------------	------------

Johnson	A-101
---------	-------

Smith	A-121
-------	-------

Mayes	A-321
-------	-------

Turner	A-176
--------	-------

Johnson	A-273
---------	-------

Jones	A-472
-------	-------

Lindsay	A-284
---------	-------

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
---------------	---------

Jones	L-17
-------	------

Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:
$$\Pi \text{ CUSTOMER_NAME (BORROW)} \cup \Pi \text{ CUSTOMER_NAME (DEPOSITOR)}$$
Output:

CUSTOMER_NAME

Johnson

Smith

Hayes

Turner

Jones

Lindsay

Jackson

Curry

Williams

Mayes

Set Intersection

Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.

It is denoted by intersection \cap .

Notation: $R \cap S$

Example: Using the above DEPOSITOR table and BORROW table

Input:

$\Pi \text{ CUSTOMER_NAME (BORROW)} \cap \Pi \text{ CUSTOMER_NAME (DEPOSITOR)}$

Output:

CUSTOMER_NAME

Smith

Jones

Set Difference

Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.

It is denoted by intersection minus (-).

Notation: $R - S$

Example: Using the above DEPOSITOR table and BORROW table

Input:

$\Pi \text{ CUSTOMER_NAME (BORROW)} - \Pi \text{ CUSTOMER_NAME (DEPOSITOR)}$

Output:

CUSTOMER_NAME

Jackson

Hayes

Willians

Curry

Cartesian Product

The Cartesian product is used to combine each row in one table with each row in the other table.
It is also known as a cross product.

It is denoted by X.

Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

Rename Operation

The rename operation is used to rename the output relation. It is denoted by rho (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

$\rho(\text{STUDENT1}, \text{STUDENT})$

Join Operation

Join operation is essentially a Cartesian product followed by a selection criterion.

Join operation denoted by \bowtie .

JOIN operation also allows joining variously related tuples from different relations.

Inner Join

Natural join between two or more relations will result in all the combination of tuples where they have equal values for the common attribute

Theta Join

The general case of JOIN operation is called a Theta join. It is denoted by symbol θ

For example:

$A \bowtie A.\text{column } 2 > B.\text{column } 2 (B)$

EQUI Join

When a theta join uses only equivalence condition, it becomes a equi join.

For example:

$A \bowtie A.\text{column } 2 = B.\text{column } 2 (B)$

Example

$A \bowtie \theta B$ Theta join can use any conditions in the selection criteria.

Left Outer Join

In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.

Example : $A \bowtie B$

Example: Select students whose ROLL_NO is greater than EMP_NO of employees and details of other students as well

STUDENT $\bowtie_{\text{STUDENT.ROLL_NO} > \text{EMPLOYEE.EMP_NO}}$ EMPLOYEE

Right Outer Join

In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.

Example: Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well

STUDENT $\bowtie_{\text{STUDENT.ROLL_NO} > \text{EMPLOYEE.EMP_NO}}$ EMPLOYEE

Full Outer Join

In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.

Example: Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well and other Students as well

STUDENT $\bowtie_{\text{STUDENT.ROLL_NO} > \text{EMPLOYEE.EMP_NO}}$ EMPLOYEE

Division Operator

Division operator $A \div B$ can be applied if and only if:

Attributes of B is proper subset of Attributes of A.

The relation returned by division operator will have attributes = (All attributes of A – All Attributes of B)

The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

Consider the relation STUDENT_SPORTS and ALL_SPORTS given in Table 2 and Table 3 above.

To apply division operator as

STUDENT_SPORTS ÷ ALL_SPORTS

Relational Calculus

Introduction

Relational calculus is a non-procedural query language, and instead of algebra, it uses mathematical predicate calculus.

Tuple relational calculus which was originally proposed by Codd in the year 1972 and

Domain relational calculus which was proposed by Lacroix and Pirotte in the year 1977

Relational calculus is a non-procedural query language. In the non-procedural query language, the user is concerned with the details of how to obtain the end results.

The relational calculus tells what to do but never explains how to do.

Types of Relational Calculus

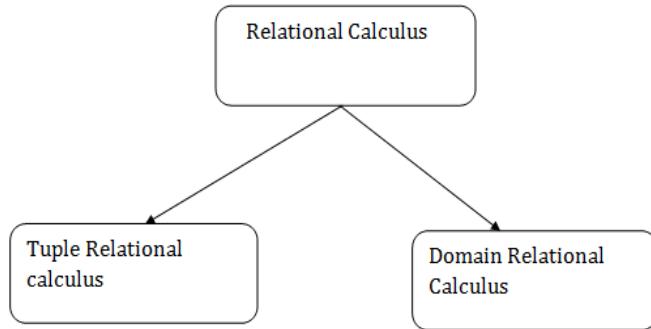


Image 45 – Relational Calculus

Reference - <https://static.javatpoint.com/dbms/images/dbms-relational-calculus.png>

Tuple Relational Calculus

In tuple relational calculus, we work on filtering tuples based on the given condition.

Syntax: { T | Condition }

In domain relational calculus, filtering is done based on the domain of the attributes and not based on the tuple values.

Syntax: { c1, c2, c3, ..., cn | F(c1, c2, c3, ..., cn)}

The tuple relational calculus is specified to select the tuples in a relation.

In TRC, filtering variable uses the tuples of a relation.

The result of the relation can have one or more tuples.

Notation:

{T | P (T)} or {T | Condition (T)}

Where

T is the resulting tuples

P(T) is the condition used to fetch T.

For example:

{ T.name | Author(T) AND T.article = 'database' }

OUTPUT: This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

The symbols used for logical operators are: **Λ for AND, ∨ for OR and ∉ for NOT.**

Variables can be constrained by quantified statements to tuples in a single relation: –

Existential Quantifier. $\exists T \in R(\text{Cond})$ will succeed if Cond succeeds for at least one tuple in T.

–

Universal Quantifier. $\forall T \in R(\text{Cond})$ will succeed if Cond succeeds for all tuples in T

Examples

$\{t \mid P(t)\}$ or $\{t \mid \text{condition}(t)\}$ —

this is also known as expression of relational calculus

Where t is the resulting tuples, P(t) is the condition used to fetch t.

$\{t \mid \text{EMPLOYEE}(t) \text{ and } t.\text{SALARY} > 10000\}$ —

implies that it selects the tuples from EMPLOYEE relation such that resulting employee tuples will have salary greater than 10000. It is example of selecting a range of values.

$\{t \mid \text{EMPLOYEE}(t) \text{ AND } t.\text{DEPT_ID} = 10\}$ —

this select all the tuples of employee name who work for Department 10.

$\{T.\text{name} \mid F \text{ ACULT Y}(T) \text{ AND } T.\text{DeptId} = 0 \text{ CS0}\}$

can be read as: “Find all tuples T field such that T is a tuple in the FACULTY relation and the value of DeptId field is ‘CS’. Return a tuple with a single field name which is equivalent to the name field of one such T tuple”.

$\{R \mid \exists T \in F \text{ ACULT Y}(T.\text{DeptId} = 0 \text{ CS0} \text{ AND } R.\text{name} = T.\text{name})\}$

can be read as: “Find all tuples R such that there exists a tuple T in FACULTY with the DeptId field value ‘CS’, and the value of the name field of R is equivalent to the name field of this tuple T.”

Example 1 Find the loan number, branch, amount of loans of greater than or equal to 10000 amount.

$\{t \mid t \in \text{loan} \wedge t[\text{amount}] \geq 10000\}$

Example 2 Find the loan number for each loan of an amount greater or equal to 10000.

$\{t \mid \exists s \in \text{loan}(t[\text{loan number}] = s[\text{loan number}] \wedge s[\text{amount}] \geq 10000)\}$

Example 3 Find the names of all customers who have a loan and an account at the bank.

$$\{t \mid \exists s \in \text{borrower}(t[\text{customer-name}] = s[\text{customer-name}]) \wedge \exists u \in \text{depositor}(t[\text{customer-name}] = u[\text{customer-name}])\}$$

Domain Relational Calculus

In domain relational calculus, filtering variable uses the domain of attributes.

Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives \wedge (and), \vee (or) and \neg (not).

It uses Existential (\exists) and Universal Quantifiers (\forall) to bind the variable.

The domain variables those will be in resulting relation must appear before | within \prec and \succ and all the domain variables must appear in which order they are in original relation or table..

Notation:

$$\{ a_1, a_2, a_3, \dots, a_n \mid P(a_1, a_2, a_3, \dots, a_n) \}$$

Where

a_1, a_2 are attributes

P stands for formula built by inner attributes

Examples

Example 1

Find the loan number, branch, amount of loans of greater than or equal to 100 amount.

$$\{ \langle l, b, a \rangle \mid \langle l, b, a \rangle \in \text{loan} \wedge (a \geq 100) \}$$

Example 2

Find the loan number for each loan of an amount greater or equal to 150.

$$\{ \langle l \rangle \mid \exists b, a (\langle l, b, a \rangle \in \text{loan} \wedge (a \geq 150)) \}$$

Example 3

Find the names of all customers having a loan at the “Main” branch and find the loan amount .

{<c, a> | $\exists l (<c, l> \in \text{borrower} \wedge \exists b (<l, b, a> \in \text{loan} \wedge (b = \text{“Main”})))}$ }

Example 4

{< name, age > | $\in \text{Student} \wedge \text{age} < 21$ }

Again, the above query will return the names and ages of the students in the table Student who are not greater than 21 years old

Example 5

{< Fname, Emp_ID > | $\in \text{Employee} \wedge \text{Salary} > 10000$ }

The result here will be returning the Fname and Emp_ID values for all the rows in the employee table where salary is greater than 10000.

RDBMS Technologies

Oracle Database Technology Features

- Data Concurrency and Consistency
- Manageability
- Backup & Recovery
- Business Intelligence
- High Availability
- Very Large Databases
- Content Management

MySQL Database Technology Features

- Data Concurrency and Consistency
- Scalability and Limit
- Backup & Recovery
- Connectivity
- High Availability
- Clients and Tools
- Workbench tool

MongoDB Database Technology Features

- Indexing
- Replication
- Backup & Recovery
- Load Balancing
- Map Reducing and Aggregation
- Stores files of any size easily without complicating your stack.
- Cloud Support

Microsoft SQL Server Database Technology Features

- Highest performing data warehouses
- End to End Mobile BI
- Backup & Recovery
- Load Balancing
- Built-in Analytics
- Mission Critical Availability
- Cloud Support

Relational Data Structure/Relational Model

The **relational model (RM)** for database management is an approach to managing data using a structure and language consistent with first-order-predicate logic, first described in 1969 by English computer scientist Edgar F Codd, where all data is represented in terms of tuples, grouped into relations. A database organized in terms of the relational model is a relational database.

Note: First-order logic—also known as predicate logic, is a collection of formal systems used in mathematics, philosophy, linguistics, and computer science.

What is a Relational Model?

RELATIONAL MODEL (RM) represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

- The table name and column names are helpful to interpret the meaning of values in each row.
- The data are represented as a set of relations.
- In the relational model, data is stored as tables.
- However, the physical storage of the data is independent of the way the data are logically organized.

Some popular Relational Database management systems are:

- DB2 and Informix Dynamic Server - IBM
- Oracle and RDB – Oracle
- SQL Server and Access - Microsoft

Relational Model Concepts

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation.
e.g., Student_Rollno, NAME,etc.
2. **Tables:** In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties: rows and columns. Rows represent records and columns represent attributes.
3. **Tuple:** It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance:** Relation instance is a finite set of tuples in the RDBMS system.
Relation instances never have duplicate tuples.
9. **Relation key:** Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain:** Every attribute has some predefined value and scope which is known as attribute domain

Table also called Relation

Primary Key Domain
Ex: NOT NULL

© guru99.com

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Column OR Attributes
Total # of column is Degree

Tuple OR Row
Total # of rows is Cardinality

Image 1- Shows some of the **Relational Model Concepts**

Reference- <https://www.guru99.com/relational-data-model-database.html>

Relational Integrity constraints

Relational Integrity constraints are referred to conditions which must be present for a valid relation. These integrity constraints are derived from the rules in the mini-world that the database represents.

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of integrity constraints

1. Domain constraints
2. Entity Integrity constraints
3. Referential integrity constraints
4. Key constraints

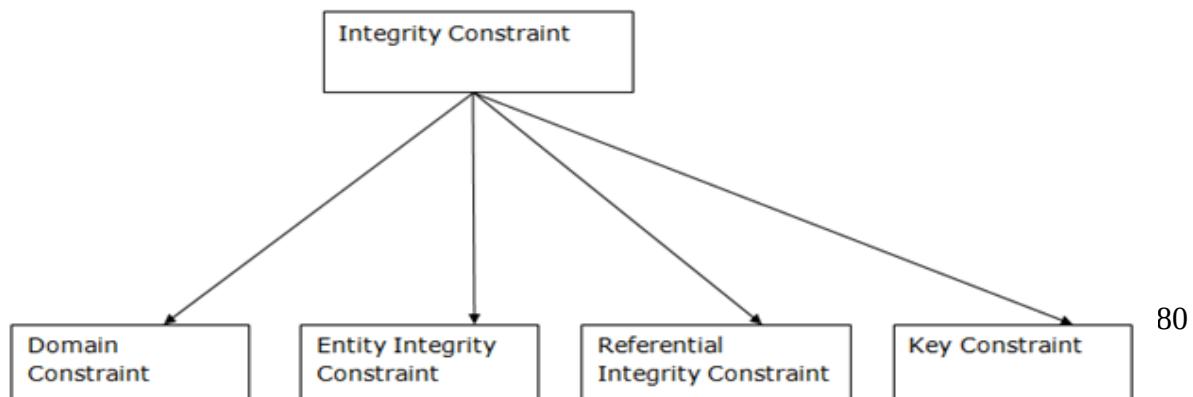


Image 2- Types of integrity constraints
Reference- <https://www.javatpoint.com/dbms-integrity-constraints>

1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMESTER	AGE
1000	Tom	1 st	17
1001	Johnson	2 nd	24
1002	Leonardo	5 th	21
1003	Kate	3 rd	19
1004	Morgan	8 th	A

Not allowed. Because AGE is an integer attribute

Image 3- Example of Domain Constraint
Reference- <https://www.javatpoint.com/dbms-integrity-constraints>

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:**EMPLOYEE**

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

Image 4- Example of Entity integrity constraint
Reference- <https://www.javatpoint.com/dbms-integrity-constraints>

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

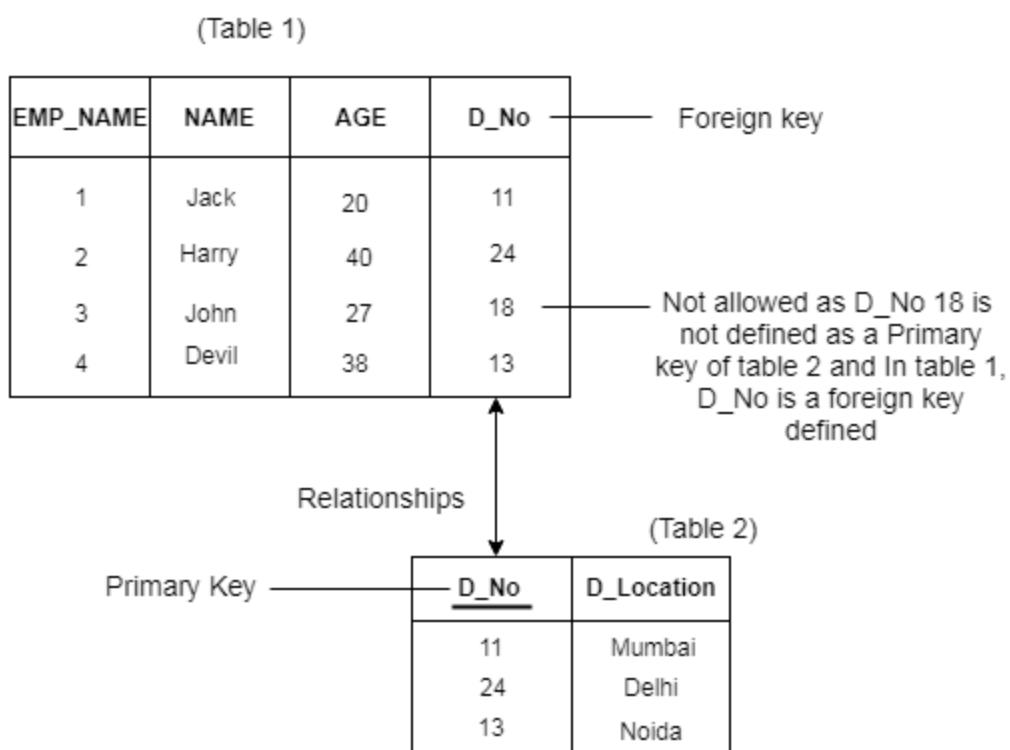
Example:

Image 5- referential integrity constraint

Reference- <https://www.javatpoint.com/dbms-integrity-constraints>

4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:

ID	NAME	SEMESTER	AGE
1000	Tom	1 st	17
1001	Johnson	2 nd	24
1002	Leonardo	5 th	21
1003	Kate	3 rd	19
1002	Morgan	8 th	22

Not allowed. Because all row must be unique

Image 6- Example of Key Constraint
Reference- <https://www.javatpoint.com/dbms-integrity-constraints>

Operations in the Relational Model

Four basic update operations performed on the relational database model are

Insert, update, delete and select.

- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples.

- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert Operation:

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Image 7- Insert
Reference- <https://www.guru99.com/relational-data-model-dbms.html>

Update Operation

You can see that in the below-given relation table CustomerName= 'Apple' is updated from Inactive to Active.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

Image 8- Update
Reference- <https://www.guru99.com/relational-data-model-dbms.html>

Delete Operation

To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.

A diagram illustrating a DELETE operation. On the left, there is a table with four rows, each representing a customer tuple. The columns are CustomerID, CustomerName, and Status. The rows are: Row 1 (CustomerID 1, Google, Active), Row 2 (CustomerID 2, Amazon, Active), Row 3 (CustomerID 3, Apple, Active), and Row 4 (CustomerID 4, Alibaba, Active). A large red arrow points from the third row (Apple) towards the right. In the center of this arrow is the word "DELETE". To the right of the arrow is another table, which is identical to the first one except that the third row (Apple) has been removed. This second table also has a red arrow pointing towards it from the left, with the word "DELETE" in its center.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

Image 9- Delete

Reference- <https://www.guru99.com/relational-data-model-dbms.html>

In the above-given example, CustomerName= "Apple" is deleted from the table.

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same database.

Select Operation

A diagram illustrating a SELECT operation. On the left, there is a table with four rows, each representing a customer tuple. The columns are CustomerID, CustomerName, and Status. The rows are: Row 1 (CustomerID 1, Google, Active), Row 2 (CustomerID 2, Amazon, Active), Row 3 (CustomerID 3, Apple, Active), and Row 4 (CustomerID 4, Alibaba, Active). A large red arrow points from the second row (Amazon) towards the right. In the center of this arrow is the word "SELECT". To the right of the arrow is another table, which is identical to the first one except that only the second row (Amazon) is present. This second table also has a red arrow pointing towards it from the left, with the word "SELECT" in its center.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
2	Amazon	Active

Image 10- Select

Reference- <https://www.guru99.com/relational-data-model-dbms.html>

In the above-given example, CustomerName="Amazon" is selected.

Advantages of using the Relational model

- **Simplicity:** A relational data model is simpler than the hierarchical and network model.

-
- **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
 - **Easy to use:** The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
 - **Query capability:** It makes possible for a high-level query language like SQL to avoid complex database navigation.
 - **Data independence:** The structure of a database can be changed without having to change any application.
 - **Scalable:** Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages of using Relational model

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

Key and Relational Data Manipulation

Keys

What are Keys?

A DBMS key is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table). They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.

Example:

Student ID	FirstName	LastName
1	Qanith	Khan
2	Rajesh	Singh
3	john	Hale

In the above-given example, Student ID is a primary key because it uniquely identifies an Student record. In this table, no other Student can have the same Student ID.

Why do we need a Key?

Here are reasons for using Keys in the DBMS system.

-
- Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys ensure that you can uniquely identify a table record despite these challenges.
 - Allows you to establish a relationship between and identify the relation between tables
 - Help you to enforce identity and integrity in the relationship.

Various Keys

Each keys have their different functionality:

- Super Key
- Primary Key
- Candidate Key
- Alternate Key
- Foreign Key
- Compound Key
- Composite Key
- Surrogate Key

Super key:

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn

199937890

AB07

James

In the above-given example, EmpSSN and EmpNum name are superkeys.

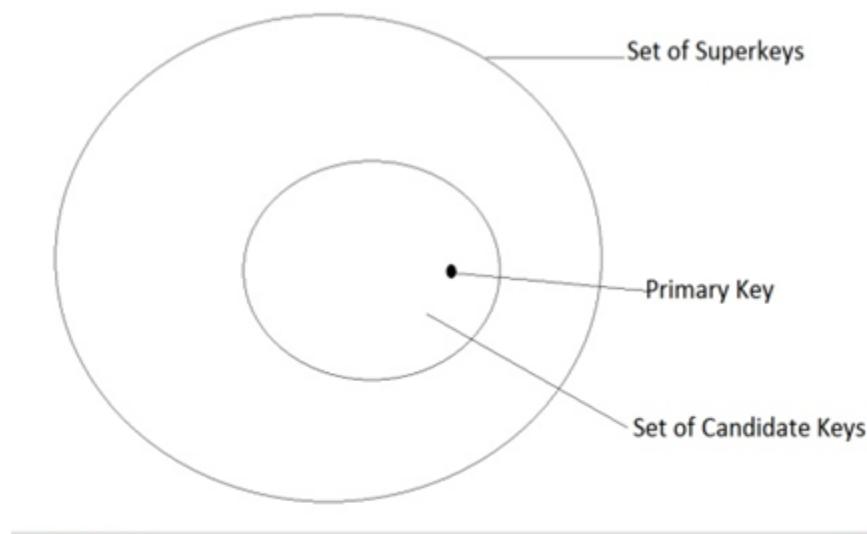


Image 11- Relation b/w keys

Reference-

https://www.slideshare.net/TechtudNetwork/relation-between-super-key-candidate-key-and-primary-key?qid=c7bb9eac-8135-480c-92ee-890bf4c744aa&v=&b=&from_search=1

Primary Key

PRIMARY KEY is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Example:

In the following example, `StudID` is a Primary Key.

StudID	Roll No	First Name	Last Name	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

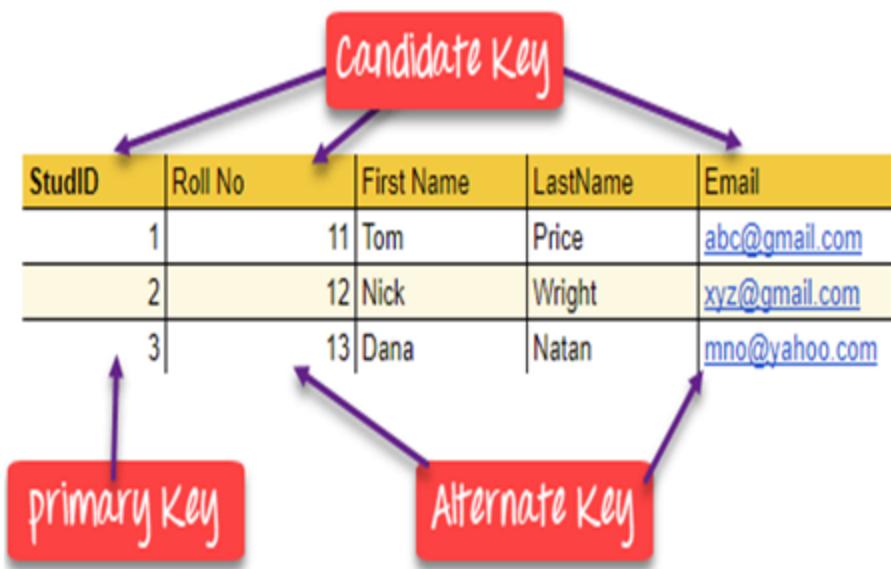


Image 12- Primary Key

Reference- <https://www.guru99.com/dbms-keys.html>

Alternate key

ALTERNATE KEYS is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate Key.

Example:

In this table, StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

StudID	Roll No	First Name	Last Name	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

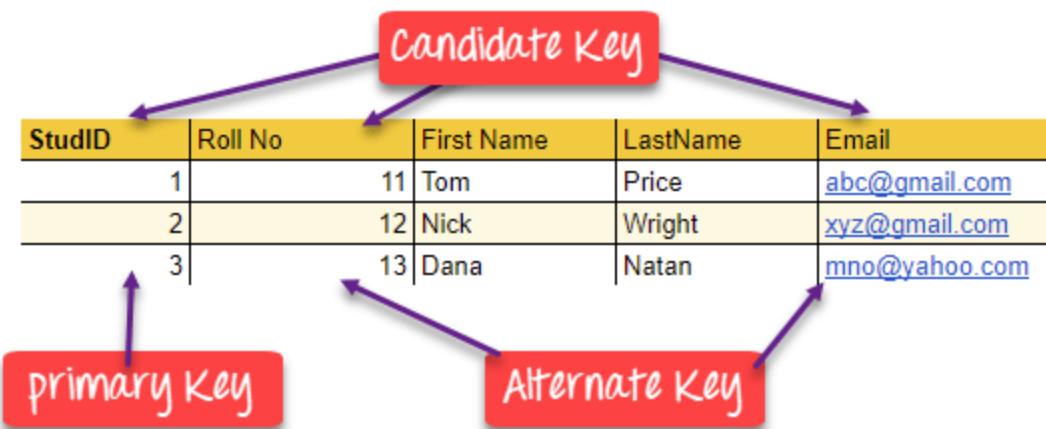


Image 13- Alternate Key

Reference-<https://www.guru99.com/dbms-keys.html>

Candidate Key

CANDIDATE KEY is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.

Properties of Candidate key:

- It must contain unique values
- Candidate key may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

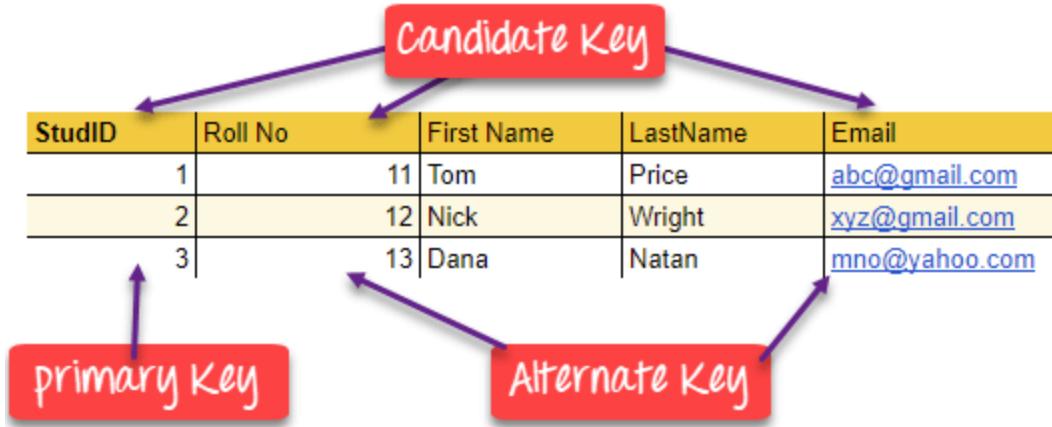


Image 14- Candidate Key

Reference-<https://www.guru99.com/dbms-keys.html>

Foreign key

FOREIGN KEY is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.

Example:

DeptCode	DeptName
001	Science
002	English

005

Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph

B009

001

Mike

Brunton

This concept is also known as Referential Integrity.

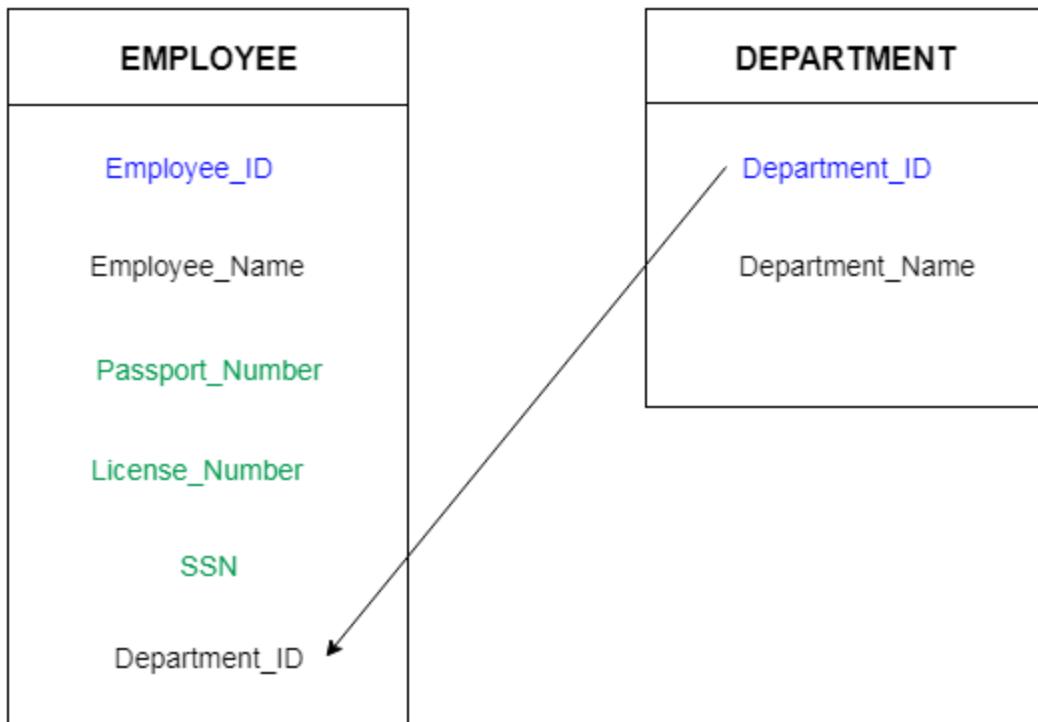


Image 15- Foreign key
Reference- <https://www.javatpoint.com/dbms-keys>

Compound key

COMPOUND KEY has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of compound key is to uniquely identify each record in the table.

Example:

OrderNo	ProductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

In this example, OrderNo and ProductID can't be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.

Composite key

COMPOSITE KEY is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table.

The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

Surrogate Key

An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key is usually an integer.

Fname	Lastname	Start Time	End Time
Anne	Smith	09:00	18:00
Jack	Francis	08:00	17:00
Anna	McLean	11:00	20:00
Shown	Willam	14:00	23:00

Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

Surrogate keys are allowed when

- No property has the parameter of the primary key.
- In the table when the primary key is too big or complicated.

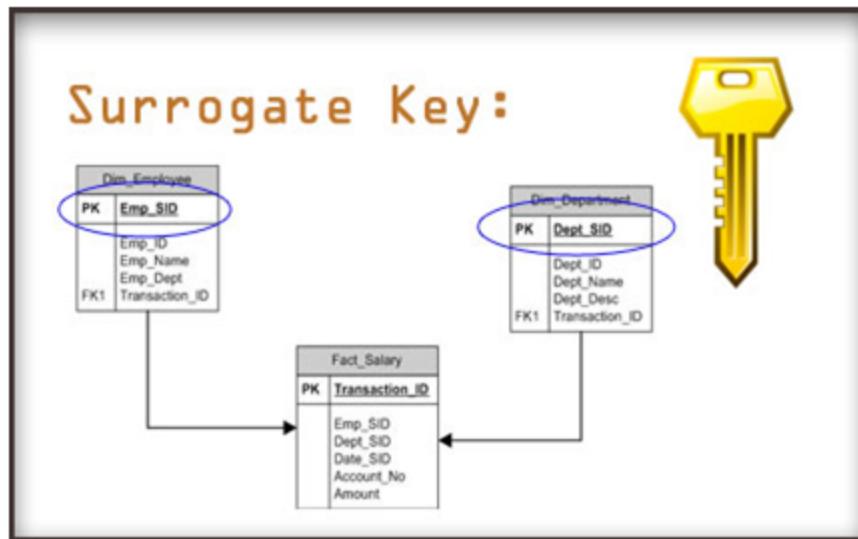


Image 16- Surrogate Key

Reference-

https://www.google.com/search?q=Surrogate+Key+&tbo=isch&ved=2ahUKEwiX99ycj-roAhUI5DgGHffQBnQQ2-cCegQIABAA&oq=Surrogate+Key+&gs_lcp=CgNpbWcQAzIECAAQzICCAAvAggAMgIIADICCAyAggAMgIIADICCAyAggAMgIIAFD-nBpY_pwaYJuhGmgAcA B4AIABswGIAbMBkgEDMC4xmAEAoAEFqgELZ3dzLXdpei1pbWc&scrlt=img&ei=89OWXtexAojI4-EP96GboAc&bih=657&biw=1366# imgrc=JVjJUIE4gOGs0M

Let's see the difference between keys

Primary Key and Foreign Key:

S.NO.	PRIMARY KEY	FOREIGN KEY
1	A primary key is used to ensure data in the	A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables.

specific column is
unique.

-
- | | |
|--|---|
| <p>2 It uniquely identifies a record in the relational database table.</p> | <p>It refers to the field in a table which is the primary key of another table.</p> |
| <p>3 Only one primary key is allowed in a table.</p> | <p>Whereas more than one foreign key are allowed in a table.</p> |
| <p>4 It is a combination of UNIQUE and Not Null constraints.</p> | <p>It can contain duplicate values and a table in a relational database.</p> |
| <p>5 It does not allow NULL values.</p> | <p>It can also contain NULL values.</p> |
-

-
- Its value cannot be
- 6 deleted from the Its value can be deleted from the child table.
 parent table.

-
- It constraint can be
- 7 implicitly defined on It constraint cannot be defined on the local or global
 the temporary tables. temporary tables.

Primary and Candidate Key:

S.NO	PRIMARY KEY	CANDIDATE KEY
------	-------------	---------------

-
- Primary key is a minimal super key. So
1. there is one and only one primary key in While in a relation there can be more
 a relation. than one candidate key.

-
- Any attribute of Primary key can not
2. contain NULL value. While in Candidate key any attribute can
 contain NULL value.

3.	Primary key can be optional to specify any relation.	But without candidate key there can't be specified any relation.
4.	Primary key specifies the important attribute for the relation.	Candidate specifies the key which can qualify for primary key.
5.	Its confirmed that a primary key is a candidate key.	But Its confirmed that a candidate key can be a primary key.

Super Key and Candidate Key

S.NO	SUPER KEY	CANDIDATE KEY
1.	Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.	Candidate Key is a proper subset of a super key.

-
- | | | |
|----|--|---|
| 2. | All super keys can't be candidate keys. | But all candidate keys are super keys. |
| 3. | Various super keys together makes the criteria to select the candidate keys. | Various candidate keys together makes the criteria to select the primary keys. |
| 4. | In a relation, number of super keys are more than number of candidate keys. | While in a relation, number of candidate keys are less than number of super keys. |
| 5. | Super key's attributes can contain NULL values. | Candidate key's attributes can also contain NULL values. |
-

Primary key and Unique key

PARAMETER	PRIMARY KEY	UNIQUE KEY
------------------	--------------------	-------------------

Basic	Used to serve as a unique identifier for each row in a table.	Uniquely determines a row which isn't primary key.
NULL value acceptance	Cannot accept NULL values.	Can accept one NULL value.
Number of keys that can be defined in the table	Only one primary key	More than one unique key
Index	Creates clustered index	Creates non-clustered index

Data Manipulation Language (DML)

Introduction to DML

- DML stands for **Data Manipulation Language**.
- It is a language used for selecting, inserting, deleting and updating data in a database.
- It is used to retrieve and manipulate data in a relational database.

DDL commands are as follows:

1. SELECT
2. INSERT
3. UPDATE
4. DELETE

DML performs read-only queries of data.

1. SELECT COMMAND

- **SELECT command** is used to retrieve data from the database.
- This command allows database users to retrieve the specific information they desire from an operational database.
- It returns a result set of records from one or more tables.

SELECT Command has many optional clauses are as stated below:

Clause	Description
WHERE	It specifies which rows to retrieve.

GROUP BY	It is used to arrange the data into groups.
HAVING	It selects among the groups defined by the GROUP BY clause.
ORDER BY	It specifies an order in which to return the rows.
AS	It provides an alias which can be used to temporarily rename tables or columns.

Syntax:

```
SELECT * FROM <table_name>;
```

Example: Select Command

```
SELECT * FROM employee;
```

OR

```
SELECT * FROM employee
```

```
where salary >=10,000;
```

2. INSERT COMMAND

- **INSERT command** is used for inserting data into a table.
- Using this command, you can add one or more records to any single table in a database.
- It is also used to add records to an existing code.

Syntax:

```
INSERT INTO <table_name> (`column_name1` <datatype>, `column_name2` <datatype>, . . . ,  
`column_name_n` <datatype>) VALUES (`value1`, `value2`, . . . , `value n`);
```

Example:

```
INSERT INTO employee (`eid` int, `ename` varchar(20), `city` varchar(20))  
VALUES ('1', 'ABC', 'PUNE');
```

3. UPDATE COMMAND

- **UPDATE command** is used to modify the records present in existing table.
- This command updates existing data within a table.
- It changes the data of one or more records in a table.

Syntax:

```
UPDATE <table_name>  
SET <column_name = value>  
WHERE condition;
```

Example: Update Command

```
UPDATE employee
```

```
SET salary=20000
```

```
WHERE ename='ABC';
```

4. DELETE COMMAND

- **DELETE command** is used to delete some or all records from the existing table.
- It deletes all the records from a table.

Syntax:

```
DELETE FROM <table_name> WHERE <condition>;
```

Example: Delete Command

```
DELETE FROM employee
```

```
WHERE emp_id = '001';
```

If we does not write the WHERE condition, then all rows will get deleted.

Relational Algebra

Relational Algebra

RELATIONAL ALGEBRA is a widely used procedural query language. It collects instances of relations as input and gives occurrences of relations as output. It uses various operations to perform this action. Relational algebra operations are performed recursively on a relation. The output of these operations is a new relation, which might be formed from one or more input relations.

- First created by Edgar F. Codd
- The main application of relational algebraic providing a theoretical foundation for relational databases, particularly query languages for such databases, chief among which is SQL.

Relational Algebraic Operations

Relational Algebra divide in various groups

1. **Unary Relational Operations**
 - 1.1 SELECT (symbol: σ)
 - 1.2 PROJECT (symbol: π)
 - 1.3 RENAME (symbol: ρ)
2. **Relational Algebra Operations From Set Theory**
 - 2.1 UNION (\cup)
 - 2.2 INTERSECTION (\cap),
 - 2.3 DIFFERENCE (-)
 - 2.4 CARTESIAN PRODUCT (\times)
3. **Binary Relational Operations**
 - 3.1 JOIN

Unary Relational Operations

SELECT (σ)

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma(σ)Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition. Select operation selects tuples that satisfy a given predicate.

$\sigma p(r)$

σ is the predicate

r stands for relation which is the name of the table

p is prepositional logic

Example 1

σ topic = "Database" (Tutorials)

Output - Selects tuples from Tutorials where topic = 'Database'.

Example 2 $\sigma \text{topic} = \text{"Database"} \text{ and } \text{author} = \text{"Edunet"}(\text{Tutorials})$

Output - Selects tuples from Tutorials where the topic is 'Database' and 'author' is Edunet.

Example 3 $\sigma \text{sales} > 50000 (\text{Customers})$

Output - Selects tuples from Customers where sales is greater than 50000

Projection(π)

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

This helps to extract the values of specified attributes to eliminate duplicate values. (π) The symbol used to choose attributes from a relation. This operation helps you to keep specific columns from a relation and discards the other columns.

Example of Projection:

Consider the following table

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active

3	Apple	Inactive
---	-------	----------

4	Alibaba	Active
---	---------	--------

Here, the projection of CustomerName and status will give

$\Pi \text{CustomerName, Status} (\text{Customers})$

CustomerName	Status
Google	Active
Amazon	Active
Apple	Inactive
Alibaba	Active

Rename(ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho ρ .

Notation – $\rho x (E)$

Where the result of expression E is saved with the name of x.

Example: $\rho(\text{RelationNew}, \text{RelationOld})$

Relational Algebra Operations From Set Theory

Union operation (\cup)

UNION is symbolized by \cup symbol. It includes all tuples that are in tables A or in B. It also eliminates duplicate tuples. So, set A UNION set B would be expressed as:

The result $\leftarrow A \cup B$

For a union operation to be valid, the following conditions must hold -

- R and S must be the same number of attributes.
- Attribute domains need to be compatible.
- Duplicate tuples should be automatically removed.

Example

Consider the following tables.

Table A		Table B	
column 1	column 2	column 1	column 2
1	1	1	1
1	2	1	3

$A \cup B$ gives

Table A ∪ B**column 1 column 2**

1 1

1 2

1 3

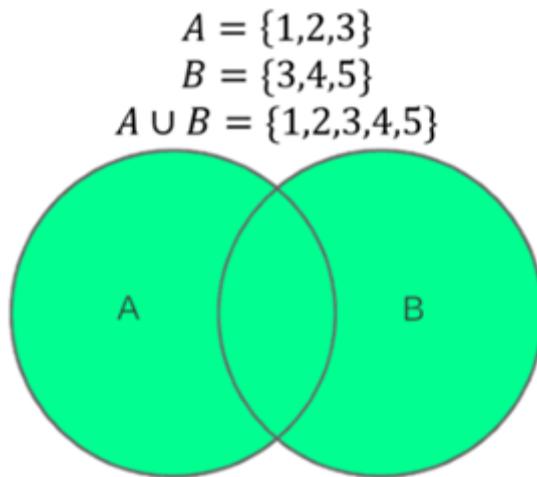


Image 17- Union Example

Reference- <https://www.codeproject.com/articles/1172312/just-enough-set-theory-when-sets-collide-part-of>

Intersection

An intersection is defined by the symbol \cap

 $A \cap B$

Defines a relation consisting of a set of all tuple that are in both A and B. However, A and B must be union-compatible.

Example: $A \cap B$

Table $A \cap B$

column 1 column 2

column 1	column 2
1	1

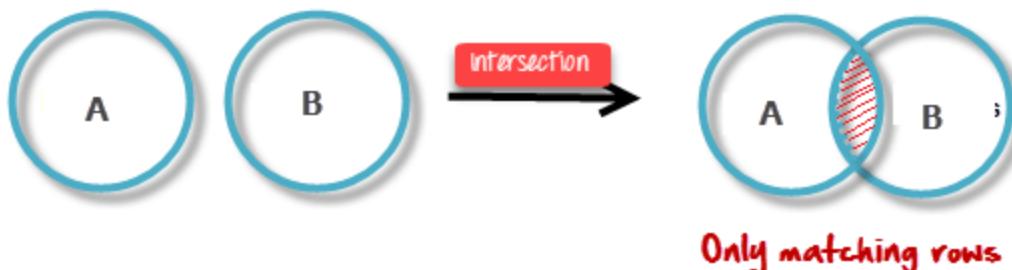


Image 18- Intersection Example
Reference- <https://www.guru99.com/relational-algebra-dbms.html>

Set Difference (-)

- Symbol denotes it. The result of $A - B$, is a relation which includes all tuples that are in A but not in B.

- The attribute name of A has to match with the attribute name in B.
- The two-operand relations A and B should be either compatible or Union compatible.
- It should be a defined relation consisting of the tuples that are in relation A, but not in B.

