INDUSTRIAL INTERNSHIP TRAINING

IN MACHINE LEARNING WITH PYTHON FROM

03/03/2021 TO 15/06/2021 AT

SOLITAIRE INFOSYS PVT. LTD. MOHALI

INTERNSHIP REPORT BEING SUBMITTED BY

ANCHAL SINGH

B. TECH STUDENT OF RAJIV GANDHI GOVT. ENGG.

COLLEGE KANGRA AT NAGROTA BAGWAN.

ROLL NO. : - 17BT030406

BRANCH: - ELECTRONICS AND COMMUNICATION ENGINEERING

UNIVERSITY: - HIMACHAL PRADESH TECHNICAL UNIVERSITY.

ACKNOWLEDGEMENT

I greatly appreciate the guidance and academic support of my mentor whose contribution in suggestions and encouragements helped me complete my internship tasks in a convincing manner in all respects. I am awfully thankful to him for his painstaking efforts in arranging experimentation and open discussions on several programming topics that enabled this duration of internship a memorable opportunity ushered on to me for learning and sharing newer dimensions to my thinking process in software Engineering as a wonderful opportunity throughout.

In the end, I would like to thank Solitaire Infosys Pvt. Ltd , faculty and other Staff for arranging needful exposure to me during my internship.

(ANCHAL SINGH)

Solitaire infosys Simple Creative innovative

S.No. 240492 Date 18-06-2021

Certificate of Training

This certificate has been awarded to Mr **Anchal Singh** from **Rajiv Gandhi Govt. Engineering College Nagrota Bagwan, Kangra** who has undertaken an internship program of **6 Months** from **03/03/2021** to **15/06/2021** in **Machine Learning With Python** Department from Solitaire Infosys Pvt. Ltd.

During the tenure of this internship with us, we found the candidate self-starter and hardworking. Also he had worked sincerely on the assignments and his performance was satisfactory to be part of the team.

We wish the Candidate success for all the future endeavors.

For Solitaire Infosys Pvt. Ltd.

Gagan Sanhuy

Human Resources Department

CERTIFIED ISO 9001 2015 COMPANY

Note: To check the authentication of certificate, please visit www.slinfy.com

COMPANY PROFILE

Solitaire Infosys private limited is a private incorporated on 6 june 2011. It is classified as non-govt. Company and registered at registrar of companies, Chandigarh its authorized share capital is Rs.900000 and its paid up capital is Rs.900000. It is involved in other computer related activities.

Complete IT solution provider

It is a leading software and web application development company, based in Mohali, which provides high quality comprehensive services to enterprises across wide range of platforms and technologies are major areas of expertise are in providing quality, cost effective software and web development.

Our focus is on understanding the diverse and mission critical need of each of our client to understand is to be able to deliver. The competence and experience of our company gives us a competitive edge by making sure we provide the best service and product to our clients. Our high quality standards unable us to deliver reliable and error free software application despite the complexity.

Campus Edge

Campus Edge is a beneficial campus resource planning software that helps institutes in a better functioning. The management, storage, and administration tasks get easier with our software. This ERP software will reduce our manual efforts and we will be able to maintain all our records more easily.

Believe in Empowering Innovation through Technology

Company design bilateral and interactive online experiences for their clients with vast knowledge of the latest industry trends and technologies.Company solution and service are created keeping in mine our client's requirements & expectations and international standard as well.

Company Expertise

Solitaire Infosys helps enterprise, whether established or startups, to build and grow customercentric digital products for mobile and web. Our clients trust our experience of our decade and our expertise that we have gained after numerous successful deliveries in various fields. We delivered satisfactory services to our clients with an aim of helping their business in growing and reaching their organizational goals.

Our Vision

We visualize becoming the most trusted and respected IT service provider across the globe with our vibrant, dynamic, and value-based IT solutions that revolve around our clients, team, and international standards. Solitaire Infosys Pvt. Ltd. envisions becoming a reliable partner to all its clients and focusing on doing everything ethically and rightly. We are always open to accept our mistakes and have the nerve to do the necessary changes.

Our Goal

To grow and flourish as a leading company, to be admired globally to provide atmost IT infrastructure and technical support services.

Delivering great work experience to our clients is what we aim at.

Solitaire Infosys Pvt. Ltd. Is an acclaimed IT service provider contributing its part in the development of many businesses around the globe. We socialize with our clients to get a superior cognizance of their business and requirements and help in fabricating websites and applications for their business. Founded in 2011 by a dynamic duo with the same aim and zeal, we have come a long way in satisfying our clients.

We are serving our clients with the world-class services for more than seven years now. The clients are delivered with the best IT solutions after we have developed a great understanding of their business and requirements. Our team works on the client projects like its own and that is the reason why we hold the edge in the league.

With every project that we deliver, we deliver our respect, creativity, quality, transparency, and teamwork to our clients. We have the experience, expertise, and capabilities to enable organizations to accelerate their service processes in every possible way. We are known for our excellent customer satisfaction, cost-effectiveness, and innovative skills that are unparalleled.

Work

How is working in solitaire Infosys special, find out about the work culture and career growth opportunities in solitaire Infosys at their office in Mohali. Follow them here and find out about their team, the fun and the work they do and stay updated with recruitment news and updates of solitaire Infosys.

Looking for in youth

As per the latest jobs posted by Solitaire Infosys they are regularly searching for job candidate with talents. They have latest openings. They freely communicated and hire fresher and experienced candidates who have filled their profiles and applied on their jobs at youth4work. What solitaire Infosys are looking for in the talents they add in their team is generally beyond just degrees and certificates but is the real passion for work a youth does. Which is generally seen by the details profiles of youth and talent matrix of youth?

SOLITAIRE INFOSYS is a leading software and web application development company, based in Mohali that provides high quality compressive services to enterprises across a wide range of platform and technologies. Our major areas of expertise are in providing quality, cost effective software or web development.

Our focus in own understanding the diverse and mission-critical needs of each of our clients. To understand is to be able to deliver. The competence and experience of our company gives us a competitive edge of making sure we provide the best services and products to our clients. Our high quality standard enables us to delivered reliable and errors-free software applications, despite their complexity.

We provide web design/web development, B2B&B2C E-commerce solutions, SEO &web promotions strategies implementation consulting services to both domestic as well as international clients‟ at the most affordable rates.

Table of figures

Table of Contents

Chapter-1

1.1 Introduction

1.1.1 Python Basic Theory

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Two major versions of Python are currently in active use:

- Python 3.x is the current version and is under active development.
- Python 2.x is the legacy version and will receive only security updates until 2020.

No new features will be implemented. Note that many projects still use Python 2, although migrating to Python 3 is getting easier.

In addition, some third-parties offer re-packaged versions of Python that add commonly used libraries and other features to ease setup for common use cases, such as **math, data analysis or scientific** use.

1.1.2 Why to Learn 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 a MUST for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain.

1.1.3 Key advantages of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. We do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive** − we 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 Characteristics of Python

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to 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.

1.3 Features of Python

- **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** – We 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.

Chapter-2

2.1 Installation of Integrated development system

During my internship period I entirely worked on Jupyter Notebook (IDE for python) which comes under the full package of Anaconda.

Here is how to get started with Anaconda System:

1. Download the Anaconda installer.
2. Double click the installer to launch.
3. Click Next.
4. Read the licensing terms and click “I Agree”.
5. Select an install for “Just Me” unless you’re installing for all users (which requires Windows Administrator privileges) and click Next.
6. Select a destination folder to install Anaconda and click the Next button.

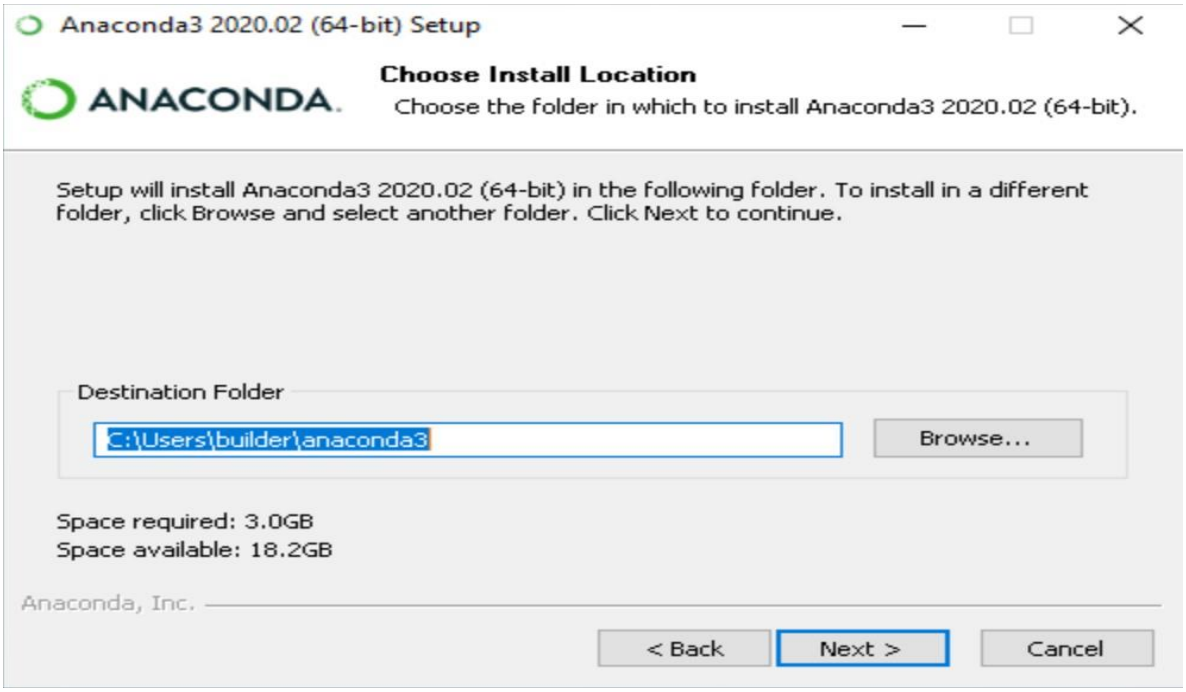


Figure-1: Anaconda installation interface

7. Choose whether to add Anaconda to your PATH environment variable. I recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

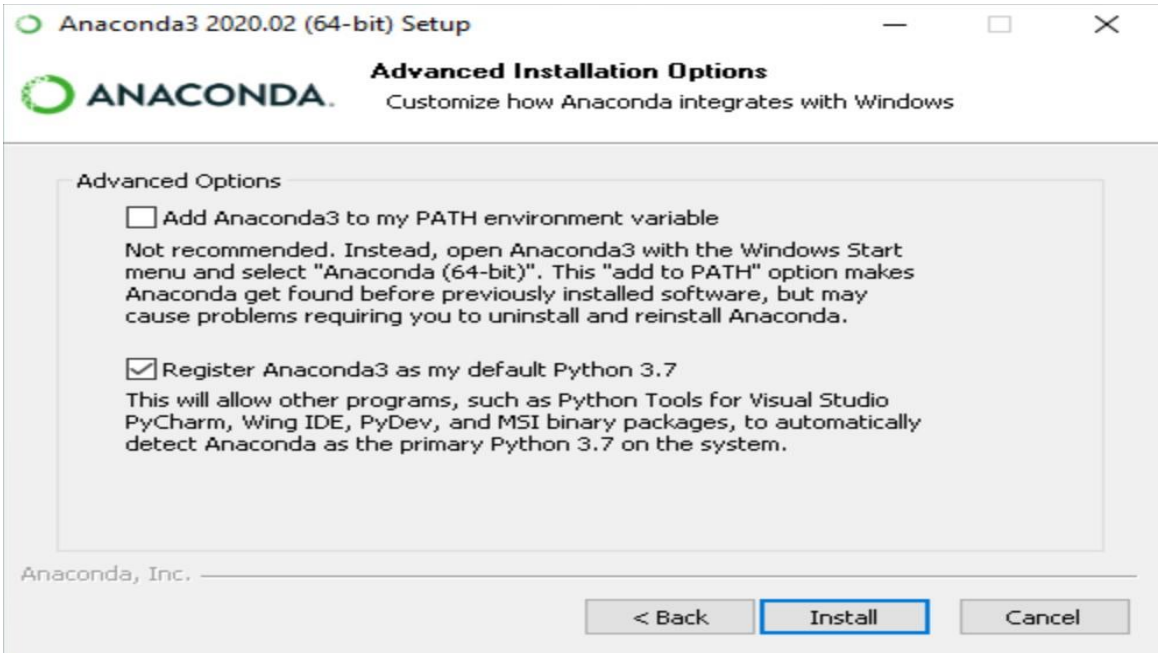


Figure-2: customizing installation interface

8. Choose whether to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked.
9. Click the Install button. If want to watch the packages Anaconda is installing, click Show Details.
10. Click the Next button.
11. Optional: To install PyCharm for Anaconda, click on the link to https://www.anaconda.com/pycharm.

