**Abnormal Events Detection and Tracking in Surveillance System Using Machine Learning**

**ABSTRACT**

The "**Abnormal Events Detection and Tracking in Surveillance System using Machine Learning**" is an innovative approach to enhance the effectiveness of surveillance systems by incorporating advanced machine learning techniques. This system aims to identify and track abnormal events within surveillance footage, such as intrusions, suspicious activities, and unusual behavior, by leveraging the capabilities of machine learning algorithms. By automating the detection and tracking of abnormal events, the proposed system significantly improves the efficiency and accuracy of surveillance operations, contributing to enhanced security and proactive incident management.

With the increasing in the number of anti-socials activates that have been taking place, security has been given utmost importance lately. Many Organizations have installed CCTVs for constant Monitoring of people and their interactions. For a developed Country with a population of 64 million, every person is captured by a camera 30 times a day. A lot of video data generated and stored for a certain time duration. A 704x576 resolution image recorded at 25fps will generate roughly 20GB per day. Constant Monitoring of data by humans to judge if the events are abnormal is near impossible task as requires a workforce and their constant attention. This creates a need to automate the same. Also, there is need to show in which frame and which part of it contain the unusual activity which aid the faster judgment of the unusual activity being abnormal. This is done by converting video into frames and analyzing the persons and their activates from the processed frame. Machine learning and Deep Learning Algorithms and techniques support us in a wide accept to make Possible.

**CHAPTER 1**

**INTRODUCTION**

* 1. **INTRODUCTION TO THE PROJECT**

Human face and human behavioural pattern play an important role in person identification. Visual information is a key source for such identifications. Surveillance videos provide such visual information which can be viewed as live videos, or it can be played back for future references. The recent trend of ‘automation’ has its impact even in the field of video analytics. Video analytics can be used for a wide variety of applications like motion detection, human activity prediction, person identification, abnormal activity recognition, vehicle counting, people counting at crowded places, etc. In this domain, the two factors which are used for person identification are technically termed as face recognition and gait recognition respectively. Among these two techniques, face recognition is more versatile for automated person identification through surveillance videos. Face recognition can be used to predict the orientation of a person’s head, which in turn will help to predict a person’s behaviour. Motion recognition with face recognition is very useful in many applications such as verification of a person, identification of a person and detecting presence or absence of a person at a specific place and time. In addition, human interactions such as subtle contact among two individuals, head motion detection, hand gesture recognition and estimation are used to devise a system that can identify and recognize suspicious behaviour among pupil in an examination hall successfully. This paper provides a methodology for suspicious human activity detection through face recognition.

Video processing is used in two main domains such as security and research. Such a technology uses intelligent algorithms to monitor live videos. Computational complexities and time complexities are some of the key factors while designing a real-time system. The system which uses an algorithm with a relatively lower time complexity, using less hardware resources and which produces good results will be more useful for time-critical applications like bank robbery detection, patient monitoring system, detecting and reporting suspicious activities at the railway station, etc

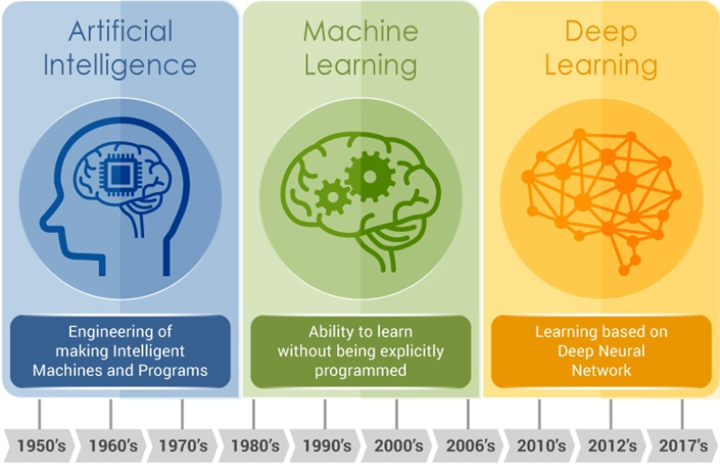
Manual monitoring of exam hall through invigilators and manual monitoring of exam hall through surveillance videos is performed throughout the world. Monitoring an examination hall is a very challenging task in terms of man power. Manual monitoring of examination halls may be prone to error during human supervision. Such a system when implemented as an ‘automatic suspicious activity detection system’ will not only help in detecting suspicious activities but also helps in minimizing such activities. Moreover, the probability of error will be much lesser. This system will serve as a useful surveillance system for educational institutions.

This paper describes a technology in which real time videos are analysed and are used for human activity analysis in an examination hall, thus helping to classify whether the particular person’s activity is suspicious or not. The system developed identifies abnormal head motions, thereby prohibiting copying. It also identifies a student moving out of his place or swapping his position with another student. Finally the system detects contact between students and hence prevents passing incriminating material among students. In our research, we have contributed upon a system that will intellectually process live video of examination halls with students and classify their activities as suspicious or not. This research proposes an intelligent algorithm that can monitor and analyse the activities of students in an examination hall and can alert the educational institute’s administration on account of any malpractices/suspicious activities.

The Suspicious Human Activity Detection system aims to identify the students who indulge in malpractices/suspicious activities during the course of an examination. The system automatically detects suspicious activities and alerts administration.

**1.2. INTRODUCTON TO ARTIFICIAL INTELLIGENCE:**

* Inventors have long dreamed of creating machines that think. When programmable computers were first conceived, people wondered whether they might become intelligent, over a hundred years before one was built. Today, **artificial intelligence (AI)** is a thriving field with many practical applications and active research topics. We look to intelligent software to automate routine labour, understand speech or images make diagnoses in medicine and support basic scientific research.
* In the early days of artificial intelligence, the field rapidly tackled and solved problems that are intellectually difficult for human beings but relatively straight-forward for computers problems that can be described by a list of formal, mathematical rules. The true challenge to artificial intelligence proved to be solving the tasks that are easy for people to perform but hard for people to describe formally problems that we solve intuitively, that feel automatic, like recognizing spoken words or face sin images.



**Fig 1.1** Evolution of Artificial Intelligence

* Ironically, abstract and formal tasks that are among the most difficult mental undertakings for a human being are among the easiest for a computer. Computers have long been able to defeat even the best human chess player, but are only recently matching some of the abilities of average human being store cognize objects or speech. A person’s everyday life requires an immense amount of knowledge about the world. Much of this knowledge is subjective and intuitive, and therefore difficult to articulate in a formal way. Computers need to capture this same knowledge in order to behave in an intelligent way. One of the key challenges in artificial intelligence is how to get this informal knowledge into a computer.
* The difficulties faced by systems relying on hard coded knowledge suggest that AI systems need the ability to acquire their own knowledge, by extracting patterns from raw data. This capability is known as **machine learning**.
* In computer science, **Artificial Intelligence** (**AI**), sometimes called **machine intelligence**, is intelligence demonstrated by machines, in contrast to the **natural intelligence** displayed by humans. Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving".
* Artificial intelligence can be classified into three different types of systems:

Analytical, Human-inspired, and Humanized artificial intelligence.

* **Analytical AI** has only characteristics consistent with cognitive intelligence; generating a cognitive representation of the world and using learning based on past experience to inform future decisions.
* **Human-inspired AI** has elements from cognitive and emotional intelligence; understanding human emotions, in addition to cognitive elements, and considering them in their decision making.
* **Humanized AI** shows characteristics of all types of competencies (i.e., cognitive, emotional, and social intelligence), is able to be self-conscious and is self-aware in interactions.
* Artificial intelligence **(AI)** is an area of computer science that aims at emphasizing the ability of machines to behave the way humans do which includes complex tasks such as observing, learning, planning and making decisions for problem-solving. It’s as simple as:

**Artificial + Intelligence**

* **Artificial** means something that’s not natural
* **Intelligence** the ability to acquire and apply knowledge and skills

**AI = Crafted intelligent behaviour by machines.**

If we delve deeper, AI can be divided into two categories:

* **General AI:** Would have all of the characteristics of human intelligence
* **Narrow AI:** Exhibits some facets of human intelligence, sort of like specialization.

|  |  |
| --- | --- |
|  |  |

**CHAPTER 3**

**OBJECTIVE OF THE PROJECT**

**3.1.1. EXISTING SYSTEM**

Traditional surveillance systems rely heavily on manual monitoring, which can be labor-intensive and error-prone. These systems lack the capability to automatically identify and track abnormal events, leading to delayed responses and missed incidents. Furthermore, they struggle with the increasing volume of surveillance data generated, making it difficult for operators to analyze footage effectively.

**3.1.2. PROPOSED SYSTEM AND ADVANTAGES**

The proposed "Abnormal Events Detection and Tracking in Surveillance System using Machine Learning" presents a comprehensive solution to the limitations of traditional surveillance systems. The system encompasses the following components:

1. **Video Analysis and Preprocessing:** Surveillance footage is analyzed using computer vision techniques to preprocess and extract relevant features from the video frames.
2. **Abnormal Event Detection:** Machine learning models, such as anomaly detection algorithms and deep learning networks, are trained on normal behavior patterns. These models then identify deviations from the established norms, flagging events as potentially abnormal.
3. **Event Tracking:** Once an abnormal event is detected, the system initiates event tracking algorithms to follow and monitor the event's progression within the video stream.
4. **Real-time Alerts and Notifications:** The system generates real-time alerts and notifications, notifying security personnel or administrators about detected abnormal events.
5. **Historical Data Analysis:** The system archives detected events and associated tracking data, enabling post-event analysis and investigation.

The proposed system revolutionizes surveillance by automating the identification and tracking of abnormal events using cutting-edge machine learning techniques. It empowers security personnel with real-time insights, enabling rapid response and proactive security measures. By incorporating machine learning, the system adapts and evolves, becoming more adept at accurately detecting and tracking abnormal events, ultimately contributing to safer environments and more efficient security management.

**ADVANATGES OF PROPOSED SYSTEM**

# Prediction model gives 99.56% accuracy

# More time saving

# Less expensive

# Easy to operate

**3.1.3 SCOPE O F THE SYSTEM**

The scope of the "Abnormal Events Detection and Tracking in Surveillance System using Machine Learning" is to create a comprehensive and efficient solution for enhancing surveillance systems' capabilities in detecting and tracking abnormal events. The system aims to automate the process of identifying and monitoring unusual or suspicious activities within video footage, contributing to improved security and incident management. The scope of the system encompasses several key areas:

**CHAPTER 4**

**SYSTEM ANALYSIS**

**4.1 Feasibility Study**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation.

**1.**  ECONOMIC FEASIBILITY

**2.**  OPERATIONAL FEASIBILITY

**3.**  TECHNICAL FEASIBILITY

1. **Economic Feasibility**

* A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.
* The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

1. **Operational Feasibility**

* Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following:
  + - Is there sufficient support for the management from the users?
    - Will the system be used and work properly if it is being developed and implemented?
    - Will there be any resistance from the user that will undermine the possible application benefits?
* This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

1. **Technical Feasibility**

* The technical issue usually raised during the feasibility stage of the investigation includes the following:
  + - Does the necessary technology exist to do what is suggested?
    - Do the proposed equipments have the technical capacity to hold the data required to use the new system?
    - Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
    - Can the system be upgraded if developed?
    - Are there technical guarantees of accuracy, reliability, ease of access and data security?
* Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source.

**CHAPTER 5**

**SOFTWARE REQUIREMENT SPECIFICATION**

**5.1. INTRODUTION**

* A **Software Requirements Specification** (**SRS**) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) – is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Non-functional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).
* **System requirements specification:** A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Projects are subject to three sorts of requirements:
  + - [**Business requirements**](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
    - **Product requirements** describe properties of a system or product (which could be one of several ways of business requirements.)
    - **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.
* Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles.

**5.2. PERFORMANCE REQUIREMENTS**

* Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.
* The requirement specification for any system can be broadly stated as given below:
* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

**5.3. FUNCTIONAL REQUIREMENTS**

The Functional Requirements Specification documents the operations and activities that a system must be able to perform.

Functional Requirements should include:

1. Descriptions of data to be entered into the system
2. Descriptions of operations performed by each screen
3. Descriptions of system reports or other outputs
4. How the system meets applicable regulatory requirements
5. Descriptions of prediction outputs

The Functional Requirements Specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

## 5.3.1 Input

# The user will only be required to input the x-ray image for which they want to know the type of the lung x-ray. They do not require registration for using the service.

**5.3.2 Output**

# The user will obtain a covid-19 label as output for which they gave input.

# 5.4 Non-functional requirements

# Non-functional requirements are often called "[quality attributes](https://en.wikipedia.org/wiki/List_of_system_quality_attributes)" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioral requirements". Informally these are sometimes called the "[utilities](https://en.wiktionary.org/wiki/ility)", from attributes like stability and portability. Qualities—that is non-functional requirements—can be divided into two main categories:

1. Execution qualities, such as safety, security, and usability, which are observable during operation (at run time).
2. Evolution qualities, such as [testability](https://en.wikipedia.org/wiki/Software_testability), maintainability, extensibility, and scalability, which are embodied in the static structure of the system.