Example

A-B

Table A - B**column 1 column 2**

1 2

$$A = \{1,2,3\}$$

$$B = \{3,4,5\}$$

$$B \setminus A = \{4,5\}$$

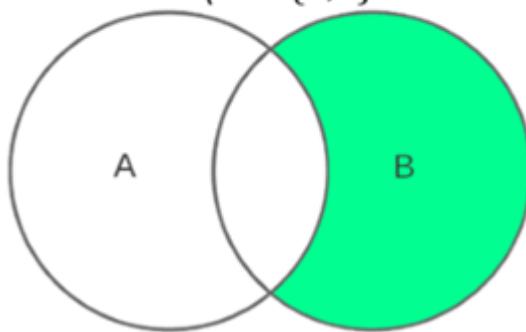


Image 19- Set Difference

Cartesian product(X)

This type of operation is helpful to merge columns from two relations. Generally, a Cartesian product is never a meaningful operation when it performs alone. However, it becomes meaningful when it is followed by other operations.

Example – Cartesian product

σ column 2 = '1' (A X B)

Output – The above example shows all rows from relation A and B whose column 2 has value 1

σ column 2 = '1' (A X B)

column 1 column 2

1	1
---	---

1	1
---	---

A1(Name, Roll No)

Name	Roll No
Anoop	1
Anurag	2

A2(Name, Roll No)

Name	Roll No
Anoop	1
Anurag	2
Ganesh	3

X



Query Output(A1 X A2)

Name	Roll No	Name	Roll No
Anoop	1	Anoop	1
Anoop	1	Anurag	2
Anoop	1	Ganesh	3
Anurag	2	Anoop	1
Anurag	2	Anurag	2
Anurag	2	Ganesh	3

Image 20- Cartesian Product

Reference- <https://www.minigranth.com/dbms-tutorial/relational-algebra/>

Binary Relational Operations

JOIN

Join operation is essentially a cartesian product followed by a selection criterion.

Join operation denoted by \bowtie .

JOIN operation also allows joining variously related tuples from different relations.

Types of JOIN

Various forms of join operation are:

1. Inner Joins:

- Theta join
- EQUI join
- Natural join

2. Outer join:

- Left Outer Join
- Right Outer Join
- Full Outer Join

Inner Join:

In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded. Let's study various types of Inner Joins:

Theta Join:

The general case of JOIN operation is called a Theta join. It is denoted by symbol θ

Example: A $\bowtie\theta$ B

Theta join can use any conditions in the selection criteria.

For example:

A \bowtie A.column 2 > B.column 2 (B)

A \bowtie A.column 2 > B.column 2 (B)

column 1	column 2
-----------------	-----------------

1	2
---	---

EQUI join:

When a theta join uses only equivalence condition, it becomes a equi join.

For example:

A \bowtie A.column 2 = B.column 2 (B)

A \bowtie A.column 2 = B.column 2 (B)

column 1	column 2
-----------------	-----------------

1	1
---	---

EQUI join is the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMS have essential performance problems.

NATURAL JOIN (\bowtie)

Natural join can only be performed if there is a common attribute (column) between the relations. The name and type of the attribute must be same.

Example

Consider the following two tables

C

Num	Square
2	4
3	9

D

Num	Cube
2	8
3	27

C \bowtie D

C \bowtie D

Num	Square	Cube
2	4	4
3	9	27

OUTER JOIN

In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

Left Outer Join(A \bowtie B)

In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.



Image 21- Left Outer Join
Reference- <https://www.guru99.com/relational-algebra-dbms.html>

Consider the following 2 Tables

A	
---	--

Num	Square
2	4
3	9
4	16

B	
---	--

Num	Cube
2	8
3	18
5	75

A \bowtie B

A \bowtie B		
Num	Square	Cube
2	4	4
3	9	9
4	16	-

Right Outer Join: (A \bowtie^R B)

In the right outer join, operation allows keeping all tuple in the right relation. However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.



Image 22- Right Outer Join

Reference- <https://www.guru99.com/relational-algebra-dbms.html>

A \bowtie B

A \bowtie B		
Num	Cube	Square

2	8	4
3	18	9
5	75	-

Full Outer Join: (A \bowtie B)

In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

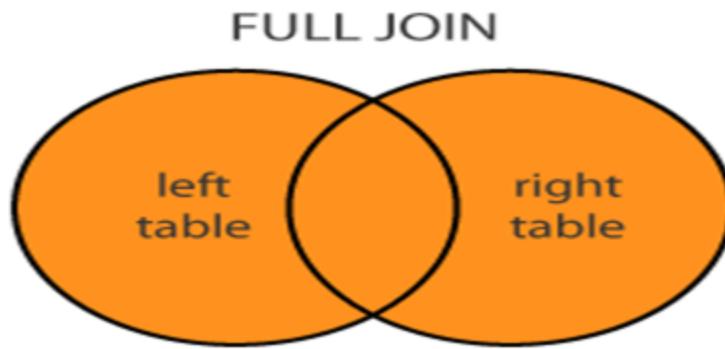


Image 23- Full Outer Join

A \bowtie B

A \bowtie B		
Num	Cube	Square
2	4	8
3	9	18

4	16	-
---	----	---

5	-	75
---	---	----

Set Operations

The SQL Set operation is used to combine the two or more SQL SELECT statements.

Types of Set Operation

1. Union
2. UnionAll
3. Intersect
4. Minus



Image 24- Set Operations
Reference- <https://www.javatpoint.com/dbms-sql-set-operation>

1. Union

- The SQL Union operation is used to combine the result of two or more SQL SELECT queries.
- In the union operation, all the number of datatype and columns must be same in both the tables on which UNION operation is being applied.
- The union operation eliminates the duplicate rows from its resultset.

Syntax

1. SELECT column_name FROM table1
2. UNION
3. SELECT column_name FROM table2;

Example:

The First table

ID	NAME
1	Jack
2	Harry
3	Jackson

The Second table

ID	NAME
3	Jackson
4	Stephan
5	David

Union SQL query will be:

1. SELECT * FROM First
2. UNION
3. SELECT * FROM Second;

The resultset table will look like:

ID	NAME
1	Jack
2	Harry

3	Jackson
4	Stephan
5	David

2. Union All

Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

Syntax:

1. SELECT column_name FROM table1
2. UNION ALL
3. SELECT column_name FROM table2;

Example: Using the above First and Second table.

Union All query will be like:

1. SELECT * FROM First
2. UNION ALL
3. SELECT * FROM Second;

The resultset table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
3	Jackson
4	Stephan
5	David

3. Intersect

-
- It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.
 - In the Intersect operation, the number of datatype and columns must be the same.
 - It has no duplicates and it arranges the data in ascending order by default.

Syntax

1. SELECT column_name FROM table1
2. INTERSECT
3. SELECT column_name FROM table2;

Example:

Using the above First and Second table.

Intersect query will be:

1. SELECT * FROM First
2. INTERSECT
3. SELECT * FROM Second;

The resultset table will look like:

ID	NAME
3	Jackson

4. Minus

- It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates and data arranged in ascending order by default.

Syntax:

1. SELECT column_name FROM table1
2. MINUS
3. SELECT column_name FROM table2;

Example

Using the above First and Second table.

Minus query will be:

1. SELECT * FROM First
2. MINUS
3. SELECT * FROM Second;

The resultset table will look like:

ID	NAME
1	John

1	Jack
2	Harry

Fundamental Operations

SELECT (σ)

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma(σ)Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition. Select operation selects tuples that satisfy a given predicate.

$\sigma p(r)$

σ is the predicate

r stands for relation which is the name of the table

p is prepositional logic

Example 1

σ topic = "Database" (Tutorials)

Output - Selects tuples from Tutorials where topic = 'Database'.

Example 2

σ topic = "Database" and author = "Edunet"(Tutorials)

Output - Selects tuples from Tutorials where the topic is 'Database' and 'author' is Edunet.

Example 3

σ sales > 50000 (Customers)

Output - Selects tuples from Customers where sales is greater than 50000

Projection(π)

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

This helps to extract the values of specified attributes to eliminate duplicate values. (Π) The symbol used to choose attributes from a relation. This operation helps you to keep specific columns from a relation and discards the other columns.

Example of Projection:

Consider the following table

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Here, the projection of CustomerName and status will give

Π CustomerName, Status (Customers)

CustomerName	Status
Google	Active
Amazon	Active

Apple	Inactive
-------	----------

Alibaba	Active
---------	--------

Cartesian product(X)

This type of operation is helpful to merge columns from two relations. Generally, a Cartesian product is never a meaningful operation when it performs alone. However, it becomes meaningful when it is followed by other operations.

Example – Cartesian product

σ column 2 = '1' (A X B)

Output – The above example shows all rows from relation A and B whose column 2 has value 1

σ column 2 = '1' (A X B)

column 1	column 2
----------	----------

1	1
---	---

1	1
---	---

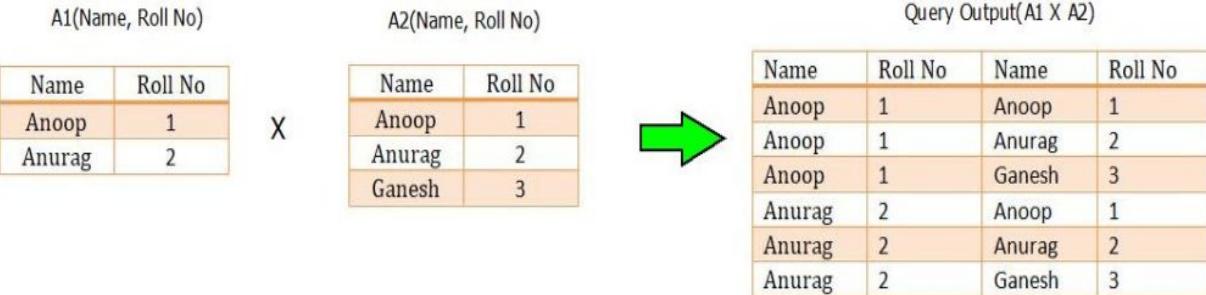


Image 20- Cartesian Product

Reference- <https://www.minigranth.com/dbms-tutorial/relational-algebra/>

Union operation (\cup)

UNION is symbolized by \cup symbol. It includes all tuples that are in tables A or in B. It also eliminates duplicate tuples. So, set A UNION set B would be expressed as:

The result $<- A \cup B$

For a union operation to be valid, the following conditions must hold -

- R and S must be the same number of attributes.
- Attribute domains need to be compatible.
- Duplicate tuples should be automatically removed.

Example

Consider the following tables.

Table A		Table B	
column 1	column 2	column 1	column 2

1	1	1	1
1	2	1	3

$A \cup B$ gives

Table A \cup B

column 1 column 2

1	1
1	2

1	3
---	---

$$\begin{aligned}A &= \{1,2,3\} \\B &= \{3,4,5\} \\A \cup B &= \{1,2,3,4,5\}\end{aligned}$$

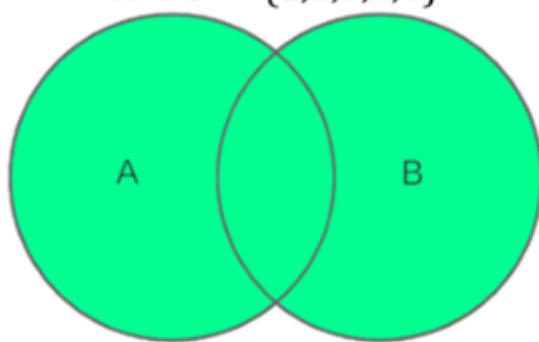


Image 17- Union Example

Reference- <https://www.codeproject.com/articles/1172312/just-enough-set-theory-when-sets-collide-part-of>

Set Difference (-)

- Symbol denotes it. The result of A - B, is a relation which includes all tuples that are in A but not in B.

- The attribute name of A has to match with the attribute name in B.
- The two-operand relations A and B should be either compatible or Union compatible.
- It should be a defined relation consisting of the tuples that are in relation A, but not in B.

Example

A-B

Table A - B

column 1	column 2
1	2

Relational Calculus

What is Relational Calculus?

Contrary to Relational Algebra which is a procedural query language to fetch data and which also explains how it is done, **Relational Calculus** is a non-procedural query language and has no description about how the query will work or the data will be fetched. It only focuses on what to do, and not on how to do it.

Relational Calculus exists in two forms:

1. Tuple Relational Calculus (TRC)
2. Domain Relational Calculus (DRC)

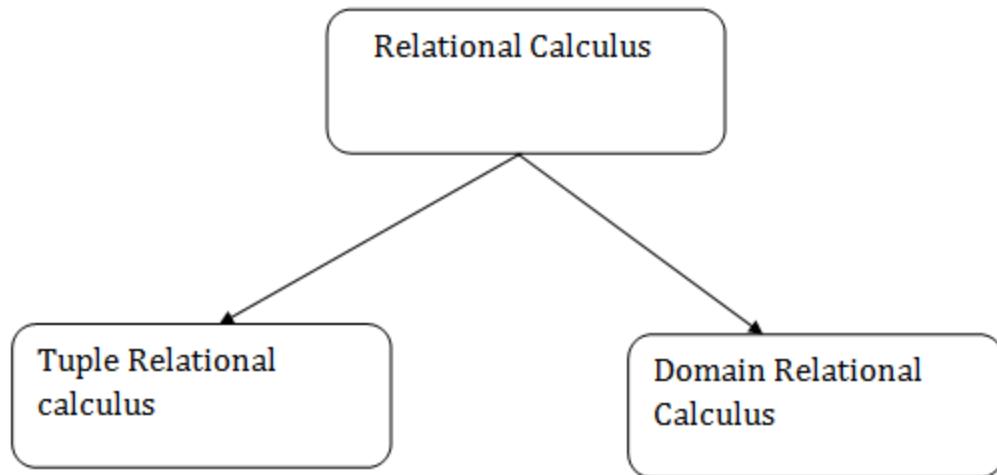


Image 25- Types of Relational Calculus

Reference- <https://www.javatpoint.com/dbms-relational-calculus>

1. Tuple Relational Calculus (TRC)

Tuple Relational Calculus is a **non-procedural query language** unlike relational algebra. Tuple Calculus provides only the description of the query but it does not provide the methods to solve it. Thus, it explains what to do but not how to do.

In Tuple Calculus, a query is expressed as

$$\{t \mid P(t)\}$$

where t = resulting tuples,

$P(t)$ = known as Predicate and these are the conditions that are used to fetch t

Thus, it generates a set of all tuples t , such that Predicate $P(t)$ is true for t .

$P(t)$ may have various conditions logically combined with OR (\vee), AND (\wedge), NOT(\neg).

It also uses quantifiers:

$\exists t \in r (Q(t))$ = "there exists" a tuple in t in relation r such that predicate $Q(t)$ is true.

$\forall t \in r (Q(t))$ = $Q(t)$ is true "for all" tuples in relation r .

Example:**Table-1: Customer**

CUSTOMER NAME	STREET	CITY
Saurabh	A7	Patiala
Mehak	B6	Jalandhar
Sumiti	D9	Ludhiana
Ria	A5	Patiala

Table-2: Branch

BRANCH NAME	BRANCH CITY
-------------	-------------

ABC

Patiala

DEF

Ludhiana

GHI

Jalandhar

Table-3: Account

ACCOUNT NUMBER	BRANCH NAME	BALANCE
----------------	-------------	---------

1111 ABC 50000

1112 DEF 10000

1113 GHI 9000

1114	ABC	7000
------	-----	------

Table-4: Loan

LOAN NUMBER	BRANCH NAME	AMOUNT
L33	ABC	10000
L35	DEF	15000
L49	GHI	9000
L98	DEF	65000

Table-5: Borrower

CUSTOMER NAME	LOAN NUMBER

Saurabh

L33

Mehak

L49

Ria

L98

Table-6: Depositor

CUSTOMER NAME	ACCOUNT NUMBER
---------------	----------------

Saurabh 1111

Mehak 1113

Sumiti 1114

Queries-1: Find the loan number, branch, amount of loans of greater than or equal to 10000 amount.

$$\{t \mid t \in \text{loan} \wedge t[\text{amount}] \geq 10000\}$$

Resulting relation:

LOAN NUMBER	BRANCH NAME	AMOUNT
L33	ABC	10000
L35	DEF	15000
L98	DEF	65000

In the above query, $t[\text{amount}]$ is known as tuple variable.

Queries-2: Find the loan number for each loan of an amount greater or equal to 10000.

$$\{t \mid \exists s \in \text{loan}(t[\text{loan number}] = s[\text{loan number}]$$
$$\wedge s[\text{amount}] \geq 10000\}$$

Resulting relation:

LOAN NUMBER

L33

L35

~~~~~L98

**Queries-3:** Find the names of all customers who have a loan and an account at the bank.

$$\{t \mid \exists s \in \text{borrower}(t[\text{customer-name}] = s[\text{customer-name}])\}$$
$$\wedge \exists u \in \text{depositor}(t[\text{customer-name}] = u[\text{customer-name}])\}$$

**Resulting relation:**

|                      |
|----------------------|
| <b>CUSTOMER NAME</b> |
|----------------------|

Saurabh

---

Mehak

**Queries-4:** Find the names of all customers having a loan at the “ABC” branch.

$\{t \mid \exists s \in \text{borrower}(t[\text{customer-name}] = s[\text{customer-name}])$

$\wedge \exists u \in \text{loan}(u[\text{branch-name}] = \text{“ABC”} \wedge u[\text{loan-number}] = s[\text{loan-number}]))\}$

Resulting relation:

| CUSTOMER NAME |
|---------------|
|               |

Saurabh

## **Domain Relational Calculus(DRC)**

**Domain Relational Calculus** is a non-procedural query language equivalent in power to Tuple Relational Calculus. Domain Relational Calculus provides only the description of the query but it does not provide the methods to solve it. In Domain Relational Calculus, a query is expressed as,

$\{ < x_1, x_2, x_3, \dots, x_n > \mid P(x_1, x_2, x_3, \dots, x_n) \}$

where,  $< x_1, x_2, x_3, \dots, x_n >$  represents resulting domains variables and  $P(x_1, x_2, x_3, \dots, x_n)$  represents the condition or formula equivalent to the Predicate calculus.

### **Predicate Calculus Formula:**

1. Set of all comparison operators
2. Set of connectives like and, or, not
3. Set of quantifiers

**Example:**

**Table-1: Customer**

| CUSTOMER NAME | STREET     | CITY       |
|---------------|------------|------------|
| Debomit       | Kadamtala  | Alipurduar |
| Sayantan      | Udaypur    | Balurghat  |
| Soumya        | Nutanchati | Bankura    |
| Ritu          | Juhu       | Mumbai     |

**Table-2: Loan**

| LOAN NUMBER | BRANCH NAME | AMOUNT |
|-------------|-------------|--------|
| L01         | Main        | 200    |
| L03         | Main        | 150    |
| L10         | Sub         | 90     |
| L08         | Main        | 60     |

**Table-3: Borrower**

| CUSTOMER NAME | LOAN NUMBER |
|---------------|-------------|
| Ritu          | L01         |
| Debomit       | L08         |

Soumya

L03

**Query-1:** Find the loan number, branch, amount of loans of greater than or equal to 100 amount.

$\{ \langle l, b, a \rangle \mid \langle l, b, a \rangle \in \text{loan} \wedge (a \geq 100) \}$

**Resulting relation:**

| LOAN NUMBER | BRANCH NAME | AMOUNT |
|-------------|-------------|--------|
| L01         | Main        | 200    |

---

|     |      |     |
|-----|------|-----|
| L03 | Main | 150 |
|-----|------|-----|

**Query-2:** Find the loan number for each loan of an amount greater or equal to 150.

$\{ \langle l \rangle \mid \exists b, a (\langle l, b, a \rangle \in \text{loan} \wedge (a \geq 150)) \}$

**Resulting relation:**

| LOAN NUMBER |
|-------------|
| L01         |

---

---

L03

**Query-3:** Find the names of all customers having a loan at the “Main” branch and find the loan amount .

{<c, a> |  $\exists l (< c, l > \in \text{borrower} \wedge \exists b (< l, b, a > \in \text{loan} \wedge (b = \text{"Main"})))$ }

**Resulting relation:**

| CUSTOMER NAME | AMOUNT |
|---------------|--------|
|---------------|--------|

Ritu 200

---

Debomit 60

---

Soumya 150

**Note:** The domain variables those will be in resulting relation must appear before | within  $<$  and  $>$  and all the domain variables must appear in which order they are in original relation or table.

**Difference between Relational Algebra and Relational Calculus:**

| S.NO | RELATIONAL ALGEBRA                                                                          | RELATIONAL CALCULUS                                            |
|------|---------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| 1.   | It is a Procedural language.                                                                | While Relational Calculus is Declarative language.             |
| 2.   | Relational Algebra means how to obtain the result.                                          | While Relational Calculus means what result we have to obtain. |
| 3.   | In Relational Algebra, The order is specified in which the operations have to be performed. | While in Relational Calculus, The order is not specified.      |
| 4.   | Relational Algebra is independent on domain.                                                | While Relation Calculus can be a domain dependent.             |

- 
5. Relational Algebra is nearer to a programming language.
- While Relational Calculus is not nearer to programming language.

---

# Data Definition language

## Introduction to DDL

- DDL stands for **Data Definition Language**.
- It is a language used for defining and modifying the data and its structure.
- It is used to build and modify the structure of your tables and other objects in the database.

## DDL commands are as follows:

1. CREATE
2. DROP
3. ALTER
4. RENAME
5. TRUNCATE

- These commands can be used to add, remove or modify tables within a database.
- DDL has pre-defined syntax for describing the data.

---

## 1. CREATE COMMAND

- **CREATE command** is used for creating objects in the database.
- It creates a new table.

### Syntax:

```
CREATE TABLE <table_name>
(
    column_name1 datatype,
    column_name2 datatype,
    .
    .
    .
    column_name_n datatype
);
```

### Example: Create command

```
CREATE TABLE employee
```

```
(

    empid INT,
    ename CHAR,
    age INT,
    city CHAR(25),
    phone_no VARCHAR(20)
```

);

## 2. DROP COMMAND

- **DROP command** allows to remove entire database objects from the database.
- It removes entire data structure from the database.
- It deletes a table, index or view.

### Syntax:

DROP TABLE <table\_name>;

OR

DROP DATABASE <database\_name>;

### Example: DROP Command

DROP TABLE employee;

OR

DROP DATABASE employees;

If you want to remove individual records, then use DELETE command of the DML statement.

## 3. ALTER COMMAND

- An **ALTER command** allows to alter or modify the structure of the database.
- It modifies an existing database object.

- Using this command, you can add additional column, drop existing column and even change the data type of columns.

**Syntax:**

```
ALTER TABLE <table_name>
ADD <column_name datatype>;
```

OR

```
ALTER TABLE <table_name>
CHANGE <old_column_name> <new_column_name>;
```

OR

```
ALTER TABLE <table_name>
DROP COLUMN <column_name>;
```

Example: Alter Command

```
ALTER TABLE employee
ADD (address varchar2(50));
```

OR

```
ALTER TABLE employee
CHANGE (phone_no) (contact_no);
```

OR

---

```
ALTER TABLE employee
```

```
DROP COLUMN age;
```

To view the changed structure of table, use 'DESCRIBE' command.

**For example:**

```
DESCRIBE TABLE employee;
```

## 4. RENAME COMMAND

- **RENAME command** is used to rename an object.
- It renames a database table.

**Syntax:**

```
RENAME TABLE <old_name> TO <new_name>;
```

**Example:**

```
RENAME TABLE emp TO employee;
```

## 5. TRUNCATE COMMAND

- **TRUNCATE command** is used to delete all the rows from the table permanently.
- It removes all the records from a table, including all spaces allocated for the records.

- 
- This command is same as DELETE command, but TRUNCATE command does not generate any rollback data.

**Syntax:**

TRUNCATE TABLE <table\_name>;

**Example:**

TRUNCATE TABLE employee;

# Operators: Select, Project, Join, Rename etc

## SELECT ( $\sigma$ )

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma( $\sigma$ )Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition. Select operation selects tuples that satisfy a given predicate.

$\sigma p(r)$

$\sigma$  is the predicate

r stands for relation which is the name of the table

p is prepositional logic

### Example 1

$\sigma$  topic = "Database" (Tutorials)

Output - Selects tuples from Tutorials where topic = 'Database'.

### Example 2

$\sigma$  topic = "Database" and author = "Edunet"( Tutorials)

Output - Selects tuples from Tutorials where the topic is 'Database' and 'author' is Edunet.

### Example 3

$\sigma$  sales > 50000 (Customers)

Output - Selects tuples from Customers where sales is greater than 50000

## Projection( $\pi$ )

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

---

This helps to extract the values of specified attributes to eliminate duplicate values. ( $\Pi$ ) The symbol used to choose attributes from a relation. This operation helps you to keep specific columns from a relation and discards the other columns.

### **Example of Projection:**

Consider the following table

| CustomerID | CustomerName | Status   |
|------------|--------------|----------|
| 1          | Google       | Active   |
| 2          | Amazon       | Active   |
| 3          | Apple        | Inactive |
| 4          | Alibaba      | Active   |

Here, the projection of CustomerName and status will give

$\Pi$  CustomerName, Status (Customers)

| CustomerName | Status |
|--------------|--------|
| Google       | Active |
| Amazon       | Active |

---

Apple                      Inactive

Alibaba                      Active

## JOIN

Join operation is essentially a cartesian product followed by a selection criterion.

Join operation denoted by  $\bowtie$ .

JOIN operation also allows joining variously related tuples from different relations.

### Types of JOIN

Various forms of join operation are:

3. Inner Joins:

- Theta join
- EQUI join
- Natural join

4. Outer join:

- Left Outer Join
- Right Outer Join
- Full Outer Join

#### Inner Join:

In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded. Let's study various types of Inner Joins:

#### Theta Join:

---

The general case of JOIN operation is called a Theta join. It is denoted by symbol  $\theta$

**Example:**  $A \bowtie \theta B$

Theta join can use any conditions in the selection criteria.

**For example:**

$A \bowtie A.\text{column } 2 > B.\text{column } 2 (B)$

**$A \bowtie A.\text{column } 2 > B.\text{column } 2 (B)$**

| column 1 | column 2 |
|----------|----------|
| 1        | 2        |

**EQUI join:**

When a theta join uses only equivalence condition, it becomes a equi join.

**For example:**

$A \bowtie A.\text{column } 2 = B.\text{column } 2 (B)$

**$A \bowtie A.\text{column } 2 = B.\text{column } 2 (B)$**

| column 1 | column 2 |
|----------|----------|
| 1        | 1        |

EQUI join is the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMS have essential performance problems.

## **NATURAL JOIN ( $\bowtie$ )**

Natural join can only be performed if there is a common attribute (column) between the relations. The name and type of the attribute must be same.

### Example

Consider the following two tables

| C |
|---|
|---|

| Num | Square |
|-----|--------|
| 2   | 4      |
| 3   | 9      |

| D |
|---|
|---|

| Num | Cube |
|-----|------|
| 2   | 8    |
| 3   | 27   |

C  $\bowtie$  D

| C $\bowtie$ D |        |      |
|---------------|--------|------|
| Num           | Square | Cube |
| 2             | 4      | 4    |
| 3             | 9      | 27   |

### OUTER JOIN

In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

#### Left Outer Join(A $\bowtie$ B)

In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.



Image 26- Left Outer Join  
Reference- <https://www.guru99.com/relational-algebra-dbms.html>

Consider the following 2 Tables

| A   |        |
|-----|--------|
| Num | Square |
| 2   | 4      |
| 3   | 9      |
| 4   | 16     |

| B   |      |
|-----|------|
| Num | Cube |
| 2   | 8    |
| 3   | 18   |
| 5   | 75   |

$A \bowtie B$ 

| <b>A <math>\bowtie</math> B</b> |               |             |
|---------------------------------|---------------|-------------|
| <b>Num</b>                      | <b>Square</b> | <b>Cube</b> |
| 2                               | 4             | 4           |
| 3                               | 9             | 9           |
| 4                               | 16            | -           |

**Right Outer Join: ( A  $\bowtie$  B )**

In the right outer join, operation allows keeping all tuple in the right relation. However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.



Image 27- Right Outer Join

Reference- <https://www.guru99.com/relational-algebra-dbms.html> $A \bowtie B$ 

| <b>A <math>\bowtie</math> B</b> |             |               |
|---------------------------------|-------------|---------------|
| <b>Num</b>                      | <b>Cube</b> | <b>Square</b> |
| 2                               | 8           | 4             |

|   |    |   |
|---|----|---|
| 3 | 18 | 9 |
| 5 | 75 | - |

### Full Outer Join: ( A $\bowtie$ B)

In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

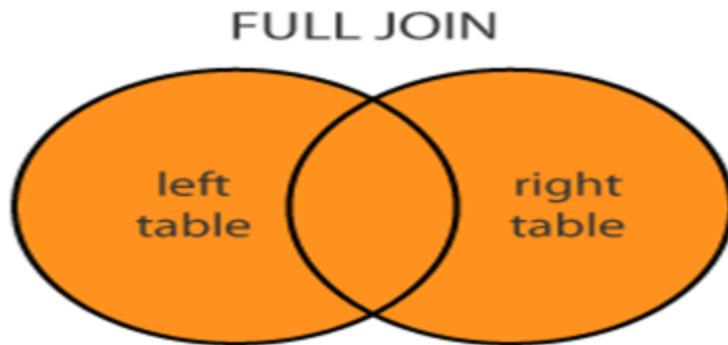


Image 28- Full Outer Join

A  $\bowtie$  B

| A $\bowtie$ B |      |        |
|---------------|------|--------|
| Num           | Cube | Square |
| 2             | 4    | 8      |
| 3             | 9    | 18     |

---

|   |    |    |
|---|----|----|
| 4 | 16 | -  |
| 5 | -  | 75 |

## Rename( $\rho$ )

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho  $\rho$ .

Notation –  $\rho x (E)$

Where the result of expression E is saved with the name of x.

**Example:**  $\rho(\text{RelationNew}, \text{RelationOld})$



---

# Introduction to MongoDB

## Brief History of MongoDB

- MongoDB was developed by Eliot Horowitz and Dwight Merriman in the year 2007
- When they experienced some scalability issues with the relational database while developing enterprise web applications at their company DoubleClick.
- According to Dwight Merriman, one of the developers of MongoDB, this name of the database was derived from the word *humongous* to support the idea of processing large amount of data.
- In 2009, MongoDB was made as an open source project, while the company offered commercial support services.
- Many companies started using MongoDB for its amazing features.
- The New York Times newspaper used MongoDB to build a web based application to submit the photos.
- In 2013, the company was officially named MongoDB Inc.

## What is MongoDB?

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and functions which is the equivalent of relational database tables. MongoDB is a database which came into light around

## MongoDB Features

Apart from most of the NoSQL default features, MongoDB does bring in some more, very important and useful features :

- 
1. MongoDB provides high performance. Input/Output operations are lesser than relational databases due to support of embedded documents(data models) and Select queries are also faster as Indexes in MongoDB supports faster queries.
  2. MongoDB has a rich Query Language, supporting all the major CRUD operations. The Query Language also provides good Text Search and Aggregation features.
  3. **Auto Replication** feature of MongoDB leads to High Availability. It provides an automatic failover mechanism, as data is restored through backup(replica) copy if server fails.
  4. Sharding is a major feature of MongoDB. Horizontal Scalability is possible due to sharding.
  5. MongoDB supports multiple Storage Engines. When we save data in the form of documents(NoSQL) or tables(RDBMS) , who saves the data? It's the Storage Engine. Storage Engines manages how data is saved in memory and on disk.

## Why Use MongoDB?

Below are the few of the reasons as to why one should start using MongoDB

- 
1. Document-oriented – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.
  2. Ad hoc queries - MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.
  3. Indexing - Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.
  4. Replication - MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.
  5. Load balancing - MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

## Difference between MongoDB & RDBMS

Below are some of the key term differences between MongoDB and RDBMS

| RDBMS  | MongoDB            | Difference                                                                                                                                                                                                                                                                                                                  |
|--------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Table  | Collection         | In RDBMS, the table contains the columns and rows which are used to store the data whereas, in MongoDB, this same structure is known as a collection. The collection contains documents which in turn contains Fields, which in turn are key-value pairs.                                                                   |
| Row    | Document           | In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB, the data is stored in documents.                                                                                                                                                                                             |
| Column | Field              | In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields.                                                                                                                                                                                                                                    |
| Joins  | Embedded documents | In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using Embedded documents. So there is no concept of joins in MongoDB. |

Apart from the terms differences, a few other differences are shown below

1. Relational databases are known for enforcing data integrity. This is not an explicit requirement in MongoDB.

- 
2. RDBMS requires that data be normalized first so that it can prevent orphan records and duplicates. Normalizing data then has the requirement of more tables, which will then result in more table joins, thus requiring more keys and indexes.

As databases start to grow, performance can start becoming an issue. Again this is not an explicit requirement in MongoDB. MongoDB is flexible and does not need the data to be normalized first.

## Where to Use MongoDB?

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub



---

## Advantages of MongoDB over RDBMS

- More scalability
- More in performance
- There is no concept of relationship.
- MongoDB is schema less.
- It is a document database in which one collection holds different documents.
- Structure of a single object is clear in MongoDB.
- There are no complex joins in MongoDB.
- MongoDB provides the facility of deep query
- It is very easy to scale.
- It uses internal memory for storing working sets
- There may be a difference between the number of fields, content and size of the document from one to another.

---

# Able to Design and Develop Dynamic Websites with PHP

In this section, we will read about:

- Decisions and loops using HTML
- Doing Repetitive task with looping
- Mixing Decisions and looping with Html
- What is a function?
- Creating and accessing, String Searching & Replacing String
- Formatting String ,String Related Library function
- Array - Anatomy of an Array
- Creating and accessing index based and Associative array
- Looping with Index based array
- Looping with associative array using each() and for each ()
- Some useful Library functions
- Working with File and Directories – Understanding File & Directory
- Opening and Closing a file. Copying, renaming and deleting a file
- Building a text editor. File uploading and Downloading
- Using Hidden field, cookies, session
- Using query string (URL rewriting)
- String matching with regular expression
- Pattern matching in PHP, replacing text, splitting a string with regular expression
- Generating images with PHP – Basics of computer graphics
- Creating image, manipulating images using text in images
- Database connectivity with MySql

---

## Decisions and loops using HTML

PHP allows us to perform actions based on some type of conditions that may be logical or comparative. Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed as asked by the user. It's just like a two-way path. If you want something then go this way or else turn that way.

To use this feature, PHP provides us with four conditional statements:

**The if Statement :** In if Statements Output will appear when only Condition must be true.

**The if-else Statement :** if-else statements allows you to display output in both the condition(if condition is true display some message otherwise display other message).

**The if-elseif-else Statement :** The if-else-if-else statement lets you chain together multiple if-else statements, thus allowing the programmer to define actions for more than just two possible outcomes.

**The switch Statement :** The switch statement is similar to a series of if statements on the same expression.

Let us now look at each one of these in details:

**if Statement:** This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed.

**Syntax :**

```
if (condition){  
    // if TRUE then execute this code  
}
```

**Example:**

```
<?php  
$x = 12;  
if ($x > 0) {
```

```
echo "The number is positive";  
}  
?>
```

**Output:**

The number is positive

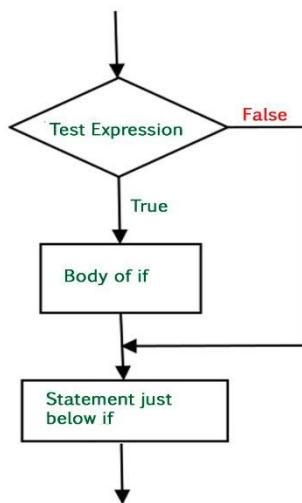
**Flowchart:**

Image 1-Flowchart for if

Reference: <https://www.geeksforgeeks.org/php-decision-making/>

**If-else Statement:** We understood that if a condition will hold i.e., TRUE, then the block of code within if will be executed. But what if the condition is not TRUE and we want to perform an action? This is where else comes into play. If a condition is TRUE then if block gets executed, otherwise else block gets executed.

Syntax:

```
if (condition) {  
    // if TRUE then execute this code  
}
```

```
else{
    // if FALSE then execute this code
}
```

Example:

```
<?php
$x = -12;
if ($x > 0) {
    echo "The number is positive";
}
else{
    echo "The number is negative";
}
?>
```

Output:

The number is negative

**Flowchart:**

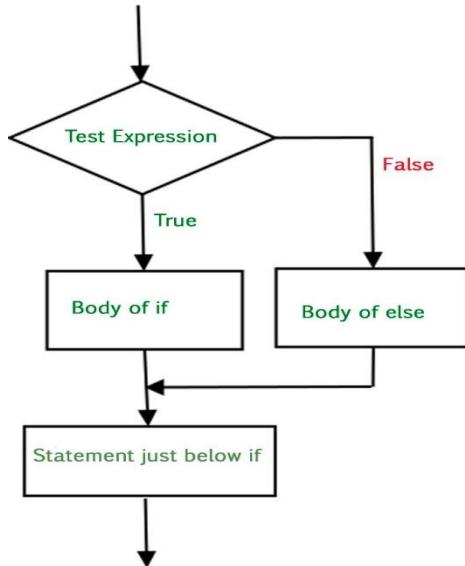


Image 2-Flowchart for if-else

Reference : <https://www.geeksforgeeks.org/php-decision-making/>

**if...elseif...else Statement:** This allows us to use multiple if...else statements. We use this when there are multiple conditions of TRUE cases.

#### Syntax:

```
if (condition) {  
    // if TRUE then execute this code  
}  
  
elseif {  
    // if TRUE then execute this code  
}  
  
elseif {  
    // if TRUE then execute this code  
}  
  
else {  
    // if FALSE then execute this code  
}
```

}

**Example:**

```
<?php  
$x = "August";  
if ($x == "January") {  
    echo "Happy Republic Day";  
}  
elseif ($x == "August") {  
    echo "Happy Independence Day!!!";  
}  
else{  
    echo "Nothing to show";  
}  
?>
```

Output:

Happy Independence Day!!!

**Flowchart:**

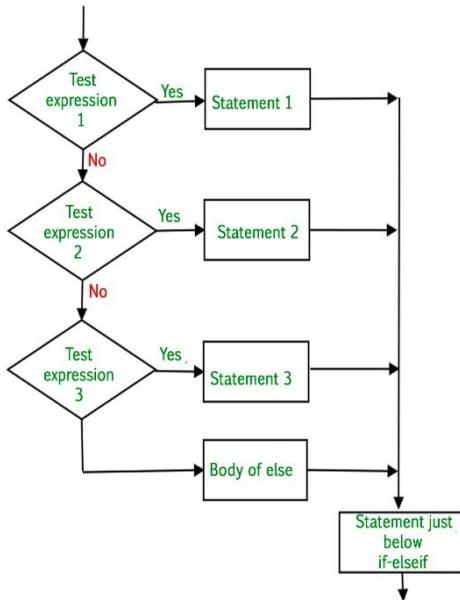


Image 3-Flowchart for if-elseif

Reference: <https://www.geeksforgeeks.org/php-decision-making/>

**switch Statement:** The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, break and default.

The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.

The **default** statement contains the code that would execute if none of the cases match.

### Syntax:

```
switch(n) {
```

---

```
case statement1:  
    code to be executed if n==statement1;  
    break;  
case statement2:  
    code to be executed if n==statement2;  
    break;  
case statement3:  
    code to be executed if n==statement3;  
    break;  
case statement4:  
    code to be executed if n==statement4;  
    break;  
.....  
default:  
    code to be executed if n != any case;
```

**Example:**

```
<?php  
$n = "February";  
switch($n) {  
    case "January":  
        echo "Its January";  
        break;  
    case "February":  
        echo "Its February";  
        break;
```

---

```
case "March":  
    echo "Its March";  
    break;  
case "April":  
    echo "Its April";  
    break;  
case "May":  
    echo "Its May";  
    break;  
case "June":  
    echo "Its June";  
    break;  
case "July":  
    echo "Its July";  
    break;  
case "August":  
    echo "Its August";  
    break;  
case "September":  
    echo "Its September";  
    break;  
case "October":  
    echo "Its October";  
    break;  
case "November":  
    echo "Its November";
```

```
break;  
case "December":  
    echo "Its December";  
    break;  
default:  
    echo "Doesn't exist";  
}  
?>
```

**Output:**

Its February

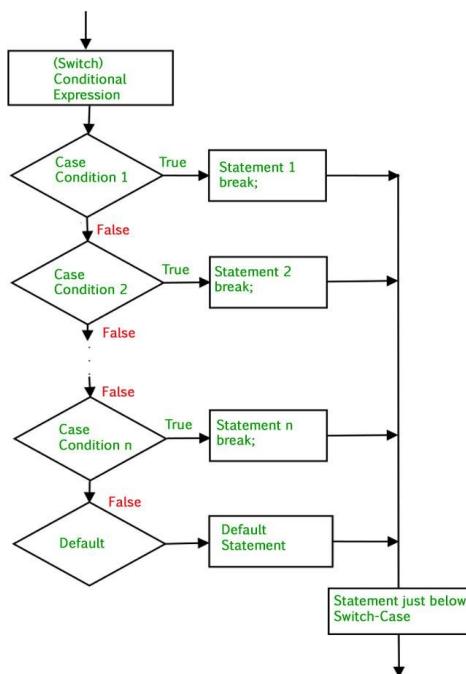
**Flowchart:**

Image 4-Flowchart for switch

Refrence: <https://www.geeksforgeeks.org/php-decision-making/>

**Ternary Operators :**

In addition to all this conditional statements, PHP provides a shorthand way of writing if...else, called Ternary Operators. The statement uses a question mark (?) and a colon (:) and takes three operands: a condition to check, a result for TRUE and a result for FALSE.

**Syntax:**

```
<?php  
$x = -12;  
// single line using ternary operator  
echo ($x > 0) ? 'The number is positive' :  
    'The number is negative';  
?>
```

**Output:** The number is negative

---

# Doing Repetitive task with looping

Loops are used to imitate a series of actions until a specified condition is fulfilled. In PHP Loops are used to execute a statement or a block of statements, multiple times until and unless a specific condition is met. This helps the user to save both time and effort of writing the same code multiple times.

**PHP supports four types of looping techniques:**

- **for loop**
- **while loop**
- **do-while loop**
- **foreach loop**

Let us now learn about each of the above mentioned loops in details:

**for loop:** This type of loops is used when the user knows in advance, how many times the block needs to execute. That is, the number of iterations is known beforehand. These type of loops are also known as entry-controlled loops. There are three main parameters to the code, namely the initialization, the test condition and the counter.

**Syntax:**

```
for (initialization expression; test condition; update expression) {  
    // code to be executed  
}
```

In for loop, a loop variable is used to control the loop. First initialize this loop variable to some value, then check whether this variable is less than or greater than counter value. If statement is true, then loop body is executed and loop variable gets updated . Steps are repeated till exit condition comes.

**Initialization Expression:** In this expression we have to initialize the loop counter to some value. for example: \$num = 1;

**Test Expression:** In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of loop and go to update expression otherwise we will exit from the for loop. For example: \$num <= 10;

**Update Expression:** After executing loop body this expression increments/decrements the loop variable by some value. for example: \$num += 2;

**Example:**