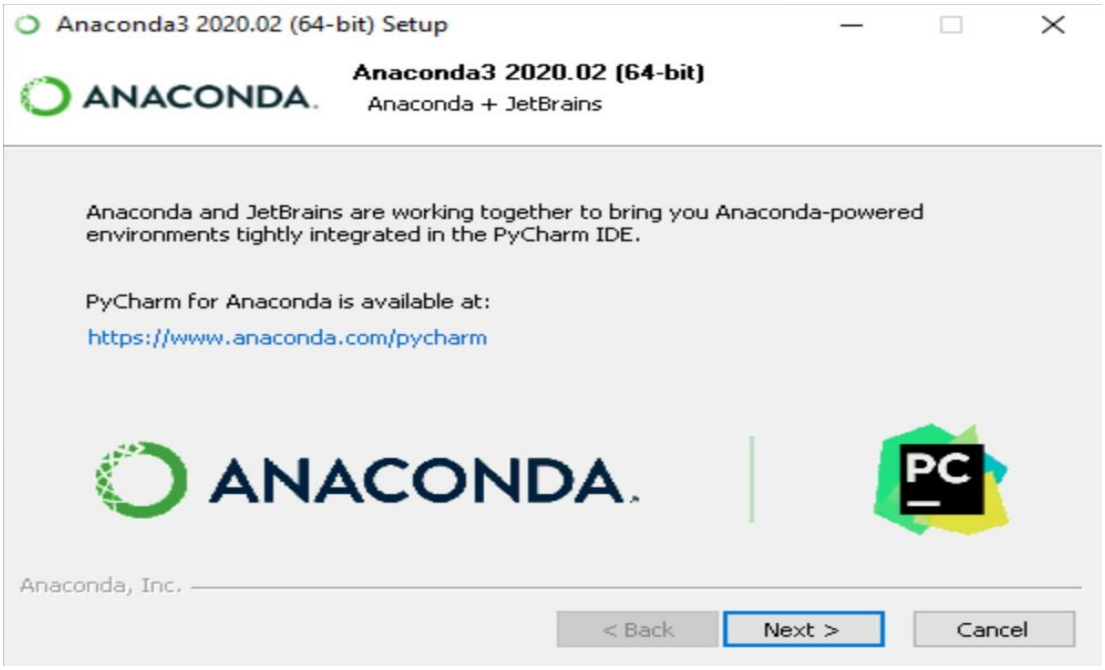


Figure-3: Anaconda installation interface

12. Or to install Anaconda without PyCharm, click the Next button.
13. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:

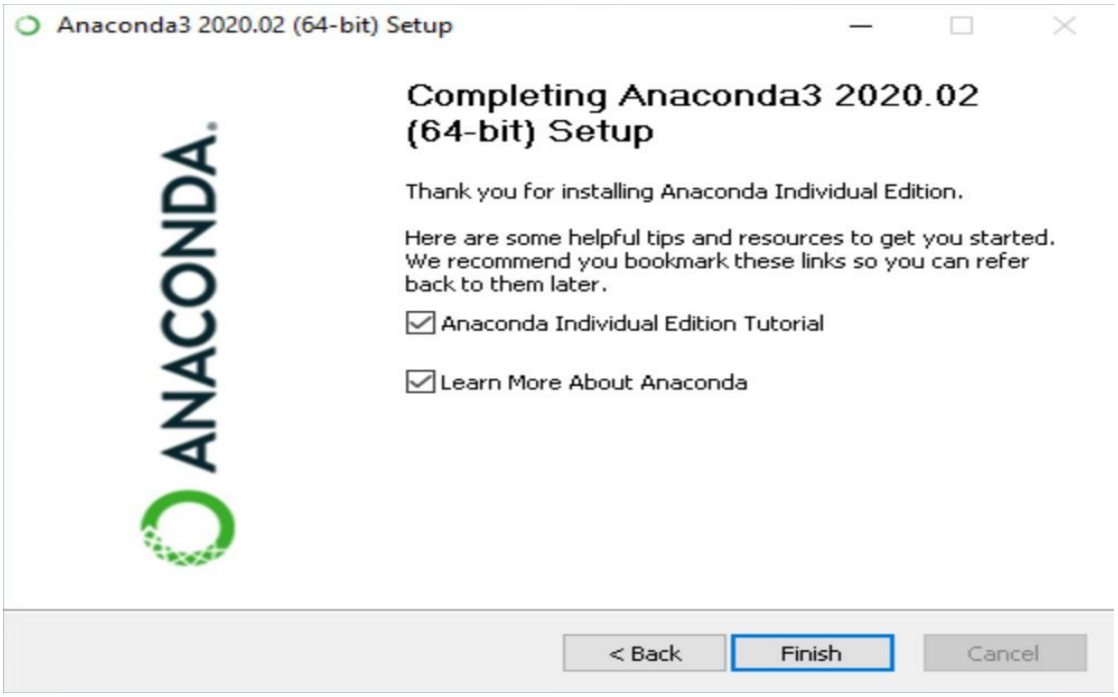


Figure-4: Final installation interface

you wish to read more about Anaconda.org and how to get started with Anaconda, check the boxes “Anaconda Individual Edition Tutorial” and “Learn more about Anaconda”. Click the Finish button.

Chapter -3

3.1 Writing First Code in Python

3.1.1 Printing statement Codes: The following codes define how we can print statements in python.

```
In [1]: print("Hello World, This is my first code in python")
        x=10
        print("x = ",x)
        print("hello")
        x=22
        print("x = ",x)
        print(x-2)
        x=x-2
        print("x = ",x)
```

```
Hello World, This is my first code in python
x =  10
hello
x =  22
20
x =  20
ADD
22
SUB
x =  20
```

3.2 Python Objects and Data Structures

3.2.1 Numbers-Here are some direct commands to perform mathematical calculations on numbers

```
In [7]: # Direct commands
        print(2+2)
        print(3-2)
        print(1/2)
        print(3*4)
        print(3**2)
        print(2**3)   #Power of numbers
        print(11%3)
        print(2.75%0.5)    #Modulus
        print(-10%3)
        print(0XAF)     #Hexadeimal multiplication
        print(2+5)
```

```
4
1
0.5
12
9
8
2
0.25
2
175
7
```

Python Supports many types of numbers as intergers(int), floating point numbers(float), complex numbers etc.

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

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

The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

```
In [12]: #ASSINGMENT
         x=3

         y=4

         a='Anchal'
         b='Singh'
         c='Manhas'
         #printing variables
         print(x)
         print(y)
         print(x+y)
         print(a+b+c)
         print(a)
         print(b)
         print(c)
```

```
3
4
7
AnchalSinghManhas
Anchal
Singh
Manhas
```

Strings are special case for assignment becase assignment of string is kept under quotes.

3.2.3 List- A list contains items separated by commas and enclosed within square brackets [ ].

```
In [15]: list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
         tinylist = [123, 'john']

         print (list)          # Prints complete list
         print (list[0])       # Prints first element of the list
         print (list[1:3])    # Prints elements starting from 2nd till 3rd
         print( list[2:])      # Prints elements starting from 3rd element
         print (tinylist * 2)  # Prints list two times
         print (list + tinylist) # Prints concatenated lists
```

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

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

3.2.4 Tuples- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

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

In [16]:
```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print (tuple)               # Prints the complete tuple
print(tuple[0] )           # Prints first element of the tuple
print (tuple[1:3])          # Prints elements of the tuple starting from 2nd till 3rd
print (tuple[2:] )          # Prints elements of the tuple starting from 3rd element
print (tinytuple * 2 )      # Prints the contents of the tuple twice
print (tuple + tinytuple)   # Prints concatenated tuples
```

```
('abcd', 786, 2.23, 'john', 70.2)
abcd
(786, 2.23)
(2.23, 'john', 70.2)
(123, 'john', 123, 'john')
('abcd', 786, 2.23, 'john', 70.2, 123, 'john')
```

3.2.5 Dictionaries- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs.

A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

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

In [17]:
```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}


print (dict['one'] )      # Prints value for 'one' key
print( dict[2] )          # Prints value for 2 key
print (tinydict)          # Prints complete dictionary
print (tinydict.keys())   # Prints all the keys
print (tinydict.values()) # Prints all the values
```

```
This is one
This is two
{'name': 'john', 'code': 6734, 'dept': 'sales'}
dict_keys(['name', 'code', 'dept'])
dict_values(['john', 6734, 'sales'])
```

3.2.6 Set- set is another type of data which conatins elements enclosed in curly braces {} and it is mutable data type. i.e. elements can be updated in set.

```
In [18]: basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
         print(basket) # duplicates will be removed
         a = set('abracadabra')
         print(a)
         a.add('z')
         print(a)
```

```
{'pear', 'orange', 'banana', 'apple'}
{'d', 'c', 'r', 'b', 'a'}
{'d', 'z', 'c', 'r', 'b', 'a'}
```

3.3 Python operators

Operators are the constructs which can manipulate the value of operands.

Consider the expression 4 + 5 = 9. Here, 4 and 5 are called operands and + is called operator.

Types of Operator

Python language supports the following types of operators.

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

3.3.1 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

3.3.2 Comparison operators

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

>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

3.3.3 Assignment Operators

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 / 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

3.3.4 Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; in the binary format their values will be 0011 1100 and 0000 1101.

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)

3.3.5 Logical Operators-

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.

3.4 Python Operators precedence:

The following table shows the precedence from highest to lowest.

Sr.No	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus (method names for the last two are +@ and -@)
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>> << Right and left bitwise shift
6	& Bitwise 'AND'

7	^ | Bitwise exclusive `OR' and regular `OR'
8	<= < > >= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators
11	is is not Identity operators
12	in not in Membership operators

Chapter-4

4.1 Python Decision Statement

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.

Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. It determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.

Following is the general form of a typical decision making structure found in most of the programming languages −

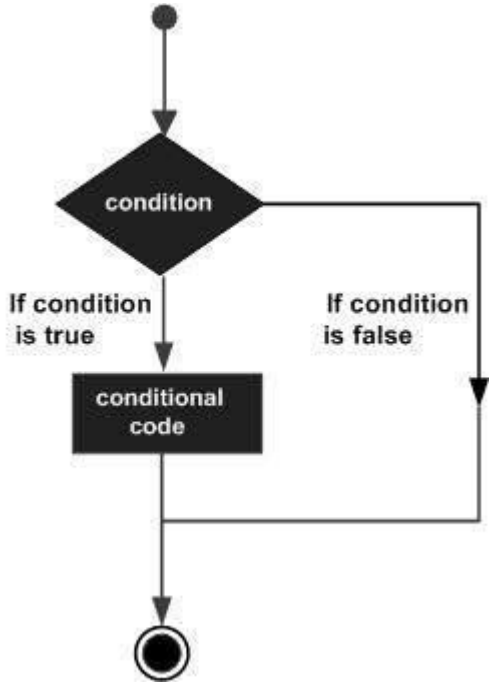


Figure-5:Conditional decision flow diagram

Python programming language assumes any **non-zero** and **non-null** values as TRUE, and if it is either **zero** or **null**, then it is assumed as FALSE value.

Python has four decision making statements:

- If statement
- If else statement
- Elif statement
- Nested if statement

4.1.1 If statement: The **if** statement contains a logical expression using which data is compared and a decision is made based on the result of the comparison. Syntax

If expression:

statement(s)

Flow diagram:

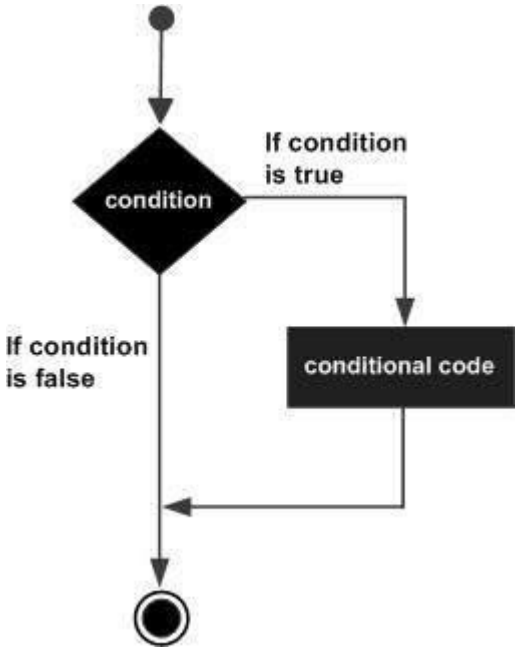


Figure-6: if condition

Example:

```
In [21]: a = 100
         if a:
            print ("a - Got a true expression value")
            print (a)

         v = 0
         if v:
            print ("2 - Got a true expression value")
            print (v)

         a - Got a true expression value
         100
```

```
In [23]: a=10
         b=input('Enter any number to test')

         if a==b:
             print('Both numbers are equal')

         Enter any number to test6
```

4.1.2 Else-if statement: An **else** statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.

The *else* statement is an optional statement and there could be at most only one **else** statement following **if**.