**5.4.1 Reliability**

# The system produces reliable output which is true

**5.4.2 Accuracy**

# The results produced by the system must be accurate.

**5.4.3 Performance**

# The performance of the system is high which produces faster output.

**5.4..4 Modifiability**

# The system can be easily modified if we want to make any new changes.

**5.5. HARDWARE REQUIREMENTS**

* The Hardware consists of the physical components of the computer that input storage processing control, output devices. Most hardware only has operating system requirements or compatibility. For example, a printer may be compatible with Windows XP but not compatible with newer versions of Windows like Windows 10, Linux, or the Apple macOS.
* The kind of hardware used in the project is
* PROCESSOR : i5
* RAM : 4GB
* HARD DISK : 5GB (minimum)

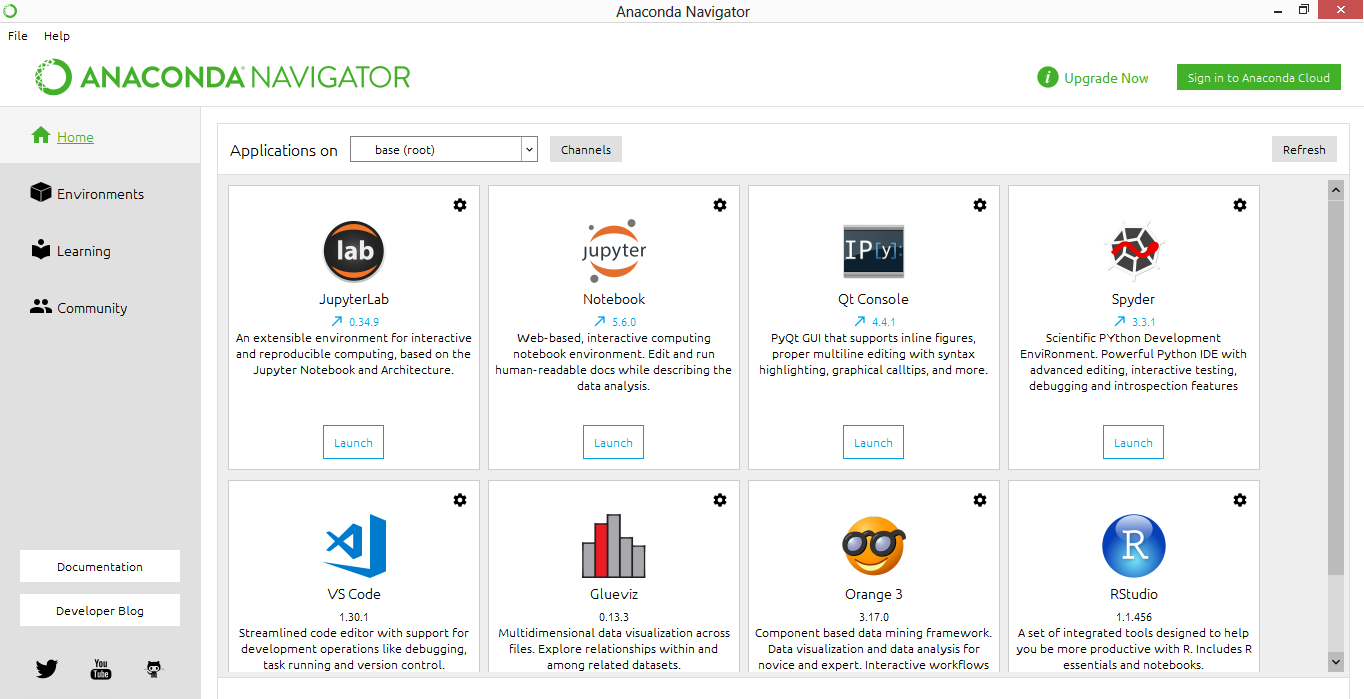
**5.6. SOFTWARE REQUIREMENTS**

* Software is a set of programs to do a particular task. Software is an essential requirement of computer systems.The system requirements or software requirements is a listing of what software programs are required to operate the program properly.
* The kind of software used in the project is
* Operating System : Windows 10
* IDE : JupyterNotebook(Anaconda)
* Coding Language : Python

**5.6.1. SOFTWARE REQUIREMENTS DESCRIPTION**

**1. Anaconda**

* **Anaconda** is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 6 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.
* Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager, called Anaconda Navigator, so it eliminates the need to learn to install each library independently.
* The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.
* The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.



**Anaconda Navigator**

### Anaconda Navigator

* Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, mac OS and Linux.
* The following applications are available by default in Navigator:

JupyterLab

Jupyter Notebook

QtConsole

Spyder

Glueviz

Orange

Rstudio

Visual Studio Code

### Conda:

* Conda is an open source, cross-platform, language-agnostic package manager and environment management systemthat installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., [R](https://en.wikipedia.org/wiki/R_(programming_language))), including multi-language projects. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

**Anaconda Cloud:**

* Anaconda Cloud is a package management service by Anaconda where you can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. You do not need to log in or to have a Cloud account, to search for public packages, download and install them.

**2. Jupyter Notebook**

* The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results.
* The Jupyter notebook combines two components:

**A web application**: A browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.

**Notebook documents**: A representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

**3. Python**

* Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community.

**FEATURES:**

* Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.
* Python's design offers some support for functional programming in the Lisp tradition. It has filter(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions.
* The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:
* Beautiful is better than ugly
* Explicit is better than implicit
* Simple is better than complex
* Complex is better than complicated
* Readability counts

**LIBRARIES:**

* Python's large standard library, commonly cited as one of its greatest strengths provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and unit testing.
* Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation wsgiref follows PEP 333, but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.
* As of March 2018, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 130,000  packages with a wide range of functionality, including:
* Graphical user interfaces
* Web frameworks
* Multimedia
* Databases
* Networking
* Automation
* Web scraping
* Documentation
* System administration
* Scientific computing
* Text processing
* Image processing

## TECHNOLOGIES USED

**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management.

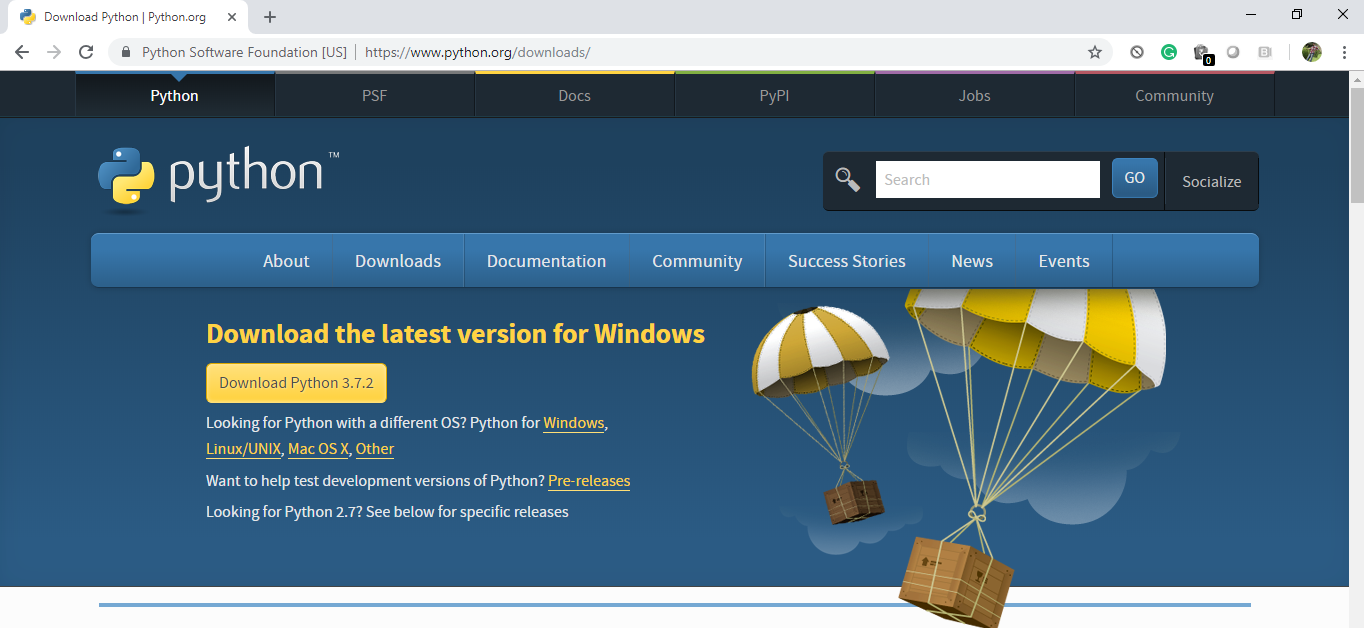
It supports multiple programming paradigms, including object oriented, imperative, functional and has a large and comprehensive standard library.

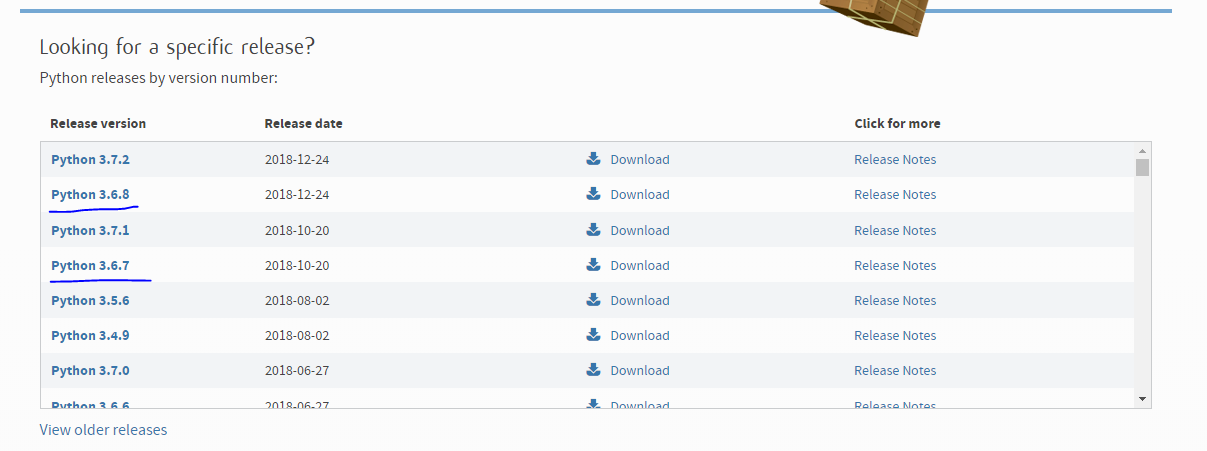
Python Installations:

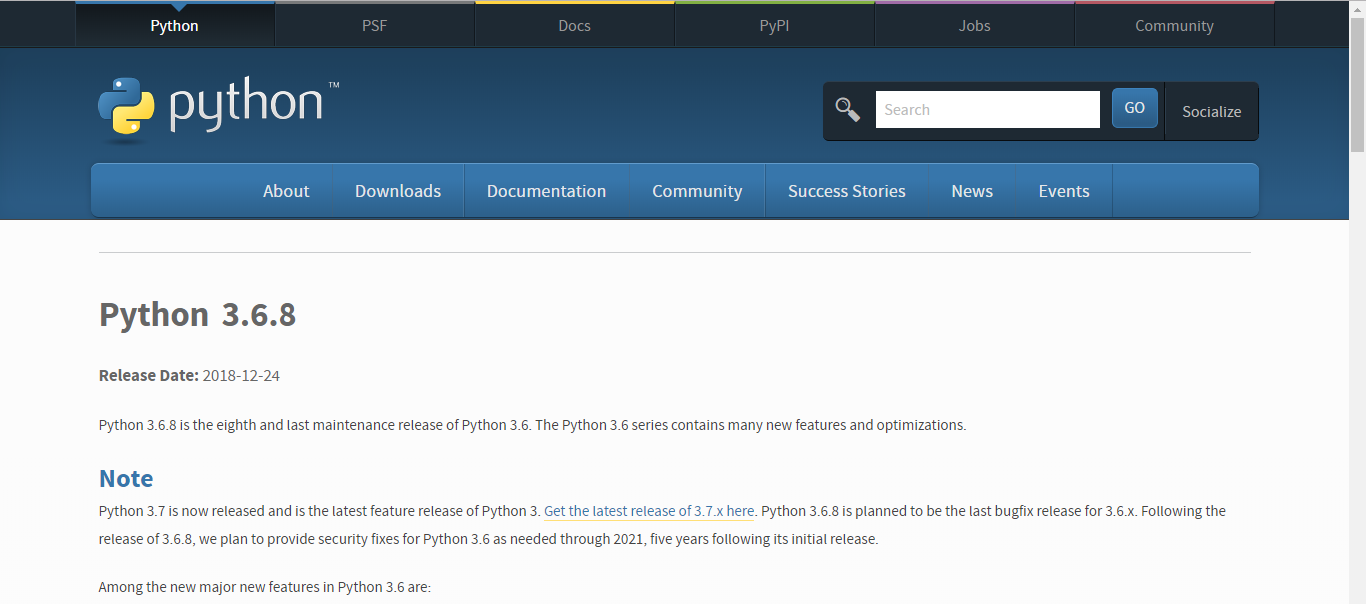
* Python Installation
* IDLE Introduction
* Pycharm Installation
* PIP Installation
* Jupyter Notebook Installation

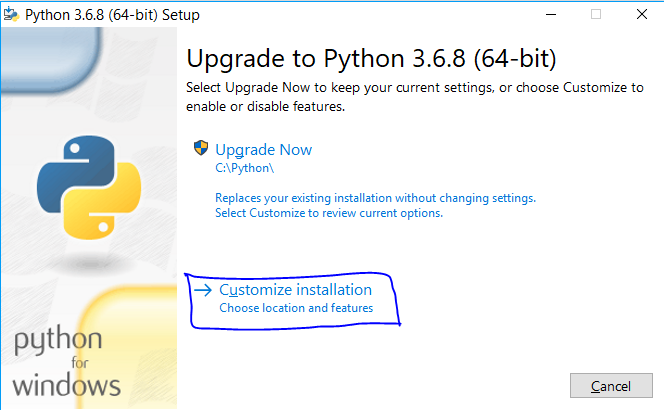
URL:

Go to www.python.org/downloads

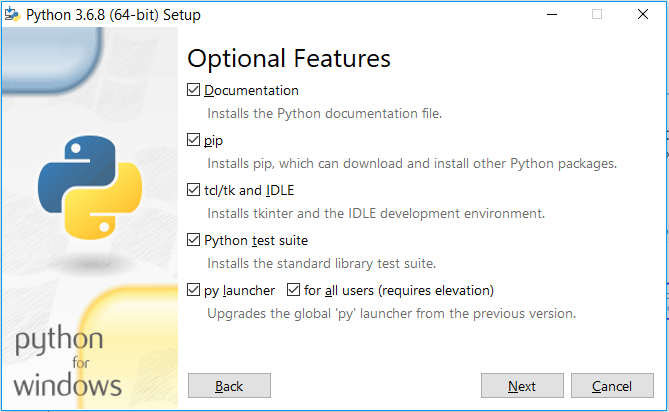


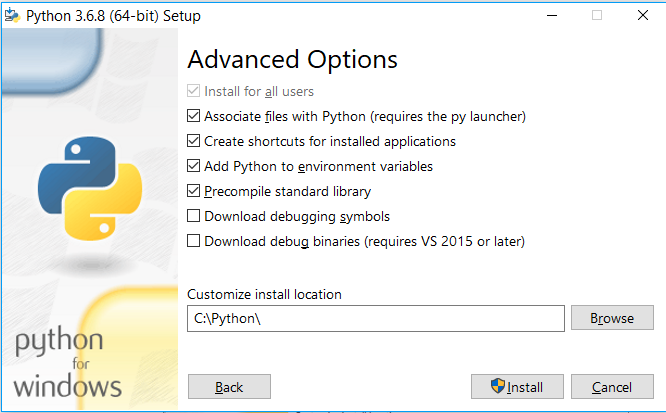
****

****

****

* Click on the .exe file
* I get upgrade as I already have python
* You will get Install now.
* Click on customize installation.

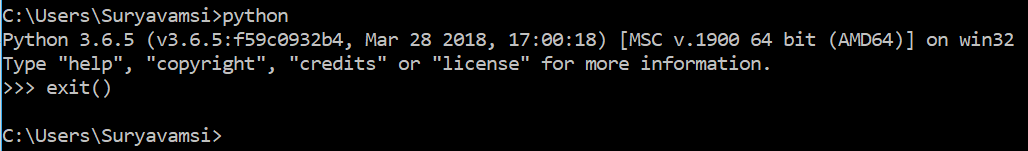
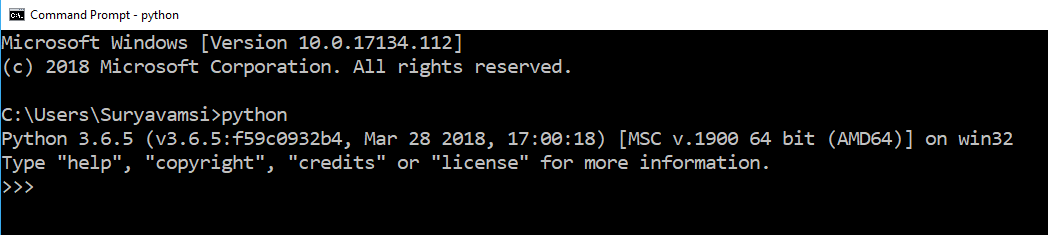




* Change the default path by selecting browse.
* Select a well-known path which you can access.
* Do not select drive just because I selected.
* Make sure all are ticked as in image.