```
<?php  
// code to illustrate for loop  
for ($num = 1; $num <= 10; $num += 2) {  
    echo "$num \n";  
}  
?>
```

Output:

1 3 5 7 9

**Flow Diagram:**

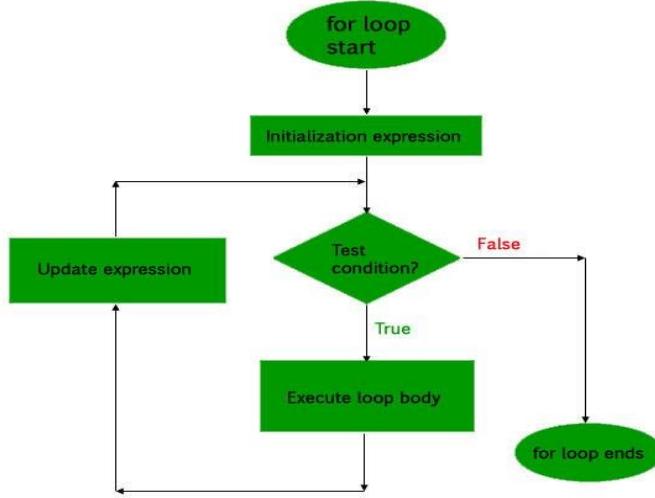


Image 5- Flowchart of for loop

References: <https://www.geeksforgeeks.org/php-loops/?ref=lbp>

**while loop:** The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if its true then it enters the loop and executes the block of statements, and goes on executing it as long as the condition holds true.

#### Syntax:

```
while (if the condition is true) {  
    // code is executed  
}
```

#### Example:

```
<?php  
// PHP code to illustrate while loops  
$num = 2;  
while ($num < 12) {  
    $num += 2;  
    echo $num, "\n";  
}  
?>
```

---

Output:

4 6 8 10 12

**Flowchart:**

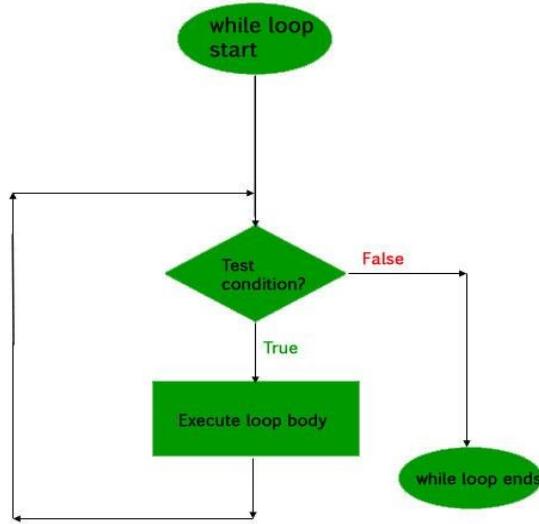


Image 6- Flowchart of While loop

Refrence: <https://www.geeksforgeeks.org/php-loops/?ref=lbp>

**do-while loop:** This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do...while loop. After executing once, the program is executed as long as the condition holds true.

Syntax:

```
do {  
    //code is executed  
} while (if condition is true);
```

**Example:**

```
<?php  
// PHP code to illustrate do...while loops
```

---

```
$num = 2;  
do {  
    $num += 2;  
    echo $num, "\n";  
} while ($num < 12);  
?>
```

**Output:**

```
4 6 8 10 12
```

**Do-While Loop:** Do while loop is very similar to the while loop, but it always executes the code block at least once and furthermore as long as the condition remains true.

This code would show the difference between while and do...while loop.

**Example :**

```
<?php  
$i = 1;  
do{  
    $i++;  
    echo "The number is " . $i . "<br>";  
}  
while($i <= 3);  
?>
```

**Output:**

```
The number is 1  
The number is 2  
The number is 3  
The number is 4
```

**Flow Chart:**

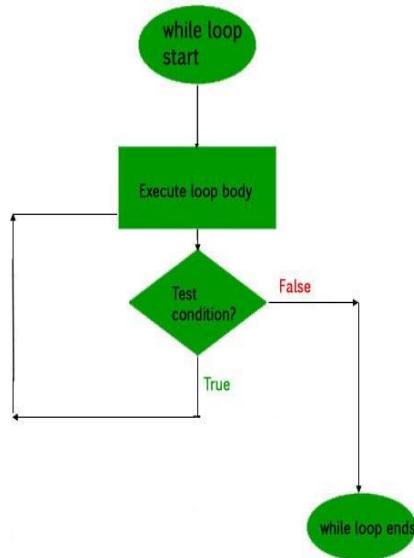


Image 7-Flow chart of do-while loop

Reference: <https://www.geeksforgeeks.org/php-loops/?ref=lbp>

### Difference Between while and do...while Loop:

The while loop differs from the do-while loop in one important way — with a while loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed.

With a do-while loop, on the other hand, the loop will always be executed once, even if the conditional expression is false, because the condition is evaluated at the end of the loop iteration rather than the beginning.

**foreach loop:** This loop is used to iterate over arrays. For every counter of loop, an array element is assigned and the next counter is shifted to the next element.

### Syntax:

```
foreach (array_element as value) {  
    //code to be executed  
}
```

**Example:**

```
<?php  
$arr = array (10, 20, 30, 40, 50, 60);  
foreach ($arr as $val) {  
    echo "$val \n";  
}  
  
$arr = array ("Ram", "Laxman", "Sita");  
foreach ($arr as $val) {  
    echo "$val \n";  
}  
?>
```

**Output:**

10 20 30 40 50 60

Ram Laxman Sita

## Mixing Decisions and looping with Html:

PHP will only process things that are enclosed within one of its valid code blocks (such as <?php and ?>). Because of this, PHP effectively ignores everything that it was not specifically told to process and can be used to our advantage.

**For example:**

```
<?php  
$var = 7;  
?  
$var = 12;<br />
```

The variable \$var has a value of:

```
<?=$var?><br />
```

Is this a valid script? Yes, the output would be the following:

```
$var = 12;
```

Although, variable \$var has a value of 7

Notice that with the second assignment of \$var, when we attempt to change the value from 7 to 12, it has no effect because it is not enclosed within valid PHP code-block syntax. So, instead of being processed, it is simply displayed to the web browser.

## PHP Conditions for HTML Code:

Let's consider we only want PHP to display a certain HTML code if the condition is true then we can do that using the following syntax

```
<?php if(conditions)  
{ ?>  
... HTML CODE ...  
<?php
```

```
} ?>
```

Although this may be confusing, it is important to remember that how PHP will process this code. To start, it will evaluate the first line of a normal if statement and then begin a code block. Then, we turn off PHP parser and jump into normal HTML code (all of which PHP will simply output to the browser and ignore) until, finally, we return to PHP code and close our if statement. The result of this technique is a way for us to control regular and standard HTML with nearly no intrusion by PHP into the syntax. Although the above example works, a special syntax is provided for instances where PHP is being used simply to control the output of standard HTML code:

```
<?php if(conditions): ?>  
... HTML CODE ...  
<?php endif; ?>
```

This syntax is identical in function to the original example provided.

### **Other valid embedded syntax :**

Beyond simple if statements, most control structures provide an alternative syntax that allows us to embed PHP code within standard HTML quickly and easily. For example, below are definitions for our repetition statements while and for (starting with while):

```
<?php while(conditions) : ?>  
... HTML CODE ...  
<?php endwhile; ?>
```

And an identical syntax for an embedded for loop:

```
<?php for(init;conditions;increment) : ?>  
... HTML CODE ...  
<?php endfor; ?>
```

---

### Embedded code in action :

Let's look at the following example

We will list, in an HTML table, all the numbers between 1 and 6 and determine under what circumstances our friend Shilpa could purchase a chocolate?

1 2 3 4 5 6

no no no no yes yes

Let's code

```
<html>
<body>
<table>
<tr>
<td align="center">1</td>
<td align="center">2</td>
<td align="center">3</td>
<td align="center">4</td>
<td align="center">5</td>
<td align="center">6</td>
</tr>
<tr>
<td align="center">no</td>
<td align="center">no</td>
<td align="center">no</td>
<td align="center">no</td>
<td align="center">yes</td>
<td align="center">yes</td>
```

```
</tr>
</table>
</body>
</html>
```

As you might have noticed that this would have been too much effort especially if it was to be done with numbers till 100, therefore, we can do it easily with the help of PHP code as well as displayed in the following example

```
<html>
<body>
<table>
<tr>
<?php for($l = 1; $l <=6; $l++) : ?>
<td align="center"><?= $l ?></td>
<?php endfor; ?>
</tr>
<tr>
<?php for($l = 1; $l <=6; $l++) : ?>
<td align="center">
<?php if($l >= 5) {
echo "yes";
} else {
echo "no";
}
?>
</td>
<?php endfor; ?>
```

```
</tr>
</table>
</body>
</html>
```

As you can see this clearly in the given example, we start by simply outputting the basic HTML code to construct the web page and begin a table. Then, we use PHP to start a for loop to count from 1 to 6. Within this loop, we display the HTML code to first start a table cell, then display the variable we are counting with (`$l`) and finally the HTML to close the table cell.

Once this has been completed, we close our row and start a new row and repeat the same looping process. This time, however, instead of simply outputting the looped variable, we use a conditional statement to determine if it is greater than or equal to the value 5 in which case we output “yes” or “no” depending on the value of `$l`. Finally, we finish the HTML for our table and web page and the script ends.

# Functions

## What is function?

Function is a group of statements that are grouped together to perform certain task. Once after defining the function, it has to be called to execute. Function can be called as "mini-program", because a function is written once and can be used as many times as needed in a program. A function can be called with the function name.

Function executes only when it is called but cannot be executed automatically during page buffering.

## Advantages of functions:

- PHP Functions reduces the memory and complexity of the program.
- Functions mainly helps in re-usability.
- PHP Functions reduces the bugs and saves the time of programmer.
- Information hiding is possible by functions.
- Once the function execution is completed it returns to the position where it was called.

## Functions in PHP are divided into two types:

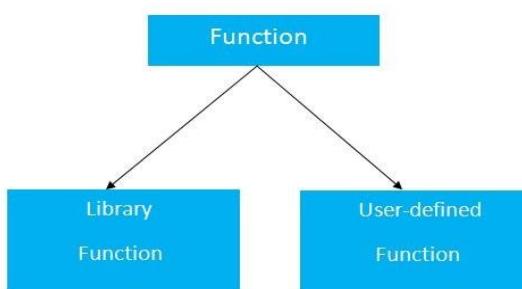


Image 8- Types of functions

---

References: <https://static.javatpoint.com/cpages/images/c-function.jpg>

### **Predefined function:**

Predefined function, which is already built-in functions in PHP. There are about 1000 built-in functions in PHP. These can be retrieved by the programmer directly.

### **User defined functions:**

User defined functions, which is built by the user based on the requirement.

### **Define a Function:**

While creating a user defined function we need to keep few things in mind:

- Any name ending with an open and closed parenthesis is a function.
- A function name always begins with the keyword function.
- To call a function we just need to write its name followed by the parenthesis
- A function name cannot start with a number. It can start with an alphabet or underscore.
- A function name is not case-sensitive.

### **Syntax:**

```
function function_name(){
```

```
    executable code;
```

```
}
```

### **Example:**

```
<?php
/* Defining a PHP Function */
function writeMessage() {
    echo "You are really a nice person, Have a nice time!";
}
/* Calling a PHP Function */
writeMessage();
```

---

```
?>
```

**This will display following result –**

You are really a nice person, Have a nice time!

### **Functions with Parameters:**

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

#### **Example:**

```
<?php  
function addFunction($num1, $num2) {  
    $sum = $num1 + $num2;  
    echo "Sum of the two numbers is : $sum";}  
addFunction(10, 20);  
?>
```

#### **Output:**

Sum of the two numbers is 30

**There are two ways to pass the arguments while calling a function from a program.**

**They are :**

- Call by value.
- Call by reference.

## **Call by value:**

While calling a function, we pass values of variables to it. Such functions are known as “Call By Values”.

In this method, the value of each variable in calling function is copied into corresponding dummy variables of the called function.

With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function.

### **Example :**

```
<?php  
    function withArg($x) {  
        echo "The x value is : $x <br />"; }  
    withArg(30);  
?>
```

### **Output:**

The x value is 30

## **Call by Reference:**

While calling a function, instead of passing the values of variables, we pass address of variables (location of variables) to the function known as “Call By References”.

In this method, the address of actual variables in the calling function are copied into the dummy variables of the called function.

With this method, using addresses we would have an access to the actual variables and hence we would be able to manipulate the actual variable in the calling function.

### **Following example depicts both the cases.**

```
<?php  
function addFive($num) {  
    $num += 6;  
}  
function addSix(&$num) {  
    $num += 7;  
}  
$orignum = 20;  
addFive( $orignum );  
echo "Original Value is $orignum<br />";  
addSix( $orignum );  
echo "Original Value is $orignum<br />";  
  
?>
```

**Output:**

The Original Value is 20

The Original Value is 27

**Recursive Function :**

Recursive function is a function which calls itself again and again until the termination condition arrive. This enables the function to repeat itself several times, outputting the result and the end of each iteration.

```
<?php  
function fact($n)  
{  
if ($n === 0)  
{  
    // our base case
```

---

```
        return 1; }
else
{
    return $n * fact($n-1); // <--calling itself.
}
echo fact(5);
?>
```

**Output:**

120

**Types of User defined functions:**

- Without Arguments
- With Arguments
- Without Return Value
- With Return Value

**Function Without Arguments:**

The information is passed to the functions by using arguments. Here, no arguments are passed to the function

**Example:**

```
<?php
function withoutArg() {
    echo "Welcome to Functions";
}
withoutArg(); // function calls
?>
```

**Output:**

## Welcome to Functions

### **Function With Arguments:**

The information is passed to the functions by using arguments. Here, arguments are passed to the function.

#### **Example:**

```
<?php  
    function withArg($x) {  
        echo "The x value is : $x <br />";}  
    withArg(30);  
?>
```

#### **Output:**

The x value is 30

### **Function Without Return Statement:**

The function won't return any value.

#### **Example:**

```
<?php  
    function withArg($x) {  
        echo "The x value is : $x";  
    }  
    withArg(30);  
?>
```

#### **Output:**

The x value is 30

### **Function With Return Statement:**

The function returns a value.

**Example:**

```
<?php  
    function withRet($x, $y) {  
        $z = $x + $y;  
        return $z;  
    }  
    echo "30 + 40 = " . withRet(30, 40) . "<br /> ";  
    echo "40 + 50 = " . withRet(40, 50);  
?  
>
```

**Output:**

30+40 = 70

40+50= 90

**Built in functions:**

Built in functions are functions that exist in PHP installation package. These built in functions are what make PHP a very efficient and productive scripting language. The built in functions can be classified into many categories.

Below is the list of the categories.

**String Functions:**

These are functions that manipulate string data.

| Function   | Description                                                                                                                                                                                                                                            | Example                                                                                            | Output                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| strtolower | Used to convert all string characters to lower case letters                                                                                                                                                                                            | echo strtolower('Benjamin');                                                                       | outputs benjamin                                                                |
| strtoupper | Used to convert all string characters to upper case letters                                                                                                                                                                                            | echo strtoupper('george w bush');                                                                  | outputs GEORGE W BUSH                                                           |
| strlen     | The string length function is used to count the number of character in a string. Spaces in between characters are also counted                                                                                                                         | echo strlen('united states of america');                                                           | 24                                                                              |
| explode    | Used to convert strings into an array variable                                                                                                                                                                                                         | \$settings = explode(';', "host=localhost; db=sales; uid=root; pwd=demo"); print_r(\$settings);    | Array ( [0] => host=localhost [1] => db=sales [2] => uid=root [3] => pwd=demo ) |
| substr     | Used to return part of the string. It accepts three (3) basic parameters. The first one is the string to be shortened, the second parameter is the position of the starting point, and the third parameter is the number of characters to be returned. | \$my_var = 'This is a really long sentence that I wish to cut short'; echo substr(\$my_var,0,12)!; | This is a re...                                                                 |

Reference: <https://www.guru99.com/php-strings.html#6>

### Numeric Functions:

Numeric functions are function that return numeric results. Numeric php function can be used to format numbers, return constants, perform mathematical computations etc.

|       |                                                                     |                                                     |                      |
|-------|---------------------------------------------------------------------|-----------------------------------------------------|----------------------|
| rand  | Used to generate a random number.                                   | <pre>&lt;?php<br/>echo rand();<br/>?&gt;</pre>      | Random number        |
| round | Round off a number with decimal points to the nearest whole number. | <pre>&lt;?php<br/>echo round(3.49);<br/>?&gt;</pre> | 3                    |
| sqrt  | Returns the square root of a number                                 | <pre>&lt;?php<br/>echo sqrt(100);<br/>?&gt;</pre>   | 10                   |
| cos   | Returns the cosine                                                  | <pre>&lt;?php<br/>echo cos(45);<br/>?&gt;</pre>     | 0.5253219888<br>1773 |

Reference: <https://www.guru99.com/functions-in-php.html>

# Strings

## What is String?

String is a collection of characters. like Tutorial, PHP, java language etc. When a number is placed into the single or double quote then it's also a string so it can't be used in arithmetic. PHP provides a lot of pre-defined function for string manipulation like strlen, strrev, HTML special characters etc.

## Some Valid String Are:

“hello”

“1245”

“436^%\$”

“hello and welcome to php”

**There are many numbers of functions that are directly concerned with manipulating strings**

- a) Search for text within a string
- b) Calculate the string length
- c) Break a string down into component parts
- d) Formatting of strings

## String Example:

```
<?php  
$str =" And welcome to php string " ;  
echo " $str ."."<br>" ;  
echo " Hello, $str! " . "<br>" ;  
echo " length:-".strlen($str);  
?>
```

## Output:

And welcome to php string

---

Hello, And welcome to php string !

length:-27

In the above example, \$str is a string variable and print the string with the help of the echo construct. strlen is a built-in function that is used to calculate the length of the string

### **Creating and Accessing Strings:**

Creating a string variable is as simple as assigning a literal string value to a new variable name.

Both single and double quote are used to create a string.

Syntax \$var="User";

\$my\_variable = " hello " ;

#### **Example:**

```
<?php  
$myvariable = " Welcome to php string " ;  
echo "$myvariable "."<br>";  
?>
```

#### **Output:**

Welcome to php string

### **Creating More Complex String Expression:**

```
<?php  
$car = "Ferrari 458";  
echo "My favourite car is $car"."<br>";  
echo "My favourite car is ${car} "."<br>";  
echo "My favourite car is ${car} "."<br>";  
?>
```

#### **Output:**

---

My favourite car is Ferrari 458

My favourite car is Ferrari 458

My favourite car is Ferrari 458

### **Searching Strings With Strstr() Function:**

Find out whether some text occurs within a string or not, using strstr() function, if string match print all the string from where string matched. If the word is not found strstr function returns false.

#### **Syntax of Strstr() Function:**

Strstr( \$expression ,search text );

#### **Example:**

```
<?php  
$string1= "Hello world!";  
echo strstr( $string1, "H" )."<br>";  
echo ( strstr( $string1, "xyz" ) ? "Yes" : "No" )."<br>";  
$string2= "Welcome to php string";  
echo strstr( $string1, "java" )."<br>"; //nothing print  
echo ( strstr( $string2, "php" ) ? "Yes" : "No" )."<br>";  
?>
```

#### **Output:**

Hello world!

No

Yes

### **Accessing Characters Within A String:**

In php we can easy to access a character at a position from string.

#### **Syntax:**

\$char = \$str [ position ] ;

**Example:**

```
<?php  
$myStr = "Welcome to the php string";  
echo $myStr[0] . "<br>"; // print "W"  
echo $myStr[6] . "<br>"; // print "e"  
$myStr[25] = '?';//Welcome to the php string?  
echo $myStr . "<br>" ;?>
```

**Output:**

W

e

Welcome to the php string?

We can easily print any character of the string and add some or replace the character with the help of index number.

**Searching & Replacing String**

str\_replace() replaces one part of a string with new parts you specify. Str\_replace() takes a minimum of three parameters: what to look for, what to replace it with, and the string to work with. It also has an optional fourth parameter, which, if passed, will be filled with the number of replacements made. Str\_replace() is a very easy way to find and replace text in a string.

Str\_replace() is a very easy way to find and replace text in a string. Here is how it works:

```
<?php  
$string = "An infinite number of monkeys";  
$newstring = str_replace("monkeys", "giraffes", $string);
```

```
print $newstring;  
?>
```

With that code, \$newstring will be printed out as "An infinite number of giraffes" - simple, really. Now consider this piece of code:

```
<?php  
$string = "An infinite number of monkeys";  
$newstring = str_replace("Monkeys", "giraffes", $string);  
print $newstring;  
?>
```

This time, \$newstring will not be "An infinite number of giraffes" as you might have expected - instead it will remain "An infinite number of monkeys". The reason for this is because the first parameter to str\_replace() is "Monkeys" rather than "monkeys", and PHP is case sensitive with strings!

There are two ways to fix the problem: either change the first letter of Monkeys to a lowercase M, or, if we're not sure which case we will find, we can switch to the case-insensitive version of str\_replace(): str\_ireplace().

```
<?php  
$string = "An infinite number of monkeys";  
$newstring = str_ireplace("Monkeys", "giraffes", $string);  
print $newstring;  
?>
```

This time around, we find that \$newstring is now "An infinite number of giraffes", as hoped. The key is that PHP will now replace "Monkeys", "monkeys", "MONKEYS", etc.

When used, the fourth parameter is passed by reference, and PHP will set it to be the number of times your string was found and replaced. Take a look at this example:

```
<?php  
$string = "He had had to have had it.";  
$newstring = str_replace("had", "foo", $string, $count);  
print "$count changes were made.\n";  
?>
```

That should output three, as PHP will replace "had" with "foo" three times.

### **Formatting String :**

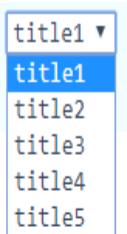
When you are using PHP to put a string on the page most of the time you will use the syntax echo, which will take the following string and display that string in HTML. You can even concatenate multiple strings or variables together to be outputted by the echo syntax.

Here is an example of using echo to output multiple options in a select box.

```
<?php  
$options = array('option1' => 'title1',  
'option2' => 'title2',  
'option3' => 'title3',  
'option4' => 'title4',  
'option5' => 'title5');  
echo '<select>';  
foreach($options as $key => $val)  
{  
echo '<option value="'. $key.'">' . $val . '</option>';  
}
```

```
echo '</select>';
```

```
?>
```

**Output:**

As you can see from the echo syntax that you can create a HTML element by using PHP. This code uses a concatenated string to insert the variables into the value attribute and a title inside the option tag. This is quite easy to read and is fine to use in normal development but what if you were using this method to display a more complicated element with more attributes such as an image tag.

This will create 5 image tags which will populate the number of different attributes.

```
<?php  
$options = array(  
    'image1' => array('class' => 'style', 'alt' => 'Image 1', 'title' => 'Image 1', 'width' => '30', 'height'  
    => '30', 'src' => 'image1.png'),  
    'image2' => array('class' => 'style', 'alt' => 'Image 2', 'title' => 'Image 2', 'width' => '40', 'height'  
    => '40', 'src' => 'image1.png'),  
    'image3' => array('class' => 'style', 'alt' => 'Image 3', 'title' => 'Image 3', 'width' => '50', 'height'  
    => '50', 'src' => 'image1.png'),  
    'image4' => array('class' => 'style', 'alt' => 'Image 4', 'title' => 'Image 4', 'width' => '60', 'height'  
    => '60', 'src' => 'image1.png'),
```

```
'image5' => array('class' => 'style', 'alt' => 'Image 5', 'title' => 'Image 5', 'width' => '70', 'height' => '70', 'src' => 'image1.png'),  
);  
foreach($options as $key => $val)  
{  
echo      '';  
}  
?>
```

### Output:



This time if you look at the echo syntax to output the image tags it's not very readable and can be hard to work out what we are joining together to create the image tag, especially with the different single and double quotes it becomes really hard to read.

The solution is to PHP built in functions sprintf() and printf() to create a formatted string to output in the HTML. Both these functions do the same thing but the printf() will echo the string directly and the sprintf() function will return a formatted string.

This is how it will be used to display the image tag.

### Printf() Function Displaying Image Tag:

```
<?php
$options = array(
'image1' => array('class' => 'style', 'alt' => 'Image 1', 'title' => 'Image 1', 'width' => '30', 'height' => '30', 'src' => 'image1.png'),
'image2' => array('class' => 'style', 'alt' => 'Image 2', 'title' => 'Image 2', 'width' => '40', 'height' => '40', 'src' => 'image1.png'),
'image3' => array('class' => 'style', 'alt' => 'Image 3', 'title' => 'Image 3', 'width' => '50', 'height' => '50', 'src' => 'image1.png'),
'image4' => array('class' => 'style', 'alt' => 'Image 4', 'title' => 'Image 4', 'width' => '60', 'height' => '60', 'src' => 'image1.png'),
'image5' => array('class' => 'style', 'alt' => 'Image 5', 'title' => 'Image 5', 'width' => '70', 'height' => '70', 'src' => 'image1.png'),
);
foreach($options as $key => $val)
{
printf('',
$val["class"],
$val["alt"],
$val["title"],
$val["width"],
$val["height"],
$val["src"]);
}
?>
```

**Output:**



### Sprintf() Function Displaying Image Tag :

```
<?php  
$options = array(  
    'image1' => array('class' => 'style', 'alt' => 'Image 1', 'title' => 'Image 1', 'width' => '30', 'height'  
    => '30', 'src' => 'image1.png'),  
    'image2' => array('class' => 'style', 'alt' => 'Image 2', 'title' => 'Image 2', 'width' => '40', 'height'  
    => '40', 'src' => 'image1.png'),  
    'image3' => array('class' => 'style', 'alt' => 'Image 3', 'title' => 'Image 3', 'width' => '50', 'height'  
    => '50', 'src' => 'image1.png'),  
    'image4' => array('class' => 'style', 'alt' => 'Image 4', 'title' => 'Image 4', 'width' => '60', 'height'  
    => '60', 'src' => 'image1.png'),  
    'image5' => array('class' => 'style', 'alt' => 'Image 5', 'title' => 'Image 5', 'width' => '70', 'height'  
    => '70', 'src' => 'image1.png'),  
);  
foreach($options as $key => $val)  
{  
    $image = sprintf('',  
        $val["class"],  
        $val["alt"],  
        $val["title"],  
        $val["width"],  
        $val["height"],
```

---

```
$val["src"]);  
echo $image;  
}  
?>
```

**Output:**

As you can see using the printf() and sprintf() functions allows you to much more readable code for outputting formatted strings

**String Related Library Function :**

The String related functions are

|                 |                                                                                              |
|-----------------|----------------------------------------------------------------------------------------------|
| strlen()        | It is used to return the length of a string.                                                 |
| strncasecmp()   | Binary safe case-insensitive string comparison                                               |
| strnatcasecmp() | It is used for case-insensitive comparison of two strings using a "natural order" algorithm  |
| strnatcmp()     | It is used for case-sensitive comparison of two strings using a "natural order" algorithm    |
| strcmp()        | It is used to compare of the first n characters.                                             |
| strpbrk()       | It is used to search a string for any of a set of characters.                                |
| stripos()       | It finds the position of the last occurrence of a case-insensitive substring in a string.    |
| strrpos()       | It finds the length of the last occurrence of a substring in a string.                       |
| strpos()        | It is used to return the position of the first occurrence of a string inside another string. |
| strrchr()       | It is used to find the last occurrence of a string inside another string.                    |
| strrev()        | It is used to reverse a string.                                                              |

Reference: [https://www.w3schools.com/php/php\\_string.asp](https://www.w3schools.com/php/php_string.asp)

### Strlen(string):

Strlen() displays the length of any string.

**Example:**

```
<?php  
echo strlen("Welcome to Strings");//will return the length of given string  
?>
```

**Output:**

18

**str\_word\_count() :**

str\_word\_count() function which enables display of the number of words in any specific string.

**Example:**

```
<?php  
echo str_word_count("Welcome to Strings");//will return the number of words in a string  
?>
```

**Output:**

3

**Strrev() :**

Strrev() is used for reversing a string. You can use this function to get the reverse version of any string.

**Example:**

```
<?php  
echo strrev("Welcome to Strings");// will return the string starting from the end  
?>
```

**Output:**

gnirts ot emocleW

# Arrays-Anatomy of Arras

## What is an Array?

- An array is a data structure that stores one or more similar type of values in a single value.
- Each array value is accessed using an ID which is called array index.
- By default the *index* always starts at zero.

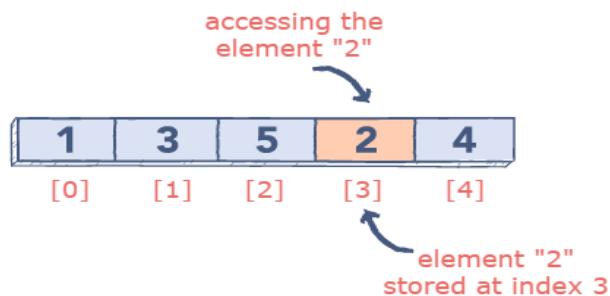


Image 9 – Array Indexing

Reference: <https://www.educative.io/edpresso/what-are-numeric-arrays-in-php>

## Advantages of arrays:

- Array can store multiple values within a single variable.
- The values of an array can be sorted easily.
- we can traverse all the elements of an array.
- Searching for an element is with arrays.
- There are more than 100 Library functions are there for arrays.

## Declaration of Array:

- An array is a data structure that stores one or more similar type of values in a single value.
- Each array value is accessed using an ID c which is called array index.
- There are three different kind of arrays
- Numeric Arrays ,Associative Arrays and Multidimensional Arrays.

```
<html>
  <body>
    <?php
      /* First method to create array. */
      $numbers = array( 1, 2, 3, 4, 5);
      echo "$numbers[0]<br>";
      echo "$numbers[1]<br>";
      echo "$numbers[2]<br>";
      /* Second method to create array. */
      $num[0] = "one";
      $num[1] = "two";
      echo "$num[0]<br>";
      echo "$num[1]<br>";
    ?>
  </body>
</html>
```

## Output:

1  
2

## Types of Arrays:

- There are three different kind of arrays
- Numeric Arrays -An array with a numeric index.

- 
- Associative Arrays-An array with strings as index.
  - Multidimensional Arrays-An array containing one or more arrays and values are accessed using multiple indices.

### Creating an Index based Array:

- These arrays can store numbers and strings.
- The index will be represented by numbers.
- By default array index starts from zero.
- **array()** function is used to create an array.

#### Example:

```
<html>
  <body>
    <?php
      /* First method to create array. */
      $numbers = array( 1, 2, 3, 4, 5);
      echo "$numbers[0]<br>";
      echo "$numbers[1]<br>";
      echo "$numbers[2]<br>";
      /* Second method to create array. */
      $num[0] = "one";
      $num[1] = "two";
      echo "$num[0]<br>";
      echo "$num[1]<br>";
    ?>
  </body>
</html>
```

#### Output:

1  
2  
3

---

one  
Two

## Creating an Associative Array:

- The associative arrays are very similar to numeric arrays in term of functionality.
- But they are different in terms of their index.
- Associative array will have their index as string.

```
<html>
<body>
<?php
/* First method to associate create array. */
$salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
echo "Salary of zara is ". $salaries['zara']. "<br />";
/* Second method to create array. */
$salaries['mohammad'] = "high";
$salaries['qadir'] = "medium";
$salaries['zara'] = "low";
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
echo "Salary of zara is ". $salaries['zara']. "<br />";
?>
</body>
</html>
```

## Output:

Salary of mohammad is 2000  
Salary of qadir is 1000  
Salary of zara is 500  
Salary of mohammad is high  
Salary of qadir is medium  
Salary of zara is low

## Creating Multi-Dimensional Array:

- A multi-dimensional array each element in the main array can also be an array.
- And each element in the sub-array can be an array.
- Values in the multi-dimensional array are accessed using multiple index.

```
<html>
  <body>
    <?php
      $marks = array(
        "mohammad" => array (
          "physics" => 35,
          "maths" => 30,
          "chemistry" => 39
        ),
        "qadir" => array (
          "physics" => 30,
          "maths" => 32,
          "chemistry" => 29)
      );
      /* Accessing multi-dimensional array values */
      echo "Marks for mohammad in physics : ";
      echo $marks['mohammad']['physics'] . "<br />";
      echo "Marks for qadir in maths : ";
      echo $marks['qadir']['maths'] . "<br />";
    ?>
  </body>
</html>
```

### Output:

Marks for mohammad in physics : 35  
Marks for qadir in maths : 32

## Looping with Index based array:

- An indexed array accessing by using loops.

- 
- We can use loops for index based array in two ways.
  - First by using for loop and secondly by using foreach.

```
<html></html>
<body>
<?php
// Creating an indexed array
$name_one = array("Zack", "Anthony", "Ram");

// Looping through an array usign foreach
echo "Looping using foreach: <br>";
foreach ($name_one as $val){
    echo $val. "<br>";
}
// count() function is used to count
// the number of elements in an array
$round = count($name_one);
echo "<br>number of elements are $round <br>";
//loop through the array using for
echo "Looping using for: <br>";
for($n = 0; $n < $round; $n++){
    echo $name_one[$n], "<br>";
}
?>
</body>
</html>
```

**Output:**

Zack  
Anthony  
Ram

**Looping with associated array:**

- An indexed array accessing by using loops.
- We can use loops for index based array in two ways.
- First by using for loop and secondly by using foreach.

### Looping with associated array using each() :

- The each() function is an inbuilt function in PHP
- It is used to get the current element key-value pair of the given .
- After returning the key and value of the current element pointer is incremented by one in the array.

#### Example:

```
<html>
<body>
<?php
$arr = array('a' => 'anny', 'b' => 'bunny');
while (list($key, $val) = each($arr))
{
    echo "$key => $val ";
}
?>
</body>
</html>
```

#### Output:

anny  
bunny

### Looping with associated array using foreach() :

- The foreach loop is mainly used for looping through the values of an array.

- 
- It loops over the array, and each value for the current array element is assigned to \$value.
  - The array pointer is advanced by one to go the next element in the array.

**Example:**

```
<html>
<body>
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
</body>
</html>
```

**Output:**

35  
37  
43

**Library functions for Arrays :**

- The Library functions allow you to access and manipulate arrays.
- Simple and multi-dimensional arrays are supported.
- There 60 functions that can support for arrays in PHP.
- Example: sizeof() , sort() , usort() ...., etc

|                      |                                                                   |
|----------------------|-------------------------------------------------------------------|
| <u>natcasesort()</u> | Sorts an array using a case insensitive "natural order" algorithm |
| <u>natsort()</u>     | Sorts an array using a "natural order" algorithm                  |
| <u>next()</u>        | Advance the internal array pointer of an array                    |
| <u>pos()</u>         | Alias of <u>current()</u> .                                       |
| <u>prev()</u>        | Rewinds the internal array pointer                                |
| <u>range()</u>       | Creates an array containing a range of elements                   |
| <u>reset()</u>       | Sets the internal pointer of an array to its first element        |
| <u>rsort()</u>       | Sorts an indexed array in descending order                        |
| <u>shuffle()</u>     | Shuffles an array                                                 |
| <u>sizeof()</u>      | Alias of <u>count()</u> .                                         |
| <u>sort()</u>        | Sorts an indexed array in ascending order                         |
| <u>uasort()</u>      | Sorts an array by values using a user-defined comparison function |
| <u>uksort()</u>      | Sorts an array by keys using a user-defined comparison function   |
| <u>usort()</u>       | Sorts an array using a user-defined comparison function           |

Reference: [https://www.w3schools.com/php/php\\_ref\\_array.asp](https://www.w3schools.com/php/php_ref_array.asp)

## **sizeof(\$arr)**

sizeof() function returns the size of the array or the number of data elements stored in the array.

### **Example:**

```
<html>
<body>
<?php
$lambdaorghinis = array("Urus", "Huracan", "Aventador");
echo "Size of the array is: ". sizeof($lambdaorghinis);
?>
</body>
</html>
```

### **Output:**

---

3

### **is\_array(\$arr) :**

is\_array(\$arr) is to check whether the provided data is in form of an array, It returns True if the variable is an array and returns False otherwise.

#### **Example:**

```
<html>
<body>
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// using ternary operator
echo is_array($lamborghinis) ? 'Array' : 'not an Array';
echo "<br>";
$mycar = "Urus";
// using ternary operator
echo is_array($mycar) ? 'Array' : 'not an Array';
?>
</body>
</html>
```

#### **Output:**

Array  
Not an array

### **in\_array(\$var, \$arr):**

in\_array(\$var, \$arr) is used to check whether a certain value is present in the array or not.

#### **Example:**

```
<html>
<body>
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// new concept car by lamborghini|
$concept = "estoque";
echo in_array($concept, $lamborghinis) ? 'Added to the Lineup' : 'Not yet!'
?>
</body>
</html>
```

---

**Output:**

Not yet!

**print\_r(\$arr) :**

print\_r(\$arr) is a function to print the array in the most descriptive way possible. This function prints the complete representation of the array, along with all the keys and values.

**Example:**

```
<html>
<body>
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
print_r($lamborghinis);
?>
</body>
</html>
```

Output:

```
Array ([0] => Urus [1] => Huracan [2] => Aventador )|
```

**array\_merge(\$arr1, \$arr2) :**

array\_merge(\$arr1, \$arr2) is to combine two different arrays into a single array. It can combine same type(indexed, associative etc) or different types of arrays into one single array.

**Example:**

```
<html>
<body>
<?php
$hatchbacks = array(
    "Suzuki" => "Baleno",
    "Skoda" => "Fabia"
);
// friends who own the above cars
$friends = array("Vinod", "Javed");
// let's merge the two arrays into one
$merged = array_merge($hatchbacks, $friends);
print_r($merged);
?>
</body>
</html>
```

Output:

```
Array () [Suzuki] => Baleno [Skoda] => Fabia [0] => Vinod [1] => Javed )|
```

### **array\_values() :**

- In an array, data is stored in form of key-value pairs, where key can be numerical or strings.
- If we want to take all the values from our array, without the keys.
- And store them in a separate array, then we can use array\_values() function.

### **Example:**

```
<html>
<body>
<?php
$hatchbacks = array(
    "Suzuki" => "Baleno",
    "Skoda" => "Fabia"
);
// friends who own the above cars
$friends = array("Vinod", "Javed");
// let's merge the two arrays into one
$merged = array_merge($hatchbacks, $friends);
//getting only the values
$merged = array_values($merged);
print_r($merged);
?>
</body>
</html>
```

Output:

```
Array ([0] => Baleno [1] => Fabia [2] => Vinod [3] => Javed )
```

### **array\_keys(\$arr) :**

array\_keys(\$arr) is function to extract extract the keys from an array.

#### **Example:**

```
<html>
<body>
<?php
$data = array("hero" => "Shubhneet", "villain" => "Karan");
print_r(array_keys($data));
?>
</body>
</html>
```

Output:

```
Array ([0] => hero [1] => villain )
```

### **array\_pop(\$arr) :**

- array\_pop(\$arr) removes the last element of the array.
- Hence it can be used to remove one element from the end.

#### **Example:**

```
<html>
<body>
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// removing the last element
array_pop($lamborghinis);
print_r($lamborghinis);
?>
</body>
</html>
```

Output:

```
Array () [0] => Urus [1] => Huracan )|
```

### **array\_push(\$arr, \$val)**

- array\_push(\$arr, \$val) function is the opposite of the array\_pop() function.
- This can be used to add a new element at the end of the array.

#### **Example:**

```
<html>
<body>
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// adding a new element at the end
array_push($lamborghinis, "Estoque");
print_r($lamborghinis);
?>
</body>
</html>
```

Output:

```
Array () [0] => Urus [1] => Huracan [2] => Aventador [3] => Estoque )|
```

### **array\_shift(\$arr) :**

- array\_shift(\$arr) is used to remove/shift the first element out of the array.

- 
- So, it is just like array\_pop() function but different in terms of the position of the element removed.

**Example:**

```
<html>
<body>
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// removing the first element
array_shift($lamborghinis);
print_r($lamborghinis);
?>
</body>
</html>
```

Output:

```
Array ([0] => Huracan [1] => Aventador )
```

**Sort(\$arr) :**

- sort(\$arr) function sorts the array elements in ascending order.
- In case of a string value array, values are sorted in ascending alphabetical order.
- Some other sorting functions are: asort(), arsort(), ksort(), krsort() and rsort().

**Example:**

```
<html>
<body>
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// sort the array
sort($lamborghinis);
print_r($lamborghinis);
?>
</body>
</html>
```

Output:

```
Array ([0] => Aventador [1] => Huracan [2] => Urus )
```

**Array\_flip(\$arr) :**

---

Array\_flip(\$arr) is function to interchange the keys and the values of a PHP associative array.

**Example:**

```
<html>
<body>
<?php
$hatchbacks = array(
    "Suzuki" => "Baleno",
    "Skoda" => "Fabia",
);
// we can directly print the result of array flipping
print_r(array_flip($hatchbacks));
?>
</body>
</html>
```

Output:

```
Array [0] [Baleno] => Suzuki [Fabia] => Skoda )|
```

**array\_reverse(\$arr) :**

- array\_reverse(\$arr) function is used to reverse the order of elements.
- Making the first element last and last element first.
- And similarly rearranging other array elements

**Example:**

```
<html>
<body>
<?php
$num = array(10, 20, 30);
// printing the array after reversing it
print_r(array_reverse($num));
?>
</body>
</html>
```

Output:

```
Array [0] => 30 [1] => 20 [2] => 10 )|
```

**array\_rand(\$arr) :**

- 
- array\_rand(\$arr) function is used to pick random data element from an array.
  - This function randomly selects one element from the given array and returns it.
  - In case of indexed array, it will return the index of the element,
  - In case of associative array, it will return the key of the selected random element.

**Example:**

```
<html>
<body>
<?php

$colors = array("red", "black", "blue", "green");

echo "Color of the day: ". $colors[array_rand($colors)];

?>
</body>
</html>

Output:
Color of the day: green|
```

**array\_slice(\$arr, \$offset, \$length) :**

- array\_slice(\$arr, \$offset, \$length) function is used to create a subset of any array.
- Using this function, we define the starting point(\$offset, which is the array index from where the subset starts).
- The length(or, the number of elements required in the subset starting from the offset).

**Example:**

---

```
<html>
<body>
<?php
$colors = array("red", "black", "blue", "green");
print_r(array_slice($colors, 1, 3));
?>
</body>
</html>

Output:
Array ([0] => black [1] => blue [2] => green )|
```

### Array\_unique() :

Array\_unique() function is used to remove duplicate values from an array.

#### Example:

```
<html>
<body>
<?php
$a=array("a"=>"red","b"=>"green","c"=>"red");
print_r(array_unique($a));
?>
</body>
</html>
```

Output:

```
Array ([a] => red [b] => green )
```

### Array\_search(\$value,\$array)

Array\_search(\$value,\$array) function used to search an array for the value and return its key.

#### Example:

```
<html>
<body>
<?php
$a=array("a"=>"red","b"=>"green","c"=>"blue");
echo array_search("red",$a);
?>
</body>
</html>
```

Output:

```
a
```

---

### Each():

- The **each()** function is an inbuilt function used to get the current element key-value which the internal pointer is currently pointing.
- After returning the key and value of the current element the internal pointer is incremented by one in the array.

### Example:

```
<html>
<body>
<?php
$arr = array('maya', 'Sham', 'Geet');
print_r (each($arr));
?>
</body>
</html>
```

Output:

```
Array ([1] => maya [value] => maya [0] => 0 [key] => 0 )|
```

---

# Working with file and Directories

## Understanding File & Directory

### Concept of file and directory

A *file* is a collection of data that is stored on disk and that can be manipulated as a single unit by its name. A *directory* is a file that acts as a folder for other files. A directory can also contain other directories (*subdirectories*); a directory that contains another directory is called the *parent* directory of the directory it contains.

### Reading the Contents of a Directory

We need three functions to perform this task: opendir(), readdir() and closedir(). The opendir() function takes one parameter, which is the directory we want to read, and returns a directory handle to be used in subsequent readdir() and closedir() calls. opendir() returns False if the directory could not be opened.

The readdir() function takes one parameter, which is the manage the opendir() returned and each time we call readdir() it returns the filename of the next file in the directory. readdir() returns False if the end of the directory has been reached. Note that readdir() returns only the names of its items, rather than full paths.

### Dealing With Directories

- Open a directory
  - `$handle = opendir('dirname');`
    - \$handle 'points' to the directory
- Read contents of directory
  - `readdir($handle)`
    - Returns name of next file in directory
    - Files are sorted as on filesystem
- Close a directory
  - `closedir($handle)`
    - Closes directory 'stream'



---

Image 1 – Dealing with directories  
Reference - <https://slideplayer.com/slide/7482728/>

## PHP opendir() function

The opendir() function in PHP is an inbuilt function which is used to open a directory handle. The path of the directory to be opened is sent as a parameter to the opendir() function and it returns a directory handle resource on success, or FALSE on failure.

Syntax: *opendir(\$path, \$context)*

Parameters Used: The opendir() function in PHP accepts two parameters.

**\$path** : It is a mandatory parameter which specifies the path of the directory to be opened.

**\$context** : It is an optional parameter which specifies the behavior of the stream.