Syntax:

If expression:

Statement (s)

Else expression:

Statement (s)

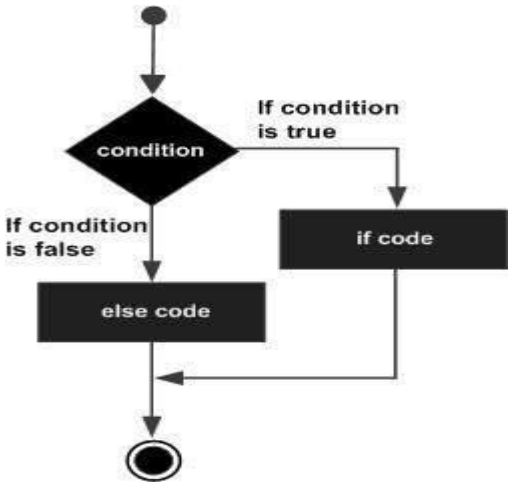


Figure-7:if else condition

Example

```
In [24]: v1 = 100
if v1:
   print( "1 - Got a true expression value")
   print( v1)
else:
   print ("1 - Got a false expression value")
   print (v1)

v2 = 0
if v2:
   print ("2 - Got a true expression value")
   print( v2)
else:
   print ("2 - Got a false expression value")
   print (v2)

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

4.1.3 Elif statement: The **elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Syntax:

if expression1:

statement (s) elif

expression2:

statement (s) elif

expression3:

statement (s) else:

statement (s)

Example

```
In [25]: var = 100
         if var == 200:
            print ("1 - Got a true expression value")
            print( var)
         elif var == 150:
            print ("2 - Got a true expression value")
            print (var)
         elif var == 100:
            print ("3 - Got a true expression value")
            print (var)
         else:
            print( "4 - Got a false expression value")
            print (var)

         3 - Got a true expression value
         100
```

4.1.4 Nested if statement: There may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, we can use the nested **if** construct.

In a nested **if** construct, we havean **if...elif...else** construct inside another **if...elif...else** construct.

Syntax

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

Example:

In [27]:
```
var = 100
if var < 200:
   print ("Expression value is less than 200")
   if var == 150:
      print ("Which is 150")
   elif var == 100:
      print ("Which is 100")
   elif var == 50:
      print ( "Which is 50")
   elif var < 50:
      print ("Expression value is less than 50")
else:
   print ( "Could not find true expression")
```

```
Expression value is less than 200
Which is 100
```

Chapter-5

5.1 Loops in Python

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement −

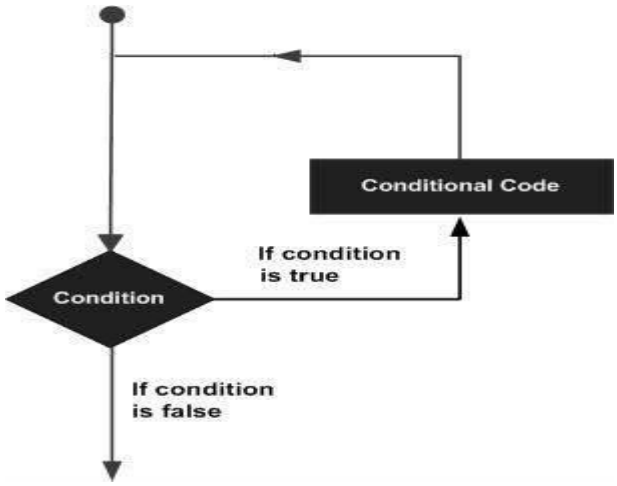


Figure-8: Conditional Looping diagram There

are few loop statements available as:

- While loop
- For loop
  Nest
  ed loops

5.1.1 While loop:

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

Syntax:

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

statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

Flow diagram:

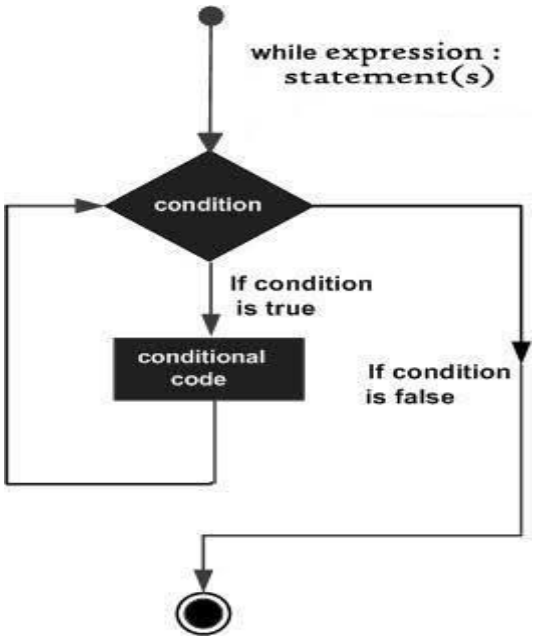


Figure-9: while Conditional loop

Example:

```
In [1]: a = 0
        while (a < 9):
           print ('The count is:', a)
           a = a + 1
```

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

5.1.2 For loop: It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax:

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

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

Flow chart

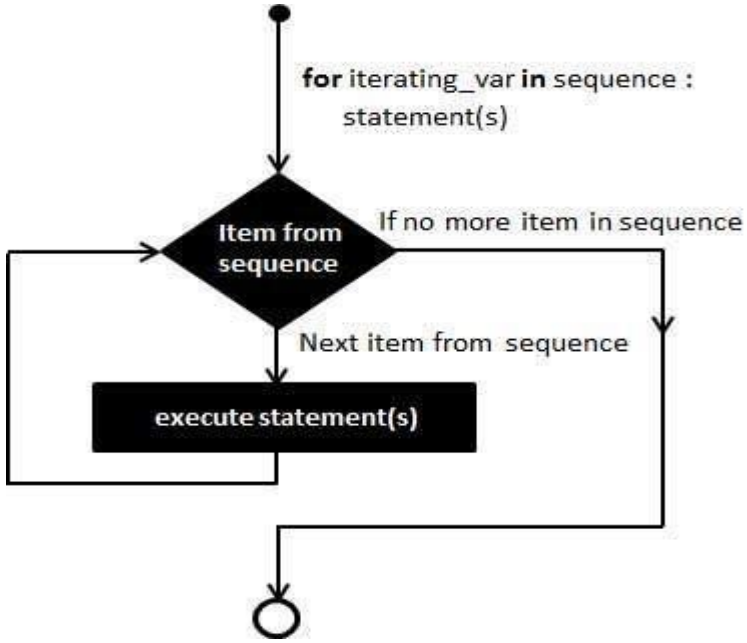


Figure-10: For looping

Example

```
In [2]: for letter in 'Python':     # First Example
            print ('Current Letter :', letter)

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

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
```

5.1.3 Nested loops: Python programming language allows to use one loop inside another loop.

Syntax

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

Example:

```
In [16]: i = 2
         while(i < 60):
            j = 2
            while(j <= (i/j)):
               if not(i%j): break
               j = j + 1
            if (j > i/j) : print (i, " is prime")
            i = i + 1
```

```
2  is prime
3  is prime
5  is prime
7  is prime
11  is prime
13  is prime
17  is prime
19  is prime
23  is prime
29  is prime
31  is prime
37  is prime
41  is prime
43  is prime
47  is prime
53  is prime
59  is prime
```

Chapter-6

6.1 Control statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements

- Break statement
- Continue Statement
- Pass statement

6.1.1 Break Statement: It terminates the current loop and resumes execution at the next statement.

The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The **break** statement can be used in both *while* and *for* loops.

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

Flow chart:

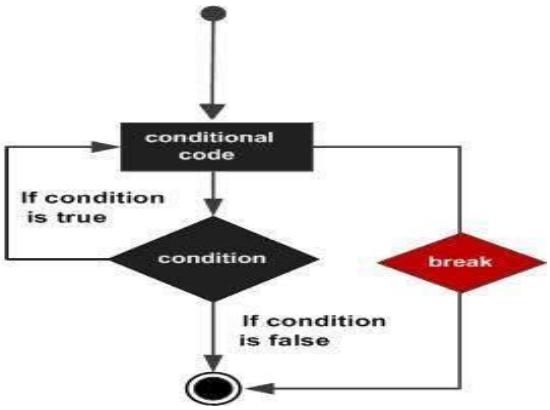


Figure-11: Break statement

Example:

```
In [17]: for letter in 'Python':     # First Example
             if letter == 'h':
                 break
             print ('Current Letter :', letter)

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

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

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

The **continue** statement can be used in both *while* and *for* loops.

Flow Chart:

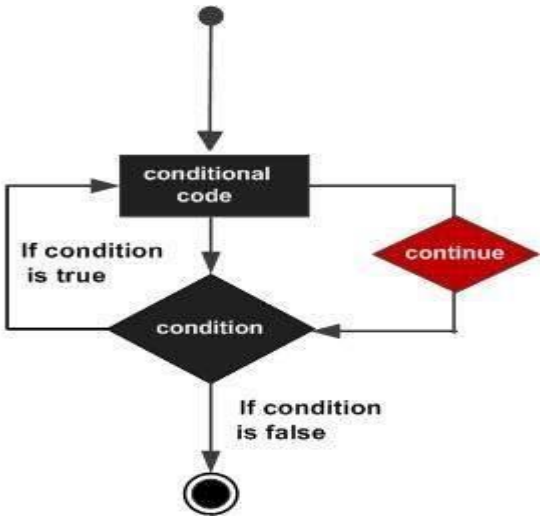


Figure-12: Continue Statement

Example:

```
In [18]: for letter in 'Python':     # First Example
             if letter == 'h':
                 continue
             print ('Current Letter :', letter)

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

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

6.1.3 Pass statement: It is used when a statement is required syntactically but we do not want any command or code to execute.

The pass statement is a *null* operation; nothing happens when it executes. The pass is also useful in places where code will eventually go, but has not been written yet.

Example:

```
In [19]: for letter in 'Python':
             if letter == 'h':
                pass
                print ('This is pass block')
             print ('Current Letter :', letter)

Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
```

Chapter-7

7.1 Functions in python

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 us many built-in functions like print(), etc. but we can also create our own functions. These functions are called *user-defined functions.*

7.1.1 Defining a Function

We can define functions to provide the required functionality. Here are 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. We 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.

Syntax

```
def functionname( parameters ):

"function_docstring"

function_suite    return

[expression]
```

e.g

```
def printme( str ):

"This prints a passed string into this function"

   print str

return
```

7.1.2 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, we can execute it by calling it from another function or directly from the Python prompt.

Example:

In [23]:
```
# Function definition
def printme( str ):
   print("This prints a passed string into this function")
   print (str)
   return;

#  calling printme function
printme(" first call to user defined function!")
printme(" second call to the same function")
```

```
This prints a passed string into this function
 first call to user defined function!
This prints a passed string into this function
 second call to the same function
```

In [24]:
```
# Function definition is here
def changeme( mylist ):
   print("This changes a passed list into this function")
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print ("Values inside the function: ", mylist)
   return

#  calling changeme function
mylist = [10,20,30];
changeme( mylist );
print ("Values outside the function: ", mylist)
```

```
This changes a passed list into this function
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```

7.2 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 we call a function, the variables declared inside it are brought into scope

```
In [25]: 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;

         #  call sum function
         sum( 10, 20 );
         print ("Outside the function global total : ", total)
```

```
Inside the function local total :  30
Outside the function global total :  0
```

7.3 Pass by reference vs value:

All parameters (arguments) in the Python language are passed by reference. It means if we change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

```
In [32]: mylist=[12,23,34]
         print("Original list is:",mylist)
         # Function definition is here
         def changeme( mylist ):
             print("This changes a passed list into this function")
             mylist.append([1,2,3,4]);

             print ("Values inside the function: ", mylist)
             return

         # Now you can call changeme function
         mylist = [10,20,30];
         changeme( mylist );
         print ("Values outside the function: ", mylist)
```

```
Original list is: [12, 23, 34]
This changes a passed list into this function
Values inside the function:  [10, 20, 30, [1, 2, 3, 4]]
Values outside the function:  [10, 20, 30, [1, 2, 3, 4]]
```

There is one more example where argument is being passed by reference and the reference is being overwritten inside the called function.

```
In [33]: # Function definition is here
         def changeme( mylist ):
            print("This changes a passed list into this function")
            mylist = [1,2,3,4]; # This would assig new reference in mylist
            print("Values inside the function: ", mylist)
            return

         # Now you can call changeme function
         mylist = [10,20,30];
         changeme( mylist );
         print("Values outside the function: ", mylist)

         This changes a passed list into this function
         Values inside the function:  [1, 2, 3, 4]
         Values outside the function:  [10, 20, 30]
```

The parameter *mylist* is local to the function changeme. Changing mylist within the function does not affect *mylist*. The function accomplishes nothing and finally this would produce the result

7.4 Function Arguments

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

- Required arguments
- Keyword arguments
- Default arguments

7.4.1 Required arguments

Required arguments are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition.