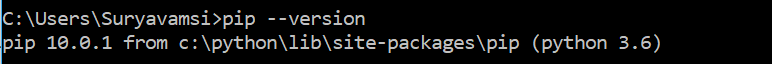
Type in ‘python’ in the cmd.

* You will get the python version and you are in python shell now.
* The three greater than symbols indicate you are in python shell
* To come out of python shell enter ctrl+z and enter or type in ‘exit()’.



Enter ‘pip --version’ in the command line

* You will get the pip installed path and the version.
* Don’t worry if versions do not match. We can upgrade pip later.

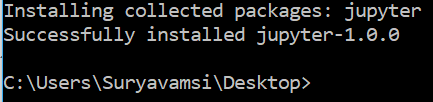


**Installing Jupyter Notebook**

* Upgrading pip to the latest . Type in the command ‘python3 –m pip install -- upgrade pip
* Type the command ‘pip install Jupyter’ in the command prompt.



* Jupyter should successfully install. Donk worry abt versions.



* Type in jupyter and a space and notebook. Jupyter will open.



* Jupyter note book should open in the browser.

**Jupyter Notebook:**

* By default jupyter notebook will be showing you your C drive user data or from the path where we opened the book.

****

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. We can analyze data in pandas with:

* Series
* Data Frames

Series is one dimensional (1-D) array defined in pandas that can be used to store any data type.

Data Frames is two-dimensional (2-D) data structure defined in pandas which consists of rows and columns.

**Matplotlib**

Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

**scikit-learn**

Scikit-learn is a machine learning library for Python. It features several regression, classification and clustering algorithms including SVMs, gradient boosting, k-means, random forests and DBSCAN. Scikit is written in Python (most of it) and some of its core algorithms are written in Cython for even better performance.

Scikit-learn is used to build models and it is not recommended to use it for reading, manipulating and summarizing data as there are better frameworks available for the purpose.

It is open source and released under BSD license.

**Seaborn:**

Seaborn is an open source, BSD-licensed Python library providing high level API for visualizing the data using Python programming language. Data can be visualized by representing it as plots which is easy to understand, explore and grasp. Such data helps in drawing the attention of key elements.

To analyse a set of data using Python, we make use of Matplotlib, a widely implemented 2D plotting library. Likewise, Seaborn is a visualization library in Python. It is built on top of Matplotlib.

**1.1 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 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.

## 1.2 History of Python

Python was developed by Guido van Rossum 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 Rossum still holds a vital role in directing its progress.

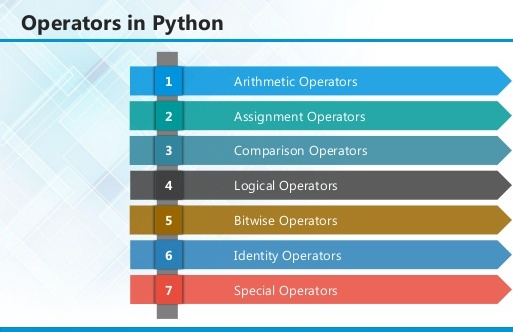
## 1.3 Python Features

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.

Python has a big list of good features:

* 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.



**2.1 ARITHMETIC OPERATORS**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + Addition | Adds values on either side of the operator. | a + b = 30 |
| - Subtraction | Subtracts right hand operand from left hand operand. | a – b = -10 |
| \* Multiplication | Multiplies values on either side of the operator | a \* b = 200 |
| / Division | Divides left hand operand by right hand operand | b / a = 2 |
| % Modulus | Divides left hand operand by right hand operand and returns remainder | b % a = 0 |
| \*\* Exponent | Performs exponential (power) calculation on operators | a\*\*b =10 to the power 20 |
| // | Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity): | 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0 |

**2.2ASSIGNMENT OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assigns values from right side operands to left side operand | c = a + b assigns value of a + b into c |
| += Add AND | It adds right operand to the left operand and assign the result to left operand | c += a is equivalent to c = c + a |
| -= Subtract AND | It subtracts right operand from the left operand and assign the result to left operand | c -= a is equivalent to c = c - a |
| \*= Multiply AND | It multiplies right operand with the left operand and assign the result to left operand | c \*= a is equivalent to c = c \* a |
| /= Divide AND | It divides left operand with the right operand and assign the result to left operand | c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a |

|  |  |  |
| --- | --- | --- |
| %= Modulus AND | It takes modulus using two operands and assign the result to left operand | c %= a is equivalent to c = c % a |
| \*\*= Exponent AND | Performs exponential (power) calculation on operators and assign value to the left operand | c \*\*= a is equivalent to c = c \*\* a |
| //= Floor Division | It performs floor division on operators and assign value to the left operand | c //= a is equivalent to c = c // a |

**2.3 IDENTITY OPERATOR**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Evaluates to true if the variables on either side of the operator point to the same object and false otherwise. | x is y, here **is** results in 1 if id(x) equals id(y). |
| is not | Evaluates to false if the variables on either side of the operator point to the same object and true otherwise. | x is not y, here **is not** results in 1 if id(x) is not equal to id(y |

**2.4 COMPARISON OPERATOR**

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

**2.5 LOGICAL OPERATOR**

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

## 2.6 Membership Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Evaluates to true if it finds a variable in the specified sequence and false otherwise. | x in y, here in results in a 1 if x is a member of sequence y. |
| not in | Evaluates to true if it does not finds a variable in the specified sequence and false otherwise. | x not in y, here not in results in a 1 if x is not a member of sequence y. |

## Python Operators Precedence

|  |  |
| --- | --- |
| **Operator** | **Description** |
| \*\* | Exponentiation (raise to the power) |
| ~ + - | Complement, unary plus and minus (method names for the last two are +@ and -@) |
| \* / % // | Multiply, divide, modulo and floor division |
| + - | Addition and subtraction |
| >> << | Right and left bitwise shift |
| & | Bitwise 'AND' |  |
| ^ | | Bitwise exclusive `OR' and regular `OR' |  |
| <= < > >= | Comparison operators |  |
| <> == != | Equality operators |  |
| = %= /= //= -= += \*= \*\*= | Assignment operators |  |
| is is not | Identity operators |  |
| in not in | Membership operators |  |
| not or and | Logical operators |  |

**3.1 LIST**

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"]

## Basic List Operations

Lists respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

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

## Built-in List Functions & Methods:

Python includes the following list functions −

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [cmp(list1, list2)](https://www.tutorialspoint.com/python/list_cmp.htm)  Compares elements of both lists. |
| 2 | [len(list)](https://www.tutorialspoint.com/python/list_len.htm)  Gives the total length of the list. |
| 3 | [max(list)](https://www.tutorialspoint.com/python/list_max.htm)  Returns item from the list with max value. |
| 4 | [min(list)](https://www.tutorialspoint.com/python/list_min.htm)  Returns item from the list with min value. |
| 5 | [list(seq)](https://www.tutorialspoint.com/python/list_list.htm)  Converts a tuple into list. |

Python includes following list methods

|  |  |
| --- | --- |
| **SN** | **Methods with Description** |
| 1 | [list.append(obj)](https://www.tutorialspoint.com/python/list_append.htm)  Appends object obj to list |
| 2 | [list.count(obj)](https://www.tutorialspoint.com/python/list_count.htm)  Returns count of how many times obj occurs in list |
| 3 | [list. extend(seq)](https://www.tutorialspoint.com/python/list_extend.htm)  Appends the contents of seq to list |
| 4 | [list.index(obj)](https://www.tutorialspoint.com/python/list_index.htm)  Returns the lowest index in list that obj appears |
| 5 | [list.insert(index, obj)](https://www.tutorialspoint.com/python/list_insert.htm)  Inserts object obj into list at offset index |
| 6 | [list.pop(obj=list[-1])](https://www.tutorialspoint.com/python/list_pop.htm)  Removes and returns last object or obj from list |
| 7 | [list.remove(obj)](https://www.tutorialspoint.com/python/list_remove.htm)  Removes object obj from list |
| 8 | [list.reverse()](https://www.tutorialspoint.com/python/list_reverse.htm)  Reverses objects of list in place |
| 9 | [list.sort([func])](https://www.tutorialspoint.com/python/list_sort.htm)  Sorts objects of list, use compare function if given |

**3.2 TUPLES**

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally we can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing −

tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

tup1 = (50,);

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

## Accessing Values in Tuples:

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0]

print "tup2[1:5]: ", tup2[1:5]

When the code is executed, it produces the following result −

tup1[0]: physics

tup2[1:5]: [2, 3, 4, 5]

## Updating Tuples:

Tuples are immutable which means you cannot update or change the values of tuple elements. We are able to take portions of existing tuples to create new tuples as the following example demonstrates −

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

tup3 = tup1 + tup2;

print tup3

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

(12, 34.56, 'abc', 'xyz')

## Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement. For example:

tup = ('physics', 'chemistry', 1997, 2000);

print tup

del tup;

print "After deleting tup : "

print tup

## Basic Tuples Operations:

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

## Built-in Tuple Functions

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [**cmp(tuple1, tuple2)**](https://www.tutorialspoint.com/python/tuple_cmp.htm):Compares elements of both tuples. |
| 2 | [**len(tuple)**](https://www.tutorialspoint.com/python/tuple_len.htm):Gives the total length of the tuple. |
| 3 | [**max(tuple)**](https://www.tutorialspoint.com/python/tuple_max.htm):Returns item from the tuple with max value. |
| 4 | [**min(tuple)**](https://www.tutorialspoint.com/python/tuple_min.htm):Returns item from the tuple with min value. |
| 5 | [**tuple(seq)**](https://www.tutorialspoint.com/python/tuple_tuple.htm):Converts a list into tuple. |

**3.2 DICTIONARY**

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

## Accessing Values in Dictionary:

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']

Result –

dict['Name']: Zara

dict['Age']: 7

## Updating Dictionary

We can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

Result −

dict['Age']: 8

dict['School']: DPS School

## Delete Dictionary Elements

We can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement. Following is a simple example –

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name']; # remove entry with key 'Name'

dict.clear(); # remove all entries in dict

del dict ; # delete entire dictionary

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

## Built-in Dictionary Functions & Methods –

Python includes the following dictionary functions −

|  |  |
| --- | --- |
| **SN** | **Function with Description** |
| 1 | [cmp(dict1, dict2)](https://www.tutorialspoint.com/python/dictionary_cmp.htm)  Compares elements of both dict. |
| 2 | [len(dict)](https://www.tutorialspoint.com/python/dictionary_len.htm)  Gives the total length of the dictionary. This would be equal to the number of items in the dictionary. |
| 3 | [str(dict)](https://www.tutorialspoint.com/python/dictionary_str.htm)  Produces a printable string representation of a dictionary |
| 4 | [type(variable)](https://www.tutorialspoint.com/python/dictionary_type.htm)  Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type. |

Python includes following dictionary methods −

|  |  |
| --- | --- |
| **SN** | **Methods with Description** |
| 1 | [**dict.clear()**](https://www.tutorialspoint.com/python/dictionary_clear.htm):Removes all elements of dictionary *dict* |
| 2 | [**dict. Copy()**](https://www.tutorialspoint.com/python/dictionary_copy.htm):Returns a shallow copy of dictionary *dict* |
| 3 | [**dict.fromkeys()**](https://www.tutorialspoint.com/python/dictionary_fromkeys.htm):Create a new dictionary with keys from seq and values *set* to *value*. |
| 4 | [**dict.get(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_get.htm):For *key* key, returns value or default if key not in dictionary |
| 5 | [**dict.has\_key(key)**](https://www.tutorialspoint.com/python/dictionary_has_key.htm):Returns *true* if key in dictionary *dict*, *false* otherwise |
| 6 | [**dict.items()**](https://www.tutorialspoint.com/python/dictionary_items.htm):Returns a list of *dict*'s (key, value) tuple pairs |
| 7 | [**dict.keys()**](https://www.tutorialspoint.com/python/dictionary_keys.htm):Returns list of dictionary dict's keys |
| 8 | [**dict.setdefault(key, default=None)**](https://www.tutorialspoint.com/python/dictionary_setdefault.htm):Similar to get(), but will set dict[key]=default if *key* is not already in dict |
| 9 | [**dict.update(dict2)**](https://www.tutorialspoint.com/python/dictionary_update.htm):Adds dictionary *dict2*'s key-values pairs to *dict* |
| 10 | [**dict.values()**](https://www.tutorialspoint.com/python/dictionary_values.htm):Returns list of dictionary *dict*'s values |

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

## Defining a Function

Simple rules to define a function in Python.

* Function blocks begin with the keyword def followed by the function name and parentheses ( ( ) ).
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
* The code block within every function starts with a colon (:) and is indented.
* The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

def functionname( parameters ):

"function\_docstring"

function\_suite

return [expression]

## Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print str

return;

# Now you can call printme function

printme("I'm first call to user defined function!")

printme("Again second call to the same function")

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

I'm first call to user defined function!

Again second call to the same function

## Function Arguments

You can call a function by using the following types of formal arguments:

* Required arguments
* Keyword arguments
* Default arguments
* Variable-length arguments

## Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

Global variables Local variables

## Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope. Following is a simple example −

total = 0; # This is global variable.

# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total = arg1 + arg2; # Here total is local variable.

print "Inside the function local total : ", total

return total;

sum( 10, 20 );

print "Outside the function global total : ", total

**Result −**

Inside the function local total : 30

Outside the function global total : 0

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

## Example:

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

def print\_func( par ):

print "Hello : ", par

return

## The *import* Statement

The *import* has the following syntax:

import module1[, module2[,... moduleN]

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script −

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

## Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

def Pots():

print "I'm Pots Phone"

Similar way, we have another two files having different functions with the same name as above −

* *Phone/Isdn.py* file having function Isdn()
* *Phone/G3.py* file having function G3()

Now, create one more file \_\_init\_\_.py in *Phone* directory −

* Phone/\_\_init\_\_.py

To make all of your functions available when you've imported Phone,to put explicit import statements in \_\_init\_\_.py as follows −

from Pots import Pots

from Isdn import Isdn

from G3 import G3

After you add these lines to \_\_init\_\_.py, you have all of these classes available when you import the Phone package.

# Now import your Phone Package.

import Phone

Phone.Pots()

Phone.Isdn()

Phone.G3()

RESULT:

I'm Pots Phone

I'm 3G Phone

I'm ISDN Phone

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.

This chapter covers all the basic I/O functions available in Python.