```
1  <?php
2
3  // Opening a directory
4  $dir_handle = opendir("/user/gfg/docs/");
5
6  if(is_resource($dir_handle))
7  {
8      echo ("Directory Opened Successfully.");
9  }
10
11 // closing the directory
12 closedir($dir_handle);
13
14 else
15 {
16     echo ("Directory Cannot Be Opened.");
17 }
18
19 ?>
```

Image 2 – opendir()  
Reference - <https://www.geeksforgeeks.org/php-opendir-function/>

```
1 <?php
2
3 // opening a directory and reading its contents
4 $dir_handle = opendir("user/gfg/sample.docx");
5
6 if(is_resource($dir_handle))
7 {
8     while(($file_name = readdir($dir_handle)) == true)
9     {
10         echo("File Name: " . $file_Name);
11         echo "<br>";
12     }
13
14     // closing the directory
15     closedir($dir_handle);
16 }
17 else
18 {
19     echo("Directory Cannot Be Opened.");
20 }
21 ?>
```

Image 3 – opendir()

Reference - <https://www.geeksforgeeks.org/php-opendir-function/>

## PHP readdir() function

The readdir() function in PHP is an inbuilt function which is used to return the name of the next entry in a directory. The method returns the filenames in the order as they are stored in the file name system.

Syntax: *readdir(dir\_handle)*

Parameters Used: The readdir() function in PHP accepts one parameter.

**dir\_handle** : It is a mandatory parameter which specifies the handle resource previously opened by the opendir() function.

```
1 <?php
2
3 // opening a directory
4 $dir_handle = opendir("user/gfg/");
5
6 // reading the contents of the directory
7 while(($file_name = readdir($dir_handle)) != false)
8 {
9 echo("File Name: " . $file_name);
10 echo "<br>" ;
11 }
12
13 // closing the directory
14 closedir($dir_handle);
15 ?>
```

Image 4 – readdir()

Reference - <https://www.geeksforgeeks.org/php-readdir-function>

```
1 <?php
2
3 // opening a directory
4 $dir_handle = opendir("user/gfg/");
5
6 if(is_resource($dir_handle))
7 {
8
9 // reading the contents of the directory
10 while(($file_name = readdir($dir_handle)) != false)
11 {
12 echo("File Name: " . $file_name);
13 echo "<br>" ;
14 }
15
16 // closing the directory
17 closedir($dir_handle);
18 }
19 else
20 {
21 echo("Failed to Open.");
22 }
23
24 else
25 {
26 echo("Invalid Directory.");
27 }
28 ?>
```

Image 5 – readdir()

Reference - <https://www.geeksforgeeks.org/php-readdir-function>

## PHP closedir() function

The directory handle to be closed is sent as a parameter to the closedir() function and the closedir() closes the directory handle. The directory handle must be previously opened by the opendir() function.

Syntax: *closedir(\$dir\_handle)*

Parameters Used: The closedir() function in PHP accepts only one parameter as described below.

**\$dir\_handle:** It is an optional parameter which specifies the directory handle resource previously opened with opendir(). If this parameter is not specified, the last link opened by opendir() is assumed and closed by closedir().

```
1 <?php
2
3 // Opening a directory
4 $dir_handle = opendir("/user/gfg/docs/");
5
6 if(is_resource($dir_handle))
7 {
8     echo("Directory Opened Successfully.");
9
10    // closing the directory
11    closedir($dir_handle);
12 }
13 else
14 {
15     echo("Directory Cannot Be Opened.");
16 }
17
18 ?>
```

Image 6 – closedir()  
Reference - <https://www.geeksforgeeks.org/php-closedir-function/>

```
1 <?php
2
3 // opening a directory and reading its contents
4 $dir_handle = opendir("user/gfg/sample.docx");
5
6 if(is_resource($dir_handle))
7 {
8     while((($file_name = readdir($dir_handle)) == true)
9     {
10         echo("File Name: " . $file_Name);
11         echo "<br>" ;
12     }
13
14     // closing the directory
15     closedir($dir_handle);
16 }
17 else
18 {
19     echo("Directory Cannot Be Opened.");
20 }
21
22 ?>
```

Image 7 – closedir()

Reference - <https://www.geeksforgeeks.org/php-closedir-function/>

## File Opening Modes

Before we look at how to open a file in PHP you need to know that a file can be opened in different modes. For example, you can open a file in read-only mode or in reading and write modes. Take a look at the table below for the different modes.

### File open modes

| Modes | Description                                                                                                                                                         |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| r     | <b>Open a file for read only.</b> File pointer starts at the beginning of the file                                                                                  |
| w     | <b>Open a file for write only.</b> Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file      |
| a     | <b>Open a file for write only.</b> The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |
| x     | <b>Creates a new file for write only.</b> Returns FALSE and an error if file already exists                                                                         |
| r+    | <b>Open a file for read/write.</b> File pointer starts at the beginning of the file                                                                                 |
| w+    | <b>Open a file for read/write.</b> Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file      |
| a+    | <b>Open a file for read/write.</b> The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |
| x+    | <b>Creates a new file for read/write.</b> Returns FALSE and an error if file already exists                                                                         |



Image 8 – File Opening Modes

Reference - <https://www.slideshare.net>

# Opening, reading and closing a file

## Opening a File

The *fopen()* function in PHP is an inbuilt function which is used to open a file or an URL. The filename and mode to be checked are sent as parameters to the *fopen()* function and it returns a file pointer resource if a match is found and a False on failure.

Syntax: *resource fopen ( \$file, \$mode, \$include\_path, \$context)*

Parameters Used: The *fopen()* function in PHP accepts four parameters.

**\$file:** It is a mandatory parameter which specifies the file.

**\$mode:** It is a mandatory parameter which specifies the access type of the file or stream.

**\$include\_path:** It is an optional parameter which is set to 1 if you want to search for the file in the include\_path (Ex. php.ini).

**\$context:** It is an optional parameter which is used to set the behavior of the stream.

```
1 <?php
2 // Opening a file using fopen()
3 // function in read only mode
4 $myfile = fopen("/home/geeks/gfg.txt", "r")
5 or die("File does not exist!");
6 ?>
```

Image 9 – Opening a file

Reference - <https://www.geeksforgeeks.org/php-fopen-function-open-file-or-url/>

```
1 <?php
2 // Opening a file using fopen()
3 // function in read/write mode
4 $myfile = fopen("gfg.txt", 'r+')
5     or die("File does not exist!");
6
7 $pointer = fgets($myfile);
8 echo $pointer;
9 fclose($myfile);
10 ?>
```

Image 10 – Opening a file

Reference - <https://www.geeksforgeeks.org/php-fopen-function-open-file-or-url/>

## Reading a File

The fread() function reads from an open file. The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

Syntax: *string fread ( \$file, \$length )*

Parameters Used: The *fread()* function in PHP accepts two parameters.

**\$file:** It is a mandatory parameter which specifies the file.

**\$length:** It is a mandatory parameter which specifies the maximum number of bytes to be read.

```
1  <?php
2  // Opening a file
3  $myfile = fopen("gfg.txt", "r");
4
5  // reading 13 bytes from the file
6  // using fread() function
7  echo fread($myfile, "13");
8
9  // closing the file
10 fclose($myfile);
11 ?>
```

Image 11 – Reading a file

Reference - <https://www.geeksforgeeks.org/php-fread-function/>

```
1  <?php
2  // Opening a file
3  $myfile = fopen("gfg.txt", "r");
4
5  // reading the entire file using
6  // fread() function
7  echo fread($myfile, filesize("gfg.txt"));
8
9  // closing the file
10 fclose($myfile);
11 ?>
```

Image 12 – Reading a file

Reference - <https://www.geeksforgeeks.org/php-fread-function/>

## Closing a File

The fclose() function in PHP is an inbuilt function which is used to close a file which is pointed by an open file pointer. The fclose() function returns true on success and false on failure. It takes the file as an argument which has to be closed and closes that file.

Syntax: *bool fclose( \$file )*

Parameters: The fclose() function in PHP accepts only one parameter which is \$file. This parameter specifies the file which has to be closed.

```
1 <?php
2
3 // opening a file using fopen() function
4 $check = fopen("gfg.txt", "r");
5
6 // closing a file using fclose() function
7 fclose($check);
8
9 ?>
```

Image 12 – Closing a file

Reference - <https://www.geeksforgeeks.org/php-fclose-function/>

```
1 <?php
2
3 // a file is opened using fopen() function
4 $check = fopen("singleline.txt", "r");
5 $seq = fgets($check);
6
7 // Outputs a line of the file until
8 // the end-of-file is reached
9 while(! feof($check))
10 {
11 echo $seq ;
12 $seq = fgets($check);
13 }
14
15 // the file is closed using fclose() function
16 fclose($check);
17
18 ?>
```

Image 13 – Closing a file

Reference - <https://www.geeksforgeeks.org/php-fclose-function/>

# Copying, renaming and deleting a file

## PHP copying a file

The copy() function in PHP is an inbuilt function which is used to make a copy of a specified file. The copy() function returns true on success and false on failure.

Syntax: *bool copy ( \$source, \$dest )*

Parameters: The copy() function in PHP accepts two parameters which are source and destination.

**\$source:** It specifies the path to the source file.

**\$dest:** It is used to specify the path to the destination file.

```
1 <?php
2
3 // Copying gfg.txt to geeksforgeeks.txt
4 echo copy("gfg.txt", "geeksforgeeks.txt");
5
6 ?>
```

Image 14 – Copying a file

Reference - <https://www.geeksforgeeks.org/php-copy-function/>

```
1 <?php
2
3 // Copying gfg.txt to geeksforgeeks.txt
4 $srcfile = '/user01/Desktop/admin/gfg.txt';
5 $destfile = 'user01/Desktop/admin/geeksforgeeks.txt';
6
7 if (!copy($srcfile, $destfile)) {
8     echo "File cannot be copied! \n";
9 }
10 else {
11     echo "File has been copied!";
12 }
13
14 ?>
```

Image 15 – Copying a file

Reference - <https://www.geeksforgeeks.org/php-copy-function/>

## PHP renaming a file

The rename() function in PHP is an inbuilt function which is used to rename a file or directory. If the new name specified by the user already exists, the rename() function overwrites it.

Syntax: *rename(oldname, newname, context)*

Parameters Used: The rename() function in PHP accepts three parameter.

**oldname :** It is a mandatory parameter which specifies the old name of the file or directory.

**newname :** It is a mandatory parameter which specifies the new name of the file or directory.

**context :** It is an optional parameter which specifies the behavior of the stream .

```
1 <?php
2 // Old Name Of The file
3 $old_name = "gfg.txt" ;
4
5 // New Name For The File
6 $new_name = "newgfg.txt" ;
7
8 // using rename() function to rename the file
9 rename( $old_name, $new_name) ;
10
11 ?>
```

Image 16 – Renaming a file

Reference - <https://www.geeksforgeeks.org/php-rename-function/>

```
1 <?php
2 // Old Name Of The file
3 $old_name = "gfg.txt" ;
4
5 // New Name For The File
6 $new_name = "newgfg.txt" ;
7
8 // Checking If File Already Exists
9 if(file_exists($new_name))
10 {
11     echo "Error While Renaming $old_name" ;
12 }
13 else
14 {
15     if(rename( $old_name, $new_name))
16     {
17         echo "Successfully Renamed $old_name to $new_name" ;
18     }
19     else
20     {
21         echo "A File With The Same Name Already Exists" ;
22     }
23 }
24 ?>
```

Image 17 – Renaming a file

Reference - <https://www.geeksforgeeks.org/php-rename-function/>

## PHP deleting a file

Deleting a file means completely erase a file from a directory so that the file is no longer exist. PHP has an PHP has an unlink() function that allows to delete a file that allows to delete a file.

Syntax: unlink( \$filename, \$context );

The PHP unlink() function takes two parameters *\$filename* and *\$context*.

```
1 <?php
2 // PHP program to delete a file named gfg.txt
3 // using unlike() function
4
5 $file_pointer = "test.txt";
6
7 // Use unlink() function to delete a file
8 if (!unlink($file_pointer)) {
9     echo ("$file_pointer cannot be deleted due to an error");
10 }
11 else {
12     echo ("$file_pointer has been deleted");
13 }
14 ?>
```

Image 18 – Deleting a file

Reference - <https://www.geeksforgeeks.org/how-to-delete-a-file-using-php/>

```
1 <?php
2 // PHP program to delete a file named gfg.txt
3 // using unlike() function
4
5 $file_pointer = fopen('gfg.txt', 'w+');
6
7 // writing on a file named gfg.txt
8 fwrite($file_pointer, 'A computer science portal for geeks!');
9 fclose($file_pointer);
10
11 // Use unlink() function to delete a file
12 if (!unlink($file_pointer)) {
13     echo ("$file_pointer cannot be deleted due to an error");
14 }
15 else {
16     echo ("$file_pointer has been deleted");
17 }
18
19 ?>
```

Image 19 – Deleting a file

Reference - <https://www.geeksforgeeks.org/how-to-delete-a-file-using-php/>

---

# Working with directories

## Reading the Contents of a Directory

Three functions to perform this task: opendir(), readdir() and closedir(). Note that readdir() returns only the names of its items, rather than full paths.

```
<?php
// open the current directory
$chandle = opendir('.');
// define an array to hold the files
$files = array();
if ($chandle) {
    // loop through all of the files
    while (false !== ($fname = readdir($chandle))) {
        // if the file is not this file, and does not start with a '.' or '..',
        // then store it for later display
        if (($fname != '.') && ($fname != '..') &&
            ($fname != basename($_SERVER['PHP_SELF']))) {
            // store the filename
            $files[] = (is_dir( "./$fname" )) ? "(Dir) {$fname}" : $fname;
        }
    }
    // close the directory
    closedir($chandle);
}
echo "<select name=\"file\">\n";
// Now loop through the files, echoing out a new select option for each one
foreach( $files as $fname )
{
    echo "<option>{$fname}</option>\n";
}
echo "</select>\n";
?>
```

Image 20 – Reading contents of a directory

## Deleting the Directory and Its Contents

The rmdir() function in PHP is an inbuilt function which is used to remove an empty directory. The directory to be deleted is sent as a parameter to the rmdir() function and it returns True on success or False on failure.

Syntax: rmdir(dirname, context)

Parameters Used: The rmdir() function in PHP accepts two parameters.

**dirname** : It is a mandatory parameter which specifies the directory to be deleted.

**context** : It is an optional parameter which specifies the behavior of the stream .

```
1 <?php
2 // creating a directory named gfg
3 mkdir('gfg');
4 $dirname= "gfg";
5
6 // removing directory using rmdir()
7 rmdir($dirname);
8 ?>
```

Image 21 – Deleting a directory

Reference - <https://www.geeksforgeeks.org/php-rmdir-function/>

```
1 <?php
2 // creating a directory named gfg
3 $dirname = "gfg";
4
5 // removing directory using rmdir()
6 if(rmdir($dirname))
7 {
8 echo ("$dirname successfully removed");
9 }
10 else
11 {
12 echo ($dirname . "couldn't be removed");
13 }
14 ?>
```

---

Image 22 – Deleting a directory

Reference - <https://www.geeksforgeeks.org/php-rmdir-function/>

# Creating New Directories

The mkdir() creates a new directory with the specified pathname. The path and mode are sent as parameters to the mkdir() function and it returns TRUE on success or FALSE on failure. The mode parameter in mkdir() function is ignored on Windows platforms.

Syntax: mkdir(path, mode, recursive, context)

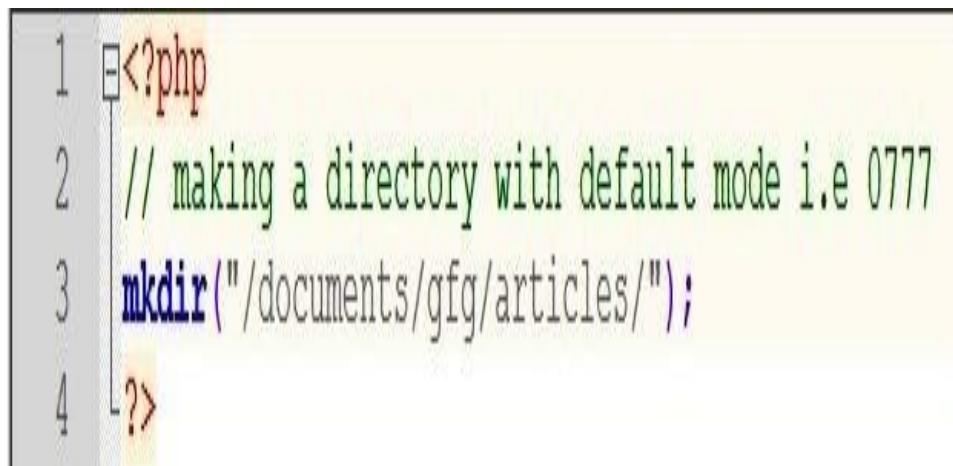
Parameters Used: The mkdir() function in PHP accepts four parameters.

**path :** It is a mandatory parameter which specifies the path.

**mode :** It is an optional parameter which specifies permission.

**recursive:** It is an optional parameter which can be used to set recursive mode.

**context :** It is an optional parameter which specifies the behavior of the stream.



```
1 <?php
2 // making a directory with default mode i.e 0777
3 mkdir("/documents/gfg/articles/");
4 ?>
```

Image 23 – Making a directory

Reference - <https://www.geeksforgeeks.org/php-mkdir-function/>

```
1 <?php
2 $nest = './node1/node2/node3/';
3 // creating a nested structure directory
4 if (!mkdir($nest, 0777, true))
5 {
6     echo('Folders cannot be created recursively');
7 }
8 else
9 {
10    echo('Folders created recursively');
11 }
12 ?>
```

Image 24 – Making a directory  
Reference - <https://www.geeksforgeeks.org/php-mkdir-function/>

---

# Building a text editor File: Uploading & Downloading

## Building a text editor File

The first thing that must be done is define the file(s). An array is used so that both files can be identified and called through \$files and later through the \$file. A value is being defined which will help pull the text from the respective submit buttons in the form fields.

```
<form name="texteditor" method="post" action="">
<div id="titleinput"><input name="edittext" value=<?php
$file = fopen("title.txt", "r");
while(! feof($file)){
echo fgets($file). "";
}
fclose($file);
?>" /></div>
<input type="submit" name="save" value="Save Title" />
</form>
```

Image 25 – Building text editor file

## Upload a file to server

A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script. Information in the **phpinfo.php** page describes the temporary directory that is used for file uploads as **upload\_tmp\_dir** and the maximum permitted size of files that can be uploaded is stated as **upload\_max\_filesize**. These parameters are set into PHP configuration file **php.ini**. The process of uploading a file follows these steps –

- The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- The user clicks the browse button and selects a file to upload from the local PC.

- 
- The full path to the selected file appears in the text field then the user clicks the submit button.
  - The selected file is sent to the temporary directory on the server.
  - The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
  - The PHP script confirms the success to the user.

As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail. An uploaded file could be a text file or image file or any document.

`$_FILES` array is used to upload files.

### Uploading & Downloading

Using `$_FILES` array to upload files.

`$_FILES["file"]["name"]` - uploaded file name

`$_FILES["file"]["type"]` - uploaded file type

`$_FILES["file"]["size"]` - uploaded file size

`$_FILES["file"]["tmp_name"]` - temporary file name stored on the server

`$_FILES["file"]["error"]` - the error code from the file uploading

Image 26 – Uploading & Downloading

Reference <https://www.etutorialspoint.com/index.php/16-how-to-get-current-filename-directory-linenumber-in-php>

## Download files in PHP

The Content-Disposition header function is used in PHP to supply a recommended filename and force the browser to display the save dialog. Before downloading the files, file types are checked. Normally, you don't necessarily need to use any server side scripting language like PHP to download images, zip files, pdf documents, exe files, etc. If such kind of file is stored in a public accessible folder, you can just create a hyperlink pointing to that file, and whenever a user click on the link, browser will automatically downloads that file. Clicking a link that points to a PDF or an Image file will not cause it to download to your hard drive directly. It will only open the

---

file in your browser. Further you can save it to your hard drive. However, zip and exe files are downloaded automatically to the hard drive by default.

Use the Content-Disposition header function in PHP to supply a recommended filename and force the browser to display the save dialog. Below is the example to download files. Before download files, file types are checked. Example using the download script.

```
<a href="download.php?file=some_file.pdf">Download here</a>

download.php download files

<?php

if (isset($_GET['file'])) {

$file = $_GET['file'];

$download_dir = 'C:/xampp/htdocs/test/file_uploads';

$fullPath = $download_dir.$file;

downloadFile( $fullPath );

}

} else {

header("HTTP/1.0 404 Not Found");

echo "<h1>Error 404: File Not Found: <br /><em>$file</em></h1>";

}

function downloadFile( $fullPath ){

// Must be fresh start

if( headers_sent() )

die('Headers Sent');
```

---

```
// Required for some browsers

if(ini_get('zlib.output_compression'))
    ini_set('zlib.output_compression', 'Off');

// File Exists?

if( file_exists($fullPath) ){

// Parse Info / Get Extension

$fsiz = filesize($fullPath);

$path_parts = pathinfo($fullPath);

$ext = strtolower($path_parts["extension"]);

// Determine Content Type

switch ($ext) {

case "pdf": $ctype="application/pdf"; break;
case "exe": $ctype="application/octet-stream"; break;
case "zip": $ctype="application/zip"; break;
case "doc": $ctype="application/msword"; break;
case "xls": $ctype="application/vnd.ms-excel"; break;
case "ppt": $ctype="application/vnd.ms-powerpoint"; break;
case "gif": $ctype="image/gif"; break;
case "png": $ctype="image/png"; break;
case "jpeg":
case "jpg": $ctype="image/jpg"; break;
```

```
default: $ctype="application/force-download";  
}  
  
header("Pragma: public"); // required  
  
header("Expires: 0");  
  
header("Cache-Control: must-revalidate, post-check=0, pre-check=0");  
  
header("Cache-Control: private",false); // required for certain browsers  
  
header("Content-Type: $ctype");  
  
header("Content-Disposition: attachment; filename=\"".basename($fullPath)."\";");  
  
header("Content-Transfer-Encoding: binary");  
  
header("Content-Length: ".$filesize);  
  
ob_clean();  
  
flush();  
  
readfile( $fullPath );  
  
} else  
  
die('File Not Found');  
  
}  
  
?>
```

### **Using query string (URL rewriting)**

#### **Passing a query string to a page**

There are two ways to pass a QueryString to a page, the first is to enter it manually. An alternative, and perhaps more useful, the way is to use HTML forms. The main thing to

---

remember with forms is that you need to use the GET method to send information via the query string.

```
<form method="GET" action="page.php">
Please enter your name: <input type="text" name="username" />
<input type="submit" value="submit" />
</form>
```

Image 27 – Passing query string to a page

### Accessing a query string element in a PHP page

In PHP, all the information passed via the query string is held in the `$_GET` super global array. To access an item, type `$_GET['varName']`, where `varName` is the name of the variable in the query string.

```
<?php
if($_GET['username'] == "") {
    // no username entered
    echo "You did not enter a name.";
} else {
    echo "Hello, " . $_GET['username'];
}
?>
<form method="GET" action="page.php">
Please enter your name: <input type="text" name="username" />
<input type="submit" value="submit" />
</form>
```

Image 28 – Accessing query string element

### Using Hidden field

A hidden field is not shown on the page. It simply keeps the text value specified in the `value` attribute. Hidden fields are good for passing additional information from the form to the server.

```

<?php
$num_to_guess = 42;
$message = "";
if (! isset ( $_POST ['guess'] )) {
$message = "Welcome!";
} else if ( $_POST ['guess'] > $num_to_guess) {
$message = $_POST ['guess'] . " is too big!";
} else if ( $_POST ['guess'] < $num_to_guess) {
$message = $_POST ['guess'] . " is too small!";
} else {
$message = "Well done!";
}
$guess = ( int ) $_POST ['guess'];
$num_tries = ( int ) $_POST ['num_tries'];
$num_tries++;
?>
<html>
<body>
<?php print $message?>
Guess number: <?php print $num_tries?><br />
<form method="post" action="<?php
print $_SERVER ['PHP_SELF']?>">
<input type="hidden" name="num_tries"
value="<?php
print $num_tries?>" /> Type your guess here: <input type="text" name="guess"
value="<?php
print $guess?>" />
</form>
</body>
</html>

```

Image 29 – Using hidden field

## Using cookies

### The Anatomy of a cookie

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies. Cookies are usually set in an HTTP header.

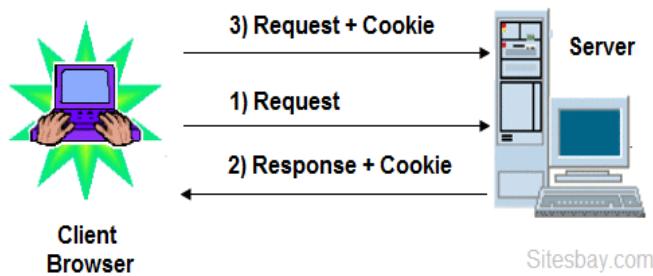


Image 30 – Cookie anatomy

Reference - <https://www.sitesbay.com/php/php-cookies-in-php>

### Setting Cookies with PHP

---

PHP provided setcookie() function to set a cookie. This function requires up to six arguments and should be called before <html> tag. For each cookie, this function has to be called separately.

Syntax: setcookie(name, value, expire, path, domain, security);

**Parameters:** The setcookie() function requires six arguments in general which are:

1. **Name:** It is used to set the name of the cookie.
2. **Value:** It is used to set the value of the cookie.
3. **Expire:** It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed.
4. **Path:** It is used to specify the path on the server for which the cookie will be available.
5. **Domain:** It is used to specify the domain for which the cookie is available.
6. **Security:** It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

Image 31 – Cookie arguments

Reference - <https://www.geeksforgeeks.org/php-cookies/>

## Accessing Cookies with PHP

PHP provides many ways to access cookies. The simplest way is to use is \$\_COOKIE variables.

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
echo $_COOKIE["name"]. "<br />";
|echo $_COOKIE["age"]. "<br />";
?>
</body>
</html>
```

Image 32 – Accessing Cookies

## Deleting Cookie with PHP

To delete a cookie you should call setcookie() with the name argument only. It is safest to set the cookie with a date that has already expired.

```
<?php
setcookie("name", "", time()*60, "/", "", 0);
setcookie("age", "31", time()*60, "/", "", 0);
?>
<html>
<head>
<title>Deleting Cookies with PHP</title>
</head>
<body>
<?php echo "Deleted Cookies" ?>
</body>
</html>
```

Image 33 – Deleting Cookies

## Using session

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session. A session creates a file in a temporary directory on the server where registered session variables and their values are stored.

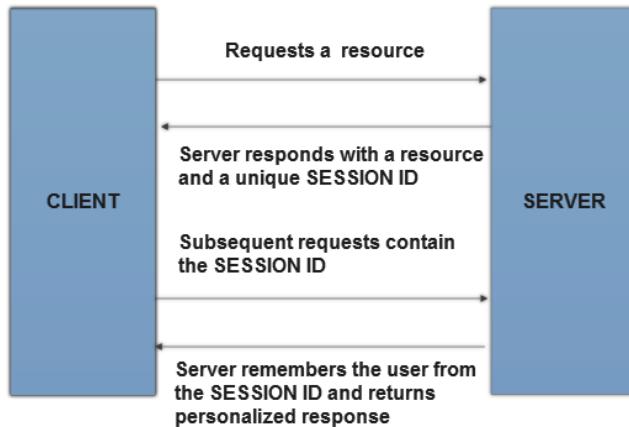


Image 34 – Session in php

Reference - <https://code.tutsplus.com/tutorials/how-to-use-sessions-and-session-variables-in-php--cms-31839>

## Starting a PHP Session

The `session_start()` function is used to start a PHP session. It is recommended to put the call to `session_start()` at the beginning of the page.

```
<?php
session_start();
?>
```

## Storing Session Data

Session data is stored in key-value pairs using the `$_SESSION[ ]` superglobal array. The stored data can be accessed during the lifetime of a session.

```
1 <?php
2
3 session_start();
4
5 $_SESSION["Rollnumber"] = "11";
6 $_SESSION["Name"] = "Ajay";
7
8 ?>
```

Image 35 – Storing session data  
Reference - <https://www.geeksforgeeks.org/php-sessions/>

## Accessing Session Data

Data stored in sessions can be easily accessed by firstly calling **session\_start()** and then by passing the corresponding key to the **\$\_SESSION** associative array.

```
1 <?php
2
3 session_start();
4
5 echo 'The Name of the student is !' . $_SESSION["Name"] . '<br>';
6 echo 'The Roll number of the student is !' . $_SESSION["Rollnumber"] . '<br>';
7
8 ?>
```

Image 36 – Accessing session data  
Reference - <https://www.geeksforgeeks.org/php-sessions/>

## Destroying Certain Session Data

To delete only a certain session data, the **unset** feature can be used with the corresponding session variable in the **\$\_SESSION** associative array.

```
1 <?php
2
3 session_start();
4
5 if(isset($_SESSION["Name"])){
6     unset($_SESSION["Rollnumber"]);
7 }
8
9 ?>
```

Image 37 – Destroying certain session data  
Reference - <https://www.geeksforgeeks.org/php-sessions/>

## Destroying Complete Session

The **session\_destroy()** function is used to completely destroy a session. The **session\_destroy()** function does not require any argument.

```
1 <?php
2
3 session_start();
4 session_destroy();
5
6 ?>
```

Image 38 – Destroying complete session data  
Reference - <https://www.geeksforgeeks.org/php-sessions/>

---

# String matching with regular expression

## What is a regular expression ?

Regular expressions are a powerful tool for examining and changing the text. They enable you to search for patterns within a string, extracting matches flexibly and precisely.

### PHP Regular Expressions

---

- Must use a delimiter: ! @ # /
- Use PHP's single quotes (no escaping \')

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <b>preg_match</b>     | Match against a pattern and extract text                |
| <b>preg_replace</b>   | Like str_replace with a pattern (and sub-patterns)      |
| <b>preg_match_all</b> | Like preg_match, but an array and count for every match |
| <b>preg_split</b>     | Like explode() but with a pattern                       |
| <b>preg_quote</b>     | Escapes text for use in a regular expression            |

Image 39 – PHP regular expressions  
Reference - <http://crackcoder.blogspot.com/2016/02/all-about-intro-regular-expressions-php.html>

## Basics of regular expression

Some characters have special meanings in regular expressions. The characters that match themselves are called literals. The characters that have special meanings are called meta characters.

| Regular Expression  | Will match...                                 |
|---------------------|-----------------------------------------------|
| foo                 | The string “foo”                              |
| ^foo                | “foo” at the start of a string                |
| foo\$               | “foo” at the end of a string                  |
| ^foo\$              | “foo” when it is alone on a string            |
| [abc]               | a, b, or c                                    |
| [a-z]               | Any lowercase letter                          |
| [^A-Z]              | Any character that is not an uppercase letter |
| (gif jpg)           | Matches either “gif” or “jpeg”                |
| [a-z]+              | One or more lowercase letters                 |
| [0-9].\[-]          | Any number, dot, or minus sign                |
| ^[a-zA-Z0-9_]{1,}\$ | Any word of at least one letter, number or _  |
| (wx)(yz)            | wy, wz, xy, or xz                             |
| [^A-Za-z0-9]        | Any symbol (not a number or a letter)         |
| ([A-Z]{3} [0-9]{4}) | Matches three letters or four numbers         |

Image 40 – PHP regular expressions

## Pattern matching in PHP

The preg\_match() function performs Perl-style pattern matching on a string. preg\_match() takes two basic and three optional parameters. preg\_match ( pattern, subject [, matches [, flags [, offset]]]). The preg\_match() function returns 1 if a match is found and 0 otherwise.

```
<?php
if (preg_match("/ell.*/", "My Halloween", $matches)) {
echo "Match was found <br />";
echo $matches[0];
}
?>
```

Image 41 – PHP pattern matching

## Replacing text

The preg\_replace() function looks for substrings that match a pattern and then replaces them with new text. preg\_replace() takes three basic parameters and an additional one. preg\_replace(pattern, replacement, subject [, limit ]).

```
<?php
$search = array ( "/(\w{6})\s\w{2})\s(\w+)/e", "/(\d{4})-(\d{2})-
(\d{2})\s(\d{2}):\d{2}:\d{2})/";
$replace = array ('$1 ".strtoupper("$2")', "$3/$2/$1 $4");
$string = "Posted by Shubhneet | 2016-03-25 02:43:41";
echo preg_replace($search, $replace, $string);
?>
```

Image 42 – Replacing text

---

# Splitting a string with a Regular Expression

## Splitting with Preg\_Split()

It will split a string, but it will allow you to use variable delimiters. It makes easy to extract interesting bits of text with unwanted (but specifiable) gunk in the middle.

```
<?php

// PHP program of preg_split() function
// split the phrase by any number of commas
// space characters include \r, \t, \n and \f

$result = preg_split("/[\s,]+/", "Geeks for Geeks");

// Display result
print_r($result);
?>
```

Image 43 – Splitting with preg\_split()

Reference - [https://www.geeksforgeeks.org/php-preg\\_split-function/](https://www.geeksforgeeks.org/php-preg_split-function/)

## Splitting without Losing

To split a string without removing anything from it, we use a flag:

PREG\_SPLIT\_DELIM\_CAPTURE. This flag inserts any captured groups into the array.

```
<?php
$str = "We123Like456Delimiters";
$regex = "~(\d+)~";
print_r(preg_split($regex,$str,-1,PREG_SPLIT_DELIM_CAPTURE));
?>
```

Image 44 – Splitting without losing

## Splitting with an Invisible Delimiter

The preg\_split function allows you to split a string with an invisible delimiter.

```
<?php
$string = ("TheDayMyVoiceBroke");
$regex = "~(?=([A-Z])~";
$words = preg_split($regex,$string);
print_r($words);
?>
```

Image 45 – Splitting with an invisible delimiter

# Generating Images with PHP

## Embedding an Image in a Page

A standard web page containing text and graphics is created through a series of HTTP requests from the web browser, each answered by a response from the web server.

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
This page contains two images.


</body>
</html>
```

Image 46 – Embedding image in a page

## The GD Extension

GD extension allows PHP to use the open source GD graphics library available from <http://www.boutell.com/gd/>.

**gd**

|                           |                             |
|---------------------------|-----------------------------|
| <b>GD Support</b>         | enabled                     |
| <b>GD Version</b>         | bundled (2.0.34 compatible) |
| <b>FreeType Support</b>   | enabled                     |
| <b>FreeType Linkage</b>   | with freetype               |
| <b>FreeType Version</b>   | 2.3.7                       |
| <b>T1Lib Support</b>      | enabled                     |
| <b>GIF Read Support</b>   | enabled                     |
| <b>GIF Create Support</b> | enabled                     |
| <b>JPG Support</b>        | enabled                     |
| <b>PNG Support</b>        | enabled                     |
| <b>WBMP Support</b>       | enabled                     |
| <b>XPM Support</b>        | enabled                     |
| <b>XBM Support</b>        | enabled                     |

Image 47 – The GD extension

Reference - <https://jgraph.net/download/manuals/chunkhtml/ch03s02.html>

## Basics of computer Graphics

An image is a rectangle of pixels that have various colors. Colors are identified by their position in the palette, an array of colors. Each entry in the palette has three separate color values—one for red, one for green, and one for blue. Each value ranges from 0 (this color not present) to 255 (this color at full intensity).

---

# Creating Image

## Creating An Image with the PHP GD Library

The 3 standard type of images that can be created from scratch with the PHP GD Library is JPG, GIF, and PNG. JPG Is Designed to Compress Full-Color Images, GIF Is Designed to Support Only 256 Colors, PNG Is Designed As An Alternative to GIF, but Does Not Support Animation. To send the image to the browser and complete the code, 3 functions are available for this purpose, one for each type of image.

**imagejpeg()**  
**imagegif()**  
**imagepng()**

The **imagedestroy()** function clear up the memory that is being taken up by storing the image.

```
1 <?php
2 header('Content-type: image/png');
3 $png_image = imagecreate(150, 150);
4 imagecolorallocate($png_image, 15, 142, 210);
5 ?>
```

Image 48 – Creating images with GD library  
Reference - <http://www.phpforkids.com/php/php-gd-library-create-image.php>

```
1 <?php
2 header('Content-type: image/png');
3 $png_image = imagecreate(150, 150);
4 imagecolorallocate($png_image, 15, 142, 210);
5 imagepng($png_image);
6 imagedestroy($png_image);
7 ?>
```

Image 49 – Creating images with GD library

Reference - <http://www.phpforkids.com/php/php-gd-library-create-image.php>

---

# Manipulating Image

## How to Draw Lines on an Image

The imageline() function itself requires 6 parameters.

The syntax is: imageline(image, x1,y1, x2, y2, color)

image = Refers to the Image Resource That the Line Will Be Applied to, x1 = x-coordinate For First Point, y1 = y-coordinate For First Point, x2 = x-coordinate For Second Point, y2 = y-coordinate For Second Point, color = Refers to the Line Color Identifier Created With imagecolorallocate()

```
1 <?php
2 header('Content-type: image/png');
3 $png_image = imagecreate(150, 150);
4 imagecolorallocate($png_image, 15, 142, 210);
5 $black = imagecolorallocate($png_image, 0, 0, 0);
6 imageline($png_image, 0, 0, 150, 150, $black);
7 imagepng($png_image);
8 imagedestroy($png_image);
9 ?>
```

Image 50 – Drawing lines on an image  
Reference - <http://www.phpforkids.com/php/php-gd-library-drawing-lines.php>

This example is to draw a thick black border around the entire image.

```

1 <?php
2 header('Content-type: image/png');
3 $png_image = imagecreate(150, 150);
4
5 imagecolorallocate($png_image, 15, 142, 210);
6 imagesetthickness($png_image, 5);
7 $black = imagecolorallocate($png_image, 0, 0, 0);
8
9 $x = 0;
10 $y = 0;
11 $w = imagesx($png_image) - 1;
12 $z = imagesy($png_image) - 1;
13
14 imageline($png_image, $x, $y, $x, $y+$z, $black);
15 imageline($png_image, $x, $y, $x+$w, $y, $black);
16 imageline($png_image, $x+$w, $y, $x+$w, $y+$z, $black);
17 imageline($png_image, $x, $y+$z, $x+$w, $y+$z, $black);
18
19 imagepng($png_image);
20 imagedestroy($png_image);
21 ?>
```

Image 51 – Drawing lines on an image

Reference - <http://www.phpforkids.com/php/php-gd-library-drawing-lines.php>

## How to Draw Shapes On An Image

Use the imagefilledrectangle() function to draw squares and rectangles, specifying the top left and bottom right corner positions. Use the imagefilledellipse() function to draw circles and ellipses, specifying the center position, width, and height of the shape. Use the imagefilledpolygon() function to draw polygons, specifying the three point of the shape, and few more.

```

<?php
header('Content-type: image/png');
$png_image = imagecreate(300, 300);
$grey = imagecolorallocate($png_image, 229, 229, 229);
$green = imagecolorallocate($png_image, 128, 204, 204);
imagefilltoborder($png_image, 0, 0, $grey, $grey);
imagefilledrectangle ($png_image, 20, 20, 80, 80, $green);
// SQUARE
imagefilledrectangle ($png_image, 100, 20, 280, 80, $green); // RECTANGLE
imagefilledellipse ($png_image, 50, 150, 75, 75, $green);
// CIRCLE
imagefilledellipse ($png_image, 200, 150, 150, 75, $green); // ELLIPSE
$poly_points = array(150, 200, 100, 280, 200, 280);
imagefilledpolygon ($png_image, $poly_points, 3, $green);
// POLYGON
imagepng($png_image);
imagedestroy($png_image);
?>
```

Image 52 – Drawing shapes on an image

---

## How to Add Image Filters & Effects

The `imagefilter()` function can be used to apply various fun and/or useful effects to an existing image or photograph.

```
<?php
header('Content-type: image/jpeg');
$jpg_image = imagecreatefromjpeg('sky.jpg');
imagefilter($jpg_image, IMG_FILTER_GRAYSCALE);
imagejpeg($jpg_image);
imagedestroy($jpg_image);
?>
```

Image 53 – Adding image filters and effects

---

# Using text in Image

To add text to an image, you need two things: the image and the font. Three functions are used to create an image from an existing image so that it can be used/edited.

