**FACE RECOGNITION BASED ATTENDANCE SYSTEM USING DEEP LEARNING**

**ABSTRACT**

This paper is about the biometric attendance management. The automatic attendance management will replace the manual method, which takes a lot of time consuming and difficult to maintain. There are many biometric processes, in that face recognition is the best method. Here we are using the Deep Learning (DL) based Convolutional Neural Network (CNN) algorithm. We are considering the human faces as a dataset that which were train using CNN. For user interface webpage created using flask framework

**Keywords:** Attendance Management, Computer Vision, Deep Learning, Human Face Images.

**CHAPTER 1**

**INTRODUCTION**

Biometrics are body measurements and calculations related to human characteristics. Biometrics authentication (or realistic authentication) is used in computer science as a form of identification and access control. It is also used to identify individuals in groups that are under surveillance.

Biometric identifiers are the distinctive, measurable characteristics used to label and describe individuals. Biometric identifiers are often categorized as physiological characteristics, which are related to the shape of the body. Examples include, but are not limited to fingerprint, palm veins, face recognition, DNA, palm print, hand geometry, iris recognition, retina and odor/scent. Behavioural characteristics are related to the pattern of behaviour of a person, including but not limited to typing rhythm, gait, keystroke, signature, behavioural profiling, and voice. Some researchers have coined the term 'behaviometrics' to describe the latter class of biometrics.

More traditional means of access control include token-based identification systems, such as a driver's license or passport, and knowledge-based identification systems, such as a password or personal identification number. Since biometric identifiers are unique to individuals, they are more reliable in verifying identity than token and knowledge-based methods; however, the collection of biometric identifiers raises privacy concerns about the ultimate use of this information.

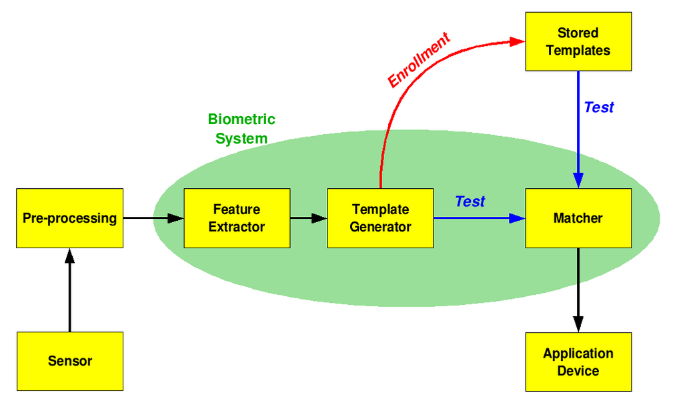
**Biometric functionality:**

Many different aspects of human physiology, chemistry or behaviour can be used for biometric authentication. The selection of a particular biometric for use in a specific application involves a weighting of several factors. Jain et al. (1999) identified seven such factors to be used when assessing the suitability of any trait for use in biometric authentication.

* Universality means that every person using a system should possess the trait.
* Uniqueness means the trait should be sufficiently different for individuals in the relevant population such that they can be distinguished from one another.
* Permanence relates to the manner in which a trait varies over time. More specifically, a trait with 'good' permanence will be reasonably invariant over time with respect to the specific matching algorithm.
* Measurability (collectability) relates to the ease of acquisition or measurement of the trait. In addition, acquired data should be in a form that permits subsequent processing and extraction of the relevant feature sets.
* Performance relates to the accuracy, speed, and robustness of technology used (see performance section for more details).
* Acceptability relates to how well individuals in the relevant population accept the technology such that they are willing to have their biometric trait captured and assessed.
* Circumvention relates to the ease with which a trait might be imitated using an artifact or substitute.

Proper biometric use is very application dependent. Certain biometrics will be better than others based on the required levels of convenience and security. No single biometric will meet all the requirements of every possible application.

The block diagram illustrates the two basic modes of a biometric system. First, in verification (or authentication) mode the system performs a one-to-one comparison of a captured biometric with a specific template stored in a biometric database in order to verify the individual is the person they claim to be. Three steps are involved in the verification of a person. In the first step, reference models for all the users are generated and stored in the model database. In the second step, some samples are matched with reference models to generate the genuine and impostor scores and calculate the threshold. The third step is the testing step. This process may use a smart card, username or ID number (e.g. PIN) to indicate which template should be used for comparison. 'Positive recognition' is a common use of the verification mode, "where the aim is to prevent multiple people from using the same identity".



Second, in identification mode the system performs a one-to-many comparison against a biometric database in an attempt to establish the identity of an unknown individual. The system will succeed in identifying the individual if the comparison of the biometric sample to a template in the database falls within a previously set threshold. Identification mode can be used either for 'positive recognition' (so that the user does not have to provide any information about the template to be used) or for 'negative recognition' of the person "where the system establishes whether the person is who she (implicitly or explicitly) denies to be". The latter function can only be achieved through biometrics since other methods of personal recognition such as passwords, PINs or keys are ineffective.