## Printing to the Screen

The simplest way to produce output is using the *print* statement where you can pass zero or more expressions separated by commas. This function converts the expressions you pass into a string and writes the result to standard output as follows −

print "Python is really a great language,", "isn't it?"

Result:

Python is really a great language, isn't it?

## Reading Keyboard Input

Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard. These functions are −

* raw\_input
* input

## The *raw\_input* Function

The *raw\_input([prompt])* function reads one line from standard input and returns it as a string (removing the trailing newline).

str = raw\_input("Enter your input: ");

print "Received input is : ", str

This prompts you to enter any string and it would display same string on the screen. When I typed "Hello Python!", its output is like this −

Enter your input: Hello Python

Received input is : Hello Python

## The *input* Function

The *input([prompt])* function is equivalent to raw\_input, except that it assumes the input is a valid Python expression and returns the evaluated result to you.

str = input("Enter your input: ");

print "Received input is : ", str

This would produce the following result against the entered input −

Enter your input: [x\*5 for x in range(2,10,2)]

Recieved input is : [10, 20, 30, 40]

## Opening and Closing Files

Until now, you have been reading and writing to the standard input and output. Now, we will see how to use actual data files.

Python provides basic functions and methods necessary to manipulate files by default. You can do most of the file manipulation using a **file** object.

## The *open* Function

Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object, which would be utilized to call other support methods associated with it.

### Syntax

file object = open(file\_name [, access\_mode][, buffering])

Here are parameter details:

* **file\_name:** The file\_name argument is a string value that contains the name of the file that you want to access.
* **access\_mode:** The access\_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).
* **buffering:** If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

Here is a list of the different modes of opening a file −

|  |  |
| --- | --- |
| **Modes** | **Description** |
| r | Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode. |
| rb | Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode. |
| r+ | Opens a file for both reading and writing. The file pointer placed at the beginning of the file. |
| rb+ | Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file. |
| w | Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| wb | Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| w+ | Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| wb+ | Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| a | Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| ab | Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| a+ | Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |
| ab+ | Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |

## The *file* Object Attributes

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object:

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| file.closed | Returns true if file is closed, false otherwise. |
| file.mode | Returns access mode with which file was opened. |
| file.name | Returns name of the file. |
| file.softspace | Returns false if space explicitly required with print, true otherwise. |

### Example

# Open a file

fo = open("foo.txt", "wb")

print "Name of the file: ", fo.name

print "Closed or not : ", fo.closed

print "Opening mode : ", fo.mode

print "Softspace flag : ", fo.softspace

This produces the following result −

Name of the file: foo.txt

Closed or not : False

Opening mode : wb

Softspace flag : 0

## The *close()* Method

The close() method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done.Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close() method to close a file.

### Syntax

fileObject.close();

### Example

# Open a file

fo = open("foo.txt", "wb")

print "Name of the file: ", fo.name

# Close opend file

fo.close()

Result −

Name of the file: foo.txt

## Reading and Writing Files

The *file* object provides a set of access methods to make our lives easier. We would see how to use *read()* and *write()* methods to read and write files.

## The *write()* Method

The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.The write() method does not add a newline character ('\n') to the end of the string **Syntax**

fileObject.write(string);

Here, passed parameter is the content to be written into the opened file. **Example**

# Open a file

fo = open("foo.txt", "wb")

fo.write( "Python is a great language.\nYeah its great!!\n");

# Close opend file

fo.close()

The above method would create *foo.txt* file and would write given content in that file and finally it would close that file. If you would open this file, it would have following content.

Python is a great language.

Yeah its great!!

## The *read()* Method

The *read()* method reads a string from an open file. It is important to note that Python strings can have binary data. apart from text data.

### Syntax

fileObject.read([count]);

Here, passed parameter is the number of bytes to be read from the opened file. This method starts reading from the beginning of the file and if *count* is missing, then it tries to read as much as possible, maybe until the end of file.

### Example

Let's take a file *foo.txt*, which we created above.

# Open a file

fo = open("foo.txt", "r+")

str = fo.read(10);

print "Read String is : ", str

# Close opend file

fo.close()

This produces the following result −

Read String is : Python is

## File Positions

The *tell()* method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.

32

The *seek(offset[, from])* method changes the current file position. The *offset* argument indicates the number of bytes to be moved. The *from* argument specifies the reference position from where the bytes are to be moved.

If *from* is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

### Example

Let us take a file *foo.txt*, which we created above.

# Open a file

fo = open("foo.txt", "r+")

str = fo.read(10);

print "Read String is : ", str

# Check current position

position = fo.tell();

print "Current file position : ", position

# Reposition pointer at the beginning once again

position = fo.seek(0, 0);

str = fo.read(10);

print "Again read String is : ", str

# Close opend file

fo.close()

This produces the following result −

Read String is : Python is

Current file position : 10

Again read String is : Python is

## Renaming and Deleting Files

Python **os** module provides methods that help you perform file-processing operations, such as renaming and deleting files.

To use this module you need to import it first and then you can call any related functions.

## The rename() Method

The *rename()* method takes two arguments, the current filename and the new filename.

### Syntax

os.rename(current\_file\_name, new\_file\_name)

### Example

Following is the example to rename an existing file *test1.txt*:

import os

# Rename a file from test1.txt to test2.txt

os.rename( "test1.txt", "test2.txt" )

## The *remove()* Method

You can use the *remove()* method to delete files by supplying the name of the file to be deleted as the argument.

### Syntax

os.remove(file\_name)

### Example

Following is the example to delete an existing file *test2.txt* −

#!/usr/bin/python

import os

# Delete file test2.txt

os.remove("text2.txt")

## Directories in Python

All files are contained within various directories, and Python has no problem handling these too. The **os** module has several methods that help you create, remove, and change directories.

## The *mkdir()* Method

You can use the *mkdir()* method of the **os** module to create directories in the current directory. You need to supply an argument to this method which contains the name of the directory to be created.

### Syntax

os.mkdir("newdir")

### Example

Following is the example to create a directory *test* in the current directory −

#!/usr/bin/python

import os

# Create a directory "test"

os.mkdir("test")

## The *chdir()* Method

You can use the *chdir()* method to change the current directory. The chdir() method takes an argument, which is the name of the directory that you want to make the current directory.

### Syntax

os.chdir("newdir")

### Example

Following is the example to go into "/home/newdir" directory −

#!/usr/bin/python

import os

# Changing a directory to "/home/newdir"

os.chdir("/home/newdir")

## The *getcwd()* Method

The *getcwd()* method displays the current working directory.

### Syntax

os.getcwd()

### Example

Following is the example to give current directory −

import os

# This would give location of the current directory

os.getcwd()

## The *rmdir()* Method

The *rmdir()* method deletes the directory, which is passed as an argument in the method.

Before removing a directory, all the contents in it should be removed.

### Syntax:

os.rmdir('dirname')

### Example

Following is the example to remove "/tmp/test" directory. It is required to give fully qualified name of the directory, otherwise it would search for that directory in the current directory.

import os

# This would remove "/tmp/test" directory.

os.rmdir( "/tmp/test" )

## File & Directory Related Methods

There are three important sources, which provide a wide range of utility methods to handle and manipulate files & directories on Windows and Unix operating systems. They are as follows −

* [File Object Methods](https://www.tutorialspoint.com/python/file_methods.htm): The *file* object provides functions to manipulate files.
* [OS Object Methods](https://www.tutorialspoint.com/python/os_file_methods.htm): This provides methods to process files as well as directories.

Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them −

* **Exception Handling:** This would be covered in this tutorial. Here is a list standard Exceptions available in Python: [Standard Exceptions](https://www.tutorialspoint.com/python/standard_exceptions.htm).
* **Assertions:** This would be covered in [Assertions in Python](https://www.tutorialspoint.com/python/assertions_in_python.htm)

List of Standard Exceptions −

|  |  |
| --- | --- |
| **EXCEPTION NAME** | **DESCRIPTION** |
| Exception | Base class for all exceptions |
| StopIteration | Raised when the next() method of an iterator does not point to any object. |
| SystemExit | Raised by the sys.exit() function. |
| StandardError | Base class for all built-in exceptions except StopIteration and SystemExit. |
| ArithmeticError | Base class for all errors that occur for numeric calculation. |
| OverflowError | Raised when a calculation exceeds maximum limit for a numeric type. |
|  |  |
| FloatingPointError | Raised when a floating point calculation fails. |
| ZeroDivisionError | Raised when division or modulo by zero takes place for all numeric types. |
| AssertionError | Raised in case of failure of the Assert statement. |
| AttributeError | Raised in case of failure of attribute reference or assignment. |
| EOFError | Raised when there is no input from either the raw\_input() or input() function and the end of file is reached. |
| ImportError | Raised when an import statement fails. |
| KeyboardInterrupt | Raised when the user interrupts program execution, usually by pressing Ctrl+c. |
| LookupError | Base class for all lookup errors. |
| IndexError  KeyError | Raised when an index is not found in a sequence.  Raised when the specified key is not found in the dictionary. |
| NameError | Raised when an identifier is not found in the local or global namespace. |
| UnboundLocalError  EnvironmentError | Raised when trying to access a local variable in a function or method but no value has been assigned to it.  Base class for all exceptions that occur outside the Python environment. |
| IOError  IOError | Raised when an input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist.  Raised for operating system-related errors. |
| SyntaxError  IndentationError | Raised when there is an error in Python syntax.  Raised when indentation is not specified properly. |
| SystemError | Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit. |
| SystemExit | Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit. |
| TypeError | Raised when an operation or function is attempted that is invalid for the specified data type. |
| ValueError | Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified. |
| RuntimeError | Raised when a generated error does not fall into any category. |
| NotImplementedError | Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented. |

## What is Exception?

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

## Handling an exception

If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

* GadFly
* mSQL
* MySQL
* PostgreSQL
* Microsoft SQL Server 2000
* Informix
* Interbase
* Oracle
* Sybase

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:

* Importing the API module.
* Acquiring a connection with the database.
* Issuing SQL statements and stored procedures.
* Closing the connection

**CHAPTER 6**

**PROJECT ARCHITECTURE**

**6.1. INTRODUCTION**

* The **Systems Development Life Cycle (SDLC)**, or *Software Development Life Cycle* in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems.
* In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**6.2. SOFTWARE MODEL OR ARCHITECTURE ANALYSIS:**

* Structured project management techniques (such as an SDLC) enhance management’s control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer.

**WHAT IS SDLC?**

* A software cycle deals with various parts and phases from planning to testing and deploying software. All these activities are carried out in different ways, as per the needs. Each way is known as a Software Development Lifecycle Model (SDLC). A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models.
* **SDLC models** \* **The Linear model (Waterfall)** - Separate and distinct phases of specification and development. - All activities in linear fashion. - Next phase starts only when first one is complete. \* **Evolutionary development** - Specification and development are interleaved (Spiral, incremental, prototype based, Rapid Application development). - Incremental Model (Waterfall in iteration), **-** RAD(Rapid Application Development) **-** Focus is on developing quality product in less time, - **Spiral Model** - We start from smaller module and keeps on building it like a spiral. It is also called Component based development. \* **Formal systems development** - A mathematical system model is formally transformed to an implementation. \* **Agile Methods.** - Inducing flexibility into development. \* **Reuse-based development** - The system is assembled from existing components.

**The General Model :**

* Software life cycle models describe phases of the software cycle and the order in which those phases are executed. There are tons of models, and many companies adopt their own, but all have very similar patterns. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

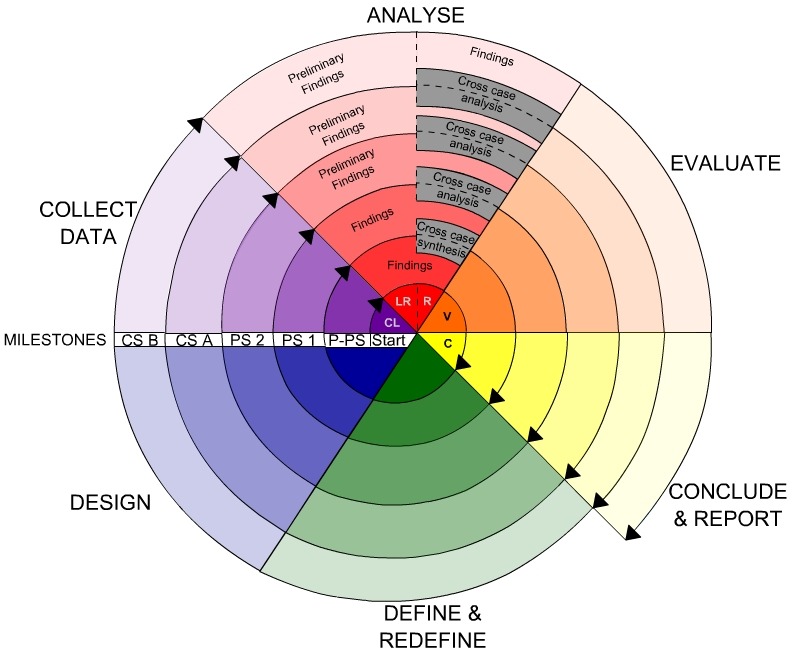
**6.3. SDLC METHODOLOGY:**

**Spiral Model**

* The spiral model is similar to the incremental model, with more emphases placed on risk analysis.  The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.  A\ software project repeatedly passes through these phases in iterations (called Spirals in this model).  The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed.  Each subsequent spirals builds on the baseline spiral. Requirements are gathered during the planning phase.  In the risk analysis phase, a process is undertaken to identify risk and alternate solutions.  A prototype is produced at the end of the  
  risk analysis phase. Software is produced in the engineering phase, along with testing at  
  the end of the phase.  The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.
* This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.
* **SPIRAL MODEL** was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.
* As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.
* The steps for Spiral Model can be generalized as follows:
* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design.A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.



**Spiral Model**

**Advantages**

* High amount of risk analysis
* Good for large and mission-critical projects.
* Software is produced early in the software life cycle.

**CHAPTER 7**

**SYSTEM DESIGN**

**7.1. INTROUDCTION**

* **System design** is transition from a user-oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walk through.
* The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

**SOFTWARE DESIGN**

In designing the software following principles are followed:

1. **Modularity and partitioning:**

Software is designed such that, each system should consist of hierarchy

of modules and serve to partition into separate function.

1. **Coupling:**

Modules should have little dependence on other modules of a system.

1. **Cohesion:**

Modules should carry out in a single processing function.

1. **Shared use:**

Avoid duplication by allowing a single module is called by other that needs

the function it provides.

#### Input design

* The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:
  + - * What data should be given as input?
      * How the data should be arranged or coded?
      * The dialog to guide the operating personnel in providing input.
      * Methods for preparing input validations and steps to follow when error occur.

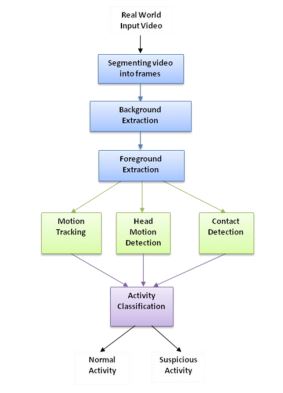
**Output design**

* A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.
* Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
  + - * Select methods for presenting information

Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives.