To call the function *printme()*, you definitely need to pass one argument, otherwise it gives a syntax error as follows −

```
In [41]: # Function definition is here
         def printme( str ):
             print( "This prints a passed string into this function")
             print (str)
             return;

         # Now you can call printme function
         printme()

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-41-9b4c4d2485a1> in <module>
               6
               7 # Now you can call printme function
         ----> 8 printme()

         TypeError: printme() missing 1 required positional argument: 'str'
```

7.4.2 Keyword arguments

Keyword arguments are related to the function calls. When we use keyword arguments in a function call, the caller identifies the arguments by the parameter name.

This allows us to skip arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters.We can also make keyword calls to the *printme()* function in the following ways −

```
In [42]: # Function definition is here
         def printme( str ):
             print( "This prints a passed string into this function")
             print (str)
             return;

         # Now you can call printme function
         printme( str = "My string")

         This prints a passed string into this function
         My string
```

```
In [43]: # Function definition is here
         def printinfo( name, age ):
             print("This prints a passed info into this function")
             print( "Name: ", name)
             print ("Age ", age)
             return;

         # Now you can call printinfo function
         printinfo( age=50, name="miki" )

         This prints a passed info into this function
         Name:  miki
         Age  50
```

7.4.3 Default arguments

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument. The following example gives an idea on default arguments, it prints default age if it is not passed

```
In [45]: # Function definition is here
         def printinfo( name, age = 35 ):
             print("This prints a passed info into this function")
             print ("Name: ", name)
             print ("Age ", age)
             return;

         # Now you can call printinfo function
         printinfo( age=50, name="miki" )
         printinfo( name="miki" )

         This prints a passed info into this function
         Name:  miki
         Age  50
         This prints a passed info into this function
         Name:  miki
         Age  35
```

7.5 Lambda Functions: These are small functions which are not defined with any name and carry a single expression whose result is returned. Lambda functions comes very handy when operating

with lists. These function are defined by the keyword **lambda** followed by the variables, a colon and the respective expression.

```
In [9]: z = lambda x: x * x
        z(8)

Out[9]: 64
```

```
In [10]: list1 = [1,2,3,4,5,6,7,8,9]
         eg = map(lambda x:x+2, list1)
         print(type(eg))
         print(list(eg))

         <class 'map'>
         [3, 4, 5, 6, 7, 8, 9, 10, 11]
```

map ( ) function basically executes the function that is defined to each of the list's element separately.

```
In [11]: def myfunc(a, b):
           return a + b

         x = map(myfunc, ('apple', 'banana', 'cherry'), ('orange', 'lemon', 'pineapple'))
         print(tuple(x))

         ('appleorange', 'bananalemon', 'cherrypineapple')
```

```
In [12]: # we can also add two lists.
         list2 = [9,8,7,6,5,4,3,2,1]
         eg2 = map(lambda x,y:x+y, list1,list2)
         print(list(eg2))

         eg3 = map(str,[12,4,4,3,13,123,44])
         print(list(eg3))

         [10, 10, 10, 10, 10, 10, 10, 10, 10]
         ['12', '4', '4', '3', '13', '123', '44']
```

filter( ) function is used to filter out the values in a list. Note that **filter ()** function returns the result in a new list.

```
In [18]: list1 = [1,2,3,4,5,6,7,8,9]
         filtered_Output=filter(lambda x:x<5,list1)
         print(list(filtered_Output))

         print(filter(lambda x:x%4==0,list1))

         [1, 2, 3, 4]
         <filter object at 0x0000024C13DB77C8>
```

Chapter-8

8.1 Object oriented programming:

The term “Object-Oriented Programming” (OOP) was coined by Alan Kay around 1966 Object-Oriented Programming (OOP), is all about creating “objects”. An object is a group of interrelated variables and functions. These variables are often referred to as properties of the object and functions are referred to as the behaviour of the objects. These objects provide a better and clear structure for the program.

For example, a car can be an object. If we consider the car as an object then its properties would be – its colour, its model, its price, its brand, etc. And its behaviour function would be acceleration, slowing down, gear change.

Another example-If we consider a dog as an object then its properties would be- his colour, his breed, his name, his weight, etc. And his behaviour/function would be walking, barking, playing, etc.

8.2 What is Class?

Class is a collection of objects. Unlike the primitive data structures, classes are data structures that the user defines.

Syntax:

Class class name:

Class body

In [1]:
```
# defining class
class batch5:
    def __init__(self,name,tech,branch):   # constructor method
        self.name=name
        self.tech=tech
        self.branch=branch

    def test(self):
        print(f'Hey {self.name},{self.tech},{self.branch}')

obj1=batch5('Anchal','python','B.tech')
obj2=batch5('Sahil','ML','B.tech')
obj1.test()
obj2.test()
```

```
Hey Anchal,python,B.tech
Hey Sahil,ML,B.tech
```

8.3 Inheritance

Inheritance Means “inheriting or transfer of characteristics from parent to child class without any modification”. The new class is called the derived/child class and the one from which it is derived is called a parent/base class.

In [2]:
```
class Director():  #This is a parent class
    def __init__(self, name, age, salary):
        self.name=name
        self.age=age
        self.salary=salary

    def config(self):
        return f'the first class instance is {self.name}'
    def test(self):
        return f'the age of {self.name} is {self.age}'
class employee(Director):  #This is a child class
    def __init__(self, name, age, salary,ide,var1):
        self.name = name
        self.age = age
        self.salary = salary
        self.ide= ide
        self.var1=var1
obj1=Director('anchal',20,239230)

print(obj1.config())
print(obj1.test())
obj2=employee('Ankit ',21,0000,'b5','python')
obj2.config()
#obj2.choice()
obj2.test()
```

```
the first class instance is anchal
the age of anchal is 20
```

Out[2]: 'the age of Ankit is 21'

In [6]:
```
class employee():
    def __init__(self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary
    def wages(self):
        return f' wages are {self.salary}'
class childemployee1(employee):
    def __init__(self,name,age,salary=215353):
        self.name = name
        self.age = age
        self.salary = salary
class childemployee2(childemployee1):
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary
emp1 = employee('a',30000,8)
emp2 = childemployee1(1,2)
emp3=childemployee2(23,45,7997)


print(emp2.salary)
print(emp2.wages())
print(emp3.wages())
```

```
215353
 wages are 215353
 wages are 7997
```

8.4 Polymorphism

While functions can take in different arguments, methods belong to the objects they act on. In Python, polymorphism refers to the way in which different object classes can share the same method name, and those methods can be called from the same place even though a variety of different objects might be passed in.

In [17]:
```
class India():
    def capital(self):
        print("New Delhi is the capital of India.")

    def language(self):
        print("Hindi is the most widely spoken language of India.")

    def type(self):
        print("India is a developing country.")
class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")

    def language(self):
        print("English is the primary language of USA.")

    def type(self):
        print("USA is a developed country.")
    def abhi(self):
        print('I am abhishek')

obj_ind = India()
obj_usa = USA()
obj_ind.language()
obj_usa.language()
```

```
Hindi is the most widely spoken language of India.
English is the primary language of USA.
```

8.5 Encapsulation

When working with classes and dealing with sensitive data, providing global access to all the variables used within the program is not a good choice. Encapsulation offers a way for us to access the required variables without providing the program full-fledged access to any of those variables.

Updating, modifying, or deleting data from variables can be done through the use of methods that are defined specifically for the purpose. The benefit of using this approach to programming is improved control over the input data and better security.

In [8]:
```
class Person:
    def __init__(self, name,reg,age=0):
        self.name = name
        self.__age = age
        self._reg=reg

    def __display__(self):    #private method
        print(self.name)
        print(self.__age)      #private instance

person = Person('aman',21)
#accessing using class method
person.__display__()
#accessing directly from outside

print(person.name)
print(person.__age)
print(person._reg)
```

```
aman
0
aman
21
```

Chapter-9

9.1 Graphical User Interface (GUI) Programming

Tkinter Programming Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

Import the Tkinter module.

Create the GUI application main window.

Add one or more of the above-mentioned widgets to the GUI application.

Enter the main event loop to take action against each event triggered by the user.

In [*]:
```
import tkinter as tk
from tkinter import *
win=tk.Tk()
win.title('Window_1')
win.geometry('500x400+300+200')
win.configure(bg='pink')
#function of buttons
def add():
    print(23+46)

def sub():
    print(56-24)

def mul():
    print(6*6)
#Buttons
b1=Button(win,text='Sahil',bg='red',bd=5,width=3,height=1,activebackground='green',command=add)
b1.grid(row=0,column=2)
b2=Button(win,text='Ankit',bg='red',bd=5,width=3,height=1,activebackground='green',command=sub)
b2.grid(row=1,column=2)
b3=Button(win,text='Anchal',bg='red',bd=5,width=3,height=1,activebackground='green',command=mul)
b3.grid(row=2,column=2)
#Label
l1=Label(win,text='Name').grid(row=0,column=0)
l1=Label(win,text='Age').grid(row=1,column=0)
l1=Label(win,text='Technolgy').grid(row=2,column=0)
# entry
e1=Entry(win, width=20).grid(row=0,column=1)
e2=Entry(win, width=20).grid(row=1,column=1)
e3=Entry(win, width=20).grid(row=2,column=1)
#add widgets to window
win.mainloop()