The first time an individual uses a biometric system is called enrolment. During enrolment, biometric information from an individual is captured and stored. In subsequent uses, biometric information is detected and compared with the information stored at the time of enrolment. Note that it is crucial that storage and retrieval of such systems themselves be secure if the biometric system is to be robust. The first block (sensor) is the interface between the real world and the system; it has to acquire all the necessary data. Most of the times it is an image acquisition system, but it can change according to the characteristics desired. The second block performs all the necessary pre-processing: it has to remove artifacts from the sensor, to enhance the input (e.g. removing background noise), to use some kind of normalization, etc. In the third block, necessary features are extracted. This step is an important step as the correct features need to be extracted in an optimal way. A vector of numbers or an image with particular properties is used to create a template. A template is a synthesis of the relevant characteristics extracted from the source. Elements of the biometric measurement that are not used in the comparison algorithm are discarded in the template to reduce the file size and to protect the identity of the enrolee. However, depending on the scope of the biometric system, original biometric image sources may be retained such as the PIV-cards used in the Federal Information Processing Standard Personal Identity Verification (PIV) of Federal Employees and Contractors.

During the enrolment phase, the template is simply stored somewhere (on a card or within a database or both). During the matching phase, the obtained template is passed to a matcher that compares it with other existing templates, estimating the distance between them using any algorithm (e.g. Hamming distance). The matching program will analyse the template with the input. This will then be output for a specified use or purpose (e.g. entrance in a restricted area), though it is a fear that the use of biometric data may face mission creep. Selection of biometrics in any practical application depending upon the characteristic measurements and user requirements. In selecting a particular biometric, factors to consider include, performance, social acceptability, ease of circumvention and/or spoofing, robustness, population coverage, size of equipment needed and identity theft deterrence. The selection of a biometric is based on user requirements and considers sensor and device availability, computational time and reliability, cost, sensor size, and power consumption.

**Multimodal biometric system:**

Multimodal biometric systems use multiple sensors or biometrics to overcome the limitations of unimodal biometric systems. For instance iris recognition systems can be compromised by aging irises and electronic fingerprint recognition can be worsened by worn-out or cut fingerprints. While unimodal biometric systems are limited by the integrity of their identifier, it is unlikely that several unimodal systems will suffer from identical limitations. Multimodal biometric systems can obtain sets of information from the same marker (i.e., multiple images of an iris, or scans of the same finger) or information from different biometrics (requiring fingerprint scans and, using voice recognition, a spoken passcode).

Multimodal biometric systems can fuse these unimodal systems sequentially, simultaneously, a combination thereof, or in series, which refer to sequential, parallel, hierarchical and serial integration modes, respectively. Fusion of the biometrics information can occur at different stages of a recognition system. In case of feature level fusion, the data itself or the features extracted from multiple biometrics are fused. Matching-score level fusion consolidates the scores generated by multiple classifiers pertaining to different modalities. Finally, in case of decision level fusion the final results of multiple classifiers are combined via techniques such as majority voting. Feature level fusion is believed to be more effective than the other levels of fusion because the feature set contains richer information about the input biometric data than the matching score or the output decision of a classifier. Therefore, fusion at the feature level is expected to provide better recognition results.

Spoof attacks consist in submitting fake biometric traits to biometric systems, and are a major threat that can curtail their security. Multi-modal biometric systems are commonly believed to be intrinsically more robust to spoof attacks, but recent studies have shown that they can be evaded by spoofing even a single biometric trait.

**Adaptive biometric systems**

Adaptive biometric systems aim to auto-update the templates or model to the intra-class variation of the operational data. The two-fold advantages of these systems are solving the problem of limited training data and tracking the temporal variations of the input data through adaptation. Recently, adaptive biometrics have received a significant attention from the research community. This research direction is expected to gain momentum because of their key promulgated advantages. First, with an adaptive biometric system, one no longer needs to collect a large number of biometric samples during the enrolment process. Second, it is no longer necessary to enrol again or retrain the system from scratch in order to cope with the changing environment. This convenience can significantly reduce the cost of maintaining a biometric system. Despite these advantages, there are several open issues involved with these systems. For miss-classification error (false acceptance) by the biometric system, cause adaptation using impostor sample. However, continuous research efforts are directed to resolve the open issues associated to the field of adaptive biometrics.

**Recent advances in emerging biometrics:**

In recent times, biometrics based on brain (electroencephalogram) and heart (electrocardiogram) signals have emerged. The research group at University of Kent led by Ramaswamy Palaniappan has shown that people have certain distinct brain and heart patterns that are specific for each individual. Another example is finger vein recognition, using pattern-recognition techniques, based on images of human vascular patterns. The advantage of such 'futuristic' technology is that it is more fraud resistant compared to conventional biometrics like fingerprints. However, such technology is generally more cumbersome and still has issues such as lower accuracy and poor reproducibility over time. This new generation of biometrical systems is called biometrics of intent and it aims to scan intent. The technology will analyse physiological features such as eye movement, body temperature, breathing etc. and predict dangerous behaviour or hostile intent before it materializes into action.