* + - * Convey information about past activities, current status or projections of the Future.
      * Signal important events, opportunities, problems, or warnings.
      * Trigger an action.

**Architecture Diagram:**



**Uml diagrams:**

**UML**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously making efforts to create a truly industry standard.

* UML stands for **Unified Modeling Language**.
* UML is different from the other common programming languages such as C++, Java, COBOL, etc.
* UML is a pictorial language used to make software blueprints.
* UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
* Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

Goals of UML

*A picture is worth a thousand words*, this idiom absolutely fits describing UML. Object-oriented concepts were introduced much earlier than UML. At that point of time, there were no standard methodologies to organize and consolidate the object-oriented development. It was then that UML came into picture.

There are a number of goals for developing UML but the most important is to define some general purpose modeling language, which all modelers can use and it also needs to be made simple to understand and use.

UML diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system. Thus it must be clear that UML is not a development method rather it accompanies with processes to make it a successful system.

In conclusion, the goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today’s complex environment.

Object-Oriented Concepts

UML can be described as the successor of object-oriented (OO) analysis and design.

An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form a hierarchy to model the real-world system. The hierarchy is represented as inheritance and the classes can also be associated in different ways as per the requirement.

Objects are the real-world entities that exist around us and the basic concepts such as abstraction, encapsulation, inheritance, and polymorphism all can be represented using UML.

UML is powerful enough to represent all the concepts that exist in object-oriented analysis and design. UML diagrams are representation of object-oriented concepts only. Thus, before learning UML, it becomes important to understand OO concept in detail.

Following are some fundamental concepts of the object-oriented world −

* **Objects** − Objects represent an entity and the basic building block.
* **Class** − Class is the blue print of an object.
* **Abstraction** − Abstraction represents the behavior of an real world entity.
* **Encapsulation** − Encapsulation is the mechanism of binding the data together and hiding them from the outside world.
* **Inheritance** − Inheritance is the mechanism of making new classes from existing ones.
* **Polymorphism** − It defines the mechanism to exists in different forms.

OO Analysis and Design

OO can be defined as an investigation and to be more specific, it is the investigation of objects. Design means collaboration of identified objects.

Thus, it is important to understand the OO analysis and design concepts. The most important purpose of OO analysis is to identify objects of a system to be designed. This analysis is also done for an existing system. Now an efficient analysis is only possible when we are able to start thinking in a way where objects can be identified. After identifying the objects, their relationships are identified and finally the design is produced.

The purpose of OO analysis and design can described as −

* Identifying the objects of a system.
* Identifying their relationships.
* Making a design, which can be converted to executables using OO languages.

There are three basic steps where the OO concepts are applied and implemented. The steps can be defined as

OO Analysis→ OO Design→ OO implementation using OO languages

The above three points can be described in detail as −

* During OO analysis, the most important purpose is to identify objects and describe them in a proper way. If these objects are identified efficiently, then the next job of design is easy. The objects should be identified with responsibilities. Responsibilities are the functions performed by the object. Each and every object has some type of responsibilities to be performed. When these responsibilities are collaborated, the purpose of the system is fulfilled.
* The second phase is OO design. During this phase, emphasis is placed on the requirements and their fulfilment. In this stage, the objects are collaborated according to their intended association. After the association is complete, the design is also complete.
* The third phase is OO implementation. In this phase, the design is implemented using OO languages such as Java, C++, etc.

**7.2. UML DIAGRAMS**

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

## UML Diagrams

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete.

UML includes the following nine diagrams, the details of which are described in the subsequent chapters.

* Class diagram
* Use case diagram
* Sequence diagram
* Activity diagram

**Use case diagram:**

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as **actors**.

## 1. USE CASE DIAGRAM

* These are a set of use cases, actors, and their relationships.
* A use case represents a particular functionality of a system.
* These controllers are known as **actors**.
* The two main components of a use case diagram are use cases and actors.



**Use case diagram for developer**

**2. SEQUENCE DIAGRAM**

* A sequence diagram is an interaction diagram.
* From the name, it is clear that the diagram deals with some sequences, which are the sequence of message flowing from one object to another.
* They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.
* A sequence diagram has objects , self message, object lifeline , message return etc …,
* In this each object has a column and the messages exchanged between them are represented by arrows.



**Sequence Diagram**

3. **Collaboration Diagram**

* Collaborationdiagram describes the flow control of a system.
* The flow can be sequential, concurrent, or branched.
* It consists of activities and links.
* Activities are nothing but the functions of a system.
* An activity is essentially a flow chart.
* This is prepared to have an idea of how the system will work when executed.
* The purpose of an activity diagram can be described as:

1. Draw the activity flow of a system.

2. Describe the sequence from one activity

to another.



**Collaboration Diagram**

**Class diagram:**

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as **actors**.



**DATASET**

**8.2. INTRODUCTION**

# A dataset is a collection of data. Most commonly a dataset corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the dataset in question. The data set lists values for each of the variables such as the height or weight of an object for each member in the dataset. A data set is organized into some type of data structure. In a database, for example, a data set might contain a collection of business data (names, salaries, contact information, sales figures, and so forth). The database itself can be considered a data set, as can bodies of data within it related to a particular type of information, such as sales data for a particular corporate department.

# The term data set originated with IBM, where its meaning was similar to that of file. In an IBM mainframe operating system, a data set s a named collection of data that contains individual data units organized (formatted) in a specific, IBM-prescribed way and accessed by a specific access method based on the data set organization. Types of data set organizations include sequential, relative sequential, indexed sequential, and partitioned. Access methods include the Virtual Sequential Access Method (VSAM) and the Indexed Sequential Access Method (ISAM).

**8.2.1 DATASET DESCRIPTION**

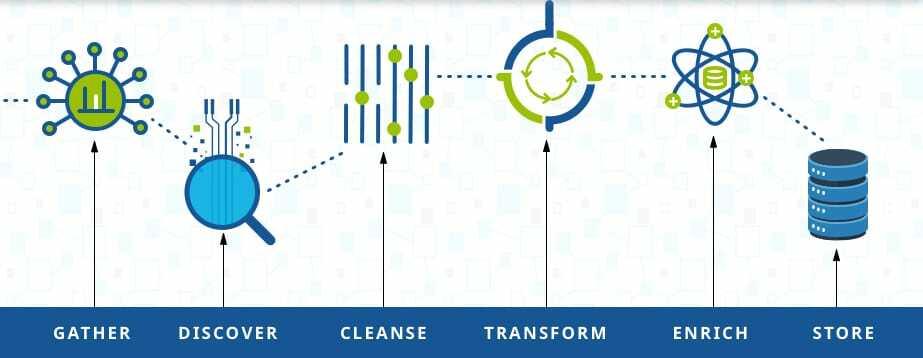
# The 2019 novel coronavirus (COVID-19) presents several unique features. While the diagnosis is confirmed using polymerase chain reaction (PCR), infected patients with pneumonia may present on chest X-ray and computed tomography (CT) images with a pattern that is only moderately characteristic for the human eye Ng, 2020. COVID-19’s rate of transmission depends on our capacity to reliably identify infected patients with a low rate of false negatives. In addition, a low rate of false positives is required to avoid further increasing the burden on the healthcare system by unnecessarily exposing patients to quarantine if that is not required. Along with proper infection control, it is evident that timely detection of the disease would enable the implementation of all the supportive care required by patients affected by COVID-19.

# In late January, a Chinese team published a paper detailing the clinical and paraclinical features of COVID-19. They reported that patients present abnormalities in chest CT images with most having bilateral involvement Huang 2020. Bilateral multiple lobular and subsegmental areas of consolidation constitute the typical findings in chest CT images of intensive care unit (ICU) patients on admission Huang 2020. In comparison, non-ICU patients show bilateral ground-glass opacity and subsegmental areas of consolidation in their chest CT images Huang 2020. In these patients, later chest CT images display bilateral ground-glass opacity with resolved consolidation Huang 2020.

# COVID is possibly better diagnosed using radiological imaging Fang, 2020 and Ai 2020.

**8.2.2 DATA PREPROCESSING**

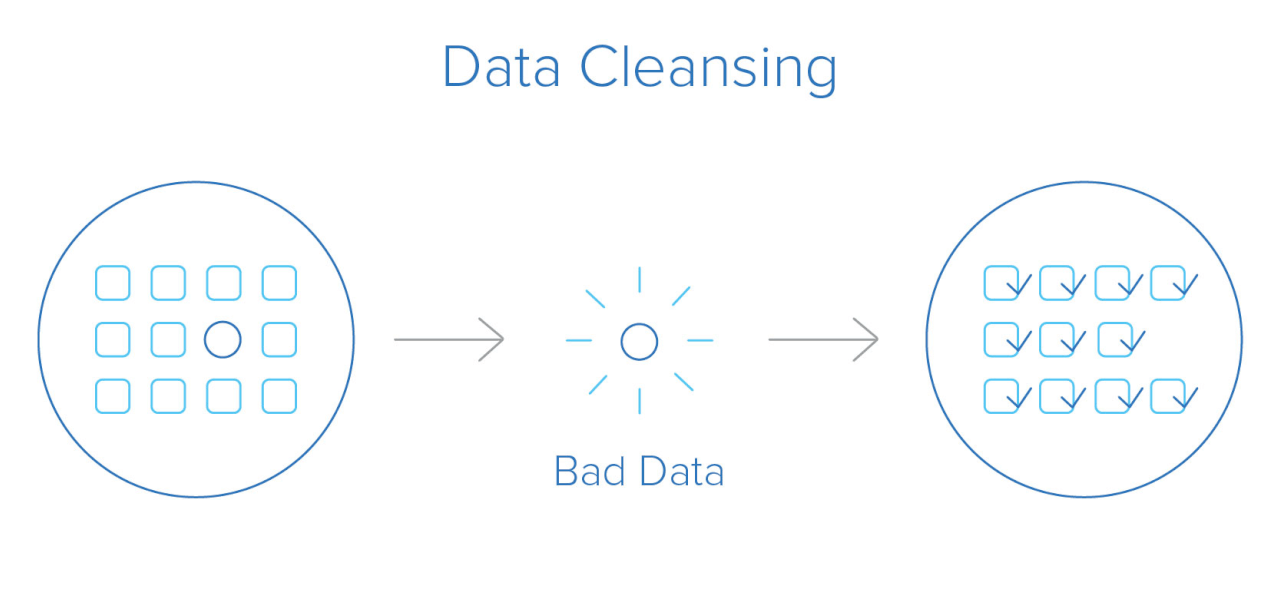
Data pre-processing is a data mining technique that is used to transform the raw data into a useful and efficient format. It is the step before applying Machine Learning Algorithms. It transforms the original data into a suitable shape to be used by a particular algorithm. Data pre-processing includes different tasks as data cleaning, feature selection, and data transformation.



**Fig 8.2.2** Data Pre-processing

**8.2.3 Data Cleaning**

# Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. There are several methods for cleaning data depending on how it is stored along with the answers being sought. Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set’s accuracy without necessarily deleting information. For one, data cleaning includes more actions than removing data, such as fixing spelling and syntax errors, standardizing data sets, and correcting mistakes such as empty fields, missing codes, and identifying duplicate data points. Data cleaning is considered a foundational element of the data science basics, as it plays an important role in the analytical process and uncovering reliable answers.



**Fig. 8.2.3** Data Cleaning

# Most importantly, the goal of data cleaning is to create data sets that are standardized and uniform to allow business intelligence and data analytics tools to easily access and find the right data for each query.

# Regardless of the type of analysis or data visualizations you need, data cleaning is a vital step to ensure that the answers you generate are accurate. When collecting data from several streams and with manual input from users, information can carry mistakes, be incorrectly inputted, or have gaps.

# Data cleaning helps ensure that information always matches the correct fields while making it easier for business intelligence tools to interact with data sets to find information more efficiently. One of the most common data cleaning examples is its application in data warehouses.

**8.2.4Data Visualization**

# Data visualization is a graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions. Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from the circle. Our culture is visual, including everything from art and advertisements to TV and movies.

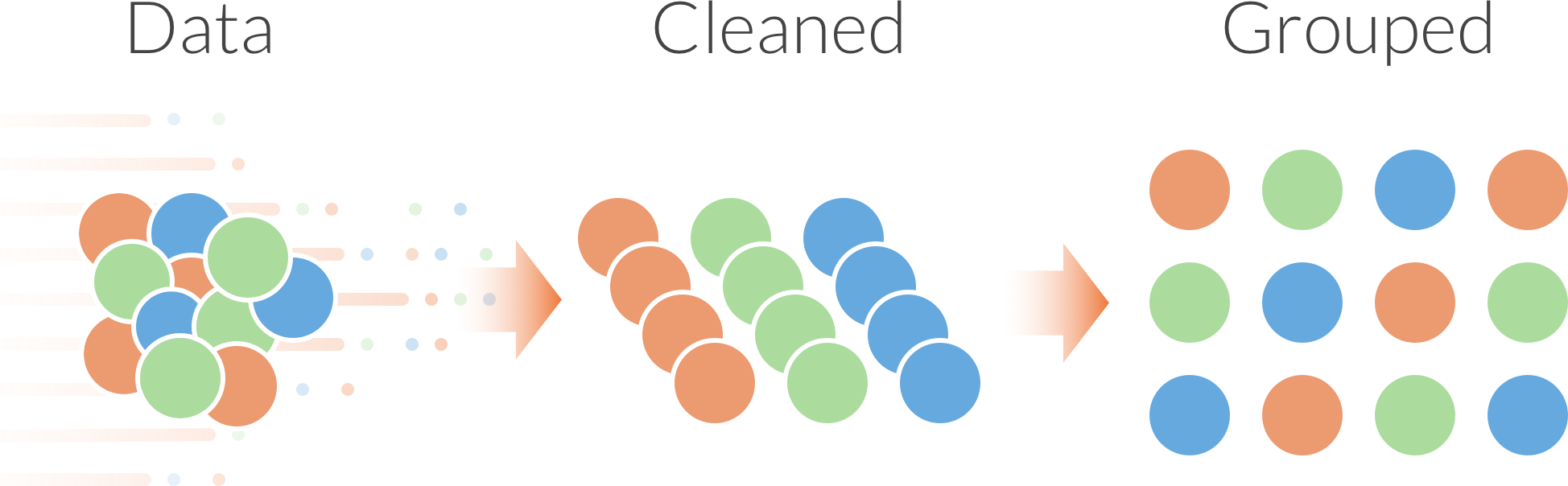
# Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It’s storytelling with a purpose. If you’ve ever stared at a massive spreadsheet of data and couldn’t see a trend, you know how much more effective a visualization can be. Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from the circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It’s storytelling with a purpose. If you’ve ever stared at a massive spreadsheet of data and couldn’t see a trend, you know how much more effective a visualization can be.