```

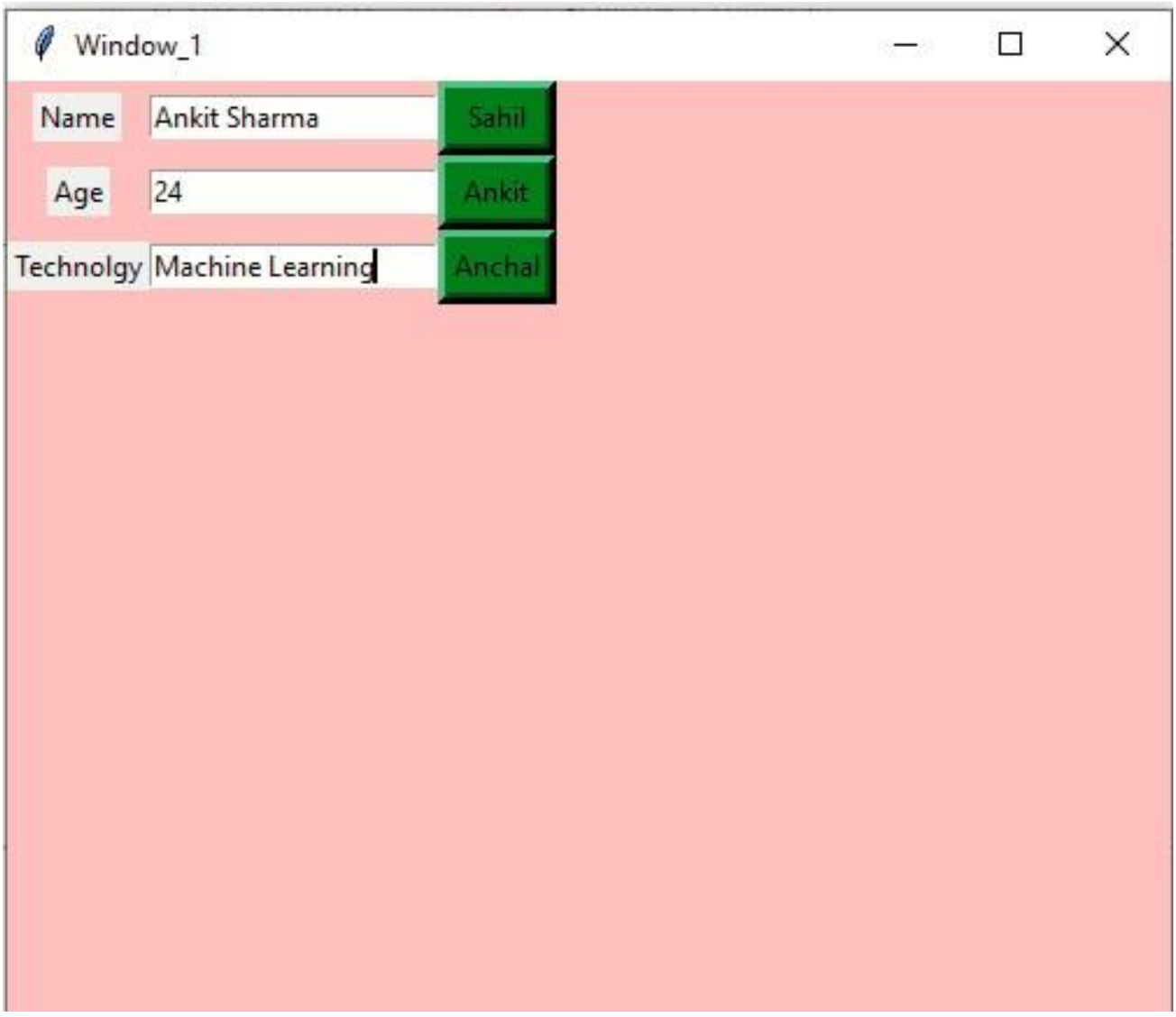


Figure 13: Simple GUI

In [*]:

```
from tkinter import *
root = Tk()
root.title('Notepad')
root.geometry('400x400+100+100')
def save_file():
    if True:
        save.set()
def donothing():

   filewin = Toplevel(root)
   button = Button(filewin, text="hit me")
   button.pack()
label_a = tk.Label(root, text="Here is a sample Window of Notepad where you can put your input text and also can save files"
label_a.pack()

label_b = tk.Label(root, text="Tkinter is a great framework to develop GUI ")
label_b.pack()

menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=save_file)
filemenu.add_command(label="Save as...", command=donothing)
filemenu.add_command(label="Close", command=donothing)
```

```
filemenu.add_separator()

filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)
editmenu = Menu(menubar, tearoff=0)
editmenu.add_command(label="Undo", command=donothing)

editmenu.add_separator()

editmenu.add_command(label="Cut", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
editmenu.add_command(label="Paste", command=donothing)
editmenu.add_command(label="Delete", command=donothing)
editmenu.add_command(label="Select All", command=donothing)

menubar.add_cascade(label="Edit", menu=editmenu)
helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Help Index", command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)

root.config(menu=menubar)
root.mainloop()
```

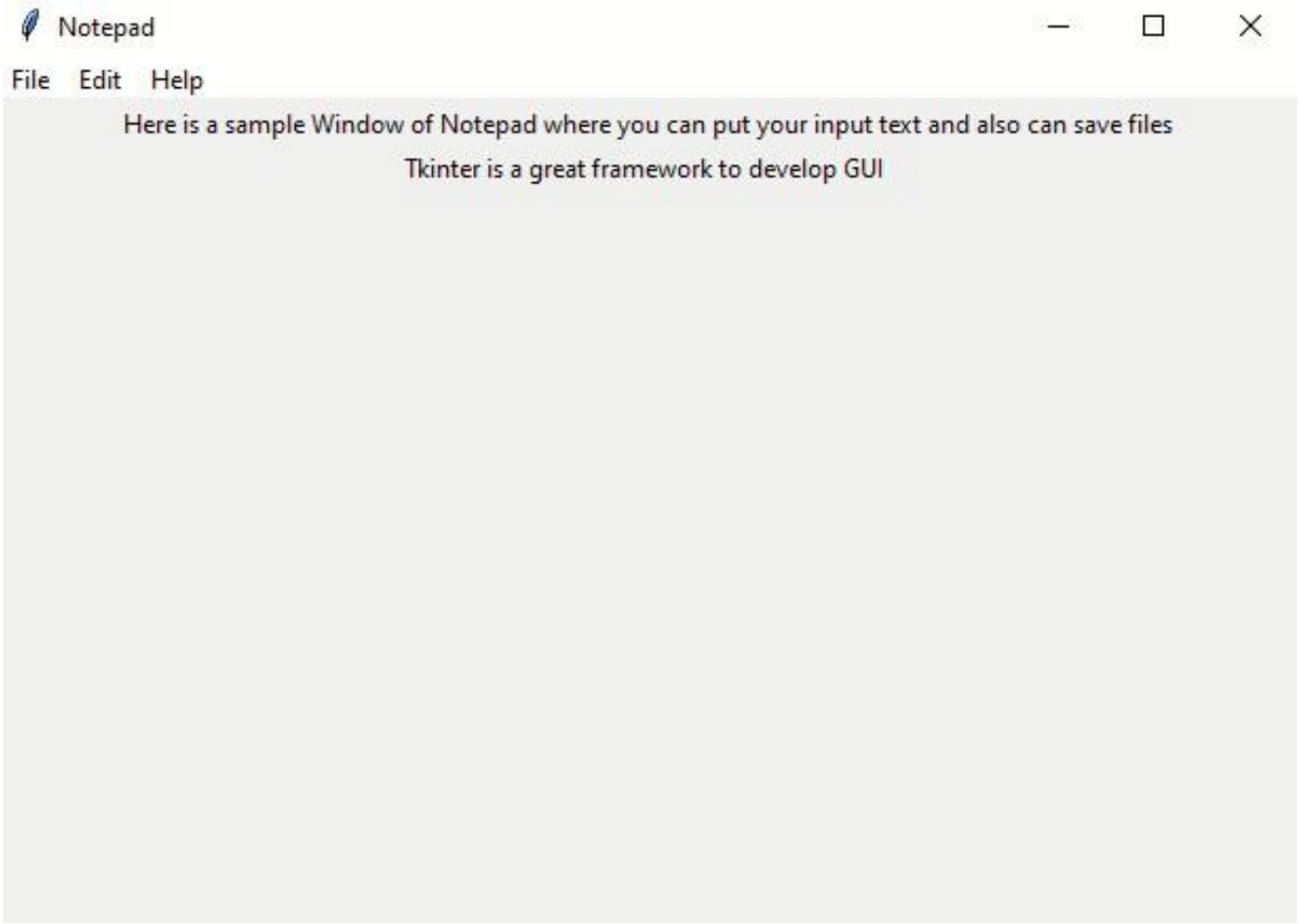


Figure 14: Notepad GUI

Chapter-10

10.1 Data Visualisation using Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter.
It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

Matplotlib was originally written by John D. Hunter in 2003. The current stable version is 2.2.0 released in January 2018.

10.2 Sinusoidal plot

```
In [2]:
# a simple plot
  # used to plot interactive plots
from matplotlib import pyplot as plt
import numpy as np
pi=3.14
x = np.arange(0, pi*2,0.2)  # generates the array of range defined
y = np.sin(x)
plt.plot(x,y)
plt.xlabel("angle")
plt.ylabel("sine")
plt.title('sine wave')
plt.show()
```

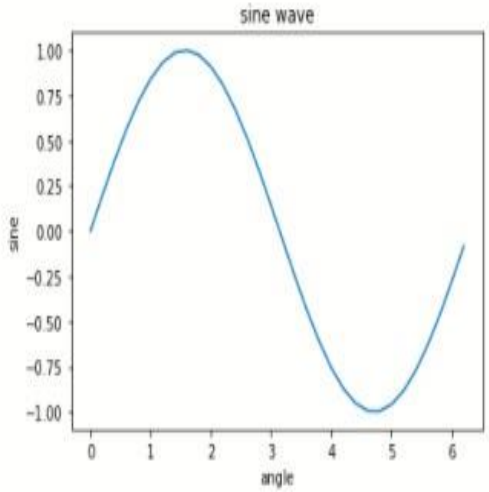


Figure 15: Simple sinusoidal plot

10.3 Parabolic Plot

```
In [5]:
# Parabolic plot
from numpy import *
from pylab import *
x = linspace(-30, 30, 30)
y = x**2
plot(x, y,color='red')
plt.xlabel('number')
plt.ylabel('sqaured value')
show()
```

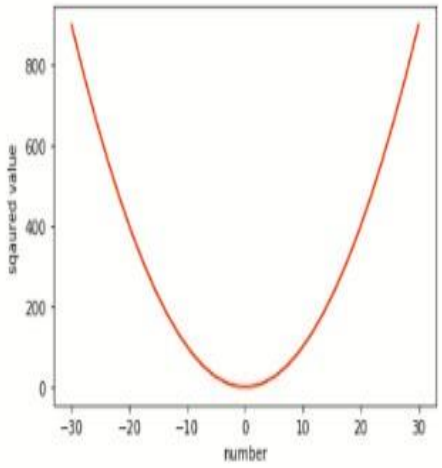


Figure 16: Parabolic plot

10.3 Subplots on single Axis

```
In [9]:
#Overlaid plots
from pylab import *
x = linspace(-6, 6, 40)

plot(x, sin(x))
plot(x, cos(x), 'r*')
plot(x, -sin(x), 'g--')
show()
```

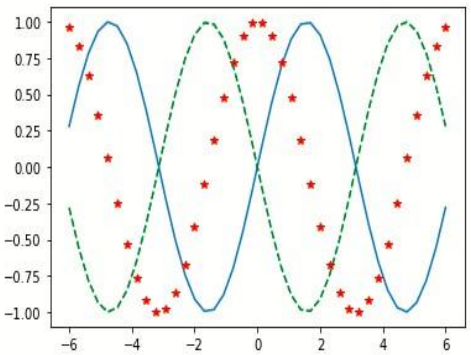


Figure 17: Overlaid plots

10.4 Bar Plots

In [24]:
```
#Bar plot
import matplotlib.pyplot as plt
import numpy as np
x=np.array([1,2,5,4])
y=np.array([4,5,6,7])
z=[1,2,4,5]
plt.bar(x,z)
plt.show()
```

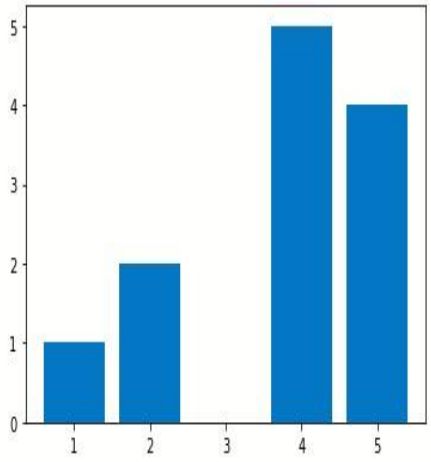


Figure 18: bar plot

10.5 Histogram plots

In [29]:
```
#Histrogram plots
import matplotlib.pyplot as plt
import numpy as np

age=[12,23,45,67,44,6,77,8,9,44,45,88,99,7,90]
bins=7
range=(0,100)
plt.hist(age,bins,range,color='green',width=10)
plt.xlabel('age')
plt.ylabel('bins')
plt.title('Histogram')
plt.show()
```

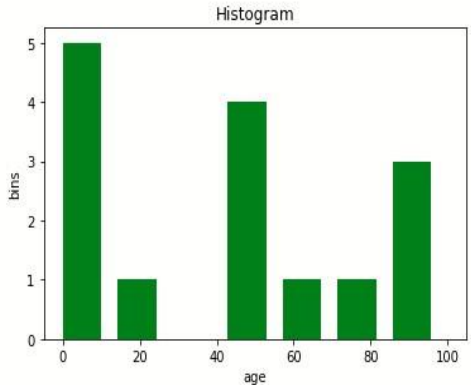


Figure 19: Histogram plot

10.6 Scatter Plots

In [38]:
```
#Scatter plots
import matplotlib.pyplot as plt
import numpy as np
import random
x=np.random.normal(5.0,1.0,200)
y=np.random.normal(5.0,2.0,200)
plt.scatter(x,y)
plt.show()
print(x)
print(y)
```