On the portability side of biometric products, more and more vendors are embracing significantly miniaturized biometric authentication systems (BAS) thereby driving elaborate cost savings, especially for large-scale deployments.

**CHAPTER 2**

**LITERATURE REVIEW**

**[1] Naveed Khan Balcoh, M. HaroonYousaf, Waqar Ahma and M. Iram Baig:** Students attendance in the classroom is very important task and if taken manually wastes a lot of time. There are many automatic methods available for this purpose i.e. Biometric attendance. All these methods also waste time because students have to make a queue to touch their thumb on the scanning device. This work describes the efficient algorithm that automatically marks the attendance without human intervention. This attendance is recorded by using a camera attached in front of classroom that is continuously capturing images of students, detect the faces in images and compare the detected faces with the database and mark the attendance.

**Summary:** Students attendance in the classroom is very important task and if taken manually wastes a lot of time. There are many automatic methods available for this purpose i.e. Biometric attendance. All these methods also waste time because students have to make a queue to touch their thumb on the scanning device. This work describes the efficient algorithm that automatically marks the attendance without human intervention.

**[2] NirmalayaKar, MrinalKanti Debbarma, Ashim Saha, and Dwijen RudraPal:** Authentication is a significant issue in system control in computer based communication. Human face recognition is an important branch of biometric verification and has been widely used in many applications, such as video monitor system, human-computer interaction, and door control system and network security. This describes a method for Student’s Attendance System which will integrate with the face recognition technology using Personal Component Analysis (PCA) algorithm. The system will record the attendance of the students in class room environment automatically and it will provide the facilities to the faculty to access the information of the students easily by maintaining a log for clock-in and clock-out time.

**Summary:** This describes a method for Student’s Attendance System which will integrate with the face recognition technology using Personal Component Analysis (PCA) algorithm. The system will record the attendance of the students in class room environment automatically and it will provide the facilities to the faculty to access the information of the students easily by maintaining a log for clock-in and clock-out time.

**[3] O. Shoewn:** In this paper, the development of an attendance management system using biometrics is proposed. Managing student attendance during lecture periods has become a difficult challenge. The ability to compute the attendance percentage becomes a major task as manual computation produces errors, and also wastes a lot of time. For the stated reason, an efficient attendance management system using biometrics is designed. This system takes attendance electronically with the help of a finger print device and the records of the attendance are stored in a database. Attendance is marked after student identification.

**Summary:** For student identification, a biometric (fingerprint) identification based system is used. This process however, eliminates the need for stationary materials and personnel for the keeping of records. Eighty candidates were used to test the system and success rate of 94% was recorded. The manual attendance system average execution time for eighty students was 17.83 seconds while it was 3.79 seconds for the automatic attendance management system using biometrics.

**[4] M. Turk and A. Pentland:** An approach to the detection and identification of human faces is presented, and a working, near-real-time face recognition system which tracks a subject's head and then recognizes the person by comparing characteristics of the face to those of known individuals is described. This approach treats face recognition as a two-dimensional recognition problem, taking advantage of the fact that faces are normally upright and thus may be described by a small set of 2-D characteristic views. Face images are projected onto a feature space ('face space') that best encodes the variation among known face images. The face space is defined by the 'Eigen faces', which are the eigenvectors of the set of faces; they do not necessarily correspond to isolated features such as eyes, ears, and noses. The framework provides the ability to learn to recognize new faces in an unsupervised manner.

**Summary:** This approach treats face recognition as a two-dimensional recognition problem, taking advantage of the fact that faces are normally upright and thus may be described by a small set of 2-D characteristic views. Face images are projected onto a feature space ('face space') that best encodes the variation among known face images. The face space is defined by the 'Eigen faces', which are the eigenvectors of the set of faces; they do not necessarily correspond to isolated features such as eyes, ears, and noses.

**[5] Rekha A. L, Chethan H. K**: The objective of this system is to present an automated system for human face recognition in a real time background for an organization to mark the attendance of their employees or student. So automated attendance using real time face recognition is a real world solution which comes with day to day activities of handling employees or student. The task is very difficult as the real time background subtraction in an image is still a challenge. In the past two decades, face detection and recognition has proven to be very interesting research field of image processing.

**Summary:** The work carried out describes an automated attendance system using video surveillance. The proposed algorithm is automatic and efficient in intelligent surveillance applications. Video surveillance is used to detect the object movement thereby the captured image undergoes face detection and recognition process and searches the student database and enters the attendance if it is valid in the list.