```
<?php
//Set the Content Type
header('Content-type: image/jpeg');
// Create Image From Existing File
$jpg_image = imagecreatefromjpeg('sky.jpg');
// Allocate A Color For The Text
$white = imagecolorallocate($jpg_image, 255, 255, 255);
// Set Path to Font File
$font_path = 'font.TTF';
// Set Text to Be Printed On Image
$text = "This is a sky!";
// Print Text On Image
imagettftext($jpg_image, 25, 0, 75, 300, $white, $font_path, $text);
// Send Image to Browser
imagejpeg($jpg_image);
// Clear Memory
imagedestroy($jpg_image);
?>
```

Image 54 – Using text in image

---

# Database Connectivity with MySql

## Introduction to RDBMS

### What is a Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

### What is RDBMS?

A Relational Database Management System (RDBMS) is a software that, enables you to implement a database with tables, columns, and indexes.

- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an
- SQL query and combines information from various tables.

## RDBMS Terminology

### What is a Table?

A table is a matrix with data. A table in a database looks like a simple spreadsheet.

### What is a Column?

One column (data element) contains data of one and the same kind, for example, the column postcode.

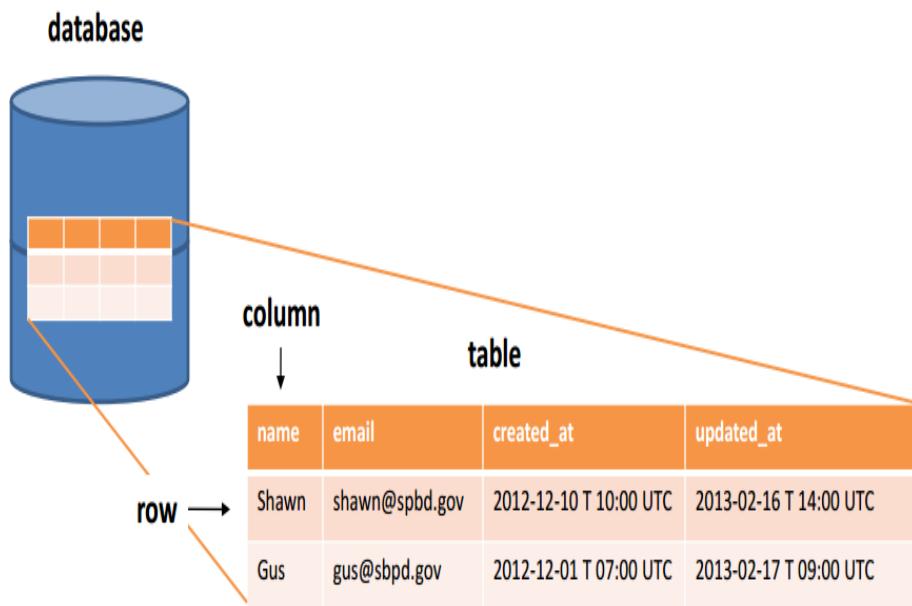


Image 55 – RDBMS

Reference - <https://code.tutsplus.com/tutorials/relational-databases-for-dummies--net-30244>

### What is a Row?

A row (= tuple, entry or record) is a group of related data, for example, the data of one subscription.

### What is Data Redundancy?

It is the existence of data that is additional to the actual data and permits correction of errors in stored or transmitted data. The additional data can simply be a complete copy of the actual data, or only select pieces of data that allow detection of errors and reconstruction of lost or damaged data up to a certain level.

### What is Primary Key?

A primary key is unique. A key value cannot occur twice in one table. With a key, you can find at most one row.

## What is Foreign Key?

A foreign key is the linking pin between two tables.

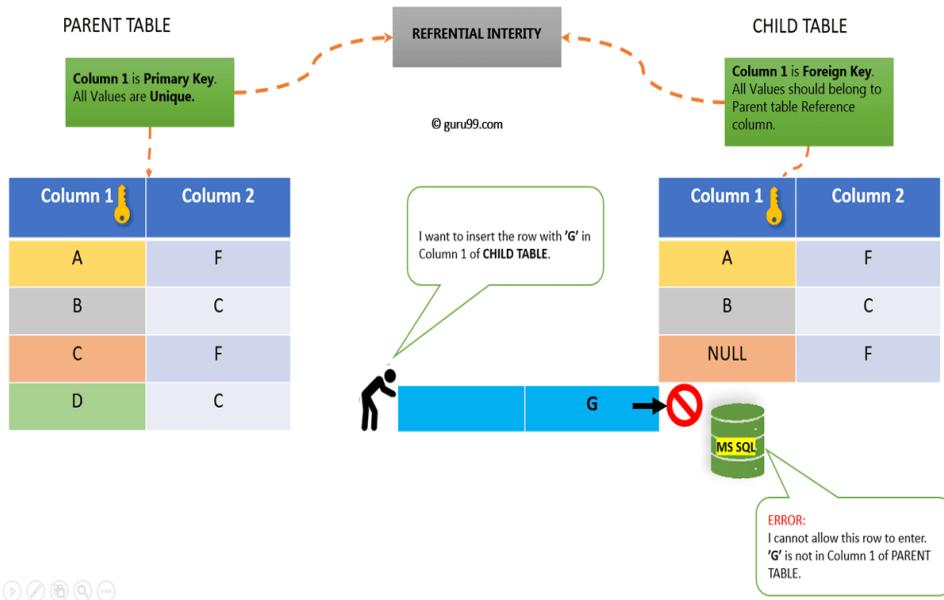


Image 56 – RDBMS terminology

Reference - <https://www.guru99.com/sql-server-foreign-key.html>

## What is Compound Key?

A compound key (composite key) is a key that consists of multiple columns because one column is not sufficiently unique.

## What is a Referential Integrity?

Referential Integrity makes sure that a foreign key value always points to an existing row.

## What is an Index?

Indexes are **special lookup tables** that the database search engine can use to speed up data retrieval. An index in a database is very similar to an index in the back of a book.

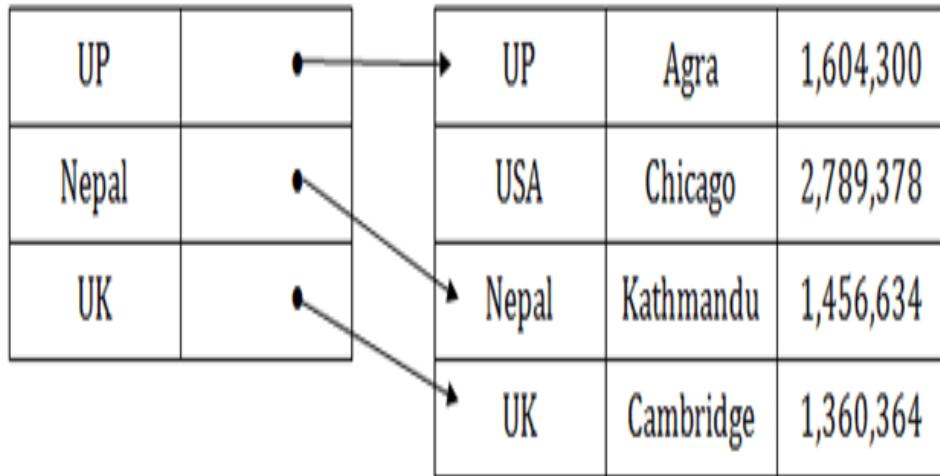


Image 57 – Indexing

Reference - <https://www.javatpoint.com/indexing-in-dbms>

## MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. Some advantages of MySQL are:

- MySQL is released under an open-source license.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.

MySQL works very quickly and works well even with large data sets. MySQL supports large databases, up to 50 million rows or more in a table. MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

## Connection with MySql Database

MySQLi Installation. For installation details, go to

<http://php.net/manual/en/mysqli.installation.php>. Open a Connection to MySQL. Close the Connection.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
If (!$conn) {
die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Image 58 – MySQL connection

## Create a MySQL Database

Create a MySQL Database Using MySQLi and PDO.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
echo "Database created successfully";
} else {
echo "Error creating database: " . $conn->error;
}
$conn->close();
?>
```

Image 59 – Create MySQL connection using MySQLi

---

## Create MySQL Tables

Create a MySQL Table Using MySQLi and PDO.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// sql to create table
$sql = "CREATE TABLE MyVisitors (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";
if ($conn->query($sql) === TRUE) {
echo "Table MyVisitors created successfully";
} else {
echo "Error creating table: " . $conn->error;
}
$conn->close();
?>
```

Image 59 – Create MySQL table

## Insert Data Into MySQL

The SQL query must be quoted in PHP. String values inside the SQL query must be quoted. Numeric values must not be quoted. The word NULL must not be quoted.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO MyVisitors (firstname, lastname, email)
VALUES ('Shubhneet', 'Goel', 'shubh@spcits.com')";
if ($conn->query($sql) === TRUE) {
echo "New record created successfully";
} else {
echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

Image 60 – Insert data into MySQL table

## Select Data From MySQL

The SELECT statement is used to select data from one or more tables.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM MyVisitors";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
while($row = $result->fetch_assoc()) {
echo "id: " . $row["id"] . " - Name: " . $row["firstname"] . " " . $row["lastname"] . "<br>";
}
} else {
echo "0 results";
}
$conn->close();
?>
```

Image 61 – Select data from MySQL

## Delete Data From MySQL

The DELETE statement is used to delete records from a table.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// sql to delete a record
$sql = "DELETE FROM MyVisitors WHERE id=3";
if ($conn->query($sql) === TRUE) {
echo "Record deleted successfully";
} else {
echo "Error deleting record: " . $conn->error;
}
$conn->close();
?>
```

Image 62 – Delete data from MySQL

## Update Data in MySQL

The UPDATE statement is utilized to update existing records in a table.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
$sql = "UPDATE MyVisitors SET lastname='Goel' WHERE id=2";
if ($conn->query($sql) === TRUE) {
echo "Record updated successfully";
} else {
echo "Error updating record: " . $conn->error;
}
$conn->close();
?>
```

Image 63 – Update data in MySQL

---

# Able to make websites, web servers, game frameworks, desktop and CLI applications and IDE using Python

In this section, we will read about:

- Introduction to Python.
- History.
- Features
- Setting Up Path
- Basic Syntax Variable
- Data Types Operator
- Conditional Statement
- Looping
- Control Statement
- String Manipulation
- Lists
- Tuple
- Functions and Methods
- Dictionaries
- Functions
- Modules
- Input and Output
- Exception Handling

# Introduction to Python

## Overview of Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- 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 from simple text processing to WWW browsers to games.

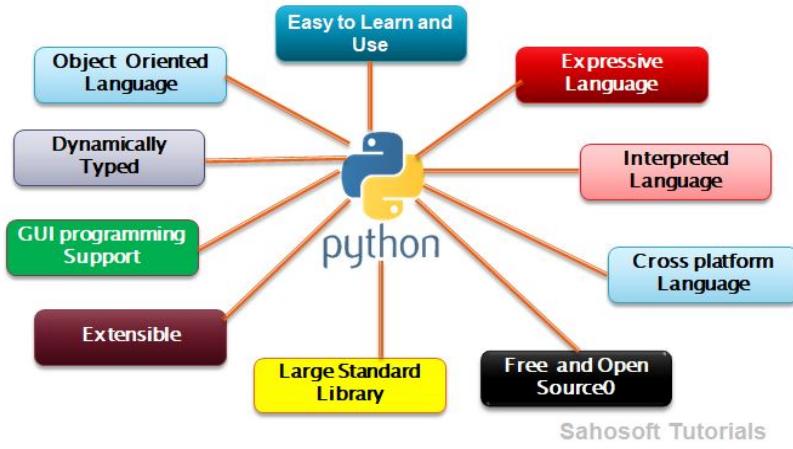


Image 1: Python- Overview of Python.  
Reference:<https://i.pinimg.com/originals/ea/fa/99/eafa991b59dde12f94e41f3622d157b2.png>

## Characteristics of Python

As we discussed the introduction to python now we are going to learn about the characteristics of python are as below:

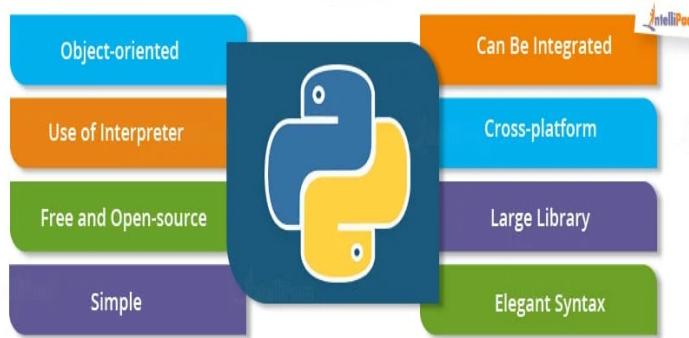


Image 2: Python - Characteristics of Python.  
Reference:<https://intellipaat.com/mediaFiles/2019/02/TutorialPython-01.jpg>

## Platform Independent

Python is platform independent. The python code can be used for any operating system like Windows, Unix, Linux, and Mac. There is no need to write different code for the different OS.

## Interpreted

The python code does not need to compile as required for other languages. Python code automatically converts the source code into byte code internally and the code is executed line by line not at once, so it takes more time to execute the code for the application.

## Simple Syntax

The Python language is simple and can be easily coded and read. The syntax of python is really simple and can be learned easily.

## High Level Language

It is a high-level language used for scripting. It means one does not need to remember the system architecture and no need for managing the memory as well.

## Free and Open Source

Python is Open Source and freely available over the internet anywhere. One does not need to take the license for it. It can be easily downloaded and use.

## Rich Library Support

Python can be integrated with other libraries that help in making the functionality work for you. Do not need to write the extra code for that.

## Advantages over other languages

One such language that is understood and preferred all over the world for development is Python. It's a general-purpose, high-level OOPs based interpreted language, used for dynamic applications, widely across the globe. Python is extremely popular for its versatility and scope of applicability.



Image 3: Python - advantages over other languages.

Reference:<https://i.redd.it/t0tidyi6ewh21.jpg>

Python is a language embraced by a large community of coders for many reasons. Many businesses are advised to choose python as their language for programming. Let's find out why:

### It is free

Python is an open-source language. The Python company is one of the largest companies and, still, is free. Using python language doesn't require a particular subscription or a custom-built

---

platform either, thus any desktop and laptop is compatible with Python. All the tools that are necessary for python coding, the supporting means, modules, and libraries are absolutely free.

The essential IDEs that is, the integrated development environments that include PTVS, Pydev with eclipse spyder python can be downloaded easily for free. Reduced costs are always beneficial to businesses.

### **It Needs Less Coding**

Python by nature has a very simple syntax. The same logic that needs 7 lines in a C++ language, requires just 3 lines in Python. Having a smaller code requires less space, less time, and is well appreciated by coders, as the rework or correction also takes lesser time. The language aces all the parameters of readability. To support itself, the language has many built-ins and libraries that make it easy to comprehend.

### **All kinds of Businesses can afford it**

Being a free platform, all small and medium level companies can use it. Companies that are in the nascent stage can use the python platform and begin their operations with cost-effective software. The ability to develop applications and software quickly makes it suitable for startups, as they can survive in the cutthroat competition by leveraging the speed of the python language.

### **It is one of the most trending languages**

Java and C++ are the native languages with the Object Oriented approach. Their use is very widespread, and efficiency is tremendous. The only problem with these languages is they are lengthy. Python, on the other hand, has all the features of object-oriented programming just like Java and C++, and is fast too. The codes are shorter and the syntax simple, thus being easy to amend, rework and optimize.

---

# History of Python

## Python Timeline/History and IEEE rankings

Python is a widely used general-purpose, high-level programming language.

- Python was conceptualized by Guido Van Rossum in the late 1980s.
- Rossum Published the first version of Python code(0.9.0) in February 1991 at CWI(Centrum Wiskunde & Informatica) in Netherland, Amsterdam.
- Python is Derived from the ABC Programming Language, which is a general-purpose Programming Language that has been developed at the CWI.
- Rossum chose the name “Python” , since he was a big fan of Monty Python’s Flying Circus.
- Python is now maintained by a core development team at the institute, although Rossum still holds a vital role in directing its progress.

The latest programming language ranking from IEEE spectrum maintains the same findings from last year’s report: Python is king for a majority of ranking metrics.

The report from IEEE Spectrum analyzes metrics from multiple data sources and ranks programming language popularity.

It ranks languages based on use cases:

web development, enterprise, mobile, and embedded. Rankings also determine a language’s popularity based on different social channels, such as Twitter, StackOverflow, GitHub, and Reddit, and even ranks by which language is currently booming in the job market.



Image 4: Python- Feature:Easy to code and Understand.  
Reference: <https://spectrum.ieee.org/image/MzExNjk1OA.jpeg>

# Features of Python

## Easy to Code and Understand

Python is high level programming language. Python is very easy to learn language as compared to other language like c, c#, java script, java etc. It is very easy to code in python language and anybody can learn python basic in a few hours or days. It is also a developer-friendly language.

### “Hello, World”

- **C**

```
#include <stdio.h>

int main(int argc, char ** argv)
{
    printf("Hello, World!\n");
}
```
- **Java**

```
public class Hello
{
    public static void main(String argv[])
    {
        System.out.println("Hello, World!");
    }
}
```
- **now in Python**

```
print "Hello, World!"
```

Image 5: Python- Features:Easy to code and Understand.  
Reference: <https://analyticsprofile.com/wp-content/uploads/2018/08/python-1.jpg>

## Expressive Language

Python is very expressive when compared to other languages. By expressive, we mean, in Python a single line of code performs a lot more than what multiple lines can perform in other languages. In simple it means that fewer lines of code are required to write a program in Python.

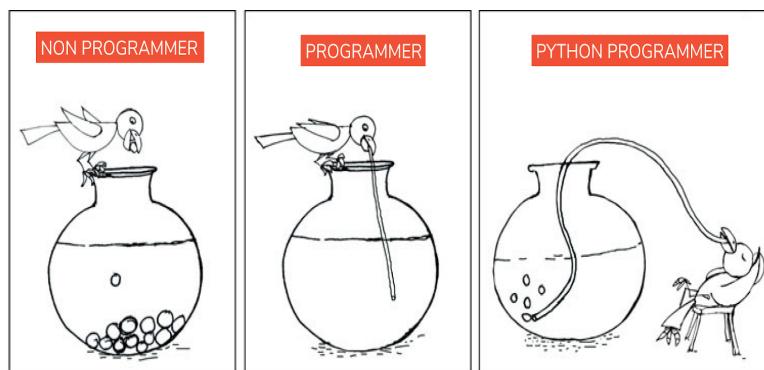


Image 6: Python-Feature:Expressive Language.  
Reference:<https://i1.faceprep.in/Companies-1/features-of-python-language.png>

## Free and Open Source

Python is free and can be easily installed by anyone and on any system. Also, Python is an open-source programming language. This means that Python's source code can be freely modified and used by anyone.

## Interpreted Language

Python is an interpreted language. An interpreter in general works very differently from a compiler. An interpreter executes a code line by line and hence it gets easy for a programmer to debug errors. Also, if you have observed, **even though your program has multiple errors, Python displays only one error at a time**. Whereas a compiler compiles the entire code at once and displays a list of errors.

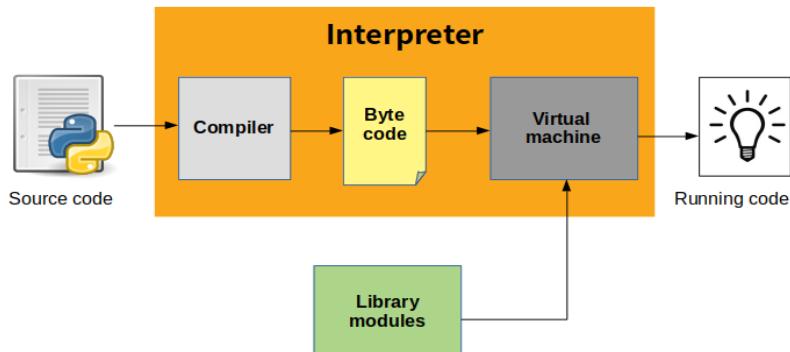


Image 7: Python-Feature:Interpreted Language.  
Reference:<https://indianpythonista.files.wordpress.com/2018/01/pjme67t.png>

## Object-Oriented Language

Like other general-purpose languages, python is also an object-oriented language. In Python, **we can easily create and use classes and objects**. Some of the other major principles of

---

object-oriented programming languages are Object, Class, Method, Inheritance, Polymorphism, Data Abstraction and Encapsulation.

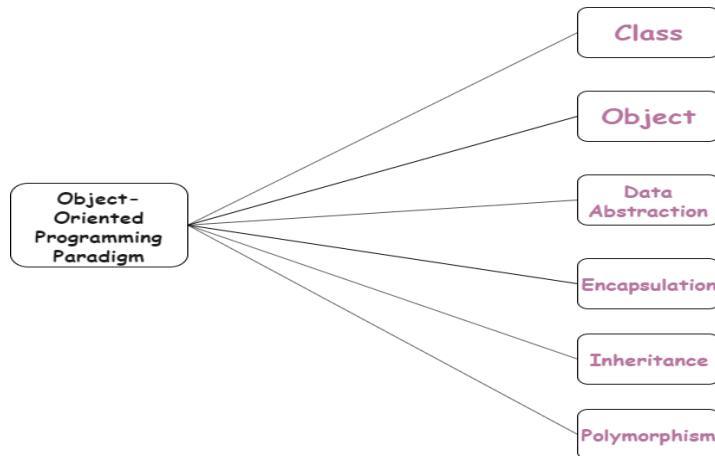


Image 8: Python-Feature:Object-Oriented Language.  
Reference:<https://cdn.askpython.com/wp-content/uploads/2019/12/Python-oop.png>

## Dynamically Typed Programming Language

Python is a dynamically typed language. This means, whenever a variable is declared, the programmer need not mention its data type. Rather, the type of the variable is decided during run time.

## Cross-Platform Language

Say, for example, you have written a piece of code in a Python IDE on Windows. Now, you want to run this code on another system. Then, you need not make any changes to the code to execute it on other machines like MAC, OS, Linux etc. The code remains exactly the same and this makes it easy for programmers to switch across platforms and work comfortably using Python.

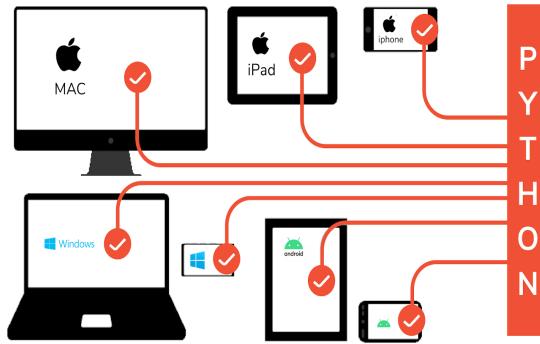


Image 9:Python-Feature: Cross-Platform Language.  
Reference:<https://i1.faceprep.in/Companies-1/python-is-a-cross-platform-language.png>

## Large Standard Library

Python has a **large standard library** and this **helps save the programmers time** as you don't have to write your own code for every single logic. There are libraries for expressions, unit-testing, web browsers, databases, CGI, image manipulation etc.

## Extensible Language

In case you want to write a part of your Python code in C++ or Java etc, then you can do it. Since Python is an extensible language, it lets you do this with ease.

# Setting Up Path

## Python Installation for Windows

List of Hardware/Software requirements:

1. Laptop/Computer with Windows/Linux OS - Ubuntu 18.04 LTS
2. Python Software Installation

Steps: Install Python software in the system.

- Open the browser and type the python.org/downloads.
- Click on [download](#).
- Choice either Windows x86-64 executable installer for 64-bit or Windows x86 executable installer for 32-bit.
- After downloading a file the below page will appear.

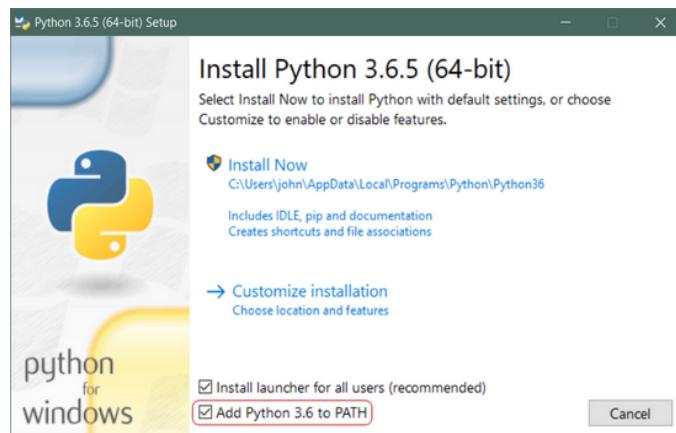


Image 10: Python- Installation of Python in windows.  
Reference:<https://files.realpython.com/media/win-install-dialog.40e3ded144b0.png>



Image 11: Python-Installation of Python In windows.  
Reference:<https://intellipaat.com/mediaFiles/2018/12/Step-4-1.png>

## How to set Python Path in Windows

To permanently modify the default **environment variables** :

My Computer > Properties > Advanced System Settings > Environment Variables > Edit

- Right-click 'My Computer'.
- Select 'Properties' at the bottom of the Context Menu.
- Select 'Advanced system settings'
- Click 'Environment Variables...' in the Advanced Tab
- Under 'System Variables': Click Edit

Add python's path to the end of the list (the paths are separated by semicolons(;) )

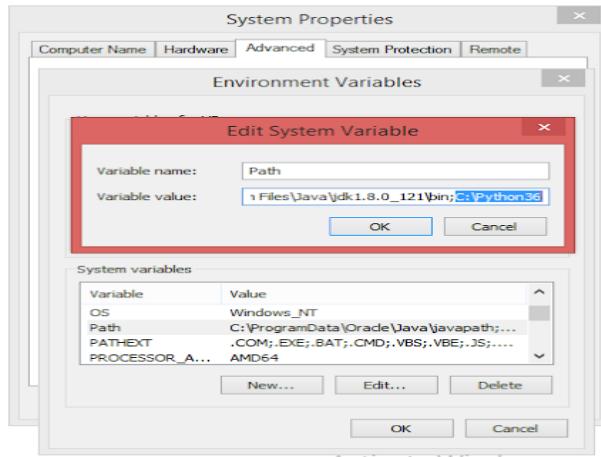


Image 12: Python-Setting up path in windows.  
Reference:<https://windowsbulletin.com/wp-content/uploads/2019/08/Add-Path-to-Python-in-Windows.jpg>

## Python Installation for Linux

If you are using Ubuntu 16.10 or newer, then you can easily install Python 3.6 with the following commands:

```
$ sudo apt-get update  
$ sudo apt-get install python3.6
```

If you're using another version of Ubuntu (e.g. the latest LTS release), we recommend using the deadsnakes PPA to install Python 3.6:

```
$ sudo apt-get install software-properties-common  
$ sudo add-apt-repository ppa:deadsnakes/ppa  
$ sudo apt-get update  
$ sudo apt-get install python3.6
```

---

## How to set Python Path in Linux

- Open your favorite terminal program;
- Open the file `~/.bashrc` in your text editor – e.g. atom `~/.bashrc`;
- Add the following line to the end:

```
export PYTHONPATH=/home/my_user/code
```

Save the file.

- Close your terminal application;
- Start your terminal application again, to read in the new settings, and type this:

```
echo $PYTHONPATH
```

- It should show something like `/home/my_user/code`.

---

# Basic Syntax Variable

## Interactive Mode Programming

Interactive mode provides us with a quick way of running blocks or a single line of Python code. Interactive mode is handy when you just want to execute basic Python commands.

To access the Python shell, open the terminal of your operating system and then type "python". Press the enter key and the Python shell will appear.

```
C:\Windows\system32>python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The >>> indicates that the Python shell is ready to execute and send your commands to the Python interpreter. The result is immediately displayed on the Python shell as soon as the Python interpreter interprets the command.

To run your Python statements, just type them and hit the enter key. You will get the results immediately, unlike in script mode. For example, to print the text "Hello World", we can type the following:

```
>>> print("Hello World")
Hello World
>>>
```

Here are other examples:

```
>>> 10
10
>>> print(5 * 20)
100
>>> "hi" * 5
```

```
'hihihihihi'  
>>>
```

## Script Mode Programming

In script mode, You write your code in a text file then save it with a .py extension which stands for "Python". Note that you can use any text editor for this, including Sublime, Atom, notepad++, etc.

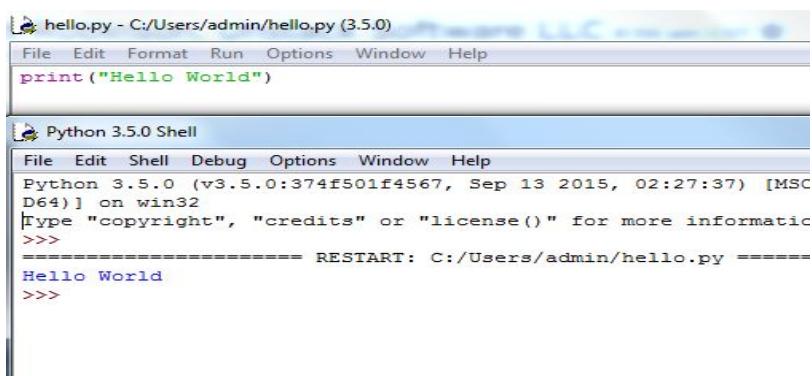
Let us create a new file from the Python shell and give it the name "hello.py". We need to run the "Hello World" program. Add the following code to the file:

```
print("Hello World")
```

Click "Run" then choose "Run Module". This will run the program:

### OUTPUT-

```
Hello World
```



The screenshot shows two windows side-by-side. The left window is titled 'hello.py - C:/Users/admin/hello.py (3.5.0)' and contains the Python code: `print("Hello World")`. The right window is titled 'Python 3.5.0 Shell' and shows the output of running the script. It displays the Python version information: 'Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC D64] on win32'. Below this, it says 'Type "copyright", "credits" or "license()" for more information'. Then it shows the command prompt '=>>>' followed by the output 'Hello World'.

Image 13:Python- Script Mode Programming.  
Reference:<https://s3.amazonaws.com/stackabuse/media/python-programming-interactive-vs-script-mode-1.jpg>

---

## Python Identifiers

An identifier is a name given to program elements such as variables, array, class and functions etc. An identifier is a sequence of letters, digits, and underscores, the first character of which can not be a digit. Following rules must be kept in mind while naming an identifier.

- The first character must be an alphabet (uppercase or lowercase) or can be an underscore.
- An identifier can not start with a digit.
- All succeeding characters must be alphabet or digit.
- No special characters like !, @, #, \$, % or punctuation symbols is allowed except the underscore “\_” .
- No two successive underscores are allowed.
- Keywords can not be used as identifiers.

**Note-** Python is a case sensitive programming language, which means “ABC” and “abc” are not the same.

## Reserved Keywords

Python keywords are set words that cannot be used as an identifier in a program. These words are known as “Reserved Words” or “Keywords”. Python keywords are standard identifiers and their meaning and purpose is predefined by the compiler.

Below is a list of available keywords in Python Programming Language.

| Keywords in Python |             |                |          |        |
|--------------------|-------------|----------------|----------|--------|
| False              | class       | <u>finally</u> | is       | return |
| None               | continue    | for            | lambda   | try    |
| True               | def         | from           | nonlocal | while  |
| and                | del         | global         | not      | with   |
| as                 | <u>elif</u> | if             | or       | yield  |
| assert             | else        | import         | pass     |        |
| break              | except      | in             | raise    |        |

Image 14:Python- Script Mode Programming.  
Reference:<https://makemeanalyst.com/wp-content/uploads/2017/06/keywords-python.png>

## Lines and Indentation

Python indentation is a way of telling the Python interpreter that a series of statements belong to a particular block of code. In languages like C, C++, Java, we use curly braces { } to indicate the start and end of a block of code. In Python, we use space/tab as indentation to indicate the same to the compiler.

- Python requires indentation as part of the syntax.
- Indentation signifies the start and end of a block of code.
- Program will not run without correct indentation.

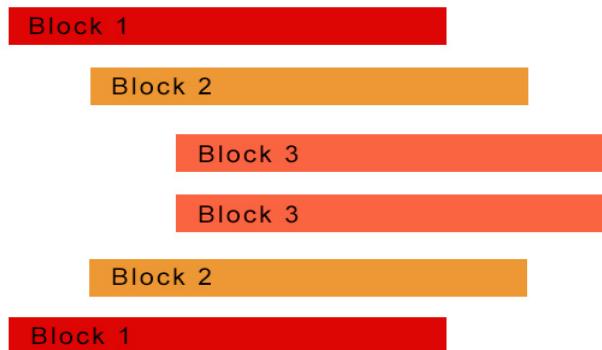


Image 15:Python- Lines and Indentation.  
Reference:<https://www.stechies.com/userfiles/images/code-block.jpg>

```
a = input('Enter the easiest programming language: ') //statement 1
if a == 'python': //statement 2
    print('Yes! You are right') //statement 3
else: //statement 4
    print('Nope! You are wrong') //statement 5
```

In the above code,

Statement 1 gets executed first, then it gets to statement 2. Now only if statement 2 is correct, we would want statement 3 to get printed. Hence Statement 3 is indented with 2 spaces. Also, statement 5 should be printed, if statement 4 is true and hence this is also indented with 2 spaces.

---

Statement 1, 2 and 4 are main statements which need to be checked and hence these 3 are indented with the same space.

## Multi Line Statement

Usually, every Python statement ends with a newline character. However, we can extend it over to multiple lines using the line continuation character (\).

When you right away use the line continuation character (\) to split a statement into multiple lines.

```
# Initializing a list using the multi-line statement
>>> my_list = [1,
... 2, 3\
... ,4,5 \
... ]
>>> print(my_list)

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

```
>>> subjects = [
... 'Maths',
... 'English',
... 'Science'
...
... ]
>>> print(subjects)

['Maths', 'English', 'Science']
```

```
>>> type(subjects)  
<class 'list'>
```

## Quotation in Python

Python accepts single ('), double ("") and triple (" " or " """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'  
sentence = "This is a sentence."  
paragraph = """This is a paragraph. It is made up  
of multiple lines and sentences.""""
```

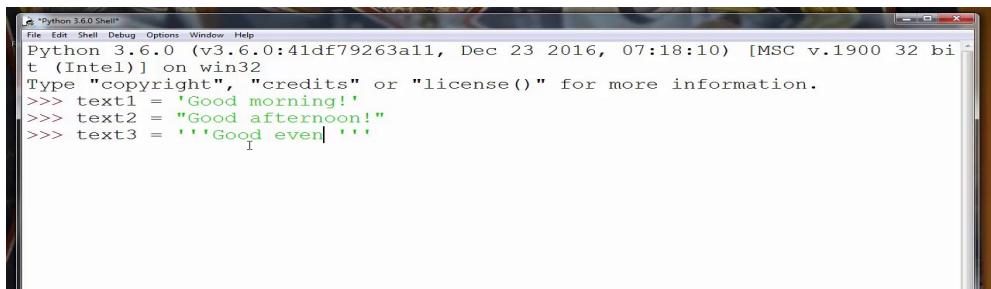


Image 16 : Python: Quotation in Python.  
Reference:<https://i.ytimg.com/vi/uSief1mVRoY/maxresdefault.jpg>

## Comments in Python

A comment in Python starts with the hash character, # , and extends to the end of the physical line. A hash character within a string value is not seen as a comment, though. To be precise, a comment can be written in three ways - entirely on its own line, next to a statement of code, and as a multi-line comment block.

Single-line comments are created simply by beginning a line with the hash (#) character, and they are automatically terminated by the end of line.

---

### For Example-

```
#This would be a comment in Python
```

Comments that span multiple lines – used to explain things in more detail – are created by adding a delimiter (""""") on each end of the comment.

```
"""
```

*This would be a multiline comment  
in Python that spans several lines and  
describes your code, your day, or anything you want it to*

```
"""
```

### Using Blank Lines

Using blank lines in functions, sparingly, to indicate logical sections. Python accepts the control-L (i.e. ^L) form feed character as whitespace; Many tools treat these characters as page separators, so you may use them to separate pages of related sections of your file.

- A line containing only whitespaces, possibly with a comment, is known as a blank line and Python totally ignores it.
- In an interactive interpreter session, you must enter a physical line to terminate a multiple statement.

### Input From the User

Python provides us with two inbuilt functions to read the input from the keyboard.

- **raw\_input ( prompt )**
- **input ( prompt )**

---

**raw\_input () :** This function works in older versions (like Python 2.x). This function takes exactly what is typed from the keyboard, converts it to string and then returns it to the variable in which we want to store. For example –

```
# Python program showing
# a use of raw_input()

g = raw_input("Enter your name : ")
print g
```

#### OUTPUT-

```
Enter your name : Priyanka Singh
Priyanka Singh
>>> |
```

**input () :** This function first takes the input from the user and then evaluates the expression, which means Python automatically identifies whether the user entered a string or a number or list. If the input provided is not correct then either syntax error or exception is raised by python. For example –

```
# Python program showing
# a use of raw_input()

val = input("Enter your value : ")
print(val)
```

#### OUTPUT-

```
Enter your value : 1234
```

```
1234
```

```
>>> |
```

## Multiple Statement on a Single Line

The semicolon ( ; ) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon –

```
a=10  
b=20  
c=a*b  
print (c)
```

## OUTPUT-

```
a=10; b=20; c=1*b; print (c)
```

## Multiple Statements Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite.

```
if expression :
```

```
    suite
```

```
elif expression :
```

```
    suite
```

```
else :
```

---

suite.

---

# Data Types Operator

In Python, operators are special symbols that designate that some sort of computation should be performed. The values that an operator acts on are called operands.

Here is an example:

```
>>> a = 10  
>>> b = 20  
>>> a + b  
30
```

In this case, the `+` operator adds the operands `a` and `b` together. An operand can be either a literal value or a variable that references an object:

```
>>> a = 10  
>>> b = 20  
>>> a + b - 5  
25
```

A sequence of operands and operators, like `a + b - 5`, is called an expression. Python supports many operators for combining data objects into expressions. These are explored below.

## Types of operators

- Python Arithmetic Operator
- Python Comparison Operator
- Python Assignment Operator
- Python Bitwise Operator
- Python Logical Operator
- Python Membership Operator
- Python Identity Operator

- 
- Python Operator Precedence

## Python Arithmetic Operators

Arithmetic Operators perform various arithmetic calculations like addition, subtraction, multiplication, division, %modulus, exponent, etc. There are various methods for arithmetic calculation in Python like you can use the eval function, declare variable & calculate, or call functions.

| Operator | Name           | Description                                                                                                                                                        |
|----------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| +        | Addition       | The operands on either side of this operator are added.                                                                                                            |
| -        | Subtraction    | The operand on the right side of this operator is subtracted from the one on the left side.                                                                        |
| *        | Multiplication | The operands on either side of this operator are multiplied                                                                                                        |
| /        | Division       | The operand on the left side of this operator is divided by the one on the right side. As a result, it returns the quotient in the form of a floating-point value. |
| %        | Modulo         | The operand on the left side of this operator is divided by the one on the right side. As a result, it returns the remainder value.                                |
| **       | Exponent       | This operator raises the left side operand to the power of the right side operand.                                                                                 |
| //       | Floor division | The operand on the left side of this operator is divided by the one on the right side. As a result, it returns the quotient in the form of an integer value.       |

## Python Comparison Operator

These operators compare the values on either side of the operand and determine the relation between them. It is also referred to as relational operators. Various comparison operators are ( ==, !=, <>, >, <=, etc)

| Operator | Name                     | Description                                                                                 |
|----------|--------------------------|---------------------------------------------------------------------------------------------|
| ==       | Equal to                 | Checks whether two operands are equal                                                       |
| !=       | Not equal to             | Checks whether two operands are not equal                                                   |
| >        | Greater than             | Checks whether the left side operand is greater than the right side operand                 |
| <        | Less than                | Checks whether the left side operand is less than the right side operand                    |
| >=       | Greater than or equal to | Checks whether the left side operand is either greater or equal to the right side operand   |
| <=       | Lesser than or equal to  | Checks whether the left side operand is either less than or equal to the right side operand |

## Python Assignment Operator

Python assignment operators are used for assigning the value of the right operand to the left operand. Various assignment operators used in Python are (+=, -=, \*=, /=, etc.)

| Operator | Name | Description |
|----------|------|-------------|
|----------|------|-------------|

|     |                           |                                                                                                                                                  |
|-----|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| =   | Assignment                | This operator assigns its right-side value to its left-side operand                                                                              |
| +=  | Addition assignment       | This operator adds left and the right side operands and assigns the result to the left side operand                                              |
| -=  | Subtraction assignment    | This operator subtracts the right operand from the left operand and assigns the result to the left side operand                                  |
| *=  | Multiplication assignment | This operator multiplies the right-side operand with the left-side operand and assigns the result to the left side operand                       |
| /=  | Division assignment       | This operator divides the left side operand by the right-side operand and assigns the quotient to the left side operand                          |
| %=  | Modulus assignment        | This operator divides the left side operand by the right side operand and assigns the remainder to the left side operand                         |
| **= | Exponentiation assignment | This operator raises the left side operand to the power of the right-side operand and assigns the result value to the left side operand          |
| //= | Floor division assignment | This operator divides the left operand by the right operand and assigns the quotient value (in the form of an integer value) to the left operand |
| &=  | Bitwise AND assignment    | This operator performs a bitwise AND operation on both the left and the right side operands and assigns the result to the left side operand      |
| =   | Bitwise OR assignment     | This operator performs a bitwise OR operation on both the left and the right side operands and assigns the result to the left side operand       |

|                        |                                |                                                                                                                                           |
|------------------------|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>^=</code>        | Bitwise XOR assignment         | This operator performs bitwise XOR operation on both the left and the right side operands and assigns the result to the left side operand |
| <code>&gt;&gt;=</code> | Bitwise right shift assignment | This operator right shifts the given value by the specified position and assigns the result to the left side operand                      |
| <code>&lt;&lt;=</code> | Bitwise left shift assignment  | This operator left shifts the given value by the specified position and assigns the result to the left side operand                       |

## Python Bitwise Operator

Bitwise operator works on bits and performs bit by bit operation. Assume if `a = 60`; and `b = 13`; Now in the binary format their values will be `0011 1100` and `0000 1101` respectively. Following table lists out the bitwise operators supported by Python language with an example each in those, 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^b = 0011 0001`

`~a = 1100 0011`

There are following Bitwise operators supported by 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)                                                         |
| $\wedge$ Binary XOR           | It copies the bit if it is set in one operand but not both.                                   | $(a \wedge b) = 49$ (means 0011 0001)                                                    |
| $\sim$ Binary Ones Complement | It is unary and has the effect of 'flipping' bits.                                            | $(\sim a) = -61$ (means 1100 0011 in 2's complement form due to a signed binary number.) |
| << Binary Left Shift          | The left operand's 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 operand's value is moved right by the number of bits specified by the right operand. | $a >> 2 = 15$ (means 0000 1111)                                                          |

## Python Logical Operator

There are **three logical operators in Python** - and, or and not. They are used to combine two or more conditional statements.

| Operator | Description                                                |
|----------|------------------------------------------------------------|
| and      | The output is true when both the expressions are true      |
| or       | The output is true if either one of the expression is true |
| not      | Reverses the output                                        |

### Python Membership Operator

Membership operators `in` and `not in` are used to find whether a value is present in a particular Python object or not.

| Operator            | Description                                                                                 |
|---------------------|---------------------------------------------------------------------------------------------|
| <code>in</code>     | Returns True if it finds a variable in the specified sequence and false otherwise.          |
| <code>not in</code> | Returns False if it does not find a variable in the specified sequence and false otherwise. |

### Python Identity Operator

Identity operators `is` and `is not` are used to check whether the memory locations of two variables/objects are the same or not. Also, we can use these operators to find if a variable/object belongs to a particular type or not.

| Operator | Description                                                                                               |
|----------|-----------------------------------------------------------------------------------------------------------|
| is       | Returns True if the operands on either side of the operator point to the same object and false otherwise. |
| is not   | Returns False if the operands on either side of the operator point to the same object and true otherwise. |

### Python Operator Precedence

The following table lists all operators from highest precedence to lowest.

| Sr.No. | Operator & Description                                                                      |
|--------|---------------------------------------------------------------------------------------------|
| 1      | **<br><br>Exponentiation (raise to the power)                                               |
| 2      | ~ + -<br><br>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  | <> == !=<br><br>Equality operators                  |
| 10 | = %= /= //=-= += *= **=<br><br>Assignment operators |
| 11 | is is not<br><br>Identity operators                 |
| 12 | in not in<br><br>Membership operators               |
| 13 | not or and<br><br>Logical operators                 |



# Conditional Statements

## Statement and Description

Decision-making statements are those that **help in deciding the flow of the program**. For example, at times, you might want to execute a block of code only if a particular condition is satisfied. Well, in this case, a decision-making statement will be of great help. To understand this, let's now look at how they function.

### If statement

This is the simplest decision-making statement in Python. It is **used to decide if a particular block of code needs to be executed or not**. Whenever the given condition is **True**, then the block of code inside it gets executed, else it does not.

#### The If statement flowchart:

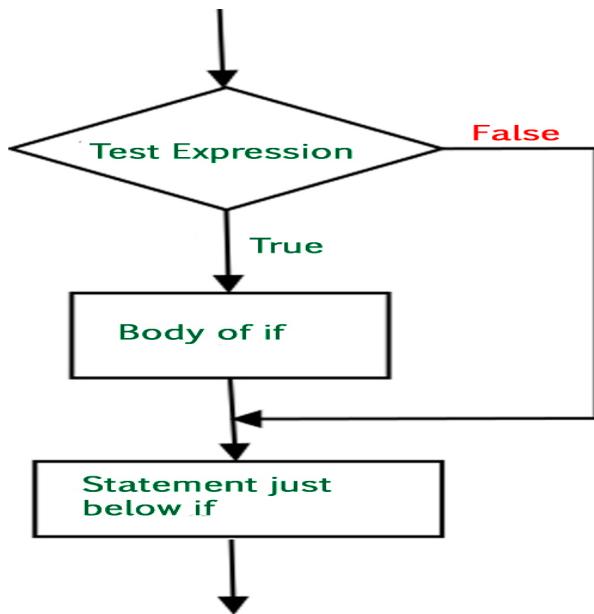


Image 17: Python - If statement in Python.  
Reference:<https://tutorialspoint.dev/image/if-statement.jpg>

### Syntax for if statement-

if condition:  
body of if

### Example of if statement