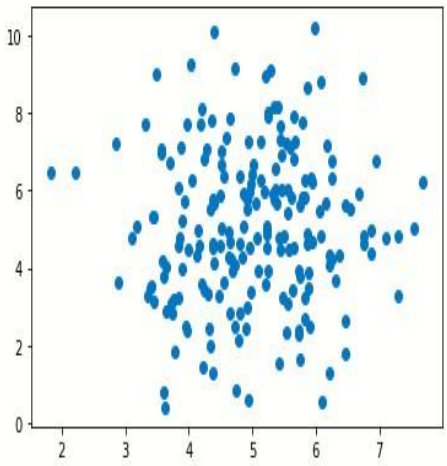


Figure 20: Scatter plot

Chapter-11

11.1 Data Visualisation using Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures
Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

11.2 Setting up Pandas with first code

In [ ]:
```
#to install Pandas Use  below command
pip install pandas
```

In [1]:
```
#A simple dataset as a dictionary
dict = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],
       "capital": ["Brasilia", "Moscow", "New Dehli", "Beijing", "Pretoria"],
       "area": [8.516, 17.10, 3.286, 9.597, 1.221],
       "population": [200.4, 143.5, 1252, 1357, 52.98] }

import pandas as pd
dataset = pd.DataFrame(dict)

```

In [2]:
```
print(dataset)
```

```
        country    capital    area  population
0        Brazil   Brasilia   8.516      200.40
1        Russia     Moscow  17.100      143.50
2         India  New Dehli   3.286     1252.00
3         China    Beijing   9.597     1357.00
4  South Africa   Pretoria   1.221       52.98
```

11.3 Read data from csv file:

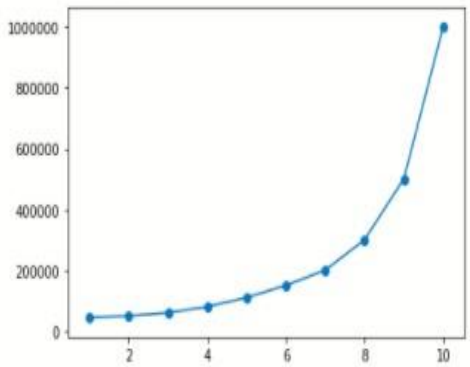
In [6]:
```
import pandas as pd
pop = pd.read_csv('countries_female_population.csv')

# Print out first 4 observations
print(pop[0:4])

# Print out fifth and sixth observation
print(pop[4:6])
```

```
       Australia   9887846   9999199   10100991   10218321   10348070   \
0        Austria  4179743.0   4158169    4190297    4220228    4246571
1        Belgium  5267437.0   5288959    5309245    5334527    5367561
2         Canada        NaN  15829173   15834015   16146667   16303914
3  Czech Republic  5264218.0   5236563    5236715    5239664    5248431

   10570420   10770864   10986535   11218144   11359807   11402769
0   4261752    4277716    4287213    4296197    4308915    4324983
1   5403126    5442557    5484429    5527684    4996609    5622157
2  16478759   16601231   16802833   16807738   17101813   17379104
3   5261005    5298196    5331165    5349616    5363971    5347235
  Australia   9887846   9999199   10100991   10218321   10348070   10570420   \
4   Denmark  2714208.0   2721084    2727505    2734113    2741613    2750422
5   Finland  2657304.0   2661379    2666839    2674534    2683230    2693213

   10770864   10986535   11218144   11359807   11402769
4   2763125    2779431    2791452    2804046    2813740
5   2703697    2714661    2726360    2736860    2748733
```

In [8]:
```
import pandas as pd
import numpy as np
dataset=pd.read_csv('Position_Salaries.csv')
x=dataset.iloc[: , 1].values   #syntax  var.iloc[nrows,ncol]
y=dataset.iloc[: , -1].values
print(x)
print(y)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[  45000   50000   60000   80000  110000  150000  200000  300000  500000
 1000000]
```

In [10]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.plot(x,y)
plt.scatter(x,y)
plt.show()
```



In [12]:
```
column_names = ["Country"] + list(range(2002, 2013))
male_pop = pd.read_csv("countries_female_population.csv",

                 index_col=0,
                 names=column_names)

female_pop = pd.read_csv("countries_female_population.csv",

                         index_col=0,
                         names=column_names)


population = male_pop + female_pop
print(population)
```

Country	2002	2003	2004	2005	2006
Australia	19775692.0	19998398	20201982	20436642	20696140
Austria	8359486.0	8316338	8380594	8440456	8493142
Belgium	10534874.0	10577918	10618490	10669054	10735122
Canada	NaN	31658346	31668030	32293334	32607828
Czech Republic	10528436.0	10473126	10473430	10479328	10496862
Denmark	5428416.0	5442168	5455010	5468226	5483226
Finland	5314608.0	5322758	5333678	5349068	5366460
France	61020146.0	61311066	61578308	64294980	64780174
Germany	84331266.0	84383602	84351314	84294444	84196068
Greece	11095774.0	11115590	11152498	11192238	11234028
Hungary	10675746.0	10647812	10625258	10608868	10584004
Iceland	286250.0	288368	290338	292814	297378
Ireland	3908814.0	3989026	4049908	4124160	4214286
Italy	58813000.0	59109694	59639274	60171142	60449646
Japan	130094000.0	130370000	130626000	130780000	130838034
Korea	47311560.0	47598266	47707908	47894342	48059150
Luxembourg	450460.0	454582	457160	460520	473972
Mexico	102286332.0	103531586	104737854	104374578	105416878
Netherlands	16266636.0	16354202	16424236	16479094	16513606
New Zealand	4010240.0	4075800	4128400	4165340	4203860
Norway	4564264.0	4592290	4616816	4644586	4676476
Poland	39743330.0	39423564	39408356	39407164	39406400
Portugal	10687938.0	10754436	10816754	10869832	10907700
Slovak Republic	5534060.0	5535710	5537858	5542664	5546616
Spain	41259904.0	42326998	43086706	43729492	44393976
Sweden	9001366.0	9027362	9058028	9090162	9122404
Switzerland	7424242.0	7477648	7525218	7572812	7613252
Turkey	NaN	69506990	70047000	70969000	71891526
United Kingdom	59507558.0	60662818	61013586	61329722	61634728
United States	283886968.0	293634376	295546918	298990198	301827564

Country	2007	2008	2009	2010	2011
Australia	21140840	21541728	21973070	22436288	22719614
Austria	8523504	8555432	8574426	8592394	8617830
Belgium	10806252	10885114	10968858	11055368	9993218
Canada	32957518	33202462	33605666	33615476	34203626
Czech Republic	10522010	10596392	10662330	10699232	10727942
Denmark	5500844	5526250	5558862	5582904	5608092
Finland	5386426	5407394	5429322	5452720	5473720
France	65175958	65541720	66416630	66769860	67197266
Germany	84027480	83887090	83636146	83397302	83278354
Greece	11279386	11319780	11367324	11415306	11419636
Hungary	10574160	10551678	10535850	10514848	10483642
Iceland	302192	309126	314600	315388	316892
Ireland	4237354	4407704	4470366	4502820	4602870
Italy	60825692	61339086	61785290	62105850	62426336
Japan	130880000	130922000	130882000	130760000	131460000
Korea	48224186	48381808	48530426	48668446	49674202
Luxembourg	480790	488384	497330	505320	514442
Mexico	106439280	107447964	108432512	109393818	110332724
Netherlands	16538956	16586652	16658782	16743026	16824634

11.4 Reading data from text files:

In [31]:
```
import pandas as pd
import numpy as np
dataset=pd.read_csv('dataset1.txt')
x=dataset.iloc[: ,0:1].values
y=dataset.iloc[: , -1].values
```

In [32]:
```
print(x)
print(y)
```

```
[['2016\t0.901696\t0.864379\t0.959785\t247']
 ['2015\t0.901896\t0.830358\t0.947688\t256']
 ['2014\t0.753941\t0.716692\t0.823655\t255']
 ['2013\t0.753234\t0.723903\t0.783208\t255']
 ['2012\t0.778848\t0.743273\t0.827198\t256']
 ['2011\t0.719219\t0.671953\t0.775855\t257']
 ['2010\t0.755883\t0.686672\t0.837381\t258']
 ['2009\t0.718968\t0.661376\t0.796495\t256']
 ['2008\t0.683499\t0.625391\t0.802568\t256']
 ['2007\t0.730754\t0.672314\t0.775615\t255']
 ['2006\t0.797153\t0.750131\t0.845594\t255']
 ['2005\t0.805097\t0.740357\t0.857118\t257']
 ['2004\t0.804828\t0.733514\t0.847314\t259']
 ['2003\t0.885766\t0.791766\t0.963670\t255']
 ['2002\t1.060945\t0.953562\t1.165773\t255']
 ['2001\t1.117587\t1.047669\t1.192748\t255']
 ['2000\t1.085899\t0.962649\t1.211827\t255']
 ['1999\t0.939475\t0.848176\t0.998502\t261']]
['2016\t0.901696\t0.864379\t0.959785\t247'
 '2015\t0.901896\t0.830358\t0.947688\t256'
 '2014\t0.753941\t0.716692\t0.823655\t255'
 '2013\t0.753234\t0.723903\t0.783208\t255'
 '2012\t0.778848\t0.743273\t0.827198\t256'
 '2011\t0.719219\t0.671953\t0.775855\t257'
 '2010\t0.755883\t0.686672\t0.837381\t258'
 '2009\t0.718968\t0.661376\t0.796495\t256'
 '2008\t0.683499\t0.625391\t0.802568\t256'
 '2007\t0.730754\t0.672314\t0.775615\t255'
```

Chapter-12

12.1 OpenCV Library:

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modelling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields − **Image**

Processing − It focuses on image manipulation.

Pattern Recognition − It explains various techniques to classify patterns.

Photogrammetry − It is concerned with obtaining accurate measurements from images.

12.2 Applications of Computer Vision

Here we have listed down some of major domains where Computer Vision is heavily used.

- **Robotics Application** ○ Localization − Determine robot location automatically ○ Navigation ○ Obstacles avoidance ○ Assembly (peg-in-hole, welding, painting) ○ Manipulation (e.g. PUMA robot manipulator)
  - ○ Human Robot Interaction (HRI) − Intelligent robotics to interact with and serve people
- **Medicine Application**
  - ○ Classification and detection (e.g. lesion or cells classification and tumour detection)
  - ○ 2D/3D segmentation
  - ○ 3D human organ reconstruction (MRI or ultrasound) ○ Vision-guided robotics surgery
- **Industrial Automation Application** ○ Industrial inspection (defect detection) ○ Assembly
  - ○ Barcode and package label reading
  - ○ Object sorting
  - ○ Document understanding (e.g. OCR)
- **Security Application** ○ Biometrics (iris, finger print, face recognition)

- Surveillance − Detecting certain suspicious activities or behaviours ○ Features of OpenCV Library ○ Using OpenCV library, you can − ○ Read and write images ○ Capture and save videos ○ Process images (filter, transform) ○ Perform feature detection ○ Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyse the video, i.e., estimate the motion in it, subtract the background, and track objects in it.
- OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc.

12.3 Reading Images with OpenCv:

In [2]:
```
#opencv is a python library which is used to deal with the image processing part
#it handles the image data and manipulates it accordingly
import cv2
image=cv2.imread('D:/Pictures/2018-10-04/037.jpg')
cv2.imshow('image1',image)

```

Figure 21: Image read output

12.4 Resizing image:

In [3]:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

import image as images
path1='D:/Pictures/2018-10-04/037.jpg'

img1=cv2.imread(path1,1)
half=cv2.resize(img1,(0,0),fx=0.5, fy=0.5)
bigger=cv2.resize(img1,(1050,1610))
streach=cv2.resize(img1,(500,600), interpolation=cv2.INTER_NEAREST)

titles=['original','half','bigger','streached']
img=[img1,half,bigger,streach]
count=4

for i in range(count):
    plt.subplot(2,2,i+1)
    plt.title(titles[i])
    plt.imshow(img[i])
    plt.show()
```