**CHAPTER 3**

**EXISTING METHOD**

In existing system biometric system is usually used in any organisation which takes a lot of time and it also requires finger print of a person to take the attendance. Sometimes which will be failed in detecting the finger print that which could cause problem in taking attendance.

**Disadvantages:**

* Time taking.
* Sometimes failed in detecting the finger print.
* Cannot be used properly for biometrics.

**CHAPTER 4**

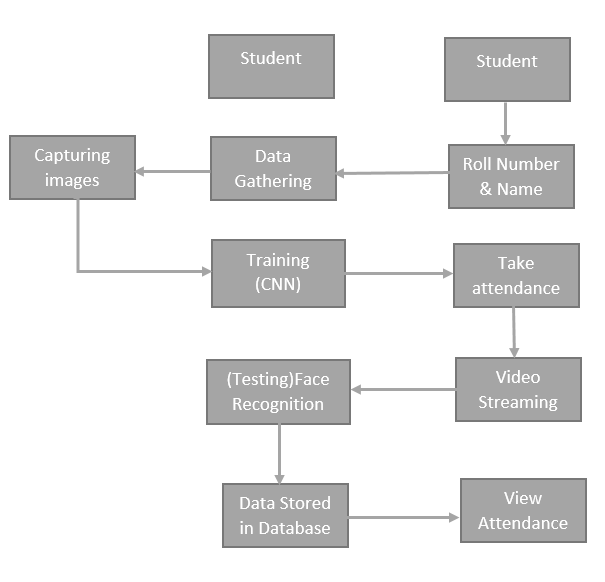
**PROPOSED METHOD**

In this paper, we have created a model that which can take the attendance of student in the allotted times by the face recognition if student not recognized I the allotted attendance taken time they will be given attendance as late coming. The automatic attendance management will replace the manual method, which takes a lot of time consuming and difficult to maintain. There are many biometric processes, in that face recognition is the best method. Here we are using the computer vision which is a field of deep learning that is used for the camera reading and writing and using Flask to create a GUI application.

**Advantages:**

* Accurate option
* Increased productivity
* Easy and safe to use
* Convenient option

**Block Diagram:**



**APPLICATIONS**

* Logical Access Control
* Physical Access Control
* Time and Attendance
* Law Enforcement
* Surveillance

**SYSTEM SPECIFICATIONS**

# **H/W Configuration:**

# Processor : I3/Intel Processor

* Hard Disk : 160GB
* RAM : 8Gb

**S/W Configuration:**

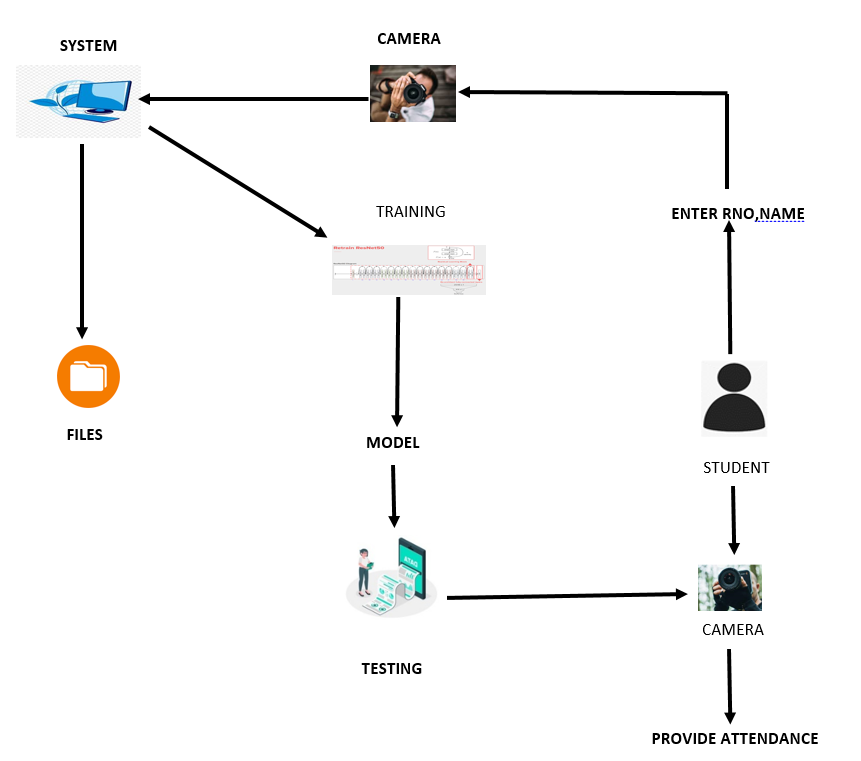
* Operating System : Windows 7/8/10 .
* IDE : Pycharm.
* Libraries Used : Numpy, CV2, OS, Flask.
* Technology : Python 3.6+.