```
#program to check if num1 is less than num 2
num1, num2 = 5, 6
if(num1 < num2):
    print("num1 is less than num2")
Output:num1 is less than num2
```

**Explanation-**Here the value 5 is less than 6, so this means the condition is True. Hence the print statement inside the body of if gets executed.

### If else statement

The statements written within the else block get executed whenever the if condition is **False**. You can imagine the flow of execution this way,

#### The if else statement flowchart:

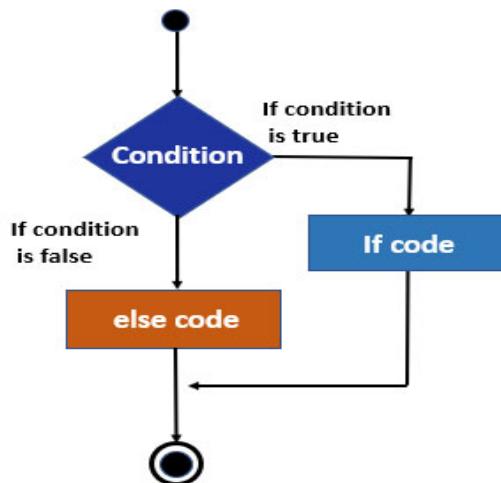


Image 18: Python -If else statement in Python.

---

Reference:<https://cdn.educba.com/academy/wp-content/uploads/2020/01/if-else-Statement-in-C-Flow.jpg>

### Syntax for if else statements

```
if condition:  
body of if  
else:  
body of else
```

### Example of if else statement-

```
#program to check if a num1 is less than num2  
  
num1, num2 = 6, 5  
  
if (num1 < num2):  
  
    print("num1 is less than num2")  
  
else:  
  
    print("num2 is less than num1")  
  
Output:num2 is lesser than num1
```

**Explanation-**In the above-given code, the if condition is False and hence the control shifts to the else block. Hence the statement written within the else block gets printed.

### Elif statement

The keyword **elif** is a combination of else and if. This statement works similar to 'else if' statements in the C and other languages. The statements in elif block get executed when the previous if condition is **False**.

### The elif statement flowchart:

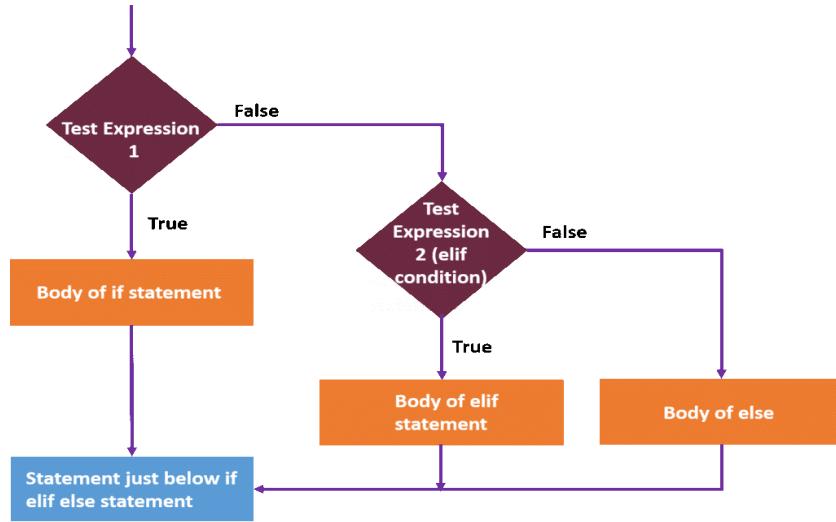


Image 19:Python - Elif Statement in Python.  
Reference:<https://intellipaat.com/mediaFiles/2019/02/python14.png>

### Syntax for elif statements

```
if condition:  
    body of if  
elif condition:  
    body of elif  
else:  
    body of else
```

### Example of elif statement-

```
num1, num2 = 5, 5  
  
if(num1 > num2):  
  
    print("num1 is greater than num2")  
  
elif(num1 == num2):
```

```
print("num1 is equal to num2")
```

```
else:
```

```
    print("num1 is less than num2")
```

```
Output:num1 is equal to num2
```

**Explanation:** Since both the values are equal, the if condition is False. Now the control shifts to elif statement and here the condition is True. Hence the statements in the elif block get executed.

### Nested if statement

One if statement inside another if statement is called Nested if statement. The structure and flow of execution can be assumed as shown below.

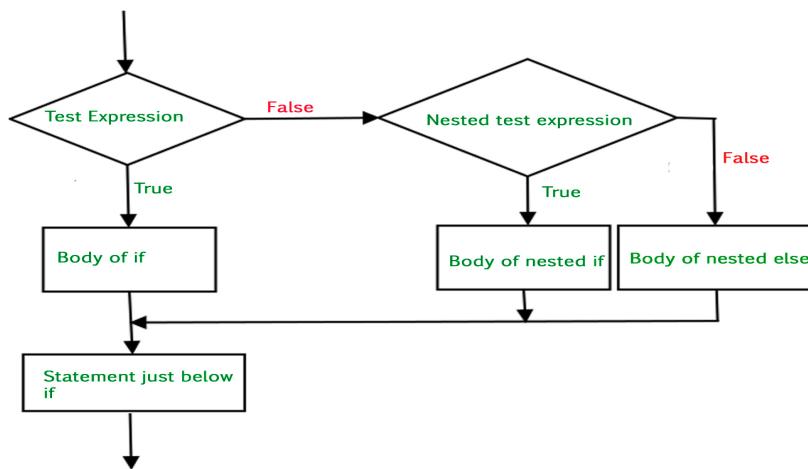


Image 20: Python - Nested if Statement in Python.  
Reference:<https://media.geeksforgeeks.org/wp-content/uploads/nested-if.jpg>

### Syntax for nested if statements

```
if condition1:  
    statements
```

```
if condition2:  
    statements  
else:  
    statements  
else:  
    statements
```

### Example of nested if statement-

```
num1 = 5  
  
if (num1 != 0):  
  
    if(num1 > 0):  
  
        print("num1 is a positive number")  
  
    else:  
  
        print("num1 is a negative number")  
  
    else:  
  
        print("num1 is neither positive nor negative")  
  
Output:num1 is a positive number
```

**Explanation:**Here, the first if condition is True. Hence the interpreter checks for the inner if condition. This is also True. Hence, the statements inside inner if get executed.

---

## Single Statement Suites

If the suite of an if clause consists only of a single line, it may go on the same line as the header statement.

Here is an example of a one-line if clause:

```
#!/usr/bin/python

var = 100

if ( var == 100 ) : print "Value of expression is 100"

print "Good bye!"
```

## OUTPUT-

```
Value of expression is 100

Good bye!
```

# Looping

Loops in Python programming function similar to loops in C, C++, Java or other languages. Python loops are used to repeatedly execute a block of statements until a given condition returns to be False. In Python, we have **3 different kinds of looping statements**, namely:

## While Loop in Python

While loop **iterates through a block of statements repetitively until the given condition returns False**. This means, while loop first checks the given condition and if the condition is **True**, then it executes the block of statements inside the while loop. Whenever the condition returns **False**, it breaks out of the loop and executes the next immediate line of code.

We use a while loop when we don't know the number of times to iterate through the block of statements.

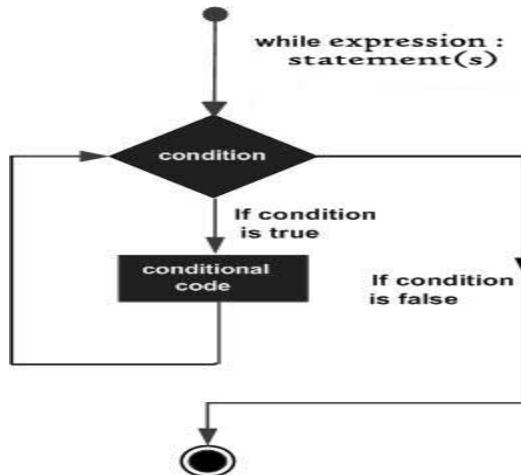


Image 21: Python - While Loop in Python.  
Reference: [https://www.tutorialspoint.com/python/images/python\\_while\\_loop.jpg](https://www.tutorialspoint.com/python/images/python_while_loop.jpg)

Syntax for while loop in Python:  
while expression:

---

statement(s)

---

**Example:** Let's say you want to calculate the sum of all numbers less than a given number (n). This is how a while loop would help

```
number = 7
sum = 0
i = 0
while (i < number):
    sum = sum + i
    i = i + 1
print (sum)
```

Output: 21

**Explanation:** Here, the while loop executes until  $i = 6$ , when  $i = 7$ , the condition  $i < \text{number}$  returns **False**. Hence it breaks out of the while loop and executes the immediate next line of code i.e the print statement.

## For Loop in Python

The **for loop in Python** is used to iterate over a sequence (list, tuple, string) or other iterable objects. Iterating over a sequence is called traversal.

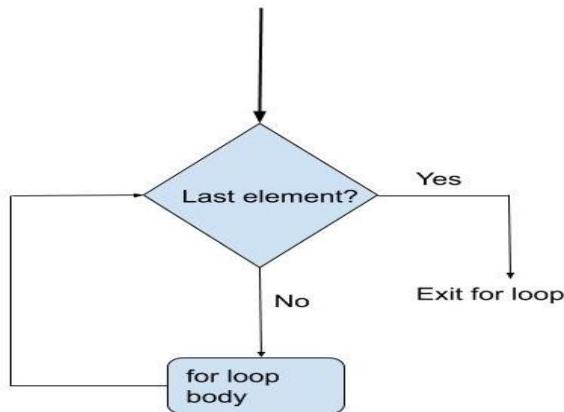


Image 22: Python - For Loop in Python.  
Reference:<https://cdn.askpython.com/wp-content/uploads/2019/07/for-loop-flow-diagram.jpg>

### Syntax of for loop-

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

### Example-

```
even_numbers = [2, 4, 6, 8, 10]  
for i in even_numbers:  
    print(even_numbers)
```

Ouput:  
[2, 4, 6, 8, 10]  
[2, 4, 6, 8, 10]  
[2, 4, 6, 8, 10]  
[2, 4, 6, 8, 10]  
[2, 4, 6, 8, 10]

---

**Example:** Now, let's say you want to print all the even numbers from 2 to 10. In this case, range function can be used this was

```
for i in range (2, 12, 2):  
    print (i)
```

Output:

```
2  
4  
6  
8  
10
```

**Explanation:** Here range function is used to iterate through the sequence 2 to 12 with a difference of 2 between the numbers. The syntax of range function used here is a range (lower limit, upper limit, the difference between numbers). In this case, the lower limit is inclusive and upper limit is exclusive of the sequence.

## Nested Loop in Python

**Loop inside a loop** is simply a nested loop. In Python, you can use any loop inside any other loop. For instance, a for in loop inside a while loop, a while inside for in and so on.

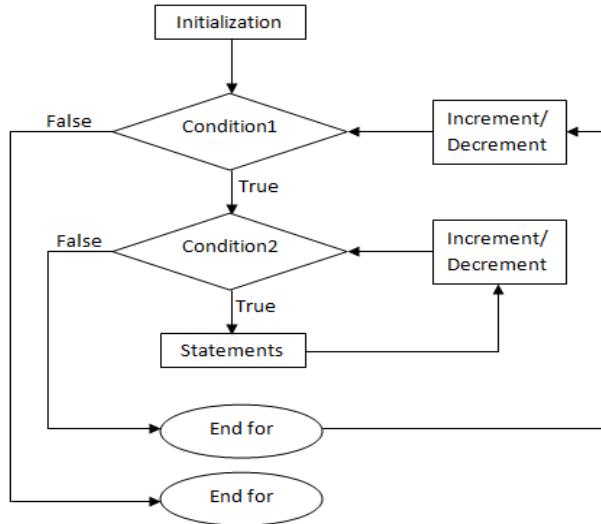


Image 23: Python - Nested Loop in Python.  
Reference:<https://i.stack.imgur.com/OBYkj.png>

Syntax for for loop inside for loop (nested for loop):

for variable in sequence:

    for variable in sequence:

        statement (s)

        statements(s)

Syntax for while loop inside while loop (nested while loop):

while condition:

    while condition:

        statement (s)

        statements(s)

**Example of nested loops in Python:** Let's say you want to print the below pattern.

```
1
1 2
1 2 3
1 2 3 4
```

```
#program to print a pattern
```

```
for i in range (1, 5):
    for j in range(i):
        print (j + 1, end = ' ')
    print ()
```

Output:

```
1
1 2
1 2 3
1 2 3 4
```

# Control Statements

Control statements in python are **used to control the flow of execution of the program based on the specified conditions**. Python supports **3 types of control statements** such as,

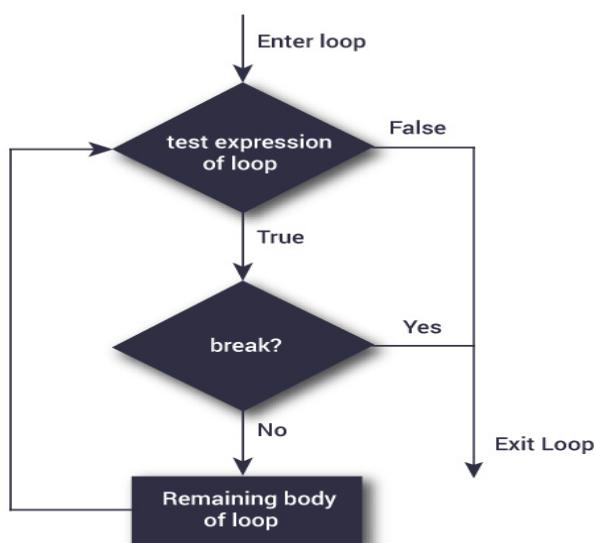
- 1) Break Statement
- 2) Continue Statement
- 3) Pass Statement

## Break Statement

The break statement in Python is used to terminate a loop. This means whenever the interpreter encounters the **break** keyword, **it simply exits out of the loop**. Once it breaks out of the loop, the control shifts to the immediate next statement.

Also, if the break statement is used inside a nested loop, it terminates the innermost loop and the control shifts to the next statement in the outer loop.

### Flowchart of Break Statement in Python:



---

Image 24: Python - Break Statement in Python.  
Reference:<https://cdn.programiz.com/sites/tutorial2program/files/flowchart-break-statement.jpg>

### Example of Break Statement in Python-

Here is an example of how break in Python can be used to exit out of the loop. Let's say you want to check if a given word contains the letter A, then how would you do it?

```
#program to check if letter 'A' is present in the input
a = input ("Enter a word")
for i in a:
    if (i == 'A'):
        print ("A is found")
        break
    else:
        print ("A not found")
```

Input: FACE Prep  
Output:  
A not found  
A is found

How is break statement useful in the above example? Our code accesses each character of the string and checks if the character is 'A'. Whenever it finds character 'A', it breaks out of the loop, hence avoiding the process of checking the remaining letters. So here break helps avoid unwanted looping.

### Continue Statement

Whenever the interpreter encounters a continue statement in Python, **it will skip the execution of the rest of the statements in that loop and proceed with the next iteration.** This means it returns the control to the beginning of the loop. Unlike the break statement, continue statement does not terminate or exit out of the loop. Rather, it continues with the next iteration. Here is the flow of execution when a continue statement is used.

#### Flowchart of Continue Statement in Python:

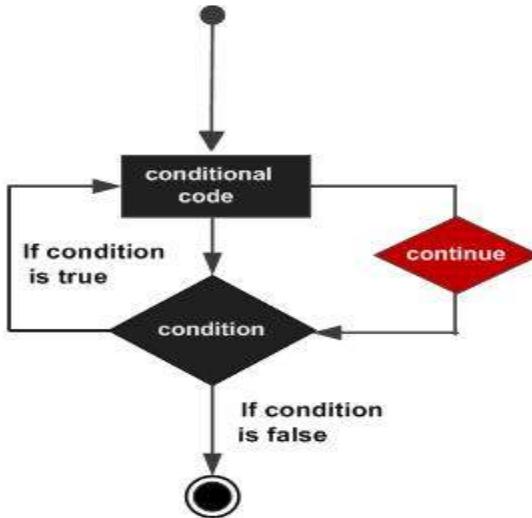


Image 25: Python - Continue Statement in Python.  
Reference:[https://www.tutorialspoint.com/python/images/cpp\\_continue\\_statement.jpg](https://www.tutorialspoint.com/python/images/cpp_continue_statement.jpg)

### Example of continue statement-

Let us see how the same above discussed example can be written using a continue statement. Here is the code.

```
#program to check if letter 'A' is present in the input

a = input ("Enter a word")

for i in a:

    if (i != 'A'):

        continue

    else:

        print ("A is found")
```

Input: Edunet Foundation

Output:

A is found

## Pass Statement

The pass statement is used to execute nothing in Python. It is a null statement in Python. It is used as a placeholder. It can be used when we have a loop or function or class that have not yet been implemented.

### Flowchart of Pass Statement in Python:

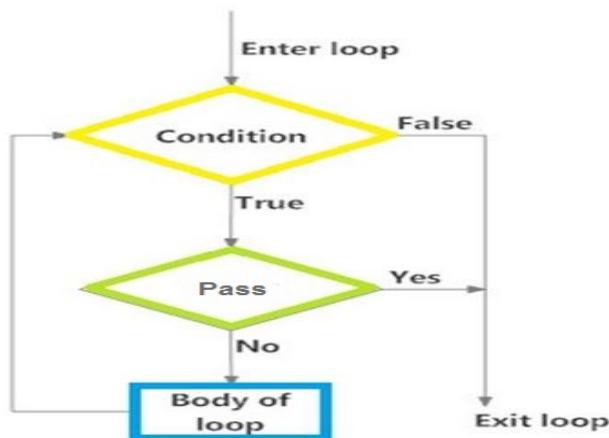


Image 26:Python - Pass Statement in Python.

Reference:<https://lh3.googleusercontent.com/proxy/1fqMIW6NoJy-E-ts0EQyJSodOhOfWHnSH81i2snZMRewSq8iUeYc5Q5hOn1LpWR6KeXjeHDDJEOf6cq609FqUoZmnSQ9AoZ1oRe1LXvKsCFJpl8>

Assume we have a loop that is not implemented yet, but needs to be implemented in the future. In this case, if you leave the loop empty, the interpreter will throw an error. To avoid this, you can use the **pass** statement to construct a block that does nothing i.e contains no statements. For example,

```
for i in 'FIVE':  
    if (i == 'A'):  
        pass  
    print (i)
```

Output:

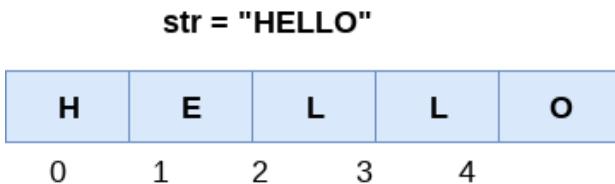
F  
I  
V  
E

---

# String Manipulation

## Introduction

- Python string is an ordered collection of characters which is used to represent and store the text-based information.
- Strings are stored as individual characters in a contiguous memory location.
- It can be accessed from both directions: forward and backward.
- Characters are nothing but symbols.
- Strings are immutable, which means that once a string is created, they cannot be changed.



str[0] = 'H'      str[:] = 'HELLO'

str[1] = 'E'      str[0:] = 'HELLO'

str[2] = 'L'      str[:5] = 'HELLO'

str[3] = 'L'      str[:3] = 'HEL'

str[4] = 'O'      str[0:2] = 'HE'

str[1:4] = 'ELL'

---

Image1 Source: <https://static.javatpoint.com/python/images/strings-indexing-and-splitting2.png>

## Create Strings

- Creating strings is easy as you only need to enclose the characters either in single or double-quotes.
- In the following example, we are providing different ways to initialize strings.
- To share an important note that you can also use triple quotes to create strings. However, programmers use them to mark multi-line strings and docstrings

```
#Python string examples - all assignments are identical.  
String_var = 'Python'  
String_var = "Python"  
String_var = """Python"""  
  
# with Triple quotes Strings can extend to multiple lines  
String_var = """ This document will help you to  
explore all the concepts  
of Python Strings!!! """  
  
# Replace "document" with "tutorial" and store in another variable  
substr_var = String_var.replace("document", "Tutorial")  
print (substr_var)
```

When you run the program, the output will be:

---

```
>>>
= RESTART: C:/Users/Ragavan/AppData/Local/Programs/Python/Python37/StringDemo.py
  This Tutorial will help you to
explore all the concepts
of Python Strings!!!
>>>
```

## Index and Slice Strings

### Index a String in Python

| P   | Y   | T   | H   | O  | N  | -  | S  | T  | R  | I  | N  | G  |
|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|
| 0   | 1   | 2   | 3   | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 |
| -13 | -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

Image2 Source: <https://cdn.techbeamers.com/wp-content/uploads/2016/03/String-Representation-in-Python.png>

- You need to know the index of a character to retrieve it from the String.
- Like most programming languages, Python allows to index from the zeroth position in Strings. But it also supports negative indexes. Index of '-1' represents the last character of the String. Similarly, using '-2', we can access the penultimate element of the string and so on.

---

```
sample_str = 'Python String'
print (sample_str[0])      # return 1st character
# output: P
print (sample_str[-1])    # return last character
# output: g
print (sample_str[-2])    # return last second character
# output: n
```

## Slice a String in Python

- To retrieve a range of characters in a String, we use ‘slicing operator,’ the colon ‘:’ sign. With the slicing operator, we define the range as [a:b]. It’ll let us print all the characters of the String starting from index ‘a’ up to char at index ‘b-1’. So the char at index ‘b’ is not a part of the output.

```
sample_str = 'Python String'
print (sample_str[3:5])          #return a range of character
# ho
print (sample_str[7:])         # return all characters from index 7
# String
print (sample_str[:6])          # return all characters before index 6
# Python
print (sample_str[7:-4])
# St
```

## Modify/Delete

- Python Strings are by design immutable. It suggests that once a String binds to a variable, it can’t be modified.
- If you want to update the String, then re-assign a new String value to the same variable.

```
sample_str = 'Python String'
sample_str[2] = 'a'

# TypeError: 'str' object does not support item assignment

sample_str = 'Programming String'
print (sample_str)

sample_str = "Python is the best scripting language."
del sample_str[1]
# TypeError: 'str' object doesn't support item deletion

del sample_str
print (sample_str)
# NameError: name 'sample_str' is not defined
```

- Similarly, we cannot modify the Strings by deleting some characters from it. Instead, we can remove the Strings altogether by using the ‘del’ command.

```
sample_str = "Python is the best scripting language."
del sample_str[1]
# TypeError: 'str' object doesn't support item deletion

del sample_str
print (sample_str)
# NameError: name 'sample_str' is not defined
```

## String Operators

There are three types of operators supported by a string, which are:

- Basic Operators (+, \*)
- Relational Operators (<, >≤, ≥, ==, !=)
- Membership Operators (in, not in)

**Concatenation (+)** - It combines two strings into one.

```
# example
var1 = 'Python'
var2 = 'String'
print (var1+var2)
# PythonString
```

**Repetition (\*)** - This operator creates a new string by repeating it a given number of times.

```
# example
var1 = 'Python'
print (var1*3)
# PythonPythonPython
```

**Slicing [ ]** - The slice operator prints the character at a given index.

```
# example
var1 = 'Python'
print (var1[2])
# t
```

**Range Slicing [x: y]** - It prints the characters present in the given range.

```
# example
var1 = 'Python'
print (var1[2:5])
# tho
```

---

**Membership (in)** - This operator returns ‘True’ value if the character is present in the given String.

```
# example
var1 = 'Python'
print ('n' in var1)
# True
```

**Membership (not in)** - It returns ‘True’ value if the character is not present in the given String.

```
# example
var1 = 'Python'
print ('N' not in var1)
# True
```

**Iterating (for)** - With this operator, we can iterate through all the characters of a string.

```
# example
for var in var1: print (var, end = "")
# Python
```

**Raw String (r/R)** - We can use it to ignore the actual meaning of Escape characters inside a string. For this, we add ‘r’ or ‘R’ in front of the String.

---

```
# example
print (r'\n')
# \n
print (R'\n')
# \n
```

## String Formatting Operators

### Python Escape Characters

- An Escape sequence starts with a backslash (\), which signals the compiler to treat it differently. Python subsystem automatically interprets an escape sequence irrespective of it is in a single-quoted or double-quoted Strings.
- Let's discuss an example-One of an important Escape sequence is to escape a single-quote or a double-quote.
- Suppose we have a string like – Python is a “widely” used language.
- The double-quote around the word “widely” disguise python that the String ends up there.
- We need a way to tell Python that the double-quotes inside the string are not the string markup quotes. Instead, they are the part of the String and should appear in the output.
- To resolve this issue, we can escape the double-quotes and single-quotes as:

```
print ("Python is a "widely" used language")
# SyntaxError: invalid syntax
# After escaping with double-quotes
print ("Python is a \"widely\" used language")
# Output: Python is a "widely" used language
```

## List of Escape Characters

- Here is the complete list of escape characters that are represented using backslash notation.

| Escape Sequence | Description                         |
|-----------------|-------------------------------------|
| \newline        | Backslash and newline ignored       |
| \\"             | Backslash                           |
| \'              | Single quote                        |
| \"              | Double quote                        |
| \a              | ASCII Bell                          |
| \b              | ASCII Backspace                     |
| \f              | ASCII Formfeed                      |
| \n              | ASCII Linefeed                      |
| \r              | ASCII Carriage Return               |
| \t              | ASCII Horizontal Tab                |
| \v              | ASCII Vertical Tab                  |
| \ooo            | Character with octal value ooo      |
| \xHH            | Character with hexadecimal value HH |

Image3 Source: <https://www.techbeamers.com/python-strings-functions-and-examples/#string-formatting-operators-in-python>

Here are some examples

```
print("C:\\\\Python32\\\\Lib")
print("This is printed\\n in two lines")
print("This is \\x48\\x45\\x58 representation")
```

---

## Output

```
C:\Python32\Lib
This is printed
in two lines
This is HEX representation
```

- Raw String to ignore escape sequence
- Sometimes we may wish to ignore the escape sequences inside a string. To do this we can place r or R in front of the string. This will imply that it is a raw string and any escape sequence inside it will be ignored.

```
print("This is \x61 \ngood example")
#Output : This is a
#Output : good example

print(r"This is \x61 \ngood example")
#Output : This is \x61 \ngood example
```

## Python Format Characters

- String '%' operator issued for formatting Strings. We often use this operator with the print() function.
- Here's a simple example.

```
print ("Employee Name: %s,\nEmployee Age:%d" % ('Alice',25))

# Employee Name: Alice,
# Employee Age: 25
```

---

## List of Format Symbols

Following is the table containing the complete list of symbols that you can use with the '%' operator.

| Symbol | Conversion                                    |
|--------|-----------------------------------------------|
| %c     | character                                     |
| %s     | string conversion via str() before formatting |
| %i     | signed decimal integer                        |
| %d     | signed decimal integer                        |
| %u     | unsigned decimal integer                      |
| %o     | octal integer                                 |
| %x     | hexadecimal integer (lowercase letters)       |
| %X     | hexadecimal integer (UPPER-case letters)      |
| %e     | exponential notation (with lowercase 'e')     |
| %E     | exponential notation (with UPPER-case 'E')    |
| %f     | floating-point real number                    |
| %g     | the shorter of %f and %e                      |
| %G     | the shorter of %f and %E                      |

Image Source : <https://www.techbeamers.com/python-strings-functions-and-examples/#string-formatting-operators-in-python>

## The `format()` Method for Formatting Strings

- The `format()` method that is available with the string object is very versatile and powerful in formatting strings. Format strings contains curly braces {} as placeholders or replacement fields which gets replaced.

- 
- We can use positional arguments or keyword arguments to specify the order.

```
# default(implicit) order
default_order = "{}, {} and {}".format('John','Bill','Sean')
print('\n--- Default Order ---')
print(default_order)

# order using positional argument
positional_order = "{1}, {0} and {2}".format('John','Bill','Sean')
print('\n--- Positional Order ---')
print(positional_order)

# order using keyword argument
keyword_order = "{s}, {b} and {j}".format(j='John',b='Bill',s='Sean')
print('\n--- Keyword Order ---')
print(keyword_order)
```

- The `format()` method can have optional format specifications. They are separated from field name using colon. For example, we can left-justify <, right-justify > or center ^ a string in the given space. We can also format integers as binary, hexadecimal etc. and floats can be rounded or displayed in the exponent format. There are a ton of formatting you can use.

---

```
# formatting integers
"Binary representation of {0} is {0:b}".format(12)

# formatting floats
"Exponent representation: {0:e}".format(1566.345)

# round off
"One third is: {0:.3f}".format(1/3)

# string alignment
"|{:<10}|{:^10}|{:>10}|".format('butter','bread','ham')
```

## Old style formatting

- We can even format strings like the old sprintf() style used in C programming language.  
We use the % operator to accomplish this.

```
x = 12.3456789
print('The value of x is %3.2f' %x)
The value of x is 12.35
print('The value of x is %3.4f' %x)
The value of x is 12.3457
```

---

## Common Python String Methods

### Unicode String support

- Regular Strings stores as the 8-bit ASCII value, whereas Unicode String follows the 16-bit ASCII standard. This extension allows the strings to include characters from the different languages of the world.
- In Python, the letter ‘u’ works as a prefix to distinguish between Unicode and usual strings.

```
print (u' Hello Python!!!')

#Hello Python
```

### Built-in String Functions

- There are numerous methods available with the string object. The format() method that we mentioned above is one of them. Some of the commonly used methods are lower(), upper(), join(), split(), find(), replace() etc. Here is a complete list of all the built-in methods to work with strings in Python.  
<https://docs.python.org/3/library/string.html>
- Built-in String Functions Example

---

```
input_word = input('Enter your string') #Input given PyThONtHeOrYmODuLe
print(input_word.isupper()) #Output : False
print(input_word.upper()) #Output : PYTHONTHEORYMODULE
print(input_word.islower()) #Output : False
print(input_word.isupper()) #Output : False
print(input_word.capitalize()) #Output : pythontheorymodule
print(input_word.split()) #Output : ['PyThONtHeOrYmODuLe']
print(input_word.split(',')) #Output : ['PyThONtHeOrYmODuLe']
print(input_word.title()) #Output : Pythontheorymodule
print(input_word.strip()) #Output : PyThONtHeOrYmODuLe
```

## Conversion Function

- `capitalize()` – Returns the string with the first character capitalized and rest of the characters in lower case.
- `lower()` – Converts all the characters of the String to lowercase
- `upper()` – Converts all the characters of the String to uppercase
- `swapcase()` – Swaps the case of every character in the String means that lowercase characters got converted to uppercase and vice-versa.
- `title()` – Returns the ‘titlecased’ version of String, which means that all words start with uppercase and the rest of the characters in words are in lowercase.
- `count( str[, beg [, end]])` – Returns the number of times substring ‘str’ occurs in the range [beg, end] if beg and end index are given else the search continues in full String Search is case-sensitive.

## Comparison Functions – Part

- `islower()` – Returns ‘True’ if all the characters in the String are in lowercase. If any of the char is in uppercase, it will return False.
- `isupper()` – Returns ‘True’ if all the characters in the String are in uppercase. If any of the char is in lowercase, it will return False.

- 
- `isdecimal()` – Returns ‘True’ if all the characters in String are decimal. If any character in the String is of other data-type, it will return False.
  - `isdigit()` – Returns ‘True’ for any char for which `isdecimal()` would return ‘True’ and some characters in the ‘No’ category. If there are any characters other than these, it will return False’.
  - `isnumeric()` – Returns ‘True’ if all the characters of the Unicode String lie in any one of the categories Nd, No, and NI. If there are any characters other than these, it will return False.

Precisely, Numeric characters are those for which Unicode property includes:  
`Numeric_Type=Digit`, `Numeric_Type=Decimal` or `Numeric_Type=Numeric`.

- `isalpha()` – Returns ‘True’ if String contains at least one character (non-empty String), and all the characters are alphabetic, ‘False’ otherwise.
- `isalnum()` – Returns ‘True’ if String contains at least one character (non-empty String), and all the characters are either alphabetic or decimal digits, ‘False’ otherwise.

## Padding Functions

- `rjust(width[,fillchar])` – Returns string filled with input char while pushing the original content on the right side. By default, the padding uses a space. Otherwise, ‘fillchar’ specifies the filler character.
- `ljust(width[,fillchar])` – Returns a padded version of String with the original String left-justified to a total of width columns. By default, the padding uses a space. Otherwise, ‘fillchar’ specifies the filler character.
- `center(width[,fillchar])` – Returns string filled with the input char while pushing the original content into the center. By default, the padding uses a space. Otherwise, ‘fillchar’ specifies the filler character.

- 
- `zfill(width)` – Returns string filled with the original content padded on the left with zeros so that the total length of String becomes equal to the input size. If there is a leading sign (+/-) present in the String, then with this function, padding starts after the symbol, not before it.

## Search Functions

- `find(str [,i [,j]])` – Searches for ‘str’ in complete String (if i and j not defined) or in a sub-string of String (if i and j are defined). This function returns the index if ‘str’ is found else returns ‘-1’. Here, i=search starts from this index, j=search ends at this index.
- `index(str[,i [,j]])` – This is same as ‘find’ method. The only difference is that it raises the ‘ValueError’ exception if ‘str’ doesn’t exist.
- `rfind(str[,i [,j]])` – This is same as `find()` just that this function returns the last index where ‘str’ is found. If ‘str’ is not found, it returns ‘-1’.
- `count(str[,i [,j]])` – Returns the number of occurrences of substring ‘str’ in the String. Searches for ‘str’ in the complete String (if i and j not defined) or in a sub-string of String (if i and j are defined). Where: i=search starts from this index, j=search ends at this index.  
var='This is a good example'

## String Substitution Functions

- `replace(old,new[,count])` – Replaces all the occurrences of substring ‘old’ with ‘new’ in the String.
  - If the count is available, then only ‘count’ number of occurrences of ‘old’ will be replaced with the ‘new’ var.
  - Where old =substring to replace, new =substring
- `split([sep[,maxsplit]])` – Returns a list of substring obtained after splitting the String with ‘sep’ as a delimiter.

- 
- Where, sep= delimiter, the default is space, maxsplit= number of splits to be done.
  - splitlines(num) – Splits the String at line breaks and returns the list after removing the line breaks.
    - Where num = if this is a positive value. It indicates that line breaks will appear in the returned list.
  - join(seq) – Returns a String obtained after concatenating the sequence ‘seq’ with a delimiter string.
    - Where: the seq= sequence of elements to join

## Misc String Functions

- lstrip([chars]) – Returns a string after removing the characters from the beginning of the String.
  - Where: Chars=this is the character to be trimmed from the String.
  - The default is whitespace character.
- rstrip() – Returns a string after removing the characters from the End of the String.
  - Where: Chars=this is the character to be trimmed from the String. The default is whitespace character.
- rindex(str[,i [,j]]) – Searches for ‘str’ in the complete String (if i and j not defined) or in a sub-string of String (if i and j are defined). This function returns the last index where ‘str’ is available.
  - If ‘str’ is not there, then it raises a ValueError exception.
  - Where: i=search starts from this index, j=search ends at this index.
- len(string) – Returns the length of given String

---

## Regular Expressions / Misc String functions

### What is Regular Expression?

- A regular expression in a programming language is a special text string used for describing a search pattern. It is extremely useful for extracting information from text such as code, files, log, spreadsheets or even documents.
- While using the regular expression the first thing is to recognize is that everything is essentially a character, and we are writing patterns to match a specific sequence of characters also referred as string. Ascii or latin letters are those that are on your keyboards and Unicode is used to match the foreign text.
- It includes digits and punctuation and all special characters like \$#@!%, etc.
- For instance, a regular expression could tell a program to search for specific text from the string and then to print out the result accordingly. Expression can include
  - Text matching
  - Repetition
  - Branching
  - Pattern-composition etc.

In Python, a regular expression is denoted as RE (REs, regexes or regex pattern) are imported through **re module**. Python supports regular expression through libraries. In Python regular expression supports various things like **Modifiers, Identifiers, and White space characters**.

| Identifiers | Modifiers | White space characters | Escape required |
|-------------|-----------|------------------------|-----------------|
|             |           |                        |                 |

|                                                                                |                                                                                              |                      |                              |
|--------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|----------------------|------------------------------|
| \d= any number (a digit)                                                       | \d represents a digit.Ex:<br>\d{1,5} it will declare digit between 1,5 like 424,444,545 etc. | \n = new line        | . + * ? [] \$<br>^ () {}   \ |
| \D= anything but a number (a non-digit)                                        | + = matches 1 or more                                                                        | \s= space            |                              |
| \s = space (tab,space,newline etc.)                                            | ? = matches 0 or 1                                                                           | \t = tab             |                              |
| \S= anything but a space                                                       | * = 0 or more                                                                                | \e = escape          |                              |
| \w = letters ( Match alphanumeric character, including "_")                    | \$ match end of a string                                                                     | \r = carriage return |                              |
| \W =anything but letters ( Matches a non-alphanumeric character excluding "_") | ^ match start of a string                                                                    | \f= form feed        |                              |
| . = anything but letters (periods)                                             | matches either or x/y                                                                        | -----                |                              |
| \b = any character except for new line                                         | [] = range or "variance"                                                                     | -----                |                              |

|    |                                        |       |  |
|----|----------------------------------------|-------|--|
| \. | {x} = this amount of<br>preceding code | ----- |  |
|----|----------------------------------------|-------|--|

## Regular Expression Syntax - RE

```
import re
```

- "re" module included with Python primarily used for string searching and manipulation
- Also used frequently for web page "Scraping" (extract large amount of data from websites)
- We will begin the expression tutorial with this simple exercise by using the expressions (w+) and (^).
- Example of w+ and ^ Expression
  - "^": This expression matches the start of a string
  - "w+": This expression matches the alphanumeric character in the string
- Here we will see an example of how we can use w+ and ^ expression in our code. We cover re.findall function later in this tutorial but for a while we simply focus on \w+ and \^ expression.

- 
- For example, for our string "python37, education is fun" if we execute the code with w+ and ^, it will give the output "python37".

```
import re
xx = "python37, education is fun"
r1 = re.findall(r"^\w+",xx)
print(r1)
```

- Remember, if you remove +sign from the w+, the output will change, and it will only give the first character of the first letter, i.e., [g]
- Example of \s expression in re.split function
- "s": This expression is used for creating a space in the string
- To understand how this regular expression works in Python, we begin with a simple example of a split function. In the example, we have split each word using the "re.split" function and at the same time we have used expression \s that allows to parse each word in the string separately.
- When you execute this code it will give you the output ['we', 'are', 'splitting', 'the', 'words'].
- Now, let see what happens if you remove "\\" from s. There is no 's' alphabet in the output, this is because we have removed '\' from the string, and it evaluates "s" as a regular character and thus split the words wherever it finds "s" in the string.
- Similarly, there are series of other regular expressions in Python that you can use in various ways in Python like \d,\D,\$,\.,\b, etc.
- Here is the complete code

---

```
import re
xx = "python37,education is fun"
r1 = re.findall(r"^\w+", xx)
print((re.split(r'\s','we are splitting the words'))))
print((re.split(r's','split the words')))
```

- Next, we will going to see the types of methods that are used with regular expressions.
- Using regular expression methods
- The "re" package provides several methods to actually perform queries on an input string.

The method we going to see are

- o re.match()
- o re.search()
- o re.findall()

**Note:** Based on the regular expressions, Python offers two different primitive operations. The match method checks for a match only at the beginning of the string while search checks for a match anywhere in the string.

### **Using re.match()**

- The match function is used to match the RE pattern to string with optional flags. In this method, the expression "`w+`" and "`\W`" will match the words starting with letter 'g' and thereafter, anything which is not started with 'g' is not identified. To check match for each element in the list or string, we run the forloop.

### **Finding Pattern in Text (re.search())**

- 
- A regular expression is commonly used to search for a pattern in a text. This method takes a regular expression pattern and a string and searches for that pattern with the string.
  - In order to use search() function, you need to import re first and then execute the code. The search() function takes the "pattern" and "text" to scan from our main string and returns a match object when the pattern is found or else not match if the pattern is not found.
  - For example here we look for two literal strings "Software testing" "python37", in a text string "Software Testing is fun". For "software testing" we found the match hence it returns the output as "found a match", while for word "python37" we could not found in string hence it returns the output as "No match".

### **re.findall()**

- **findall()** module is used to search for “all” occurrences that match a given pattern. In contrast, search() module will only return the first occurrence that matches the specified pattern. findall() will iterate over all the lines of the file and will return all non-overlapping matches of pattern in a single step.
- For example, here we have a list of e-mail addresses, and we want all the e-mail addresses to be fetched out from the list, we use the re.findall method. It will find all the e-mail addresses from the list.
- Here is the complete code

---

```
import re
list = ["python37 get", "python37 give", "guru Selenium"]
for element in list:
    z = re.match("(g\w+)\W(g\w+)", element)
    if z:
        print((z.groups()))

patterns = ['software testing', 'python37']
text = 'software testing is fun?'
for pattern in patterns:
    print('Looking for "%s" in "%s" ->' % (pattern, text), end=' ')
    if re.search(pattern, text):
        print('found a match!')
    else:
        print('no match')
abc = 'python37@google.com,careerpython37@hotmail.com,users@yahooemail.com'
emails = re.findall(r'[\w\.-]+@[\\w\.-]+', abc)
for email in emails:
    print(email)
```

---

# List

- The most commonly used data structure in Python is List. Python list is a container like an array that holds an ordered sequence of objects. The object can be anything from a string to a number or the data of any available type.
- A list can also be both homogenous as well as heterogeneous. It means we can store only integers or strings or both depending on the need. Next, every element rests at some position (i.e., index) in the list. The index can be used later to locate a particular item. The first indexes begin at zero, next are one, and so forth.

## Create a list in Python

- There are multiple ways to form a list in Python. Let's start with the most efficient one.
- Subscript operator
  - The square brackets [ ] represent the subscript operator in Python. It doesn't require a symbol lookup or a function call. Hence, it turns out the fastest way to create a list in Python.
- You can specify the elements inside [ ], separated by commas.
- Create a Python list using subscript operator

---

```
# empty list
my_list = []

# list of integers
my_list = [1, 2, 3]

# list with mixed datatypes
my_list = [1, "Hello", 3.4]

# nested list - Also, a list can even have another list as an item.
# This is called nested list.
my_list = ["mouse", [8, 4, 6], ['a']]
```

## List() constructor

- Python includes a built-in list() method a.k.a constructor.
- It accepts either a sequence or tuple as the argument and converts into a Python list.
- Let's start with an example to create a list without any element.
- Create Python list using list()

```
# Create Python list using list()
# Syntax
theList = list([n1, n2, ...] or [n1, n2, [x1, x2, ...]])
```

- We can supply a standard or nested sequence as the input argument to the list() function.  
Let's first create an empty list.