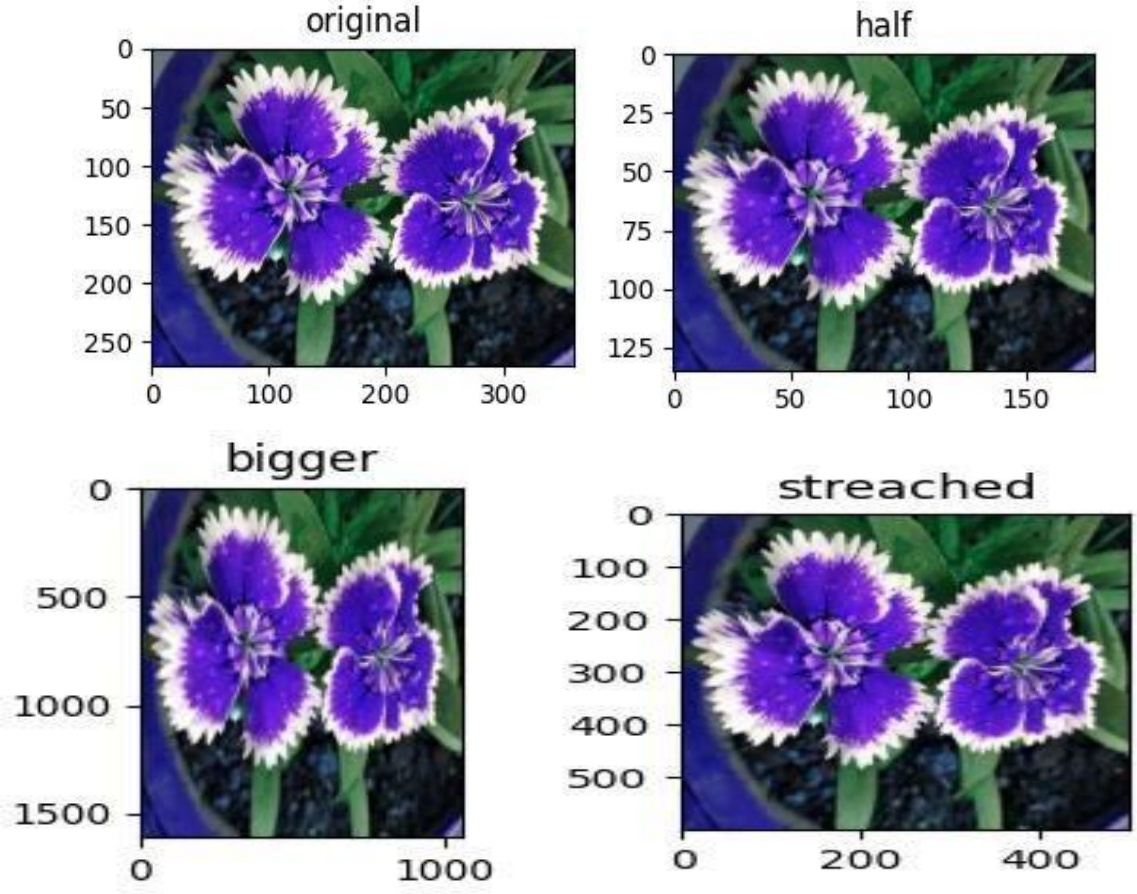


Figure 21: Image rescaling

12.5 Conversion of color image to gray:

In [*]:

```
import cv2

# reading the image
image_org=cv2.imread('D:/Pictures/2018-10-04/037.jpg')
image = cv2.imread('D:/Pictures/2018-10-04/037.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imshow('Original Image', image_org)
cv2.imshow('grey_image', image)

cv2.waitKey(0)
cv2.destroyAllWindows()


```

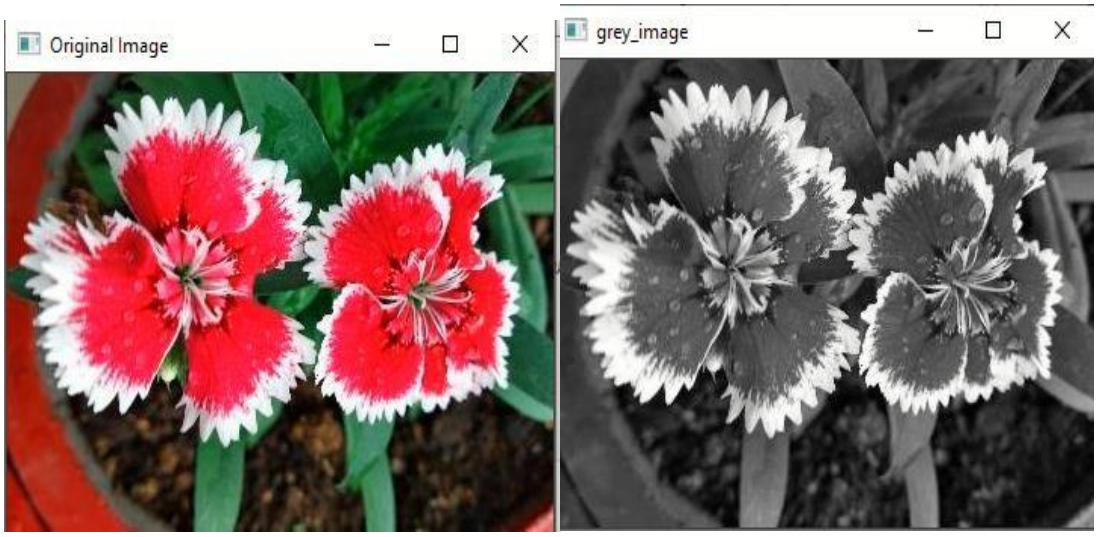


Figure 22: RGB Image conversion to grey

12.6 Reading Image from Android Phone:

In [ ]:

```
# Import essential libraries
import requests
import cv2
import numpy as np
import imutils

# Replace the below URL with your own. Make sure to add "/shot.jpg" at last.
url = "http://192.168.1.26:8080/shot.jpg"

# While loop to continuously fetching data from the Url
while True:
    img_resp = requests.get(url)
    img_arr = np.array(bytearray(img_resp.content), dtype=np.uint8)
    img = cv2.imdecode(img_arr, -1)
    img = imutils.resize(img, width=800, height=1200)
    cv2.imshow("Android_cam", img)

    # Press Esc key to exit
    if cv2.waitKey(1) == 27:
        break

cv2.destroyAllWindows()
```

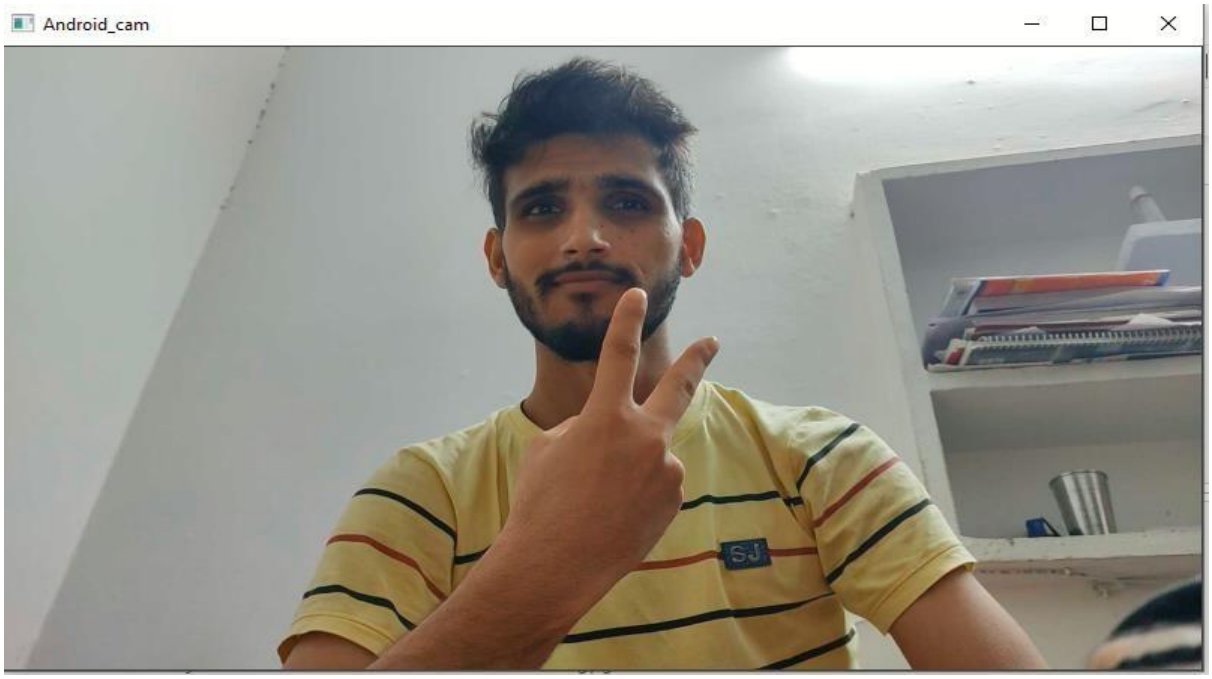


Figure 23: Image read from Android-1

Figure 24: Image read from Android-2

12.7 A Mini project on Image converter:

This mini project using openCV library is capable of converting images from one color code to another (i.e gray, RGB) and in adddition to that it can also be used to convert PNG images to JPG and vice-versa.

In addition to that it can handle more operations like reshping and resizing of images and many more upon which I am still working.

In [ ]:

```
from tkinter import *
import tkinter as tk
# To get the dialog box to open when required
from tkinter import filedialog
from tkinter import filedialog as fd
from tkinter import messagebox
import numpy as np
import cv2
# Loading Python Imaging Library
from PIL import ImageTk, Image

# Create a window
root = Tk()


# Set Title as Image Loader
root.title("PhotoEdit")
# To set background  of root

# Add image file
bg = PhotoImage(file="a.jpg")
#
# Create Canvas
canvas1 = Canvas(root, width=400,
                 height=400)

canvas1.pack(fill="both", expand=True)

# Display image
canvas1.create_image(0, 0, image=bg,
                     anchor="nw")

# Add Text
canvas1.create_text(200, 250, text="Welcome")

image1= tk.PhotoImage(file="a.jpg")
label_for_image= Label(root, image=image1)
label_for_image.pack()
```

```
label_for_image= Label(root, image=image1)
label_for_image.pack()

# Set the resolution of window
root.geometry("900x300+100+100")

# Allow Window to be resizable
root.resizable(width=True, height=True)
# Specify Grid
Grid.rowconfigure(root, 0, weight=0)
Grid.columnconfigure(root, 0, weight=1)

# Grid.rowconfigure(root, 1, weight=1)
Grid.columnconfigure(root, 1, weight=1)

# Grid.rowconfigure(root, 2, weight=0)
Grid.columnconfigure(root, 2, weight=1)

# Grid.rowconfigure(root, 3, weight=0)
Grid.columnconfigure(root, 3, weight=1)

# Grid.rowconfigure(root, 4, weight=0)
Grid.columnconfigure(root, 4, weight=1)

# Grid.rowconfigure(root, 5, weight=0)
Grid.columnconfigure(root, 5, weight=1)

# Grid.rowconfigure(root, 6, weight=0)
Grid.columnconfigure(root, 6, weight=1)


def open_img():
    global x
    global img
    # Select the Imagename from a folder
    x = openfilename()

    # opens the image
    img = Image.open(x)
```

```
def openfilename():
    global filename
    # open file dialog box to select image
    # The dialogue box has a title "Open"
    filename = filedialog.askopenfilename(title='"pen')
    return filename


def png_convert():
    global img1

    if filename.endswith(".jpg"):

        img1 = Image.open(filename)

        # after converting the image save to desired
        # location with the Extersion .png
        export_filename = fd.asksaveasfilename(defaultextension=".png")
        img1.save(export_filename)

        # displaying the Messaging box with the Success
        messagebox.showinfo("success ", "your Image converted to Png")
    else:

        # if Image select is not with the Format of .jpg
        # then display the Error
        Label_2 = Label(root, text="Error!", width=20,
                        fg="red", font=("bold", 15))
        Label_2.place(x=80, y=280)
        messagebox.showerror("Something Went Wrong...try again later!")


def jpg_converter():
    global img1

    if filename.endswith(".png"):
        img1 = Image.open(filename)
        export_filename = fd.asksaveasfilename(defaultextension=".jpg")
        img1.save(export_filename)
```

```
    print('Height : ', new[0])
    print('Width : ', new[1])
    print('layers : ', new[2])


# Create buttons and place it into the window using grid layout
btn1 = Button(root, text='open image', command=open_img, bg='#68FF33', bd=10, activebackground='blue').grid(
    row=0,
    column=0, sticky=NSEW)
btn2 = Button(root, text='Convert to png', command=png_convert, bg='#68FF33', bd=10,
              activebackground='blue').grid(row=0,
                                            column=1, sticky=NSEW)

btn2a = Button(root, text='Convert to jpg', command=jpg_converter, bg='#68FF33', bd=10,
               activebackground='blue').grid(row=0,
                                             column=2, sticky=NSEW)

btn3 = Button(root, text='Convert to gray', command=gray_converter, bg='#68FF33', bd=10,
              activebackground='blue').grid(row=0,
                                            column=3, sticky=NSEW)
btn4 = Button(root, text='Resize to half', command=half_resizer, bg='#68FF33', bd=10,
              activebackground='blue').grid(row=0,
                                            column=4, sticky=NSEW)
btn5 = Button(root, text='Resize to double', command=size_doubler, bg='#68FF33', bd=10,
              activebackground='blue').grid(row=0,
                                            column=5, sticky=NSEW)
btn6 = Button(root, text='Pixel Read', command=pixel_read, bg='#68FF33', bd=10, activebackground='blue').grid(
    row=0,
    column=6, sticky=NSEW)
root.mainloop()