**PROBLEM STATEMENT**

In manual attendance recording is not accurate because no one can taken properly, this causes the fake attendance and time consuming.

To overcome the above problem we introduce the biometric based attendance to reducing the time and face count.

**ARCHITECTURE**



**SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS**

Installing Python:

1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



1. Once the download is complete, run the exe for install Python. Now click on Install Now.
2. You can see Python installing at this point.
3. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

Installing PyCharm:

1. To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and Click the "DOWNLOAD" link under the Community Section.



1. Once the download is complete, run the exe for install Pycharm. The setup wizard should have started. Click “Next”.
2. On the next screen, Change the installation path if required. Click “Next”.
3. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
4. Choose the start menu folder. Keep selected Jet Brains and click on “Install”.
5. Wait for the installation to finish.
6. Once installation finished, you should receive a message screen that Pycharm is installed. If you want to go ahead and run it, click the “Run Pycharm Community Edition” box first and click “Finish”.
7. After you click on "Finish," the Following screen will appear.



9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like numpy, pandas, seaborn, scikit-learn, matplotlib.pyplot)

Ex: pip install numpy



**MODULES**

**System**

**Data Gathering:**

After clicking the capture button the web cam will be opened for taking images and it captures 200 frames, after completion of taking images web cam will be automatically closed.

**Training:**

All captured images are stored into a Trainingimage folder. For training we use LBPHFaceRecognizer\_create() method which is used for recognising the face features and we use CNN algorithm for classifying the faces. After training with CNN the model will be saved in to directory.

**Testing and considering the attendance**

Whenever a student made a click on the button provided, a web cam will gets opened.

The web cam has ability to capture the image of the particular students face and then the image is converted into greyscale and it undergoes for scaling

The scaled image is converted into the form of vectors with the help of LBPHFaceRecognizer\_create

Now the converted data will be helpful to predict the outcomes.

There are 4 conditions to collect the attendance

1: Before 10AM – Early come 2. After 10AM – Late

3. Before 4PM – Early out 4. After 4PM- Normal out

After checking the conditions data will be stored into MySQL database.

**STUDENT**

**Enter Data:**

After student adding the details like Roll number and Name.

After entering the details student click on the image capture button.

**Take Attendance**

Here student can select the options like take attendance button then only system can access the attendance. After that system stores the information into MYSQL database.

**View Attendance**

After entering the roll number student can view their attendance details of overall present hours in the college.

**STEPS FOR EXECUTING THE PROJECTS**

1. Install the required packages

2. Using TkInter enter the data of the student.

3. Capture the image of student.

4. Train the images using the model.

5. Test by taking picture of student.

6. Attendance is registered by the taken picture.

**SYSTEM DESIGN**

**UML Diagrams**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful insists the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

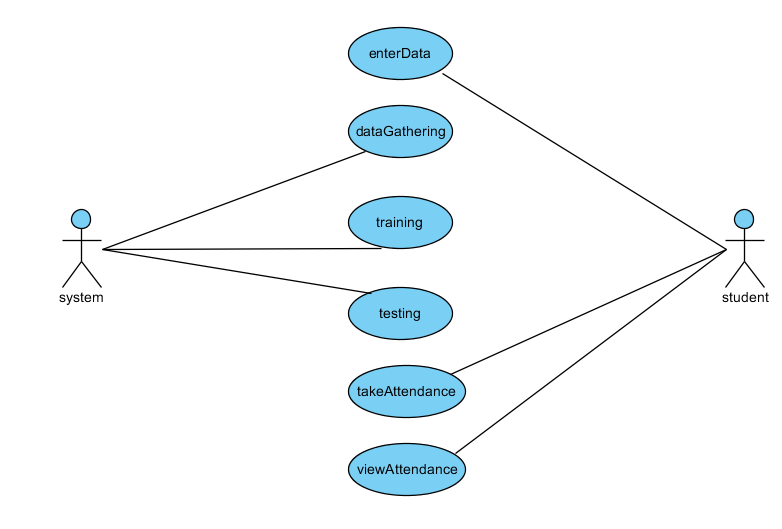
**Goals:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

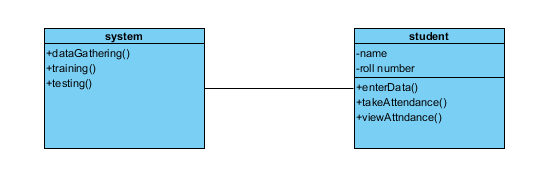
**Use Case Diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