**Fig 8.2.4** Data Visualization

**8.2.5 Data Transformation**

# Data transformation is the process of converting data from one format to another, typically from the format of a source system into the required format of a destination system. Data transformation is a component of most data integration and data management tasks, such as data wrangling and data warehousing. One step in the ELT/ETL process, data transformation may be described as either “simple” or “complex,” depending on the kinds of changes that must occur to the data before it is delivered to its target destination. The data transformation process can be automated, handled manually, or completed using a combination of the two.



**Fig. 8.2.5** Data Transformation

# Today, the reality of big data means that data transformation is more important for businesses than ever before. An ever-increasing number of programs, applications, and devices continually produce massive volumes of data. And with so much disparate data streaming in from a variety of sources, data compatibility is always at risk. That’s where the data transformation process comes in: it allows companies and organizations to convert data from any source into a format that can be integrated, stored, analyzed, and ultimately mined for actionable business intelligence.

**8.3. SAMPLE CODE:**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from imutils import paths

import matplotlib.pyplot as plt

import datetime

from tkinter.filedialog import askopenfilename

import cv2

import shutil

import os

from imageai.Prediction.Custom import CustomImagePrediction

import os

main = tkinter.Tk()

main.title("Abnormal Events Detection and tracking in Surveillance System")

main.geometry("1200x1200")

global filename

execution\_path = os.getcwd()

prediction = CustomImagePrediction()

prediction.setModelTypeAsResNet()

prediction.setModelPath("model.h5")

prediction.setJsonPath("model\_class.json")

prediction.loadModel(num\_objects=2)

def upload():

global filename

filename = askopenfilename(initialdir = "videos")

pathlabel.config(text=filename)

def generateFrame():

global filename

text.delete('1.0', END)

if not os.path.exists('frames'):

os.mkdir('frames')

else:

shutil.rmtree('frames')

os.mkdir('frames')

vidObj = cv2.VideoCapture(filename)

count = 0

success = 1

while success:

success, image = vidObj.read()

if count < 500:

cv2.imwrite("frames/frame%d.jpg" % count, image)

text.insert(END,"frames/frame."+str(count)+" saved\n")

print("frames/frame."+str(count)+" saved")

#pathlabel.config(text="frames/frame."+str(count)+" saved")

else:

break

count += 1

pathlabel.config(text="Frame generation process completed. All frames saved inside frame folder")

def detectActivity():

imagePaths = sorted(list(paths.list\_images("frames")))

count = 0

option = 0;

text1.delete('1.0', END)

for imagePath in imagePaths:

predictions, probabilities = prediction.predictImage(imagePath, result\_count=1)

for eachPrediction, eachProbability in zip(predictions, probabilities):

if float(eachProbability) > 80:

count = count + 1;

if float(eachProbability) < 80:

count = 0

if count > 10:

option = 1

print(imagePath+" is predicted as "+eachPrediction+" with probability : " +str(eachProbability))

text1.insert(END,imagePath+" is predicted as "+eachPrediction+" with probability : " +str(eachProbability)+"\n\n")

count = 0;

print(imagePath+" processed")

if option == 0:

text1.insert(END,"No suspicious activity found in given footage")

font = ('times', 20, 'bold')

title = Label(main, text='Abnormal Events Detection and tracking in Surveillance System')

title.config(bg='brown', fg='white')

title.config(font=font)

title.config(height=3, width=80)

title.place(x=5,y=5)

font1 = ('times', 14, 'bold')

upload = Button(main, text="Upload CCTV Footage", command=upload)

upload.place(x=50,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='brown', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=300,y=100)

depthbutton = Button(main, text="Generate Frames", command=generateFrame)

depthbutton.place(x=50,y=150)

depthbutton.config(font=font1)

userinterest = Button(main, text="Detect Suspicious Activity Frame", command=detectActivity)

userinterest.place(x=280,y=150)

userinterest.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=25,width=50)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=200)

text.config(font=font1)

text1=Text(main,height=25,width=50)

scroll=Scrollbar(text1)

text1.configure(yscrollcommand=scroll.set)

text1.place(x=550,y=200)

text1.config(font=font1)

main.config(bg='brown')

main.mainloop()

**CHAPTER-9**

**SYSTEM TESTING**

**9.1. INTRODUCTION**

* Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing represents an interesting anomaly for the software. During earlier definition and development phases, it was attempted to build software from an abstract concept to a tangible implementation. No system is error free because it is so till the next error drops up during any phase of the development or usage of the product.
* The phase involves the testing of development system using various data. Preparation of the test data plays a vital role in system testing. After preparing the test data, the system under study was tested using those data. while testing the system, by using the test data, errors were found and corrected by using the following testing steps and corrections were also noted for future use.

**9.2. SOFTWARE TESTING**

* As the coding is completed according to the requirement we have to test the quality of the software. Software testing is a critical element of the software quality assurance and represents the ultimate review of specification, design and coding. In addition, data collected as testing is conducted to provide a good indication of software reliability and some indications of software quality as a whole. To assure the software quality we conduct both white box testing and black box testing.

**1. WHITE BOX TESTING**

* White box testing is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.
* Usually Test Engineers are involved in the black box testing.

**2. BLACK BOX TESTING**

* Black box testing is the testing process in which tester can perform testing on an application with having internal structural knowledge.
* Usually The Developers are involved in white box testing.

**9.3. SYSTEM TESTING**

* It is designed to uncover weakness that was not detected in the earlier tests. The total system is tested for recovery and fallback after various major failures to ensure that no data are lost. An acceptance test is done to validity and reliability of the system. The philosophy behind the testing is to find error in project. There are many test cases designed with this .the flow of testing is as follows.

**1. Code testing**

* Specification testing is done to check if the program does with it should do and how it should behave under various condition or combinations and submitted for processing in the system and it is checked if any overlaps occur during the processing. This strategy examines the logic of the program. Here only syntax of the code is tested. In code testing syntax errors are corrected, to ensure that the code is perfect.

**2. Unit testing**

* The first level of testing is called unit testing. Here different modules are tested against the specification produced running the design of the modules. Unit testing is done to test the working of individual modules with test oracles. Unit testing comprises a set of tests performed by an individual programmer prior to integration of the units into a large system. A program unit is usually small enough that the programmer who developed it can test it in a great detail. Unit testing focuses first on the modules to locate errors. These errors are verified and corrected so that the unit perfectly fitsto the project.

**System Testing**

* The next level of testing is system testing and acceptance testing. This testing is done to check if the system has met its requirements and to find the external behavior of the system. System testing involves two kinds of activities.

1. **Integration testing**
2. **Acceptance testing**

* The next level of testing is called the integration testing. In this many tested modules are combined into subsystems, which were then tested. Test case data is prepared to check the control flow of all the modules and to exhaust all possible inputs to the program situations like treating the modules when there is no data entered in the test box is also tested.

1. **Integration testing**

* This testing strategy dictates the order in which modules must be available, and exerts strong influence on the order in which the modules must be written, debugged and unit tested. In integration testing, all modules on which unit testing is performed are integrated together and tested.

1. **Acceptance testing**

* This testing is performed finally by user to demonstrate that the implemented system satisfies its requirements. The user gives various inputs to get required outputs.

1. **Specification Testing**

* This is done to check if the program does what it should do and how it should behave under various conditions or combination and submitted for processing in the system and it is checked if any overlaps occur during the processing.

**OUTPUT ANALYSIS**

**SCREENS**:

In this project we need to detect person behaviour as suspicious or not, now a day’s every where CCTV cameras are installed which capture videos and store at centralized server and manually scanning those videos to detect suspicious activity from human required lots of human efforts and time. To overcome from such issue author is asking to automate such process using Machine Learning Algorithms.

To automate that process first we need to build training model using huge amount of images (all possible images which describe features of suspicious activities) and ‘Convolution Neural Network’ using TENSOR FLOW Python module.

Then we can upload any video and then application will extract frames from uploaded video and then that frame will be applied on train model to predict its class such as ‘suspicious or normal’.

To implement above concept we need to install python 3.5 version in 64 bit laptop. I will send this software with code. While software installation u need to select checkbox saying add path to system variable. This option will show on first or second screen of installation. Once you install software execute below commands. Your system must connect to internet.

pip install tensorflow

pip install numpy

pip install scipy

pip install opencv-python

pip install pillow

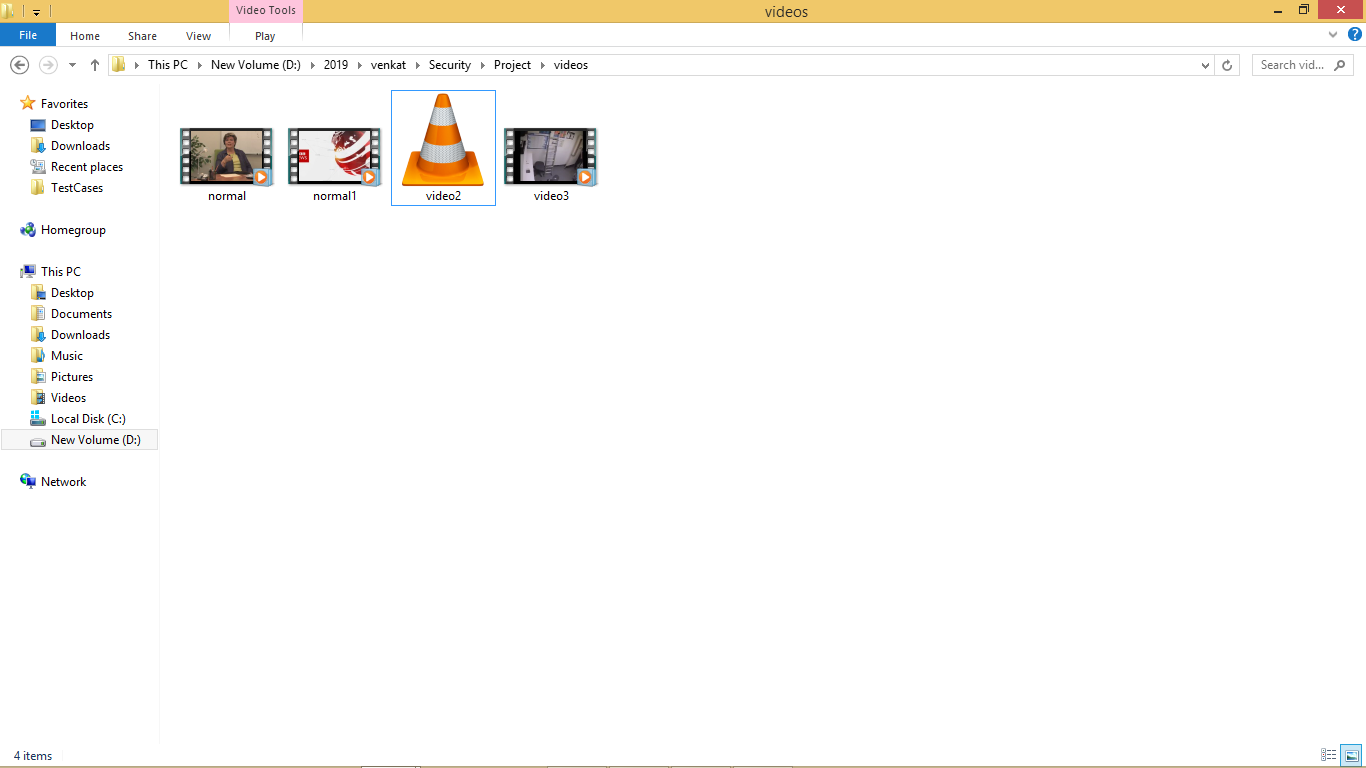
pip install matplotlib

pip install h5py

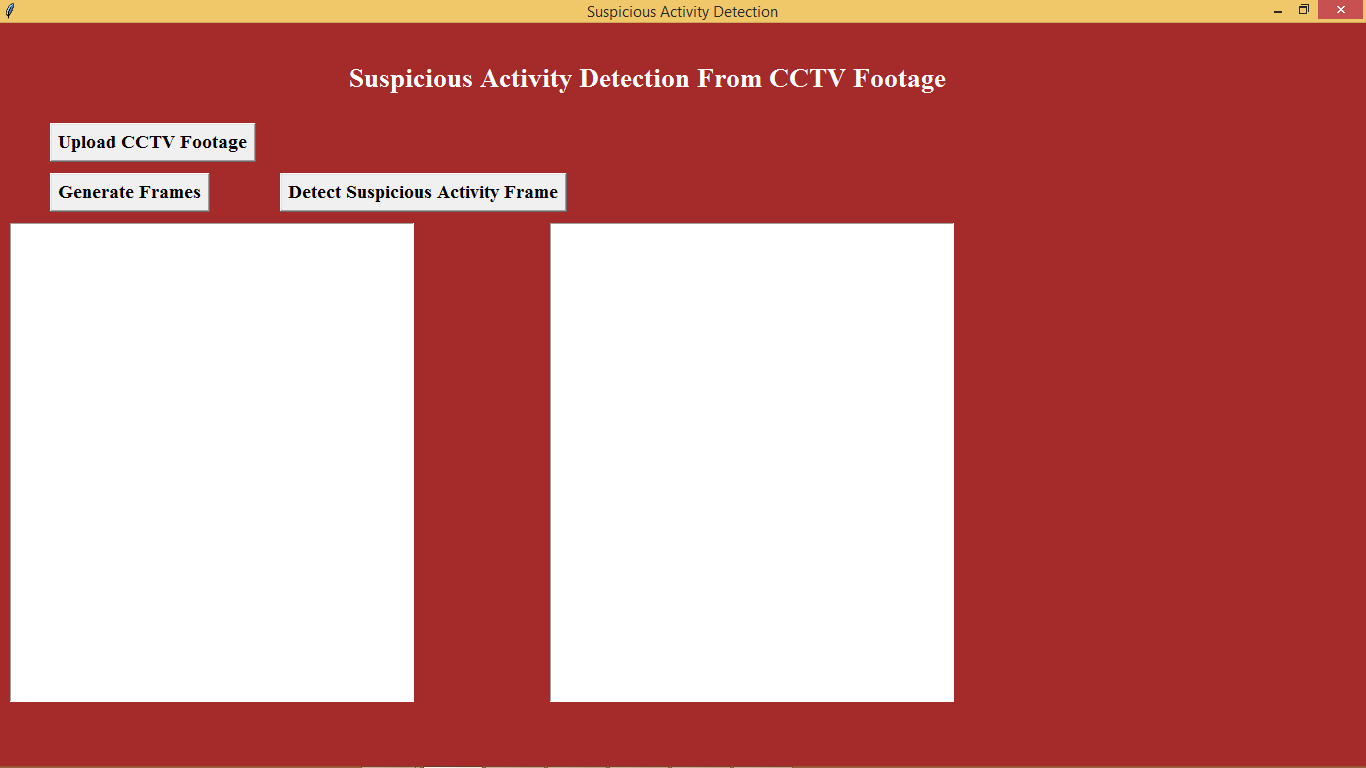
pip install keras

pip install <https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.2/imageai-2.0.2-py3-none-any.whl>