```
#theList = list()
#empty list
len(theList)
```

- Note- The len() function returns the size of the list.

### List comprehension

- Python supports a concept known as “List Comprehension.” It helps in constructing lists in an entirely natural and easy way.
- A list comprehension has the following syntax:

```
#Syntax - How to use List Comprehension
theList = [expression(iter) for iter in oldList if filter(iter)]
```

- It has square brackets grouping an expression followed by a for-in clause and zero or more if statements. The result will always be a list.
- Let's first see a simple example

```
theList = [iter for iter in range(5)]
print(theList)
```

---

## Creating a multi-dimensional list

- You can create a sequence with a pre-defined size by specifying an initial value for each element.

```
init_list = [0]*3
print(init_list)
[0, 0, 0]
```

With the above concept, you can build a two-dimensional list.

```
two_dim_list = [ [0]*3 ] *3
```

The above statement works, but Python will only create the references as sublists instead of creating separate objects.

```
two_dim_list = [ [0]*3 ] *3
print(two_dim_list)
Output: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
two_dim_list[0][2] = 1
print(two_dim_list)
Output:[[0, 0, 1], [0, 0, 1], [0, 0, 1]]
```

We changed the value of the third item in the first row, but the same column in other rows also got affected.

## Extending a list

- 
- Python allows lists to re-size in many ways. You can do that just by adding two or more of them.

```
L1 = ['a', 'b']
L2 = [1, 2]
L3 = ['Learn', 'Python']
L1 + L2 + L3
#Output : ['a', 'b', 1, 2, 'Learn', 'Python']
```

**List Extend ()** - Alternately, you can join lists using the extend() method.

```
L1 = ['a', 'b']
L2 = ['c', 'd']
L1.extend(L2)
print(L1)
#Output : ['a', 'b', 'c', 'd']
```

**List Append()** - Next, you can append a value to a list by calling the append() method. See the below example.

```
L1 = ['x', 'y']
L1.append(['a', 'b'])
L1
#Output :['x', 'y', ['a', 'b']]
```

## How to access elements from a list?

There are various ways in which we can access the elements of a list.

---

## List Index

- We can use the index operator [] to access an item in a list. Index starts from 0. So, a list having 5 elements will have index from 0 to 4.
- Nested list are accessed using nested indexing.

```
my_list = ['p', 'r', 'o', 'b', 'e']
# Output: p
print(my_list[0])

# Output: o
print(my_list[2])

# Output: e
print(my_list[4])

# Error! Only integer can be used for indexing
# my_list[4.0]

# Nested List
n_list = ["Happy", [2,0,1,5]]

# Nested indexing

# Output: a
print(n_list[0][1])

# Output: 5
print(n_list[1][3])
```

## Negative indexing

- Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

```
my_list = ['p', 'r', 'o', 'b', 'e']

# Output: e
print(my_list[-1])

# Output: p
print(my_list[-5])
```

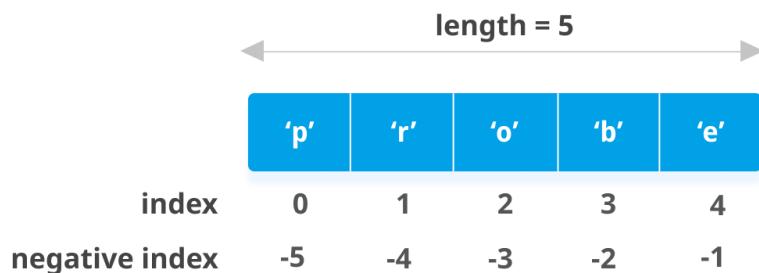


Image source: <https://cdn.programiz.com/sites/tutorial2program/files/python-list-index.png>

## Slice lists in Python

- We can access a range of items in a list by using the slicing operator (colon).

---

```
my_list = ['p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z']
# elements 3rd to 5th
print(my_list[2:5])

# elements beginning to 4th
print(my_list[:-5])

# elements 6th to end
print(my_list[5:])

# elements beginning to end
print(my_list[:])
```

- Slicing can be best visualized by considering the index to be between the elements as shown below. So if we want to access a range, we need two indices that will slice that portion from the list.

## Change or add elements to a list

- List are mutable, meaning, their elements can be changed unlike string or tuple.
- We can use assignment operator (=) to change an item or a range of items.

```
# mistake values
odd = [2, 4, 6, 8]

# change the 1st item
odd[0] = 1

# Output: [1, 4, 6, 8]
print(odd)

# change 2nd to 4th items
odd[1:4] = [3, 5, 7]

# Output: [1, 3, 5, 7]
print(odd)
```

We can add one item to a list using `append()` method or add several items using `extend()` method.

```
odd = [1, 3, 5]

odd.append(7)

# Output: [1, 3, 5, 7]
print(odd)

odd.extend([9, 11, 13])

# Output: [1, 3, 5, 7, 9, 11, 13]
print(odd)
```

We can also use `+` operator to combine two lists. This is also called concatenation.

---

```
odd = [1, 3, 5]

# Output: [1, 3, 5, 9, 7, 5]
print(odd + [9, 7, 5])

#Output: ["re", "re", "re"]
print(["re"] * 3)
```

The \* operator repeats a list for the given number of times.

```
odd = [1, 9]
odd.insert(1,3)

# Output: [1, 3, 9]
print(odd)

odd[2:2] = [5, 7]

# Output: [1, 3, 5, 7, 9]
print(odd)
```

## Delete or Remove elements from a list

- We can delete one or more items from a list using the keyword del. It can even delete the list entirely.

---

```
my_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']

# delete one item
del my_list[2]

# Output: ['p', 'r', 'b', 'l', 'e', 'm']
print(my_list)

# delete multiple items
del my_list[1:5]

# Output: ['p', 'm']
print(my_list)

# delete entire list
del my_list

# Error: List not defined
print(my_list)
```

- We can use remove () method to remove the given item or pop() method to remove an item at the given index.
- The pop() method removes and returns the last item if index is not provided. This helps us implement lists as stacks (first in, last out data structure).
- We can also use the clear() method to empty a list.

```
my_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']
my_list.remove('p')

# Output: ['r', 'o', 'b', 'l', 'e', 'm']
print(my_list)

# Output: 'o'
print(my_list.pop(1))

# Output: ['r', 'b', 'l', 'e', 'm']
print(my_list)

# Output: 'm'
print(my_list.pop())

# Output: ['r', 'b', 'l', 'e']
print(my_list)

my_list.clear()

# Output: []
print(my_list)
```

Finally, we can also delete items in a list by assigning an empty list to a slice of elements.

```
my_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']
my_list[2:3] = []
my_list
#Output:['p', 'r', 'b', 'l', 'e', 'm']
my_list[2:5] = []
my_list
#Output:['p', 'r', 'm']
```

---

## Python List Methods

- Methods that are available with list object in Python programming are tabulated below.
- They are accessed as list.method(). Some of the methods have already been used above.

### Python List Methods

- append() - Add an element to the end of the list
- extend() - Add all elements of a list to the another list
- insert() - Insert an item at the defined index
- remove() - Removes an item from the list
- pop() - Removes and returns an element at the given index
- clear() - Removes all items from the list
- index() - Returns the index of the first matched item
- count() - Returns the count of number of items passed as an argument
- sort() - Sort items in a list in ascending order
- reverse() - Reverse the order of items in the list
- copy() - Returns a shallow copy of the list

### Some examples of Python list methods

---

```
my_list = [3, 8, 1, 6, 0, 8, 4]
# Output: 1
print(my_list.index(8))

# Output: 2
print(my_list.count(8))

my_list.sort()

# Output: [0, 1, 3, 4, 6, 8, 8]
print(my_list)

my_list.reverse()

# Output: [8, 8, 6, 4, 3, 1, 0]
print(my_list)
```

## List Comprehension: Elegant way to create new List

- List comprehension is an elegant and concise way to create a new list from an existing list in Python.
- List comprehension consists of an expression followed by for statement inside square brackets.
- Here is an example to make a list with each item being increasing power of 2.

```
pow2 = [2 ** x for x in range(10)]
# Output: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
print(pow2)
```

This code is equivalent to

```
pow2 = []
for x in range(10):
    pow2.append(2 ** x)
```

- 
- A list comprehension can optionally contain more for or if statements. An optional if statement can filter out items for the new list. Here are some examples.

```
pow2 = [2 ** x for x in range(10) if x > 5]
pow2
#Output: [64, 128, 256, 512]

odd = [x for x in range(20) if x % 2 == 1]
odd
#Output: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

[x+y for x in ['Python ','C '] for y in ['Language','Programming']]
#Output: ['Python Language', 'Python Programming', 'C Language', 'C Programming']
```

## Other List Operations in Python

**List Membership Test** - We can test if an item exists in a list or not, using the keyword in.

```
my_list = ['p','r','o','b','l','e','m']

# Output: True
print('p' in my_list)

# Output: False
print('a' in my_list)

# Output: True
print('c' not in my_list)
```

**Iterating Through a List** - Using a for loop we can iterate though each item in a list.

---

```
for fruit in ['apple', 'banana', 'mango']:  
    print("I like", fruit)
```

---

# Tuples

- A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas, in a list, elements can be changed.

## Creating a Tuple

- A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. The parentheses are optional, however, it is a good practice to use them.
- A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).

```
# Empty tuple
my_tuple = ()
print(my_tuple) # Output: ()

# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple) # Output: (1, 2, 3)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple) # Output: (1, "Hello", 3.4)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
# Output: ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)
```

A tuple can also be created without using parentheses. This is known as tuple packing.

---

```
my_tuple = 3, 4.6, "dog"
print(my_tuple)    # Output: 3, 4.6, "dog"
# tuple unpacking is also possible
a, b, c = my_tuple
print(a)          # 3
print(b)          # 4.6
print(c)          # dog
```

- Creating a tuple with one element is a bit tricky.
- Having one element within parentheses is not enough. We will need a trailing comma to indicate that it is, in fact, a tuple.

```
my_tuple = ("hello")
print(type(my_tuple))    # <class 'str'>
# Creating a tuple having one element
my_tuple = ("hello",)
print(type(my_tuple))    # <class 'tuple'>
# Parentheses is optional
my_tuple = "hello",
print(type(my_tuple))    # <class 'tuple'>
```

## Advantages of Tuple over List

- Since tuples are quite similar to lists, both of them are used in similar situations as well.
- However, there are certain advantages of implementing a tuple over a list. Below listed are some of the main advantages:
  - We generally use tuple for heterogeneous (different) datatypes and list for homogeneous (similar) datatypes.
  - Since tuples are immutable, iterating through tuple is faster than with list. So there is a slight performance boost.

- o Tuples that contain immutable elements can be used as a key for a dictionary.  
With lists, this is not possible.
- o If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

## Access Tuple Elements

There are various ways in which we can access the elements of a tuple.

### 1. Indexing

- We can use the index operator [] to access an item in a tuple where the index starts from 0.
- So, a tuple having 6 elements will have indices from 0 to 5. Trying to access an element outside of tuple (for example, 6, 7,...) will raise an IndexError.
- The index must be an integer; so we cannot use float or other types. This will result in TypeError.
- Likewise, nested tuples are accessed using nested indexing, as shown in the example below.

```
my_tuple = ('p', 'e', 'r', 'm', 'i', 't')
print(my_tuple[0])      # 'p'
print(my_tuple[5])      # 't'
# IndexError: list index out of range
# print(my_tuple[6])
# Index must be an integer
# TypeError: list indices must be integers, not float
# my_tuple[2.0]
# nested tuple
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
# nested index
print(n_tuple[0][3])      # 's'
print(n_tuple[1][1])      # 4
```

## 2. Negative Indexing

- Python allows negative indexing for its sequences.
- The index of -1 refers to the last item, -2 to the second last item and so on.

```
my_tuple = ('p', 'e', 'r', 'm', 'i', 't')
# Output: 't'
print(my_tuple[-1])
# Output: 'p'
print(my_tuple[-6])
```

## 3. Slicing

- We can access a range of items in a tuple by using the slicing operator - colon ":".

```
my_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
# elements 2nd to 4th
# Output: ('r', 'o', 'g')
print(my_tuple[1:4])
# elements beginning to 2nd
# Output: ('p', 'r')
print(my_tuple[:-7])
# elements 8th to end
# Output: ('i', 'z')
print(my_tuple[7:])
# elements beginning to end
# Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
print(my_tuple[:])
```

Slicing can be best visualized by considering the index to be between the elements as shown below. So if we want to access a range, we need the index that will slice the portion from the tuple.

---

## Performing Operations - Modifying, Deleting Python Tuple

### 1. Changing a Tuple

- Unlike lists, tuples are immutable.
- This means that elements of a tuple cannot be changed once it has been assigned.  
But, if the element is itself a mutable datatype like list, its nested items can be changed.
- We can also assign a tuple to different values (reassignment).

```
my_tuple = (4, 2, 3, [6, 5])
# TypeError: 'tuple' object does not support item assignment
# my_tuple[1] = 9
# However, item of mutable element can be changed
my_tuple[3][0] = 9      # Output: (4, 2, 3, [9, 5])
print(my_tuple)
# Tuples can be reassigned
my_tuple = ('p','r','o','g','r','a','m','i','z')
# Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
print(my_tuple)
```

- We can use + operator to combine two tuples. This is also called concatenation.
- We can also repeat the elements in a tuple for a given number of times using the \* operator.
- Both + and \* operations result in a new tuple.

```
# Concatenation
# Output: (1, 2, 3, 4, 5, 6)
print((1, 2, 3) + (4, 5, 6))
# Repeat
# Output: ('Repeat', 'Repeat', 'Repeat')
print(("Repeat",) * 3)
```

## 2. Deleting a Tuple

- We cannot change the elements in a tuple. That also means we cannot delete or remove items from a tuple.
- But deleting a tuple entirely is possible using the keyword del.

```
my_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
# can't delete items
# TypeError: 'tuple' object doesn't support item deletion
# del my_tuple[3]
# Can delete an entire tuple
del my_tuple
# NameError: name 'my_tuple' is not defined
print(my_tuple)
```

## Tuple Methods

- Methods that add items or remove items are not available with tuple. Only the following two methods are available.
- Python Tuple Methods

- **count(x)**      Returns the number of items x
  - **index(x)**      Returns the index of the first item that is equal to x

---

Some examples of Python tuple methods:

```
my_tuple = ('a','p','p','l','e',)
print(my_tuple.count('p'))  # Output: 2
print(my_tuple.index('l'))  # Output: 3
```

## Other Tuple Operations

- Tuple Membership Test - We can test if an item exists in a tuple or not, using the keyword **in**.

```
my_tuple = ('a','p','p','l','e',)
# In operation
# Output: True
print('a' in my_tuple)
# Output: False
print('b' in my_tuple)
# Not in operation
# Output: True
print('g' not in my_tuple)
```

- Iterating Through a Tuple - Using a **for loop** we can iterate through each item in a tuple.

```
for name in ('John','Kate'):
    print("Hello",name)

# Output:
Hello John
Hello Kate
```

---

# Functions and Methods

## What Is a Function in Python?

- A function in Python is a logical unit of code containing a sequence of statements indented under a name given using the “def” keyword.
- Functions allow you to create a logical division of a big project into smaller modules. They make your code more manageable and extensible.
- Basically, there are three types of functions:
  - Python Built-in functions (an already created, or predefined, function)
  - User-defined function (a function created by users as per the requirements)
  - Anonymous function (a function having no name)

### Function in Python - Create & Def Statement

#### Create a Function – Syntax

```
#Single line function:  
def single_line(): statement  
  
#Python function with docstring:  
def fn(arg1, arg2,...):  
    """docstring"""  
    statement1  
    statement2  
  
#Nested Python function:  
def fn(arg1, arg2,...):  
    """docstring"""  
    statement1  
    statement2  
    def fn_new(arg1, arg2,...):  
        statement1  
        statement2  
    ...  
    ...
```

## Def Statement

- The “def” keyword is a statement for defining a function in Python.
- You start a function with the def keyword, specify a name followed by a colon (:) sign.
- The “def” call creates the function object and assigns it to the name given.
- You can further re-assign the same function object to other names.
- Give a unique name to your function and follow the same rules as naming the identifiers.
- Add a meaningful docstring to explain what the function does. However, it is an optional step.
- Now, start the function body by adding valid Python statements each indented with four spaces.
- You can also add a statement to return a value at the end of a function. However, this step is optional.

- 
- Since def is a statement, so you can use it anywhere a statement can appear – such as nested in an if clause or within another function.
  - Example :

```
if test:  
def test(): # First definition  
...  
else:  
def test(): # Alternate definition  
...  
...
```

## Call a Function

### Example of a Function Call & Polymorphism in Python

#### How to Call a Function in Python?

- By using the def keyword, you learned to create the blueprint of a function which has a name, parameters to pass and a body with valid Python statements.
- You can do so by calling it from the Python script or inside a function or directly from the Python shell.
- To call a function, you need to specify the function name with relevant parameters, and that's it.
- Follow the below example to learn how to call a function in Python.
- Python function for modular programming

Example of a Function Call:

---

It's a simple example where a function “typeOfNum()” has nested functions to decide on a number is either odd or even.

```
def typeOfNum(num): # Function header
    # Function body
    if num % 2 == 0:
        def message():
            print("You entered an even number.")
        else:
            def message():
                print("You entered an odd number.")
            message()
        # End of function
    typeOfNum(2) # call the function
    typeOfNum(3) # call the function again
```

## Polymorphism in Python

- In Python, functions polymorphism is possible as we don't specify the argument types while creating functions.
- The behavior of a function may vary depending upon the arguments passed to it.
- The same function can accept arguments of different object types.
- If the objects find a matching interface, the function can process them.

Example

```
def product(x, y) : return x * y
print(product(4, 9)) # function returns 36
print(product('Python!', 2)) # function returns
# Python!Python!
print(product('Python 2 or 3?', '3')) # TypeError occurs
```

---

The above example clarifies that we can pass any two objects to the product() function which supports the '\*' operator.

Some points which you should remember are as follows:

- Python is a dynamically typed language which means the types correlate with values, not with variables. Hence, the polymorphism runs unrestricted.
- That's one of the primary differences between Python and other statically typed languages such as C++ or Java.
- In Python, you don't have to mention the specific data types while coding.
- However, if you do, then the code limits to the types anticipated at the time of coding.
- Such code won't allow other compatible types that may require in the future.
- Python doesn't support any form of function overloading.

## How to Pass Parameters to a Function

### Parameters in a Function

- Parameters are the variables used in the function definition whereas arguments are the values we pass to the function parameters.
- Python supports different variations of passing parameters to a function.
- The argument gets assigned to the local variable name once passed to the function.
- Changing the value of an argument inside a function doesn't affect the caller.
- If the argument holds a mutable object, then changing it in a function impacts the caller.
- We call the passing of immutable arguments as Pass by Value because Python doesn't allow them to change in place.
- The passing of mutable arguments happens to be Pass by Pointer in Python because they are likely to be affected by the changes inside a function.

---

### Example: Immutable vs. Mutable

```
def test1(a, b) :  
    a = 'Garbage' # 'a' receives an immutable object  
    b[0] = 'Python' # 'b' receives a list object  
    # list is mutable  
    # it can undergo an in place change  
def test2(a, b) :  
    a = 'Garbage 2'  
    b = 'Python 3' # 'b' now is made to refer to new  
    # object and therefore argument 'y'  
    # is not changed  
    arg1 = 10  
    arg2 = [1, 2, 3, 4]  
    test1(arg1, arg2)  
    print("After executing test 1 =>", arg1, arg2)  
    test2(arg1, arg2)  
    print("After executing test 2 =>", arg1, arg2)  
  
#After execution, the above code prints the following.  
  
After executing test 1 => 10 ['Python', 2, 3, 4]  
After executing test 2 => 10 ['Python', 2, 3, 4]
```

### Example: How to avoid changing the mutable argument

```
def test1(a, b) :  
    a = 'Garbage'  
    b[0] = 'Python'  
    arg1 = 10  
    arg2 = [1, 2, 3, 4]  
    print("Before test 1 =>", arg1, arg2)  
    test1(arg1, arg2[:]) # Create an explicit copy of mutable object  
    # 'y' in the function.  
    # Now 'b' in test1() refers to a  
    # different object which was initially a  
    # copy of 'arg2'  
    print("After test 1  =>", arg1, arg2)  
  
#After execution, the above code prints the following.  
  
Before test 1 => 10 [1, 2, 3, 4]  
After test 1  => 10 [1, 2, 3, 4]
```

## Standard arguments

- The standard arguments are those which you pass as specified in a Python function definition. It means without changing their order and without skipping any of them.

```
def fn(value):  
    print(value)  
    return  
fn()
```

- Executing the above code throws the below error as we've not passed the single argument required.
- TypeError: fn() missing 1 required positional argument: 'value'

---

## Keyword-based arguments

- When you assign a value to the parameter (such as param=value) and pass to the function (like fn(param=value)), then it turns into a keyword argument.
- If you pass the keyword arguments to a function, then Python determines it through the parameter name used in the assignment.
- See the below example.

```
def fn(value):
    print(value)
    return
fn(value=123) # output => 123
fn(value="Python!") # output => Python!
```

While using keyword arguments, you should make sure that the name in the assignment should match with the one in the function definition. Otherwise, Python throws the `TypeError` as shown below.

```
fn(value1="Python!") # wrong name used in the keyword argument
```

The above function call causes the following error.

```
TypeError: fn() got an unexpected keyword argument 'value1'
```

## Arguments with Default Values

- Python functions allow setting the default values for parameters in the function definition. We refer them as the default arguments.

- 
- The callee uses these default values when the caller doesn't pass them in the function call.
  - The below example will help you clearly understand the concept of default arguments.

```
def daysInYear(is_leap_year=False):
    if not is_leap_year:
        print("365 days")
    else:
        print("366 days")
    return
daysInYear()
daysInYear(True)
```

Here, the parameter “is\_leap\_year” is working as a default argument. If you don’t pass any value, then it assumes the default which is False.

The output of the above code is:

365 days

366 days

## Variable Arguments

- You may encounter situations when you have to pass additional arguments to a Python function. We refer them as variable-length arguments.
- The Python’s print() is itself an example of such a function which supports variable arguments.
- To define a function with variable arguments, you need to prefix the parameter with an asterisk (\*) sign. Follow the below syntax.

---

```
def fn([std_args,] *var_args_tuple ):
    """docstring"""
    function_body
    return_statement
    Check out the below example for better clarity.
    def inventory(category, *items):
        print("%s [items=%d]: " % (category, len(items)), items)
        for item in items:
            print("-", item)
    return
inventory('Electronics', 'tv', 'lcd', 'ac', 'refrigerator', 'heater')
inventory('Books', 'python', 'java', 'c', 'c++')
```

## Global, Local and Nonlocal variables

### Global Variables

- In Python, a variable declared outside of the function or in global scope is known as a global variable. This means that a global variable can be accessed inside or outside of the function.

Example 1: Create a Global Variable

```
x = "global"
def foo():
    print("x inside:", x)
    foo()
    print("x outside:", x)

#Output be:
x inside: global
x outside: global
```

In the above code, we created x as a global variable and defined a foo() to print the global variable x. Finally, we call the foo() which will print the value of x.

---

What if you want to change the value of x inside a function?

```
x = "global"
def foo():
    x = x * 2
    print(x)
foo()

#Output be:

UnboundLocalError:
local variable 'x' referenced before assignment
```

- The output shows an error because Python treats x as a local variable and x is also not defined inside foo().

## Local Variables

- A variable declared inside the function's body or in the local scope is known as local variable.

Accessing local variable outside the scope

```
def foo():
    y = "local"
    foo()
    print(y)

#Output
NameError: name 'y' is not defined
```

---

## Create a Local Variable

- Normally, we declare a variable inside the function to create a local variable.

```
def foo():
    y = "local"
    print(y)
foo()

#Output:
local
```

## Global and local variables

Here, we will show how to use global variables and local variables in the same code.

Using Global and Local variables in the same code

```
x = "global"
def foo():
    global x
    y = "local"
    x = x * 2
    print(x)
    print(y)
foo()

#Output
global global
local
```

- 
- In the above code, we declare x as a global and y as a local variable in the foo(). Then, we use multiplication operator \* to modify the global variable x and we print both x and y.
  - After calling the foo(), the value of x becomes global global because we used the x \* 2 to print two times global. After that, we print the value of local variable y i.e local.

Example: Global variable and Local variable with same name

```
x = 5
def foo():
    x = 10
    print("local x:", x)
foo()
print("global x:", x)

#Output
local x: 10
global x: 5
```

In the above code, we used the same name x for both global variable and local variable. We get a different result when we print the same variable because the variable is declared in both scopes, i.e. the local scope inside foo() and global scope outside foo().

When we print the variable inside foo() it outputs local x: 10. This is called the local scope of the variable.

Similarly, when we print the variable outside the foo(), it outputs global x: 5. This is called the global scope of the variable.

## Nonlocal Variables

- 
- Nonlocal variable are used in nested function whose local scope is not defined. This means that the variable can be neither in the local nor the global scope.
  - We use nonlocal keyword to create nonlocal variable.

Example: Create a nonlocal variable

```
def outer():
    x = "local"
def inner():
    nonlocal x
    x = "nonlocal"
    print("inner:", x)
    inner()
    print("outer:", x)
outer()
#Output
inner: nonlocal
outer: nonlocal
```

In the above code, there is a nested function inner(). We use nonlocal keyword to create a nonlocal variable. The inner() function is defined in the scope of another function outer().

**Note:** If we change value of nonlocal variable, the changes appear in the local variable.

## Global Keyword

### What is the global keyword

- In Python, global keyword allows you to modify the variable outside of the current scope. It is used to create a global variable and make changes to the variable in a local context.
- Rules of global Keyword

- 
- The basic rules for global keyword in Python are:
    - When we create a variable inside a function, it is local by default.
    - When we define a variable outside of a function, it is global by default. You don't have to use global keyword.
    - We use global keyword to read and write a global variable inside a function.
    - Use of global keyword outside a function has no effect.
    - Use of global Keyword

Example 1: Accessing global Variable from inside a Function

```
c = 1 # global variable
def add():
    print(c)
add()
```

Example 2: Modifying Global Variable From Inside the Function

```
c = 1 # global variable
def add():
    c = c + 2 # increment c by 2
    print(c)
add()
```

Example 3: Changing Global Variable from inside a Function using global

```
c = 0 # global variable
def add():
    global c
    c = c + 2 # increment by 2
    print("Inside add():", c)
add()
print("In main:", c)
```

## Global in Nested Functions

- Here is how you can use a global variable in nested function.

Example: Using a Global Variable in Nested Function

```
def foo():
    x = 20
    def bar():
        global x
        x = 25
        print("Before calling bar: ", x)
        print("Calling bar now")
        bar()
        print("After calling bar: ", x)
    foo()
    print("x in main: ", x)
```

## Name Resolution in a Python Function

- It is essential to understand how name resolution works in case of a def statement.
- Here are a few points you should keep in mind.
- The name assignments create or change local names.
- The LEGB rule comes in the picture for searching the name reference.
  - o local – L

- 
- o then enclosing functions (if any) – E
  - o next comes the global – G
  - o and the last one is the built-in – B

To gain more understanding, run through the below example:

```
#var = 5
def fn1():
    #var = [3, 5, 7, 9]
    def fn2() :
        #var = (21, 31, 41)
        print(var)
    fn2()
    #uncomment var assignments one-by-one and check the output
fn1()
print(var)
#After uncommenting the first "var" assignment
#the output is:
5
5
#Next, after uncommenting the second "var"
#assignment as well, the output is:
[3, 5, 7, 9]
5
#Finally, if we uncomment the last "var" assignment,
#then the result is as follows.
(21, 31, 41)
5
```

## Scope Lookup in Functions

- Python functions can access names in all available enclosing def statements..

```
x = 101 # global scope name - unused
def fn1():
    x = 102 # Enclosing def local
    def fn2():
        print(x) # Reference made in nested def
    fn2() # Prints 102: enclosing def local
fn1()
```

---

The scope lookup remains in action even if the enclosing function has already returned.

```
def fn1():
    print('In fn1')
    X = 100
    def fn2():
        print('In fn2')
        print(X) # Remembers X in enclosing def scope
        return fn2 # Return fn2 but don't call it
    action = fn1() # Make, return function
    action() # Call fn2() now: prints 100
```

## Return Values

- In Python functions, you can add the “return” statement to return a value.
- Usually, the functions return a single value. But if required, Python allows returning multiple values by using the collection types such as using a tuple or list.
- This feature works like the call-by-reference by returning tuples and assigning the results back to the original argument names in the caller.

```
def returnDemo(val1, val2):
    val1 = 'Windows'
    val2 = 'OS X'
    # return multiple values in a tuple
    return val1, val2
var1 = 4
var2 = [2, 4, 6, 8]
print("before return =>", var1, var2)
var1, var2 = returnDemo(var1, var2)
print("after return =>", var1, var2)

#Output

before return => 4 [2, 4, 6, 8]
after return => Windows OS X
```

## Function Examples

- **General Function** - Check out a general function call example.

```
def getMin(*varArgs):
    min = varArgs[0]
    for i in varArgs[1:]:
        if i < min :
            min = i
        return min
    min = getMin(21, -11, 17, -23, 6, 5, -89, 4, 9)
    print(min)
#The output is as follows.
-89
```

- **Recursive Function** - Example of the recursive function.

---

```
def calcFact(num):
    if(num != 1):
        return num * calcFact(num-1)
    else:
        return 1
print(calcFact(4))
```

## Functions as Objects

- Yes, Python treats everything as an object and functions are no different.
- You can assign a function object to any other names.

```
def testFunc(a, b) : print('testFunc called')
fn = testFunc
fn(22, 'bb')
```

You can even pass the function object to other functions.

```
def fn1(a, b) : print('fn1 called')
def fn2(fun, x, y) : fun(x, y)
fn2(fn1, 22, 'bb')
```

You can also embed a function object in data structures.

---

```
def fn1(a) : print('fn1', a)
def fn2(a) : print('fn2', a)
listOfFuncs = [(fn1, "First function"), (fn2, "Second function")]
for (f, arg) in listOfFuncs : f(arg)
```

You can return a function object from another function.

```
def FuncLair(produce) :
    def fn1() : print('fn1 called')
    def fn2() : print('fn2 called')
    def fn3() : print('fn3 called')
    if produce == 1 : return fn1
    elif produce == 2 : return fn2
    else : return fn3
    f = FuncLair(2) ; f()
```

## Function Attributes

- Python functions also have attributes.
- You can list them via the `dir()` built-in function.
- The attributes can be system-defined.
- Some of them can be user-defined as well.
- The `dir()` function also lists the user-defined attributes.

```
def testFunc():
    print("I'm just a test function.")
    testFunc.attr1 = "Hello"
    testFunc.attr2 = 5
    testFunc()
```

- 
- You can utilize the function attributes to archive state information instead of using any of the globals or nonlocals names.
  - Unlike the nonlocals, attributes are accessible anywhere the function itself is, even from outside its code.

---

## Dictionary

Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair.

- Dictionaries are optimized to retrieve values when the key is known.

### Create a dictionary

- Creating a dictionary is as simple as placing items inside curly braces {} separated by comma.
- An item has a key and the corresponding value expressed as a pair, key: value.
- While values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique.

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

# using dict()
my_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

We can also create a dictionary using the built-in function dict().

---

## Access Items in a Dictionary

- While indexing is used with other container types to access values, dictionary uses keys.  
Key can be used either inside square brackets or with the get() method.
- The difference while using get() is that it returns None instead of KeyError, if the key is not found.

```
my_dict = {'name':'Alice', 'age': 26}
# Output: Jack
print(my_dict['name'])
# Output: 26
print(my_dict.get('age'))
# Trying to access keys which doesn't exist throws error
# my_dict.get('address')
# my_dict['address']

#output
Alice
26
```

## Change or Add elements in a dictionary

- Dictionaries are mutable. We can add new items or change the value of existing items using assignment operator.
- If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

---

```
my_dict = {'name':'Alice', 'age': 26}
# update value
my_dict['age'] = 27
#Output: {'age': 27, 'name': 'Alice', }
print(my_dict)
# add item
my_dict['address'] = 'Downtown'
# Output: {'address': 'Downtown', 'age': 27, 'name': 'Alice', }
print(my_dict)

#Output
{'name': 'Alice', 'age': 27}
{'name': 'Alice', 'age': 27, 'address': 'Downtown'}
```

## Delete or remove elements from a dictionary

- We can remove a particular item in a dictionary by using the method `pop()`. This method removes an item with the provided key and returns the value.
- The method, `popitem()` can be used to remove and return an arbitrary item (key, value) from the dictionary. All the items can be removed at once using the `clear()` method.
- We can also use the `del` keyword to remove individual items or the entire dictionary itself.

```
# create a dictionary
squares = {1:1, 2:4, 3:9, 4:16, 5:25}
# remove a particular item
# Output: 16
print(squares.pop(4))
# Output: {1: 1, 2: 4, 3: 9, 5: 25}
print(squares)
# remove an arbitrary item
# Output: (1, 1)
print(squares.popitem())
# Output: {2: 4, 3: 9, 5: 25}
print(squares)
# delete a particular item
del squares[5]
# Output: {2: 4, 3: 9}
print(squares)
# remove all items
squares.clear()

#Output
16
{1: 1, 2: 4, 3: 9, 5: 25}
(1, 1)
{2: 4, 3: 9, 5: 25}
{2: 4, 3: 9}
{}
```

## Python Dictionary Methods

Methods that are available with dictionary are tabulated below. Some of them have already been used in the above examples.

- `clear()` - Remove all items form the dictionary.
- `copy()` - Return a shallow copy of the dictionary.
- `fromkeys(seq[, v])` - Return a new dictionary with keys from seq and value equal to v (defaults to None).
- `get(key[,d])` - Return the value of key. If key doesnot exit, return d (defaults to None).
- `items()` - Return a new view of the dictionary's items (key, value).
- `keys()` - Return a new view of the dictionary's keys.

- 
- `pop(key[,d])` - Remove the item with key and return its value or d if key is not found. If d is not provided and key is not found, raises `KeyError`.
  - `popitem()` - Remove and return an arbitrary item (key, value). Raises `KeyError` if the dictionary is empty.
  - `setdefault(key[,d])` - If key is in the dictionary, return its value. If not, insert key with a value of d and return d (defaults to `None`).
  - `update([other])` - Update the dictionary with the key/value pairs from other, overwriting existing keys.
  - `values()` - Return a new view of the dictionary's values

```
marks = {}.fromkeys(['Math', 'English', 'Science'], 0)
# Output: {'English': 0, 'Math': 0, 'Science': 0}
print(marks)
for item in marks.items():
    print(item)
# Output: ['English', 'Math', 'Science']
list(sorted(marks.keys()))
```

## Dictionary Comprehension

- Dictionary comprehension is an elegant and concise way to create new dictionary from an iterable in Python.
- Dictionary comprehension consists of an expression pair (key: value) followed by for statement inside curly braces {}.
- Here is an example to make a dictionary with each item being a pair of a number and its square.

---

```
squares = {x: x*x for x in range(6)}
# Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
print(squares)

#This code is equivalent to

squares = {}
for x in range(6):
    squares[x] = x*x
```

- A dictionary comprehension can optionally contain more for or if statements.
- An optional if statement can filter out items to form the new dictionary.
- Here are some examples to make dictionary with only odd items.

```
odd_squares = {x: x*x for x in range(11) if x%2 == 1}
# Output: {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print(odd_squares)
```

## Other Dictionary Operations

**Dictionary Membership Test** - We can test if a key is in a dictionary or not using the keyword `in`. Notice that membership test is for keys only, not for values.

---

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
# Output: True
print(1 in squares)
# Output: True
print(2 not in squares)
# membership tests for key only not value
# Output: False
print(49 in squares)
```

**Iterating Through a Dictionary** - Using a for loop we can iterate though each key in a dictionary.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
for i in squares:
    print(squares[i])
```

## Built-in Functions with Dictionary

Built-in functions like all(), any(), len(), cmp(), sorted() etc. are commonly used with dictionary to perform different tasks.

- all() - Return True if all keys of the dictionary are true (or if the dictionary is empty).
- any() - Return True if any key of the dictionary is true. If the dictionary is empty, return False.
- len() - Return the length (the number of items) in the dictionary.
- cmp() - Compares items of two dictionaries.
- sorted() - Return a new sorted list of keys in the dictionary.

---

Here are some examples that uses built-in functions to work with dictionary.

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
# Output: 5
print(len(squares))
# Output: [1, 3, 5, 7, 9]
print(sorted(squares))
```

---

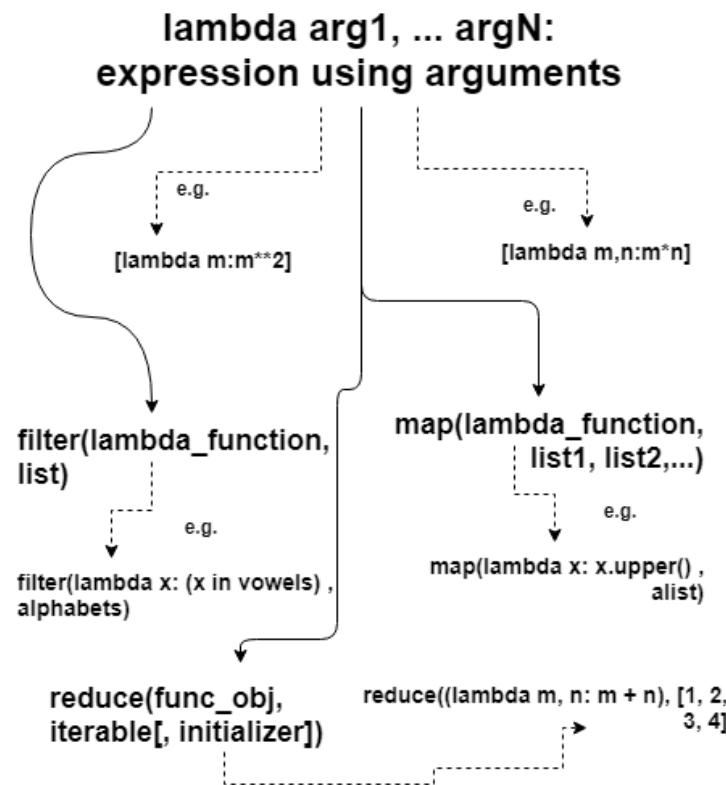
# Functions

## Introduction / Overview

- In Python, you have a couple of ways to make functions:
  - a) Use Def keyword: It creates a function object and assigns it to a name.
  - b) Use lambda: It creates an inline function and returns it as a result.
- A lambda function is a lightweight anonymous function. It can accept any number of arguments but can only have a single expression.

## What is lambda in Python?

- Lambda is an unnamed function. It provides an expression form that generates function objects.
- This expression form creates a function and returns its object for calling it later.



## How to create a lambda function?

### Syntax

It has the following signature:

```
lambda arg1, arg2, ... argN: expression using arguments
```

- The body of a lambda function is akin to what you put in a def body's return statement. The difference here is that the result is a typed-expression, instead of explicitly returning it.

- 
- Note: Lambda function can't include any statements. It only returns a function object which you can assign to any variable.
  - The lambda statement can appear in places where the def is not allowed. For example – inside a list literal or a function call's arguments, etc.

### Example

```
lambda inside a list

list = [lambda m:m**2, lambda m,n:m*n, lambda m:m**4]
print(alist[0](10), alist[1](2, 20), alist[2](3))
# Output: 100 40 81

lambda inside a dictionary

key = 'm'
aDict = {'m': lambda x:2*x, 'n': lambda x:3*x}
print(aDict[key](9))
# Output: 18
```

## Why Use Lambda Functions in Python?

- A lambda function is not an absolute necessity in Python, but using a lambda function in certain situations definitely makes it a bit easier to write the code. Not just that, it also makes the written code a bit cleaner. Now, in what all situations using a lambda function is beneficial? Following are some of the situations where using a lambda function is preferred.
- Lambda functions in Python are very useful in defining the in-line functions where the regular functions, defined using the def keyword, won't work syntactically. Since a

---

lambda function is an expression rather than a statement, it can be used in places where a regular function is not allowed by the Python syntax, for instance, in places such as inside a Python list or in a function's call argument.

- As observed in the above example, the same operation performed by a regular function with the function body of at least three to four lines can be performed by a lambda function which only takes one line. Then, why use a long function to perform a simple operation when it can be done in a single line expression?
- So, to summarize, a lambda function behaves like a regular function, takes an argument, and returns a value but is not bound to any name or identifier. There is no need to use the return statement in a lambda function in Python; it will always return the value obtained by evaluating the lambda expression in Python.

### **Properties of Python Lambda Functions**

- Anonymous functions created using the `lambda` keyword can have any number of arguments, but they are syntactically restricted to just one expression, that is, they can have only one expression.
- Lambda function in Python can be used wherever a function object is required.
- Lambda functions do not require any return statement; they always return a value obtained by evaluating the lambda expression in Python.
- Python Lambda functions are widely used with some Python built-in functions such as `map()`, `reduce()`, etc. Extending Python lambda functions

### **Built in Functions**

- We can extend the utility of lambda functions by using it with the `filter` and `map` functions.

- 
- It is possible by passing the lambda expression as an argument to another function. We refer to these methods as higher-order functions as they accept function objects as arguments.
  - Python provides two built-in functions like filter(), map() which can receive lambda functions as arguments.

### **Map functions over iterables – map()**

- The map() function lets us call a function on a collection or group of iterables.
- We can also specify a Python lambda function in the map call as the function object.
- The map() function has the following signature.

map(function\_object, iterable1, iterable2,...)

- It expects variable-length arguments: first is the lambda function object, and rest are the iterables such as a list, dictionary, etc.

### **What does the map() function do?**

- The map function iterates all the lists (or dictionaries etc.) and calls the lambda function for each of their elements.
- What does the map() function return?
- The output of map() is a list which contains the result returned by the lambda function for each item it gets called.
- Below is a simple example illustrating the use of map() function to convert elements of lists into uppercase.

---

```
# Python lambda demo to use map() for adding elements of two lists

alist = ['learn', 'python', 'step', 'by', 'step']
output = list(map(lambda x: x.upper(), alist))

# Output: ['LEARN', 'PYTHON', 'STEP', 'BY', 'STEP']

print(output)
```

Let's have another example illustrating the use of map() function for adding elements of two lists.

```
# Python lambda demo to use map() for adding elements of two lists

list1 = [1, 2, 3, 4]
list2 = [100, 200, 300, 400]
output = list(map(lambda x, y: x+y, list1, list2))

# Output: [101, 202, 303, 404]

print(output)
```

### Select items in iterables – filter()

- The filter() function selects an iterable's (a list, dictionary, etc.) items based on a test function.
- We can also filter a list by using the Python lambda function as the function object.
- The filter function has the following signature.  
`filter(function_object, list)`
- It expects two parameters: first is the lambda function object and the second is a list.

### What does the filter() function do?

- The filter function iterates the list and calls the lambda function for each element.

---

## What does the filter() function return?

- It returns a final list containing items for which the lambda function evaluates to True.
- Below is a simple example illustrating the use of filter() function to determine vowels from the list of alphabets.

```
# Python lambda demo to filter out vowels from a list

alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
vowels = ['a', 'e', 'i', 'o', 'u']
output = list(filter(lambda x: (x in vowels) , alphabets))

# Output: ['a', 'e', 'i']
print(output)
```

## Aggregate items in iterables – reduce()

- The reduce method continuously applies a function on an iterable (such as a list) until there are no items left in the list. It produces a non-iterable result, i.e., returns a single value.
- This method helps in aggregating data from a list and returning the result. It can let us do a rolling calculation over successive pairs of values in a sequence.
- We can also pass a Python lambda function as an argument to the reduce method.
- The reduce() function has the following syntax.

```
reduce(func_obj, iterable[, initializer])
```

Below is a simple example where the reduce() method is computing the sum of elements in a list.