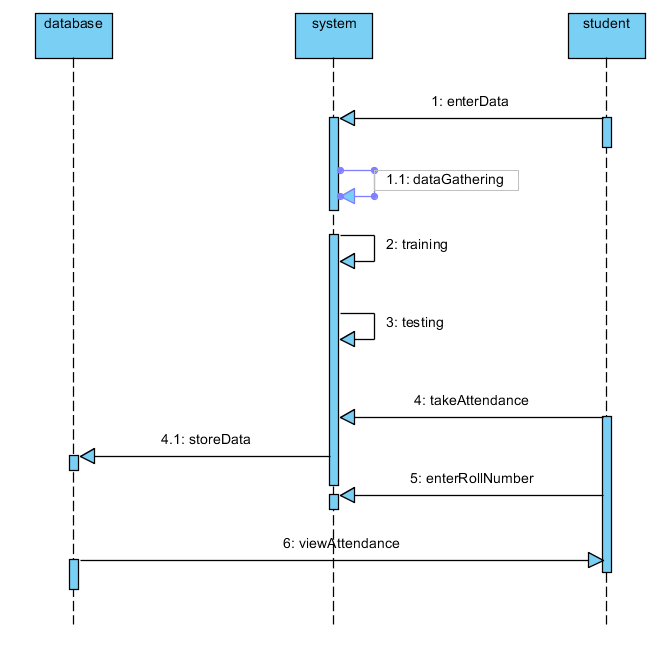
**Class Diagram:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



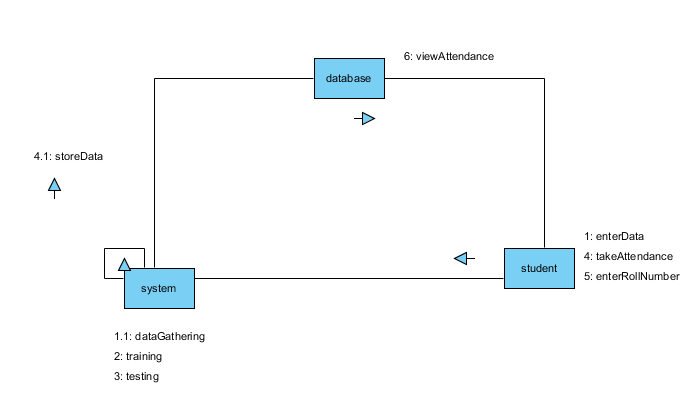
**Sequence Diagram:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and time diagrams.

****

**Collaboration Diagram:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

****

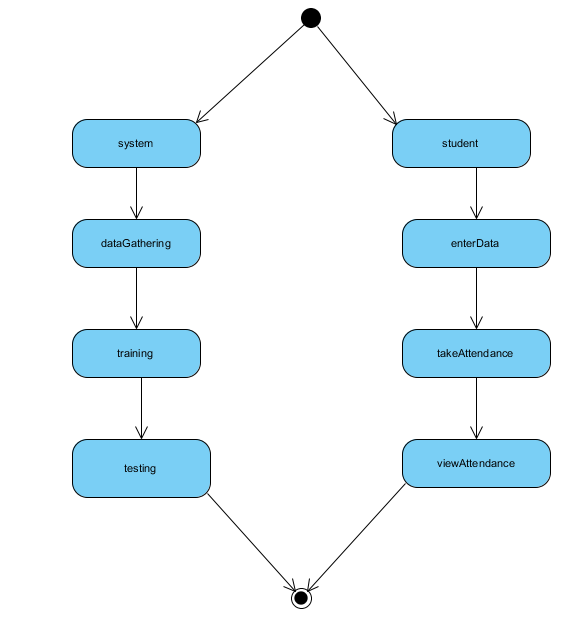
**Deployment diagram**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.



**Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**Component diagram**:

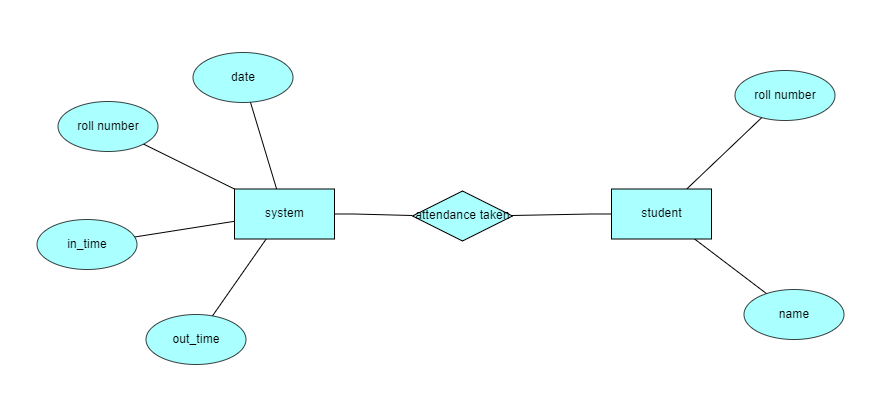
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.



**ER Diagram:**

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

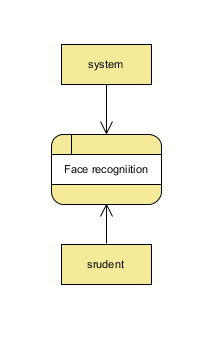
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.