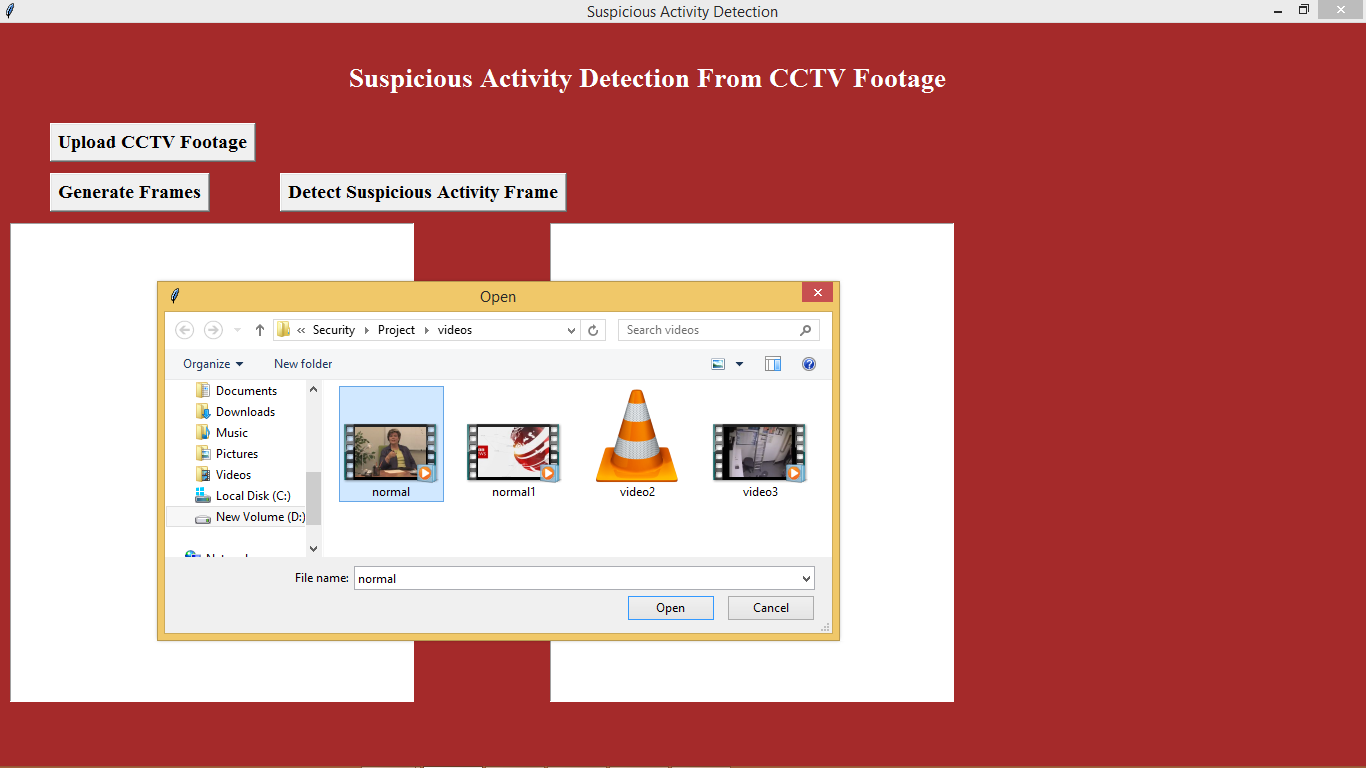
After executing above commands all software will be installed. To monitor i am using below videos. For training i used human images who cover their faces to perform suspicious activity and if any video contains person covering their faces then application will detect it as a suspicious activity.



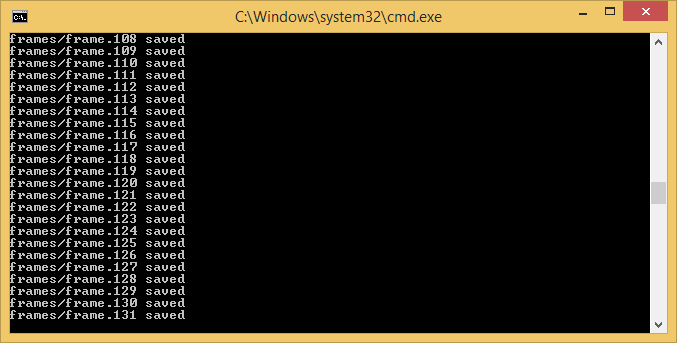
Double click on ‘run.bat’ file from project folder to start project execution. We will get below screen



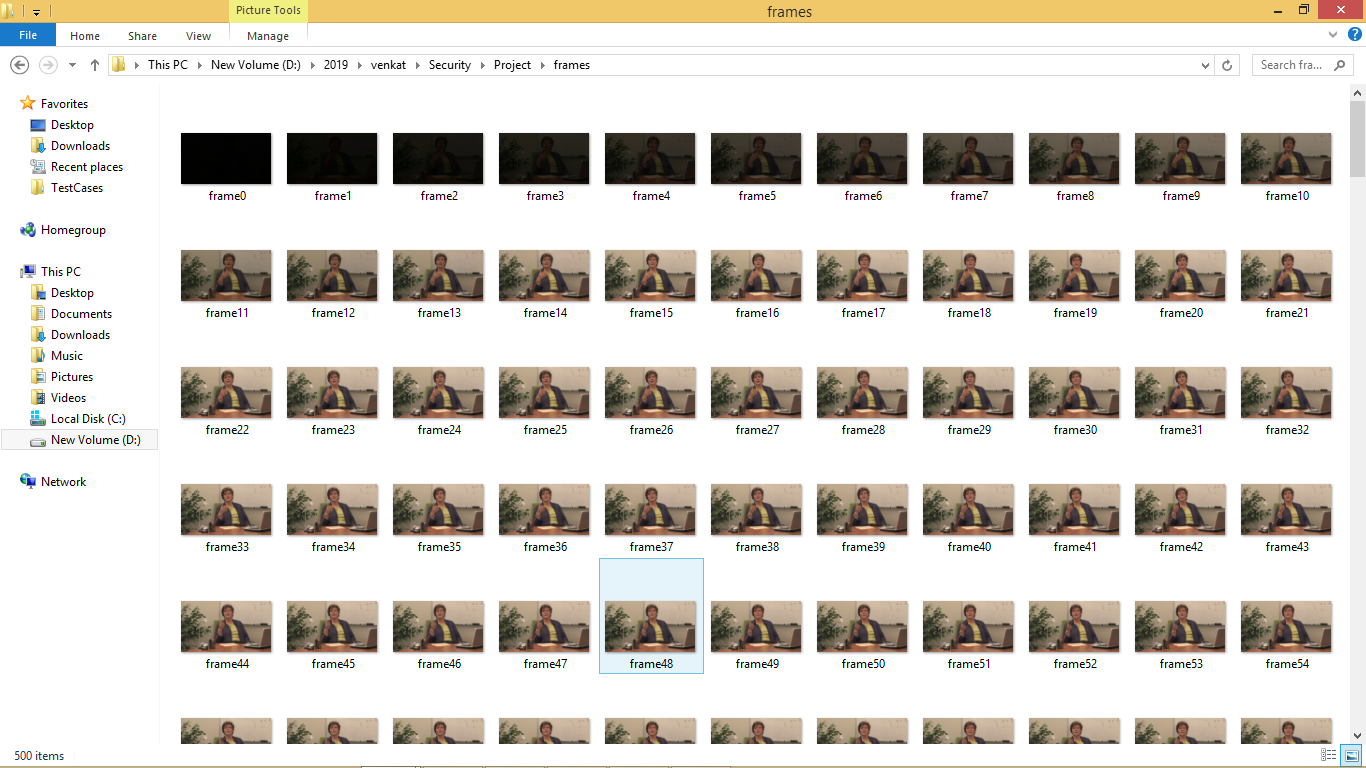
Click on ‘Upload CCTV Footage’ button to upload video



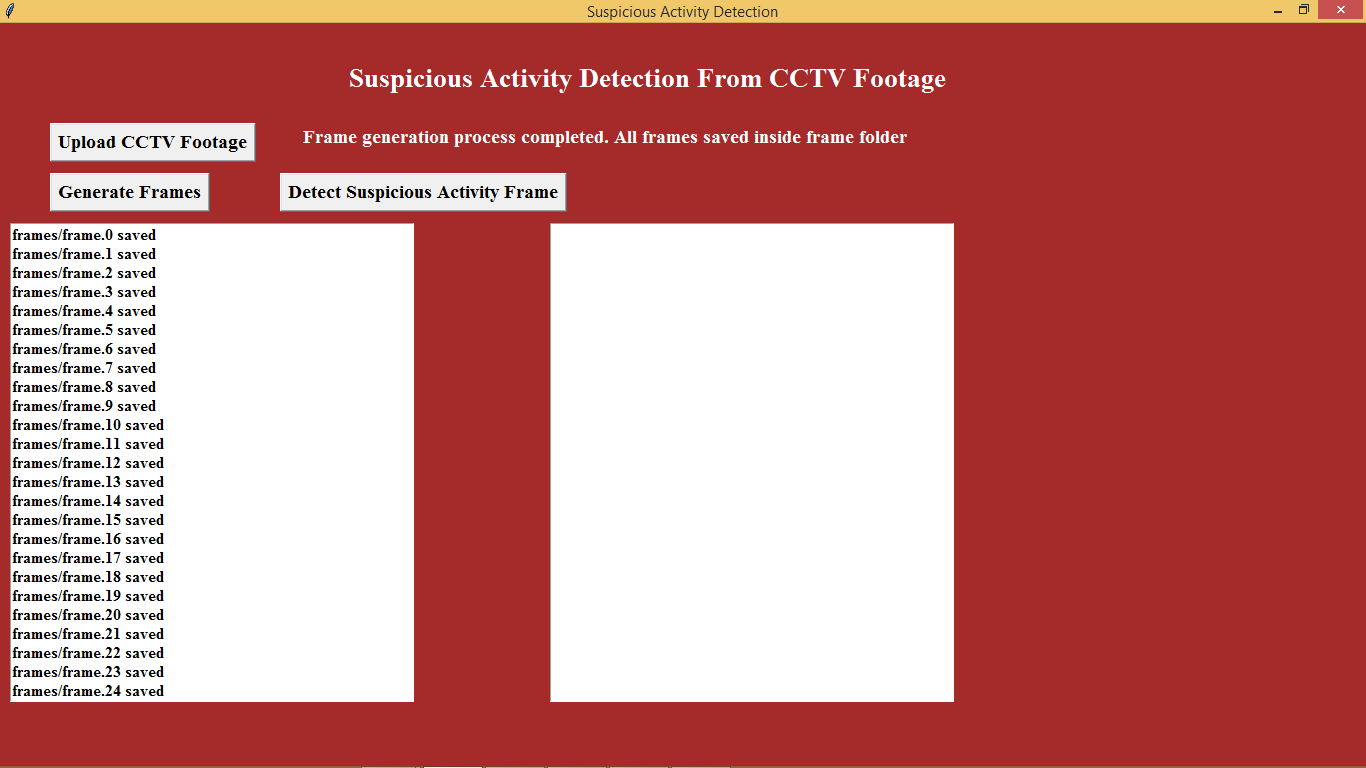
In above screen i am uploading one normal video. After uploading video click on ‘Generate Frames’ button to generate frame



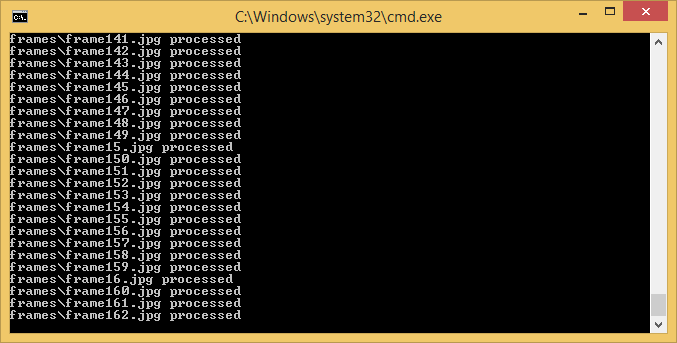
In above black screen we can see extracted frames are saving inside ‘frames’ folder frame no. Now we see frames folder below which has images from video



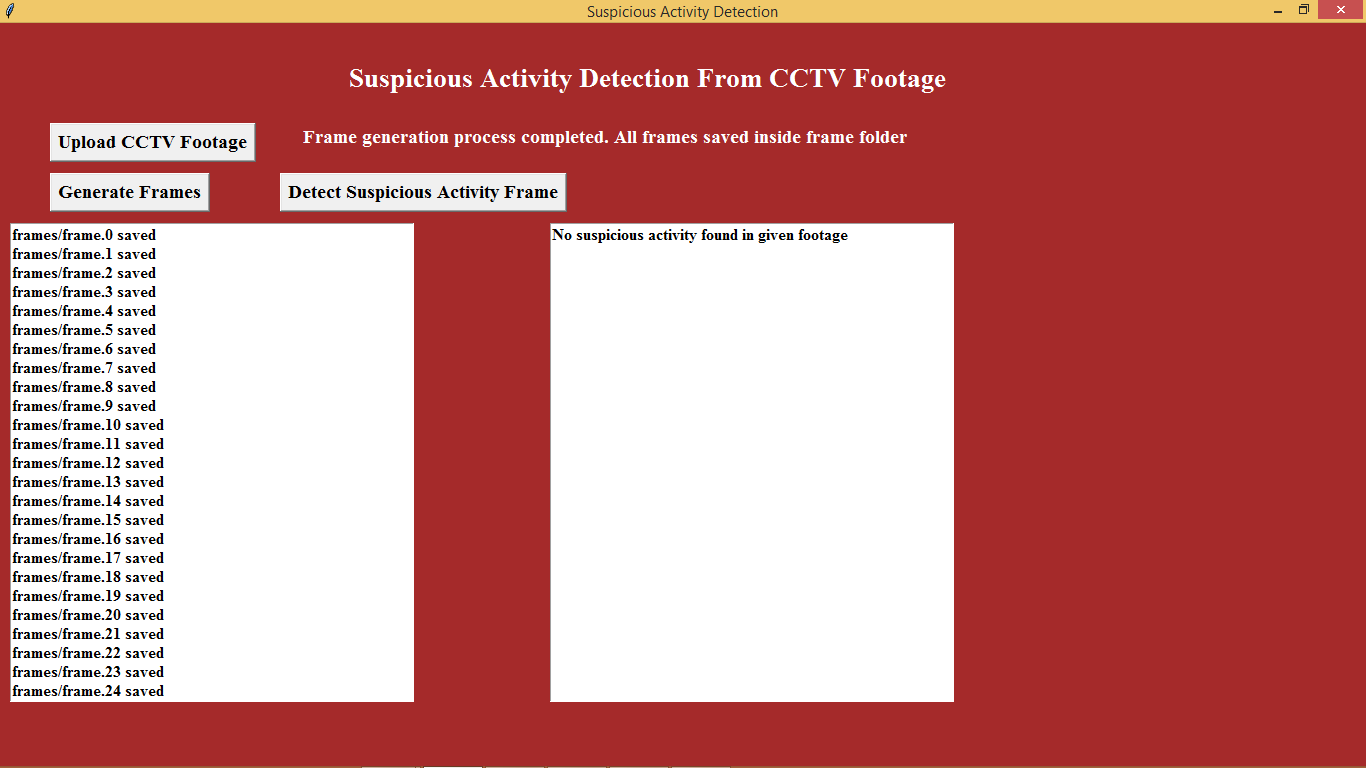
In above folder screen we can see all images from video extracted. After frame extraction will get below screen