---

```
from functools import reduce
def fn(m, n) : return m + n
print(reduce((lambda m, n: m + n), [1, 2, 3, 4]))
print(reduce(fn, [1, 2, 3, 4]))
```

#After executing the above code, you see the following output.

```
10
10
```

---

# Modules

## What are modules in Python?

- Modules refer to a file containing Python statements and definitions.
- A file containing Python code, for e.g.: example.py, is called a module and its module name would be example.
- We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code.
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.
- Generally, it is a good practice to create modules which have a fixed purpose. It increases readability and increases productivity and bug reporting.

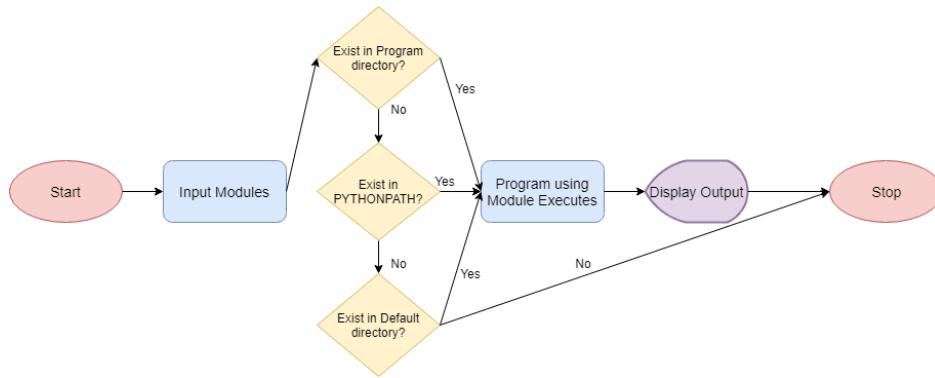
Let us create a module. Type the following and save it as example.py.

```
# Python Module example
def add(a, b):
    """This program adds two numbers
    and return the result"""
    result = a + b
    return result
```

Here, we have defined a function add() inside a module named example. The function takes in two numbers and returns their sum.

## Python Module: Mechanism

- For systems where Python is pre-installed or when installed using the system package manager such as apt-get, dnf, zypper, etc. or using package environment managers like Anaconda the following applies.
- When we import modules, the python interpreter locates them from three locations:
- The directory from the program is getting executed
- The directory specified in the PYTHONPATH variable (A shell variable or an environment variable)
- The default directory (It depends on the OS distribution.)
- The Flowchart for the above mechanism is as follows:



## import modules in Python?

- We can import the definitions inside a module to another module or the interactive interpreter in Python.
- We use the import keyword to do this. To import our previously defined module example we type the following in the Python prompt.

```
import example
```

- This does not enter the names of the functions defined in example directly in the current symbol table. It only enters the module name example there.
- Using the module name we can access the function using the dot. Operator.
- For example:

```
example.add(4,5.5)  
  
#output  
9.5
```

- Python has a ton of standard modules available.
- You can check out the full list of Python standard modules and what they are for. These files are in the Lib directory inside the location where you installed Python.
- Standard modules can be imported the same way as we import our user-defined modules.

**There are various ways to import modules. They are listed as follows.**

**import statement** - We can import a module using import statement and access the definitions inside it using the dot operator as described above. Here is an example.

```
# import statement example  
# to import standard module math  
import math  
print("The value of pi is", math.pi)
```

---

**Import with renaming** - We can import a module by renaming it as follows.

```
# import module by renaming it
import math as m
print("The value of pi is", m.pi)
```

We have renamed the math module as m. This can save us typing time in some cases.

- Note that the name math is not recognized in our scope. Hence, math.pi is invalid, m.pi is the correct implementation.

**Python from...import statement** - We can import specific names from a module without importing the module as a whole. Here is an example.

```
# import only pi from math module
from math import pi
print("The value of pi is", pi)
```

- We imported only the attribute pi from the module.
- In such case we don't use the dot operator. We could have imported multiple attributes as follows.

```
from math import pi, e
pi #3.141592653589793
e #2.718281828459045
```

---

**Import all names** - We can import all names(definitions) from a module using the following construct.

```
# import all names from the standard module math
from math import *
print("The value of pi is", pi)
```

We imported all the definitions from the math module. This makes all names except those beginning with an underscore, visible in our scope.

Importing everything with the asterisk (\*) symbol is not a good programming practice. This can lead to duplicate definitions for an identifier. It also hampers the readability of our code.

### Python Module Search Path

- While importing a module, Python looks at several places. Interpreter first looks for a built-in module then (if not found) into a list of directories defined in sys.path. The search is in this order.
- The current directory,
- PYTHONPATH (an environment variable with a list of directory).
- The installation-dependent default directory.

```
import sys
sys.path

Output:
[ '',
'C:\\\\Python33\\\\Lib\\\\idlelib',
'C:\\\\Windows\\\\system32\\\\python33.zip',
'C:\\\\Python33\\\\DLLs',
'C:\\\\Python33\\\\lib',
'C:\\\\Python33',
'C:\\\\Python33\\\\lib\\\\site-packages']
```

We can add modify this list to add our own path.

### Reloading a module

- The Python interpreter imports a module only once during a session. This makes things more efficient. Here is an example to show how this works.
- Suppose we have the following code in a module named my\_module.

```
# This module shows the effect of
# multiple imports and reload
print("This code got executed")
```

Now we see the effect of multiple imports.

```
import my_module
```

This code got executed

```
import my_module  
import my_module
```

We can see that our code got executed only once. This goes to say that our module was imported only once.

Now if our module changed during the course of the program, we would have to reload it. One way to do this is to restart the interpreter. But this does not help much.

- Python provides a neat way of doing this. We can use the reload() function inside the imp module to reload a module. This is how its done.

```
import imp  
import my_module
```

This code got executed

```
import my_module  
imp.reload(my_module)
```

This code got executed

```
<module 'my_module' from '.\\my_module.py'>
```

## The dir() built-in function

- 
- We can use the `dir()` function to find out names that are defined inside a module.
  - For example, we have defined a function `add()` in the module `example` that we had in the beginning.

```
dir()
```

Here, we can see a sorted list of names (along with `add`). All other names that begin with an underscore are default Python attributes associated with the module

For example,

```
data = dir()
print(data)

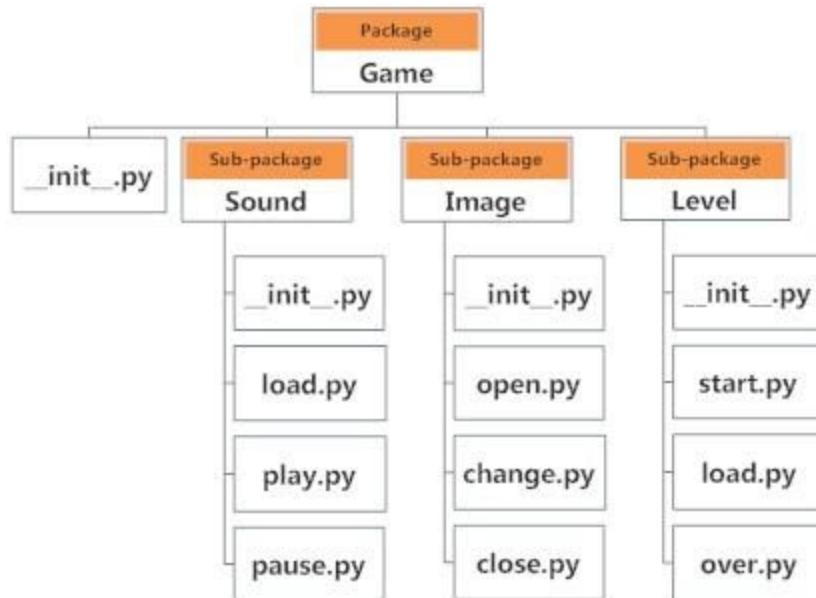
#Output:
['__annotations__', '__builtins__',
 '__doc__', '__file__', '__loader__',
 '__name__', '__package__', '__spec__']
```

---

# Package

## What are packages?

- We don't usually store all of our files in our computer in the same location. We use a well-organized hierarchy of directories for easier access.
- Similar files are kept in the same directory, for example, we may keep all the songs in the "music" directory. Analogous to this, Python has packages for directories and modules for files.
- As our application program grows larger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project (program) easy to manage and conceptually clear.
- Similar, as a directory can contain sub-directories and files, a Python package can have sub-packages and modules.
- A directory must contain a file named `__init__.py` in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.
- Here is an example. Suppose we are developing a game, one possible organization of packages and modules could be as shown in the figure below.



## Importing module from a package

- We can import modules from packages using the dot (.) operator.
- For example, if want to import the start module in the above example, it is done as follows.

```
import Game.Level.start
```

- Now if this module contains a function named select\_difficulty(), we must use the full name to reference it.

```
Game.Level.start.select_difficulty(2)
```

- 
- If this construct seems lengthy, we can import the module without the package prefix as follows.

```
from Game.Level import start
```

- We can now call the function simply as follows.

```
start.select_difficulty(2)
```

- Yet another way of importing just the required function (or class or variable) from a module within a package would be as follows.

```
from Game.Level.start import select_difficulty
```

Now we can directly call this function.

```
select_difficulty(2)
```

Although easier, this method is not recommended. Using the full namespace avoids confusion and prevents two same identifier names from colliding.

While importing packages, Python looks in the list of directories defined in sys.path, similar as for module search path.

# Input /Output

## Python Input, Output and Import

- Python provides numerous built-in functions that are readily available to us at the Python prompt.
- Some of the functions like input() and print() are widely used for standard input and output operations respectively. Let us see the output section first.
- Python Output Using print() function
- We use the print() function to output data to the standard output device (screen).
- An example of its use is given below.

```
print('This sentence is output to the screen')
```

The actual syntax of the print() function is:

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

Here, objects is the value(s) to be printed. The sep separator is used between the values. It defaults into a space character.

- After all values are printed, end is printed. It defaults into a new line.
- The file is the object where the values are printed and its default value is sys.stdout (screen).

---

## Output formatting

- Sometimes we would like to format our output to make it look attractive. This can be done by using the str.format() method. This method is visible to any string object.

```
x = 5; y = 10
print('The value of x is {} and y is {}'.format(x,y))
#Output
The value of x is 5 and y is 10
```

Here, the curly braces {} are used as placeholders. We can specify the order in which they are printed by using numbers (tuple index).

```
print('I love {0} and {1}'.format('bread','butter'))
print('I love {1} and {0}'.format('bread','butter'))
#Output
I love bread and butter
I love butter and bread
```

We can even use keyword arguments to format the string.

```
print('Hello {name}, {greeting}'.format(greeting = 'Goodmorning', name = 'John'))
Hello John, Goodmorning
```

We can also format strings like the old sprintf() style used in C programming language. We use the % operator to accomplish this.

---

```
x = 12.3456789
print('The value of x is %3.2f' %x)
#The value of x is 12.35
print('The value of x is %3.4f' %x)
#The value of x is 12.3457
```

## Python Input

- The value of variables was defined or hard coded into the source code.
- To allow flexibility, we might want to take the input from the user. In Python, we have the `input()` function to allow this. The syntax for `input()` is:

```
input([prompt])
```

- Where `prompt` is the string we wish to display on the screen. It is optional.

```
num = input('Enter a number: ')
Enter a number: 10
Num : 10
```

- Here, we can see that the entered value 10 is a string, not a number. To convert this into a number we can use `int()` or `float()` functions.

---

```
int('10') #10
float('10') #10.0
```

This same operation can be performed using the eval() function. But eval takes it further. It can evaluate even expressions, provided the input is a string

```
int('2+3')
eval('2+3')
```

## What is a file?

- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since, random access memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.
- When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed.
- In Python, file processing takes place in the following order.
  - Open a file that returns a filehandle.
  - Use the handle to perform read or write action.
  - Close the filehandle.
- Before you do a read or write operation to a file in Python, you need to open it first. And as the read/write transaction completes, you should close it to free the resources tied with the file.

---

## How to open a file?

- Python has a built-in function `open()` to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

```
# open file in current directory
f = open("test.txt")

# specifying full path
f = open("C:/Python33/README.txt")
```

- We can specify the mode while opening a file. In mode, we specify whether we want to read 'r', write 'w' or append 'a' to the file. We also specify if we want to open the file in text mode or binary mode.
- The default is reading in text mode. In this mode, we get strings when reading from the file.
- On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.
- Python `open()` file method

```
file_object = open(file_name [, access_mode] [, buffering])
```

Below are the parameter details.

- <access\_mode>- It's an integer representing the file opening mode, e.g., read, write, append, etc. It's an optional parameter. By default, it is set to read-only <r>. In this mode, we get data in text form after reading from the file.
- On the other hand, the binary mode returns bytes. It's preferable for accessing the non-text files like an image or the Exe files. See the table in the next section. It lists down the available access modes.
- <buffering>- The default value is 0, which means buffering won't happen. If the value is 1, then line buffering will take place while accessing the file. If it's higher than 1, then the buffering action will run as per the buffer size. In the case of a negative value, the default behavior is considered.
- <file\_name>- It's a string representing the name of the file you want to access.

## Python File Modes

| MODES | DESCRIPTION                                                                                                                                        |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <r>   | It opens a file in read-only mode while the file offset stays at the root.                                                                         |
| <rb>  | It opens a file in (binary + read-only) modes. And the offset remains at the root level.                                                           |
| <r+>  | It opens the file in both (read + write) modes while the file offset is again at the root level.                                                   |
| <rb+> | It opens the file in (read + write + binary) modes. The file offset is again at the root level.                                                    |
| <w>   | It allows write-level access to a file. If the file already exists, then it'll get overwritten. It'll create a new file if the same doesn't exist. |
| <wb>  | Use it to open a file for writing in binary format. Same behavior as for write-only mode.                                                          |
| <w+>  | It opens a file in both (read + write) modes. Same behavior as for write-only mode.                                                                |

|       |                                                                                                                            |
|-------|----------------------------------------------------------------------------------------------------------------------------|
| <wb+> | It opens a file in (read + write + binary) modes. Same behavior as for write-only mode.                                    |
| <a>   | It opens the file in append mode. The offset goes to the end of the file. If the file doesn't exist, then it gets created. |
| <ab>  | It opens a file in (append + binary) modes. Same behavior as for append mode.                                              |
| <a+>  | It opens a file in (append + read) modes. Same behavior as for append mode.                                                |
| <ab+> | It opens a file in (append + read + binary) modes. Same behavior as for append mode.                                       |

```
# equivalent to 'r' or 'rt'  
f = open("test.txt")  
  
# write in text mode  
f = open("test.txt", 'w')  
  
# read and write in binary mode  
f = open("img.bmp", 'r+b')
```

- Unlike other languages, the character 'a' does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings).
- Moreover, the default encoding is platform dependent. In windows, it is 'cp1252' but 'utf-8' in Linux.
- So, we must not also rely on the default encoding or else our code will behave differently in different platforms.

- 
- Hence, when working with files in text mode, it is highly recommended to specify the encoding type.

```
f = open("test.txt", mode = 'r', encoding = 'utf-8')
```

### The Python file object attributes

- When you call the Python open() function, it returns an object, which is the filehandle. Also, you should know that Python files have several linked attributes. And we can make use of the filehandle to list the attributes of a file.
- For more information on file attributes, please run down through the below table.

| Attribute        | Description                                                                                                                      |
|------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <file.closed>    | For a closed file, it returns true whereas false otherwise.                                                                      |
| <file.mode>      | It returns the access mode used to open a file.                                                                                  |
| <file.name>      | Returns the name of a file                                                                                                       |
| <file.softspace> | It returns a boolean to suggest if a space char will get added before printing another value in the output of a <print> command. |

### Example: Python file attribute in action

---

```
#Open a file in write and binary mode.
fob = open("app.log", "wb")

#Display file name.
print("File name: ", fob.name)

#Display state of the file.
print("File state: ", fob.closed)

#print the opening mode.
print("Opening mode: ", fob.mode)

#Output the softspace value.
print("Softspace flag: ", fob.softspace)
```

## How to close a file?

- It's always the best practice to close a file when your work gets finished. However, Python runs a garbage collector to clean up the unused objects. But you must do it on your own instead leave it for the GC.
- The close() file method
- Python provides the <close()> method to close a file.
- While closing a file, the system frees up all resources allocated to it. And it's rather easy to achieve.
- Close operation in Python
- The most basic way is to call the Python close() method.

```
f = open("app.log",encoding = 'utf-8')
# do file operations.
f.close()
```

## Close with try-catch

Say, if an exception occurs while performing some operations on the file. In such a case, the code exits without closing the file. So it's better to put the code inside a <try-finally> block.

```
try:  
    f = open('app.log', encoding = 'utf-8')  
    # do file operations.  
finally:  
    f.close()
```

So, even if there comes an exception, the above code will make sure your file gets appropriately closed.

## Auto close using ‘with’

Another way to close a file is by using the WITH clause. It ensures that the file gets closed when the block inside the WITH clause executes. The beauty of this method is that it doesn't require to call the close() method explicitly.

```
with open('app.log', encoding = 'utf-8') as f:  
    #do any file operation.
```

## Perform Write operation

- While you get ready for writing data to a file, first of all, open it using a mode (read/write/append). View the list of all available file modes here.

- 
- You can even do the same using the append mode. Also, if you've used the <w> mode, then it'll erase the existing data from the file. So you must note this fact while you choose it.
  - The write() file method
  - Python provides the write() method to write a string or sequence of bytes to a file. This function returns a number, which is the size of data written in a single Write call.

### Example: Read/Write to a File in Python

```
with open('app.log', 'w', encoding = 'utf-8') as f:  
    #first line  
    f.write('my first file\n')  
  
    #second line  
    f.write('This file\n')  
  
    #third line  
    f.write('contains three lines\n')  
with open('app.log', 'r', encoding = 'utf-8') as f:  
    content = f.readlines()  
    for line in content:  
        print(line)
```

### Perform Read operation

- To read data from a file, first of all, you need to open it in reading mode. Then, you can call any of the methods that Python provides for reading from a file.
- Usually, we use Python <read(size)> function to read the content of a file up to the size. If you don't pass the size, then it'll read the whole file.
- Example: Read from a File in Python

```
with open('app.log', 'w', encoding = 'utf-8') as f:  
    #first line  
    f.write('my first file\n')  
    #second line  
    f.write('This file\n')  
    #third line  
    f.write('contains three lines\n')  
    f = open('app.log', 'r', encoding = 'utf-8')  
    # read the first 10 data  
    print(f.read(10))  
    #'my first f'  
    print(f.read(4))  
    #'ile\n'  
    # read in the rest till end of file  
    print(f.read())  
    #'This file\ncontains three lines\n'  
    # further reading returns empty sting#''  
    print(f.read())
```

## Set File offset in Python

- Tell() Method - This method gives you the current offset of the file pointer in a file.

```
file.tell()  
#The tell() method doesn't require any argument
```

- Seek() Method - This method can help you change the position of a file pointer in a file.

```
file.seek(offset[, from])  
#The <offset> argument represents the size of the displacement.  
#The <from> argument indicates the start point
```

---

## Renaming and deleting files in Python

- While you were using the <read/write> functions, you may also need to <rename/delete> a file in Python. So, there comes a <os> module in Python, which brings the support of file <rename/delete> operations.
- So, to continue, first of all, you should import the <os> module in your Python script.
- The rename() file method

```
#Syntax  
os.rename(cur_file, new_file)
```

- The <rename()> method takes two arguments, the current filename and the new filename.
- Following is the example to rename an existing file <app.log> to <app1.log>.

```
import os  
#Rename a file from <app.log> to <app1.log>  
os.rename( "app.log", "app1.log" )  
The remove() file method  
os.remove(file_name)
```

The <remove()> method deletes a file which it receives in the argument.

Following is the example to delete an existing file, the <app1.log>.Example:

---

```
import os
#Delete a file <app1.log>
os.remove( "app1.log" )
```

---

## Python File object methods

- So far, we've only shared with you a few of the functions that you can use for file handling in Python. But there is more to the story of Python file handling.
- Python's open() method returns an object which we call as the filehandle. Python adds a no. of functions that we can call using this object.

| Function                    | Description                                                                                                                                                                  |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <file.close()>              | Close the file. You need to reopen it for further access.                                                                                                                    |
| <file.flush()>              | Flush the internal buffer. It's same as the <stdio>'s <fflush()> function.                                                                                                   |
| <file.fileno()>             | Returns an integer file descriptor.                                                                                                                                          |
| <file.isatty()>             | It returns true if file has a <tty> attached to it.                                                                                                                          |
| <file.next()>               | Returns the next line from the last offset.                                                                                                                                  |
| <file.read(size)>           | Reads the given no. of bytes. It may read less if EOF is hit.                                                                                                                |
| <file.readline(size)>       | It'll read an entire line (trailing with a new line char) from the file.                                                                                                     |
| <file.readlines(size_hint)> | It calls the <readline()> to read until EOF. It returns a list of lines read from the file.<br>If you pass <size_hint>, then it reads lines equalling the <size_hint> bytes. |
| <file.seek(offset[, from])> | Sets the file's current position.                                                                                                                                            |
| <file.tell()>               | Returns the file's current position.                                                                                                                                         |
| <file.truncate(size)>       | Truncates the file's size. If the optional size argument is present, the file is truncated to (at most) that size.                                                           |
| <file.write(string)>        | It writes a string to the file. And it doesn't return any value.                                                                                                             |

|                             |                                                                                                                                          |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <file.writelines(sequence)> | Writes a sequence of strings to the file.<br>The sequence is possibly an iterable object producing strings, typically a list of strings. |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------|

## Python Copy File – 9 Ways

- shutil copyfile() method
- shutil copy() method
- shutil copyfileobj() method
- shutil copy2() method
- os popen method
- os system() method
- threading Thread() method
- subprocess call() method
- subprocess check\_output() method

### 1. **shutil copyfile() method**

- This method copies the content of the source to the destination only if the target is writable. If you don't have the right permissions, then it will raise an IOError.
- It works by opening the input file for reading while ignoring its file type.
- Next, it doesn't treat special files any differently and won't create their clones.
- The copyfile() method makes use of lower-level function copyfileobj() underneath. It takes file names as arguments, opens them and passes file handles to copyfileobj(). There is one optional third argument in this method which you can use to specify the buffer length. It'll then open the file for reading in chunks of the specified buffer size. However, the default behavior is to read the entire file in one go.

---

```
copyfile(source_file, destination_file)
```

- Following are the points to know about the copyfile() method.
- It copies the contents of the source to a file named as the destination.
- If the destination isn't writable, then the copy operation would result in an IOError exception.
- It will return the SameFileError if both the source and destination files are the same.
- However, if the destination pre-exists with a different name, then the copy will overwrite its content.
- Error 13 will occur if the destination is a directory which means this method won't copy to a folder.
- It doesn't support copying files such as character or block devices and the pipes.

---

```
# Python Copy File - Sample Code
from shutil import copyfile
from sys import exit
source = input("Enter source file with full path: ")
target = input("Enter target file with full path: ")
# adding exception handling
try:
    copyfile(source, target)
except IOError as e:
    print("Unable to copy file. %s" % e)
    exit(1)
except:
    print("Unexpected error:", sys.exc_info())
    exit(1)
print("\nFile copy done!\n")
while True:
    print("Do you like to print the file ? (y/n): ")
    check = input()
    if check == 'n':
        break
    elif check == 'y':
        file = open(target, "r")
        print("\nHere follows the file content:\n")
        print(file.read())
        file.close()
        print()
        break
    else:
        continue
```

---

## 2. shutil copy() method

```
copyfile(source_file, [destination_file or dest_dir])
```

- The copy() method functions like the “cp” command in Unix. It means if the target is a folder, then it’ll create a new file inside it with the same name (basename) as the source file. Also, this method will sync the permissions of the target file with the source after copying its content. It too throws the SameFileError if you are copying the same file.

```
import os
import shutil

source = 'current/test/test.py'
target = '/prod/new'

assert not os.path.isabs(source)
target = os.path.join(target, os.path.dirname(source))

# create the folders if not already exists
os.makedirs(target)

# adding exception handling
try:
    shutil.copy(source, target)
except IOError as e:
    print("Unable to copy file. %s" % e)
except:
    print("Unexpected error:", sys.exc_info())
```

### copy() vs copyfile()

- The copy() also sets the permission bits while copying the contents whereas the copyfile() only copies the data.

- 
- The copy() will copy a file if the destination is a directory whereas the copyfile() will fail with error 13.
  - Interestingly, the copyfile() method utilizes the copyfileobj() method in its implementation whereas the copy() method makes use of the copyfile() and copymode() functions in turn.
  - Point-3 makes it apparent that copyfile() would be a bit faster than the copy() as the latter has an additional task (preserving the permissions) at hand.

### 3. shutil copyfileobj() method

- This method copies the file to a target path or file object. If the target is a file object, then you need to close it explicitly after the calling the copyfileobj(). It assumes an optional argument (the buffer size) which you can use to supply the buffer length. It is the number of bytes kept in memory during the copy process. The default size that system use is 16KB.

```
from shutil import copyfileobj
status = False
if isinstance(target, string_types):
    target = open(target, 'wb')
    status = True
try:
    copyfileobj(self.stream, target, buffer_size)
finally:
    if status:
        target.close()
```

---

#### 4. shutil copy2() method

- However, the copy2() method functions like the copy(). But it also gets the access and modification times added in the meta-data while copying the data. Copying the same file would result in SameFileError.

```
from shutil import *
import os
import time
from os.path import basename

def displayFileStats(filename):
    file_stats = os.stat(basename(filename))
    print('\tMode :', file_stats.st_mode)
    print('\tCreated :', time.ctime(file_stats.st_ctime))
    print('\tAccessed:', time.ctime(file_stats.st_atime))
    print('\tModified:', time.ctime(file_stats.st_mtime))

os.mkdir('test')

print('SOURCE:')
displayFileStats(__file__)

copy2(__file__, 'testfile')

print('TARGET:')
displayFileStats(os.path.realpath(os.getcwd() + './test/testfile'))
```

#### copy() vs copy2()

- The copy() only sets permission bits whereas copy2() also updates the file metadata with timestamps.
- The copy() method calls copyfile() and copymode() internally whereas copy2() replaces the call to copymode() with copystat().

---

## 5. shutil.copymode()

```
copymode(source, target, *, follow_symlinks=True)
```

- It intends to copy the permission bits from source to the target files.
- The file contents, owner, and group remain unchanged. The arguments passed are strings.
- If the follow\_symlinks arg is false and the first two args are symlinks, then copymode() will try to update the target link, not the actual file it is pointing.

## 6. shutil.copystat()

```
copystat(source, target, *, follow_symlinks=True)
```

- It attempts to preserve the permission bits, last used time/update time, and flags of the target file.
- The copystat() includes the extended attributes while copying on Linux. The file contents/owner/group remain unchanged.
- If the follow\_symlinks arg is false and the first two args are symlinks, then copystat() will update them, not the files they point.

## 7. os popen() method

- This method creates a pipe to or from the command. It returns an open file object which connects to a pipe. You can use it for reading or writing according to the file mode, i.e., ‘r’ (default) or ‘w’.

---

```
os.popen(command[, mode[, bufsize]])
mode - It can be 'r' (default) or 'w'.
```

- bufsize – If its value is 0, then no buffering will occur. If the bufsize is 1, then line buffering will take place while accessing the file. If you provide a value greater than 1, then buffering will occur with the specified buffer size. However, for a negative value, the system will assume the default buffer size.

```
#For Windows OS.
import os
os.popen('copy 1.txt.py 2.txt.py')

#For Linux OS.
import os
os.popen('cp 1.txt.py 2.txt.py')
```

## 8. os system() method

- The system() method allows you to instantly execute any OS command or a script in the subshell.
- You need to pass the command or the script as an argument to the system() call. Internally, this method calls the standard C library function.
- Its return value is the exit status of the command.

```
#For Windows OS.  
import os  
os.system('copy 1.txt.py 2.txt.py')  
  
#For Linux OS.  
import os  
os.system('cp 1.txt.py 2.txt.py')
```

Python file copy using threading library in Async manner

- If you want to copy a file asynchronously, then use the below method. In this, we've used Python's threading module to run the copy operation in the background.
- While using this method, please make sure to employ locking to avoid deadlocks. You may face it if your application is using multiple threads reading/writing a file.

```
import shutil  
from threading import Thread  
  
src="1.txt.py"  
dst="3.txt.py"  
  
Thread(target=shutil.copy, args=[src, dst]).start()
```

## 9. Use subprocess's call() method to copy a file in Python

- The subprocess module gives a simple interface to work with child processes. It enables us to launch subprocesses, attach to their input/output/error pipes, and retrieve the return values.
- The subprocess module aims to replace the legacy modules and functions like – os.system, os.spawn\*, os.popen\*, popen2.\*.

- 
- It exposes a call() method to invoke system commands to execute user tasks.

```
import subprocess
src="1.txt.py"
dst="2.txt.py"
cmd='copy "%s" "%s"' % (src, dst)
status = subprocess.call(cmd, shell=True)
if status != 0:
    if status < 0:
        print("Killed by signal", status)
    else:
        print("Command failed with return code - ", status)
else:
    print('Execution of %s passed!\n' % cmd)
```

#### 10. Use subprocess's check\_output() method to copy a file in Python

- With subprocess's check\_output() method, you can run an external command or a program and capture its output. It also supports pipes.

```
import os, subprocess
src=os.path.realpath(os.getcwd() + "./1.txt.py")
dst=os.path.realpath(os.getcwd() + "./2.txt.py")
cmd='copy "%s" "%s"' % (src, dst)
status = subprocess.check_output(['copy', src, dst], shell=True)
print("status: ", status.decode('utf-8'))
```

---

# Exception Handling

## Exception Handling: Error vs. Exception

### What is Error?

- The error is something that goes wrong in the program, e.g., like a syntactical error.
- It occurs at compile time. Let's see an example.

```
if a<5
File "<interactive input>", line 1
    if a < 5
        ^
SyntaxError: invalid syntax
```

### What is Exception?

- The errors also occur at runtime, and we know them as exceptions. An exception is an event which occurs during the execution of a program and disrupts the normal flow of the program's instructions.
- In general, when a Python script encounters an error situation that it can't cope with, it raises an exception.
- When a Python script raises an exception, it creates an exception object.
- Usually, the script handles the exception immediately. If it doesn't do so, then the program will terminate and print a traceback to the error along with its whereabouts.

```
1 / 0
Traceback (most recent call last):
File "<string>", line 301, in run code
File "<interactive input>", line 1, in <module>
ZeroDivisionError: division by zero
```

## Handle Exceptions with Try-Except

### What is Try-Except Statement?

- We use the try-except statement to enable exception handling in Python programs.
- Inside the try block, you write the code which can raise an exception.
- And the code that handles or catches the exception, we place in the except clause.
- Python Exception Handling Syntax
- Following is the syntax of a Python try-except-else block.

```
try:
    You do your operations here;
    .....
except ExceptionI:
    If there is ExceptionI, then execute this block.
except ExceptionII:
    If there is ExceptionII, then execute this block.
    .....
else:
```

Here is a checklist for using the Python try statement effectively.

- If there is no exception then execute this block.

- 
- A single try statement can have multiple except statements depending on the requirement.  
In this case, a try block contains statements that can throw different types of exceptions.
  - We can also add a generic except clause which can handle all possible types of exceptions.
  - We can even include an else clause after the except clause. The instructions in the else block will execute if the code in the try block doesn't raise an exception.
  - We'll perform a WRITE operation on it. Upon execution, it'll throw an exception.

```
try:  
    fob = open("test", "r")  
    fob.write("It's my test file to verify exception handling in Python!!")  
except IOError:  
    print "Error: can't find the file or read data"  
else:  
    print "Write operation is performed successfully on the file"  
  
#The above code produces the following output.  
  
Error: can't find file or read data
```

## Handling All Types of Exceptions with Except

- If we use a bare “except” clause, then it would catch all types of exceptions.
- However, neither it's a good programming practice nor does anyone recommend it.
- It is because that such a Python try-except block can handle all types of exceptions. But it'll not help the programmer to find what exception caused the issue.
- You can go through the below code to see how to catch all exceptions.
- Example

---

```
try:  
    You do your operations here;  
    .....  
except:  
    If there is any exception, then execute this block.  
    .....  
else:  
    If there is no exception then execute this block.
```

## Handling Multiple Exceptions with Except

- We can define multiple exceptions with the same except clause. It means that if the Python interpreter finds a matching exception, then it'll execute the code written under the except clause.
- In short, when we define except clause in this way, we expect the same piece of code to throw different exceptions. Also, we want to take the same action in each case.
- Please refer to the below example.
- Example

```
try:  
    You do your operations here;  
    .....  
except (Exception1[, Exception2[,...ExceptionN]]]):  
    If there is any exception from the given exception list,  
    then execute this block.  
    .....  
else:  
    If there is no exception then execute this block
```

## Handle Exceptions with Try-Finally

### What is Try-Finally Statement?

- We can also enable Python exception handling with the help of try-finally statement.
- With try block, we also have the option to define the “finally” block. This clause allows defining statements that we want to execute, no matter whether the try block has raised an exception or not.
- This feature usually comes in the picture while releasing external resources.
- Here is the coding snippet for help.

```
try:  
    You do your operations here;  
    .....  
    Due to any exception, this may be skipped.  
finally:  
    This would always be executed.  
    .....
```

---

## Raise Exception with Arguments

### What is Raise?

- We can forcefully raise an exception using the raise keyword.
- We can also optionally pass values to the exception and specify why it has occurred.
- Raise Syntax.

```
raise [Exception [, args [, traceback]]]
```

- Where, Under the “Exception” – specify its name.
- The “args” is optional and represents the value of the exception argument.
- The final argument, “traceback,” is also optional and if present, is the traceback object used for the exception.
- Let’s take an example to clarify this.
- Raise Example

```
raise MemoryError
Traceback (most recent call last):
...
MemoryError

>>> raise MemoryError("This is an argument")
Traceback (most recent call last):
...
MemoryError: This is an argument

>>> try:
    a = int(input("Enter a positive integer value: "))
    if a <= 0:
        raise ValueError("This is not a positive number!!")
    except ValueError as ve:
        print(ve)

Following Output is displayed if we enter a negative number:
Enter a positive integer: -5
This is not a positive number!!
```

## Create Custom Exceptions

### What is a Custom Exception?

- A custom exception is one which the programmer creates himself.
- He does it by adding a new class. The trick here is to derive the custom exception class from the base exception class.
- Most of the built-in exceptions do also have a corresponding class.
- Create Exception Class in Python

```
class UserDefinedError(Exception):
    ...
    pass

raise UserDefinedError
Traceback (most recent call last):
...
__main__.UserDefinedError

raise UserDefinedError("An error occurred")
Traceback (most recent call last):
...
__main__.UserDefinedError: An error occurred
```

## Python Built-in Exceptions

| Exception          | Cause of Error                                        |
|--------------------|-------------------------------------------------------|
| AirthmeticError    | For errors in numeric calculation.                    |
| AssertionError     | If the assert statement fails.                        |
| AttributeError     | When an attribute assignment or the reference fails.  |
| EOFError           | If there is no input or the file pointer is at EOF.   |
| Exception          | It is the base class for all exceptions.              |
| EnvironmentError   | For errors that occur outside the Python environment. |
| FloatingPointError | It occurs when the floating point operation fails.    |
| GeneratorExit      | If a generator's <close()> method gets called.        |

|                     |                                                                                         |
|---------------------|-----------------------------------------------------------------------------------------|
| ImportError         | It occurs when the imported module is not available.                                    |
| IOError             | If an input/output operation fails.                                                     |
| IndexError          | When the index of a sequence is out of range.                                           |
| KeyError            | If the specified key is not available in the dictionary.                                |
| KeyboardInterrupt   | When the user hits an interrupt key (Ctrl+c or delete).                                 |
| MemoryError         | If an operation runs out of memory.                                                     |
| NameError           | When a variable is not available in local or global scope.                              |
| NotImplementedError | If an abstract method isn't available.                                                  |
| OSError             | When a system operation fails.                                                          |
| OverflowError       | It occurs if the result of an arithmetic operation exceeds the range.                   |
| ReferenceError      | When a weak reference proxy accesses a garbage collected reference.                     |
| RuntimeError        | If the generated error doesn't fall under any category.                                 |
| StandardError       | It is a base class for all built-in exceptions except <StopIteration> and <SystemExit>. |
| StopIteration       | The <next()> function has no further item to be returned.                               |
| SyntaxError         | For errors in Python syntax.                                                            |
| IndentationError    | It occurs if the indentation is not proper.                                             |

---

|                       |                                                                |
|-----------------------|----------------------------------------------------------------|
| TabError              | For inconsistent tabs and spaces.                              |
| SystemError           | When interpreter detects an internal error.                    |
| SystemExit            | The <sys.exit()> function raises it.                           |
| TypeError             | When a function is using an object of the incorrect type.      |
| UnboundLocalErr<br>or | If the code using an unassigned reference gets executed.       |
| UnicodeError          | For a Unicode encoding or decoding error.                      |
| ValueError            | When a function receives invalid values.                       |
| ZeroDivisionError     | If the second operand of division or modulo operation is zero. |

---

## References

1. Introduction to the Internet by Scott D.James
2. <https://www.w3schools.com/html/>
3. <https://www.w3docs.com/learn-html.html>
4. <https://www.tutorialspoint.com/html/index.htm>
5. <https://www.educba.com/what-is-css/>
6. [https://www.tutorialspoint.com/css/css\\_tutorial.pdf](https://www.tutorialspoint.com/css/css_tutorial.pdf)
7. <https://code.tutsplus.com/tutorials/10-css3-properties-you-need-to-be-familiar-with--net-16417>
8. <https://www.hostinger.in/tutorials/difference-between-inline-external-and-internal-css>
9. <https://www.javatpoint.com/inline-css>
10. <https://www.w3schools.in/bootstrap4/intro/>
11. <https://www.uplers.com/blog/what-are-the-pros-cons-of-foundation-and-bootstrap/>
12. <https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/#uses>
13. <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>
14. <https://www.ibm.com/in-en/cloud/learn/cloud-computing>
15. <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>
16. <https://aws.amazon.com/about-aws/global-infrastructure/>
17. <https://www.eztalks.com/webinars/what-is-a-live-webinar-and-how-does-it-work.html>
18. <https://www.ventureharbour.com/webinar-software-10-best-webinar-platforms-compared/>
19. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4432776/>

- 
20. <https://www.gatevidyalay.com/types-of-attributes/>
  21. <https://codeandwork.github.io/courses/cs/sqlErdSimpleQueries.html>
  22. <https://www.csetutor.com/er-diagram-in-dbms-concept-importance-example/>
  23. <https://www.javatpoint.com/dbms-keys>
  24. <https://www.guru99.com/dbms-keys.html#7>
  25. <https://www.javatpoint.com/unique-key-in-sql>
  26. <https://www.javatpoint.com/dbms-mapping-constraints>
  27. <https://www.studytonight.com/dbms/er-diagram.php>
  28. <https://tutorialwing.com/mapping-constraints-in-dbms-for-relationship-types/>
  29. <https://prepinsta.com/dbms/mapping-constraints/>
  30. [https://docs.oracle.com/cd/B28359\\_01/server.111/b28318/part\\_3.htm](https://docs.oracle.com/cd/B28359_01/server.111/b28318/part_3.htm)
  31. <https://dev.mysql.com/doc/refman/8.0/en/features.html>
  32. <https://docs.mongodb.com/manual/reference/command/features/>
  33. <https://www.microsoft.com/en-us/sql-server/sql-server-2017-features>
  34. <https://www.studytonight.com/>
  35. <https://www.tutorialspoint.com/>
  36. <https://www.guru99.com/>
  37. <https://www.javatpoint.com/>
  38. <https://www.geeksforgeeks.org/>
  39. [https://www.w3schools.com/php/php\\_if\\_else.asp](https://www.w3schools.com/php/php_if_else.asp)
  40. <https://www.geeksforgeeks.org/php-decision-making/>

- 
41. <https://www.geeksforgeeks.org/php-loops/?ref=lbp>
  42. <http://shubhneet.com/mixing-decisions-and-looping-with-html/>
  43. [https://www.tutorialspoint.com/php/php\\_functions.htm](https://www.tutorialspoint.com/php/php_functions.htm)
  44. <https://www.c-sharpcorner.com/UploadFile/d9da8a/call-by-value-and-call-by-reference-in-php/>
  45. <https://www.splessons.com/lesson/php-functions/>
  46. [https://www.w3schools.in/php/functions/#PHP\\_Default\\_Argument\\_Value](https://www.w3schools.in/php/functions/#PHP_Default_Argument_Value)
  47. <https://www.javatpoint.com/php-recursive-function>
  48. <https://www.javatpoint.com/php-string-functions>
  49. <http://shubhneet.com/creating-and-accessing-string/>
  50. [https://www.javatpoint.com/php-string-str\\_replace-function](https://www.javatpoint.com/php-string-str_replace-function)
  51. <https://www.elated.com/formatting-php-strings-printf-sprintf/>
  52. [https://www.w3schools.com/php/php\\_string.asp](https://www.w3schools.com/php/php_string.asp)
  53. [https://www.tutorialspoint.com/php/php\\_arrays.htm](https://www.tutorialspoint.com/php/php_arrays.htm)
  54. <https://www.geeksforgeeks.org/php-arrays/>
  55. <https://www.studytonight.com/php/php-array-functions>
  56. [http://dsl.org/cookbook/cookbook\\_8.html](http://dsl.org/cookbook/cookbook_8.html)
  57. <https://www.geeksforgeeks.org/>
  58. [https://www.tutorialspoint.com/php/php\\_file\\_uploading.htm](https://www.tutorialspoint.com/php/php_file_uploading.htm)
  59. <https://www.tutorialrepublic.com/php-tutorial/php-file-download.php>
  60. [https://www.w3schools.com/php/php\\_mysql\\_intro.asp](https://www.w3schools.com/php/php_mysql_intro.asp)
  61. <https://www.tutorialspoint.com/mysql/mysql-connection.htm>

- 
62. <https://beginnersbook.com/2018/01/introduction-to-python-programming/>
  63. [https://www.tutorialspoint.com/python/python\\_overview.htm](https://www.tutorialspoint.com/python/python_overview.htm)
  64. <http://net-informations.com/python/iq/objects.htm>
  65. <https://elearningindustry.com/advantages-of-python-programming-languages>
  66. <https://www.javatpoint.com/python-history>
  67. <https://www.faceprep.in/python/python-installation/>
  68. <https://www.python.org/downloads/>
  69. <https://docs.python-guide.org/starting/install3/linux/>
  70. <https://www.edureka.co/blog/add-python-to-path/>
  71. [https://www.tutorialspoint.com/python/python\\_basic\\_syntax.htm](https://www.tutorialspoint.com/python/python_basic_syntax.htm)
  72. <https://www.pythontutorforbeginners.com/basics/python-syntax-basics>
  73. <https://www.programiz.com/python-programming/operators>
  74. <https://www.geeksforgeeks.org/decision-making-python-else-nested-elif/>
  75. <https://www.faceprep.in/python/loops-in-python/>
  76. <https://www.geeksforgeeks.org/loops-and-loop-control-statements-continue-break-and-pass-in-python/>
  77. Reference1 : <https://docs.python.org/3/tutorial/>
  78. Reference 2 : <https://intellipaat.com/blog/tutorial/python-tutorial/>
  79. Reference 3 : <https://www.techbeamers.com/python-programming-tutorials/>
  80. Reference 4 : <https://www.programiz.com/python-programming>
  81. Reference 5: <https://realpython.com/>
  82. Reference 6: <https://www.phptpoint.com/python-tutorial/>

- 
- 83. Reference 7: <https://www.guru99.com/python-tutorials.html>
  - 84. Reference 8 : <https://www.datacamp.com/>
  - 85. Reference 9 : <https://www.learnpython.org>