****

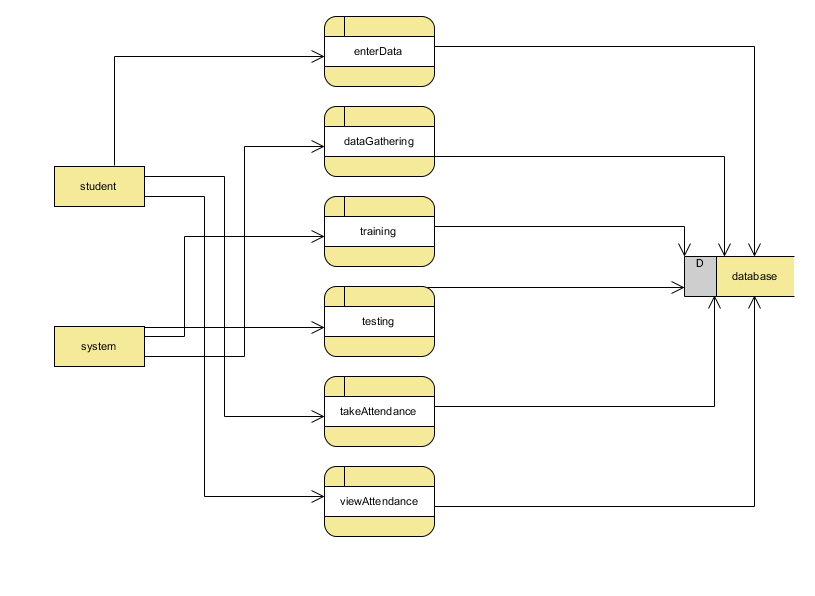
**DFD Diagram:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

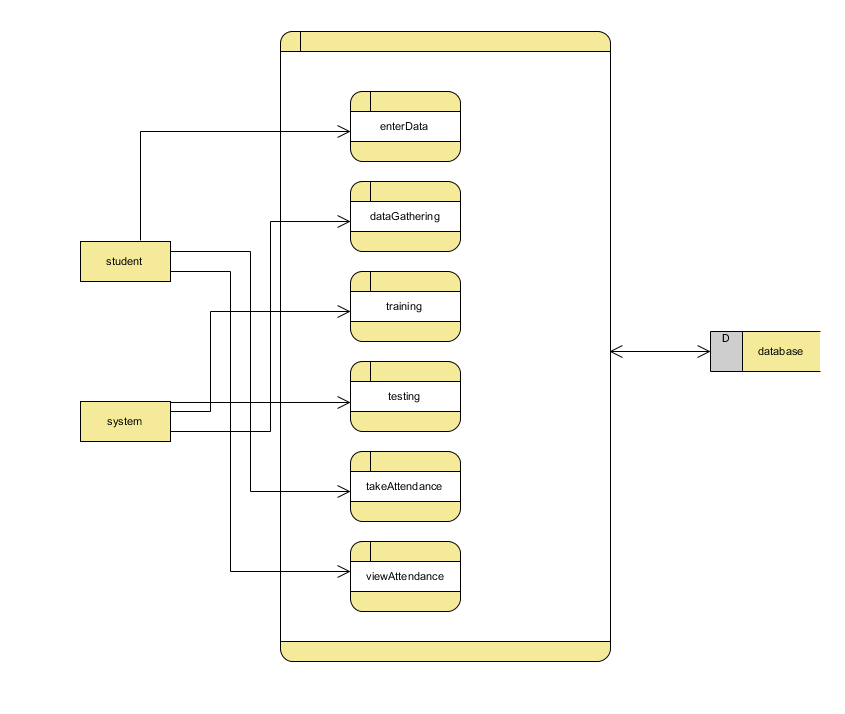
**CONTEXT LEVEL DIAGRAM**



**LEVEL-1 DIAGRAM**



**LEVEL-2 DIAGRAM**



# **INTRODUCTION TO PYTHON**

* **Python**

### **What is a Script?**

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

**Scripts are reusable**

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

**Scripts are editable**

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

**You will need a text editor**

Just about any text editor will suffice for creating Python script files.

You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word*,*or just about any word processor if you want to.

**Difference between a script and a program**

**Script:**

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

**Program:**

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

**Python**

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

**Python concepts**

If you’re not interested in the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-is to interface with C++ (via SWIG)

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical 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 Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Possum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

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

**Dynamic vs. Static**

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

**Variables**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

**Standard Data Types**

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## **Python Numbers**

Number data types store numeric values. Number objects are created when you assign a value to them

## **Python Strings**

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## **Python Lists**

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

## **Python Tuples**

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

## **Python Dictionary**

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

# **20 Python libraries**

**1.** Requests. The most famous http library written by Kenneth remits. It’s a must have for every python developer.

**2.** Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.

**3.** Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

**4.** Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

**5.** SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

**6.** Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.

**7.** Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

**8.** Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

**9.** Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

**10.** Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyser.

**11.** Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

**12.** Piglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of mine craft was made

**13.** Pit. A GUI toolkit for python. It is my second choice after python for developing GUI’s for my python scripts.

**14.** Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

**15.** Scaly. A packet sniffer and analyser for python made in python.

**16.** Pywin32. A python library which provides some useful methods and classes for interacting with windows.

**17.** Notch. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

**18.** Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

**19.** Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

**20.** I Python. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**Numpy**

Humpy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

**Matplotlib**

• High quality plotting library.