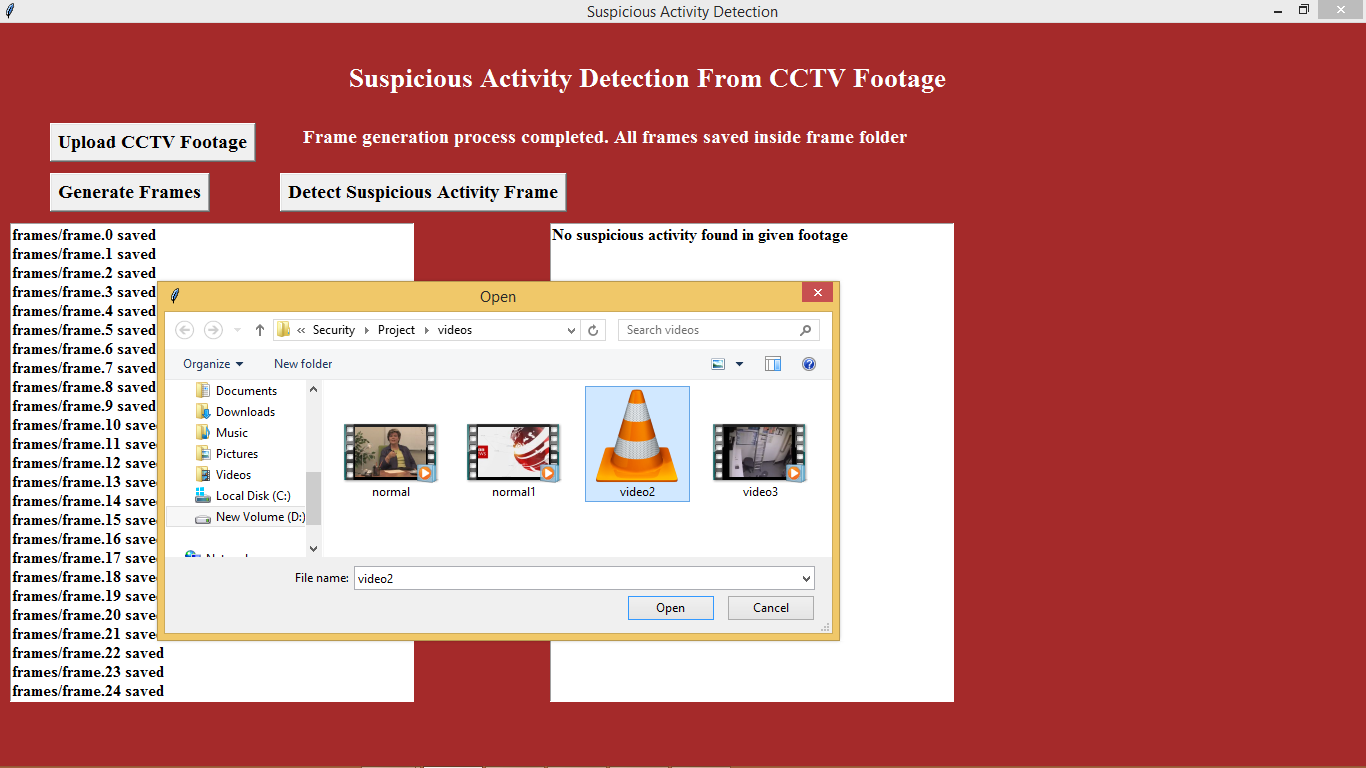
Now click on ‘Detect Suspicious Activity Frame’ button to start monitoring frames for suspicious activity



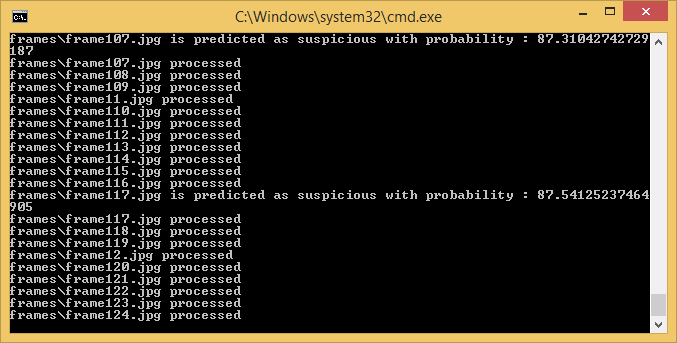
In above black console window we can see processing of each frame to detect suspicious activity.



In above screen we can see frames scanned and no suspicious activity found. Now we will upload another video and check status



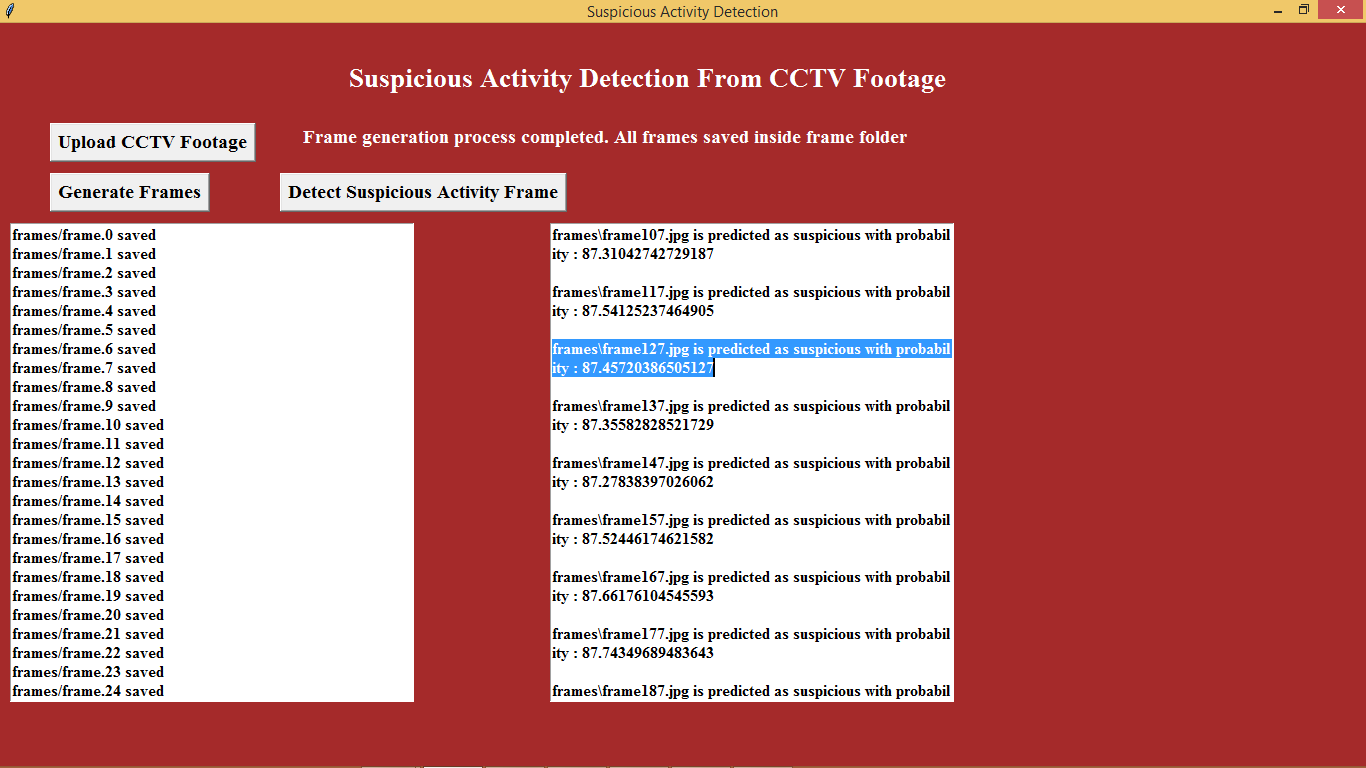
In above screen i am uploading ‘Video2’ and then extract frames



In above screen for uploaded video, we can see suspicious activity found at farme117.jpg. After scanning all images, we will get below details screen. Now in below screen we can see frame117.jpg image from frames folder



In above screen frame117 showing one image of a person with face covering. Similarly, we can see all frames details in below screen which has such activities



In above screen in right text area, we can see details of all frames which has such activities.

Note: you too can upload your own videos and check but your videos must have person covering their faces or doing shop lifting robbers videos. Your videos must be like similar one which i used in this project

**CHAPTER-11**

**CONCLUSION AND FUTURE ENHANCEMENT**

**CONCLUSION:**

In conclusion, the development of the "Abnormal Events Detection and Tracking in Surveillance System using Machine Learning" represents a significant advancement in the field of surveillance technology. By harnessing the power of machine learning algorithms and real-time video analysis, this system has the potential to revolutionize the way abnormal events are detected and managed within surveillance environments.

The system's core features, including anomaly detection, event tracking, real-time alerts, and historical data analysis, offer a comprehensive solution to the limitations of traditional surveillance systems. The integration of machine learning brings efficiency, accuracy, and automation to the process of identifying and responding to abnormal activities, contributing to enhanced security measures and proactive incident management.

**Future Enhancements:**

While the proposed system has the potential to make a substantial impact, there are several avenues for future enhancements and improvements:

**1. Improved Accuracy:** Further refinement of machine learning models is crucial to minimize false positives and negatives in abnormal event detection. Continual learning from new data and feedback can enhance the system's accuracy.

**2.Multi-modal Analysis:** Incorporating other data sources, such as audio and sensor data, can provide additional context for abnormal event detection, improving the overall accuracy of the system.

**3.Edge Computing:** Exploring edge computing solutions can enable faster processing and response times by performing some computations directly on the cameras or edge devices, reducing latency.

**CHAPTER-12**

**BIBLIOGRAPHY**

**BOOKS REFERRED**

# [1] Y. Wang, T. Liu, J. Zhou, J. Guan, Video anomaly detection based on spatio-

# temporal relationships among objects, Neurocomputing 532 (2023) 141–151.

# [2] W. Hyun, W.J. Nam, S.W. Lee, Dissimilate-and-assimilate strategy for video

# anomaly detection and localization, Neurocomputing 522 (2023) 203–213.

# [3] N. Li, J.-X. Zhong, X. Shu, H. Guo, Weakly-supervised anomaly detection in video

# surveillance via graph convolutional label noise cleaning, Neurocomputing 481

# (2022) 154–167.

# [4] N. Li, F. Chang, C. Liu, A self-trained spatial graph convolutional network for

# unsupervised human-related anomalous event detection in complex scenes, IEEE

# Transactions on Cognitive and Developmental Systems (2022), 1–1.

# [5] S. Zhang, Z. Wei, J. Nie, L. Huang, S. Wang, Z. Li, A Review on Human Activity

# Recognition Using Vision-Based Method, Journal of Healthcare Engineering (2017)

# 1–31.

# [6] Z. Fu, X. Lai, S.M. Naqvi, Enhanced detection reliability for human tracking based

# video analytics (2019).

# [7] S. Yan, J.S. Smith, W. Lu, B. Zhang, Abnormal event detection from videos using a

# two-stream recurrent variational autoencoder, IEEE Transactions on Cognitive and

# Developmental Systems 12 (1) (2020) 30–42.

# [8] A.B. Chan, N. Vasconcelos, Modeling, clustering, and segmenting video with

# mixtures of dynamic textures, IEEE Trans. on Pattern Analysis and Machine

# Intelligence, vol. 30(5) (2008) 909–926.

# [9] Y. Yang, Z. Fu, S.M. Naqvi, A two-stream information fusion approach to abnormal

# event detection in video, IEEE International Conference on Acoustics, Speech and

# Signal Processing (ICASSP) (2022) 5787–5791.

# [10] Z. Fu, F. Angelini, J. Chambers, S.M. Naqvi, Multi-level cooperative fusion of GM-

# PHD filters for online multiple human tracking, IEEE Transactions on Multimedia

# 21 (9) (2019) 2277–2291.

# [11] Y. Du, W. Wang, L. Wang, Hierarchical recurrent neural network for skeleton based

# action recognition (2015)

# [12] F. Angelini, Y. Jiawei, S.M. Naqvi, Privacy-Preserving Online Human Behaviour

# Anomaly Detection Based On Body Movements and Objects Positions (2019).

# [13] N. Li, F. Chang, C. Liu, Human-related anomalous event detection via spatial-

# temporal graph convolutional autoencoder with embedded long short-term

# memory network, Neurocomputing 490 (2022) 482–494.

# [14] S. Yan, Y. Xiong, D. Lin, Spatial temporal graph convolutional networks for

# skeleton-based action recognition., AAAI Conference on Artificial Intelligence

# (2018).

# [15] A. Markovitz, G. Sharir, I. Friedman, L. Zelnik-Manor, S. Avidan, Graph Embedded

# Pose Clustering for Anomaly Detection (2020).

# [16] Y. Yang, Z. Fu, S.M. Naqvi, Enhanced adversarial learning based video anomaly

# detection with object confidence and position, 13th International Conference on

# Signal Processing and Communication Systems (ICSPCS) (2019).

# [17] W. Sultani, C. Chen, M. Shah, Real-World Anomaly Detection in Surveillance

# Videos (2018).

# [18] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, J. Gall, A survey on human motion

# analysis from depth data (2013) 149–187.

# [19] M. Hasan, J. Choi, J. Neumann, A.K. Roy-Chowdhury, L.S. Davis, Learning

# temporal regularity in video sequences, IEEE International Conference on

# Computer Vision and Pattern Recognition (2016).

# [20] Y.S. Chong, Y.H. Tay, Abnormal event detection in videos using spatiotemporal

# autoencoder, CoRR (2017).

# [21] M. Astrid, M.Z. Zaheer, S.-I. Lee, Pseudobound: Limiting the anomaly

# reconstruction capability of one-class classifiers using pseudo anomalies,

# Neurocomputing 534 (2023) 147–160.

# [22] W. Liu, W. Luo, D. Lian, S. Gao, Future frame prediction for anomaly detection – a

# new baseline, IEEE Conference on Computer Vision and Pattern Recognition

# (CVPR) (2018).

# [23] J.T. Zhou, L. Zhang, Z. Fang, J. Du, X. Peng, Y. Xiao, Attention-driven loss for

# anomaly detection in video surveillance, IEEE Transactions on Circuits and

# Systems for Video Technology 30 (12) (2020) 4639–4647.

# [24] X. Zeng, Y. Jiang, W. Ding, H. Li, Y. Hao, Z. Qiu, A hierarchical spatio-temporal

# graph convolutional neural network for anomaly detection in videos, CoRR

# 2112.04294 (2021).

# [25] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv (2018).