**Python class and objects**

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behaviour. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

**Inheritance**

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by sub classing a class, you can change the behaviour of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behaviour of your classes without changing the superclass at all.

**Operator Overloads**

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behaviour closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

**Exceptions**

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

. • Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

**User-Defined Exceptions**

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class. To make a custom exception, simply inherit the base exception and define what it will do.

**Python modules**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

**Functions in Python**

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

## There are three types of functions in python:

## Help (), min (), print ().

## **Python Namespace**

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.   
Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* **global names** of a module
* **local names** in a function or method invocation
* **built-in names**: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

**Garbage Collection**

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

**Python XML Parser**

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register call-backs for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

**Python Web Frameworks**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

**Common web framework functionality**

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## **Comparing web frameworks**

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

## **Web framework resources**

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
* Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?” The votes aren't as important as the list of the many frameworks that are available to Python developers.

## **Web frameworks learning checklist**

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

**SYSTEM STUDY**

**Feasibility Study**

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* Economical Feasibility
* Technical Feasibility
* Social Feasibility

**Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

### **SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**Types of Tests**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# **Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CONCLUSION**

In our proposed work, we have created a model that which can take the attendance of student in the allotted times by the face recognition if student not recognized I the allotted attendance taken time they will be given attendance as late coming. We have used Flask Framework, where the information about the student are stored and a model is trained and then the student picture is captured which is tested and attendance is taken to the student by the captured face image.

**FUTURE WORK**

In the future, we can extend the idea and can apply in different fields, like educational, corporate offices, and in many work places. That which can be easy to consider the attendance of a person with a lesser time.

**REFERENCES**

[1] Naveed Khan Balcoh, M. HaroonYousaf, Waqar Ahma and M. Iram Baig, Algorithm for efficient Attendance Management: Face Recognition Based approach, International Journal of Computer Science Issue, Vol.9, Issue 4, No 1, July 2012.

[2] NirmalayaKar, MrinalKanti Debbarma, Ashim Saha, and Dwijen RudraPal, Study of implementing Automated Attendance System using Implementing Automated Attendance System Using face recognition Technique, International Journal of Computer and Communication Engineering, Vol 1, No 2,July 2012

[3] O. Shoewn, Development of Attendance Management System using Biometrics. The Pacific Journal of Science and Technology Volume 13, No 1, May 2012

[4] M. Turk and A. Pentland (1991) “Face recognition using Eigen faces”. Proc.IEEE conference on computer vision and Pattern Recognition

[5] W.Zhao, R. Chellapa, P.J.Phillips and A.Rosenfld, “Face Recognition: A Literature Survey, vol. 35, No 4, Dec 2003, pp.399-458

[6] R.L. Hsu, Mottalec M.A and A.K.Jain,”Face Detection in colour images”, Proceedings International Conference on Image Processing, Oct 2001, pp. 1046-1049

[7] ToufiqP. Ahmed Egammal and Anurag mittal (2006),”A Framework for feature selection for Background Subtraction”, in Proceedings of IEEE computer Society Conference on Computer Vision and Pattern Recognition.

[8] M.H.Yang, N.Ahuja and D.Kriegmao, “Face recognition using kernel Eigen faces”, IEEE International Conference on Image Processing, vol.1, pp. 10-13, Sept. 2000

[9] Rekha A. L, Chethan H. K, “Automated Attendance System Management System Using Face Recognition through Video Surveillance”, Volume 1, Issue 11, July-2014

[10] Anil K. Jain, Arun Ross and Salil Prabhakar, “An introduction to biometric recognition”, Circuits and Systems for Video Technology, IEEE Transcations on Volume 14, Issu 1, Jan 2004 Page(s):4-20

[11] H. M. El Barkey, “Face detection using fast neural networks and image decomposition”, Neuro computing, Vol.11, no 3, pp 1039- 1046, 2002

[12] R. L. Hsu, Mottale M.A and A.K. Jain, “Face detection in colour images”, Proceedings International Conference on Image Processing (ICIP), Oct 2001, pp.1046-1049

[13] Prabhjot Singh and Anjana Sharma, “Face Recognition using Principal Component Analysis in MATLAB”, International Journal of Scientific Research in Computer Science and Engineering, vol.3,Issue 1, ISSN 2320-7639