```

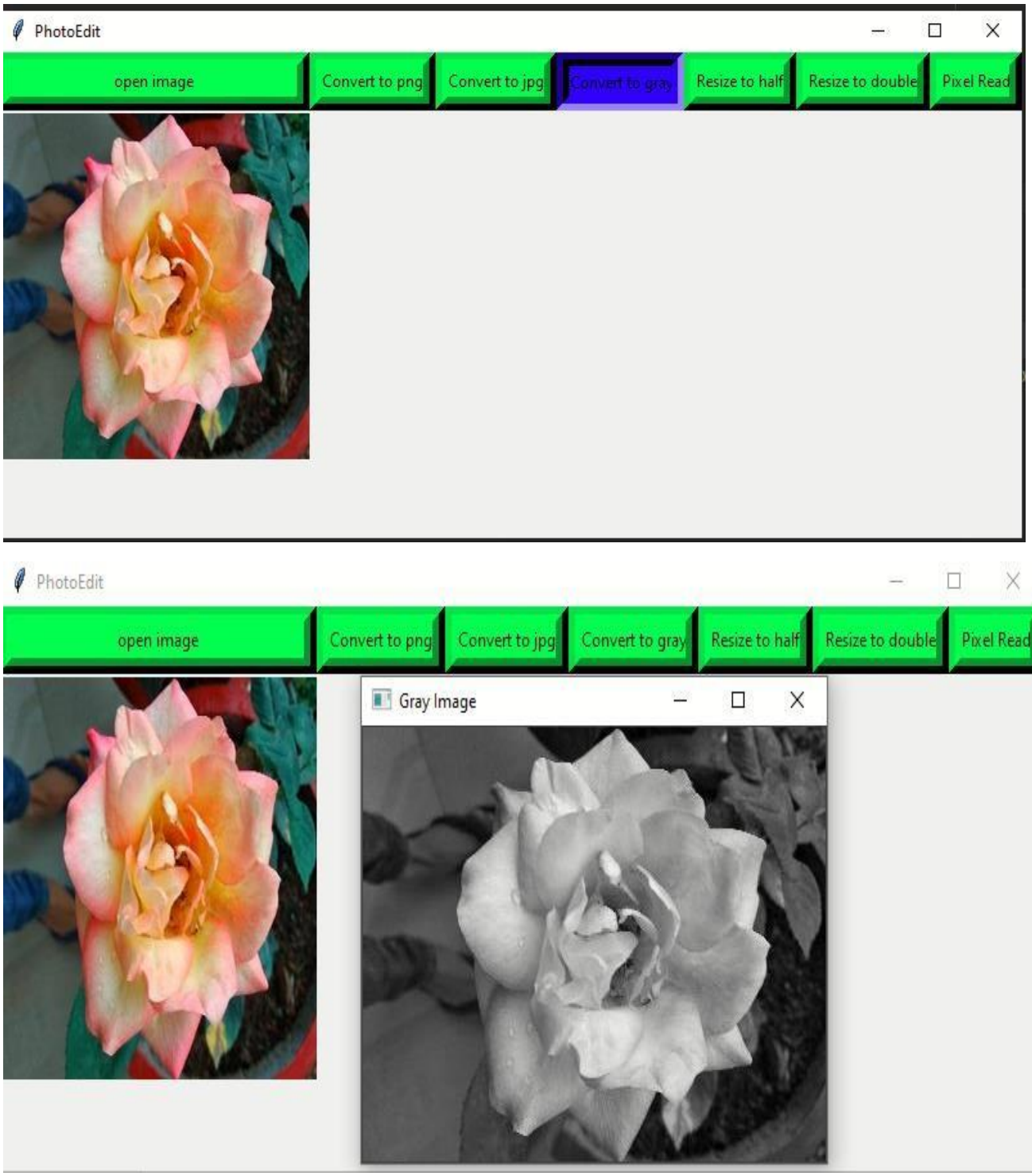


Figure 25: Sample outputs of image converter

Chapter-13

13.1 Introduction to Machine Learning:

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyse the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future.

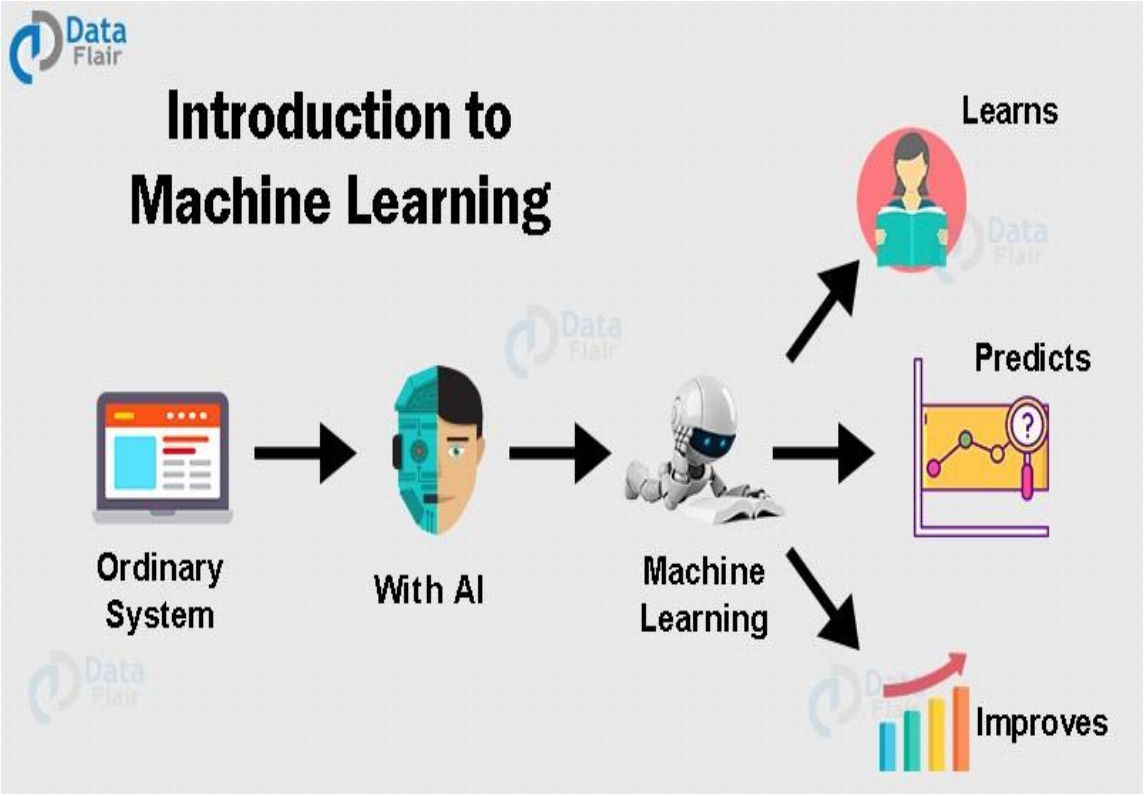


Figure 26: Machine learning process overview

13.2 Machine Learning – Types of Techniques:

There are three types of ML Techniques:

13.2.1 Supervised Learning:-Supervised learning as the name suggests getting supervised by someone. It is a learning in which the machine uses data which is already tagged with the correct answer. After that, the machine is provided with a new set of data. With the help of supervised learning, the algorithm analyses the training data (set of training examples) and produces a correct outcome from labelled data. Here the machine has already learned the things from previous data. So, now is the time it uses the learning wisely.

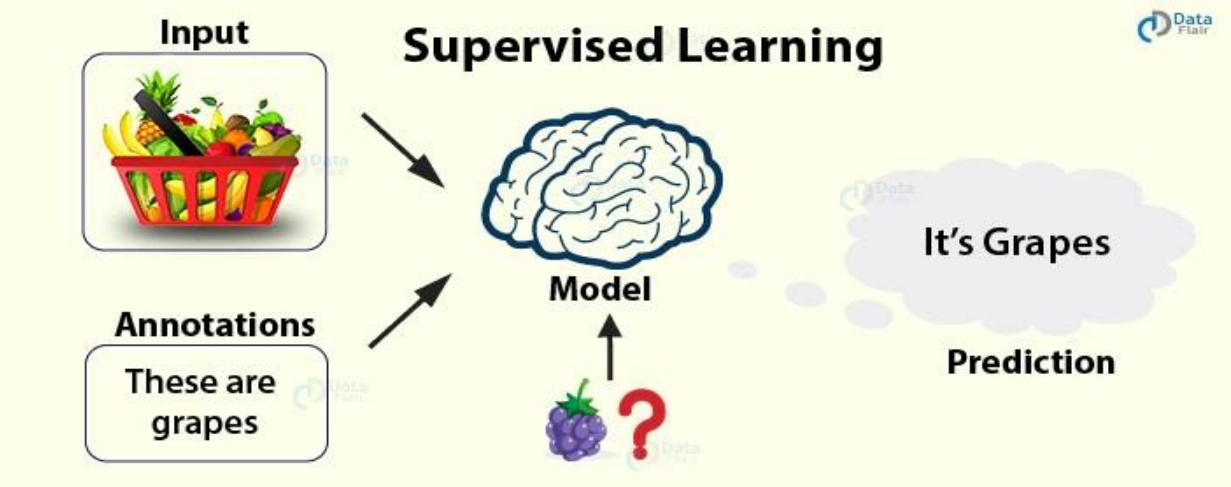


Figure 27: Supervised learning process overview

For example – if we take a fruit basket, the machine will first classify the fruit with its shape and colour and would confirm the fruit name. If one searches for grapes, then machine learning from its training data (basket containing fruits) will use the prior knowledge.
It will then apply the knowledge to test data and will then provide you with the results.

13.2.2 Unsupervised Learning:- In unsupervised learning, the training of the machine is done using the information which is neither classified nor labelled. The machine learning algorithm acts on information without guidance. It groups unsorted information according to similarities, patterns, and differences without any prior training or supervision. Since there is no training given to the machine, the machine itself finds the hidden structure in unlabelled data and interprets it.
For example – a wooden stick with a cap can be a pen and with no cap a pencil. With no learning and no training, the machine tries to interpret itself.

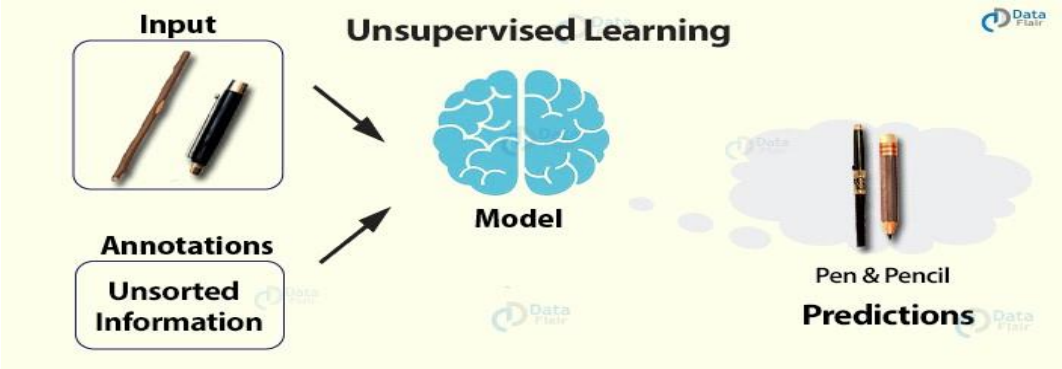


Figure 28: Unsupervised learning process overview

13.2.3 Reinforcement Learning:- Reinforcement learning is a very interesting kind of learning. There's no answer key which can tell what's right. But, the reinforcement learning agent still decides how to act to perform its task. This machine learning technique is all about taking actions that are suitable and maximize the reward in a particular situation. It is when the learner receives rewards and punishments for their actions.

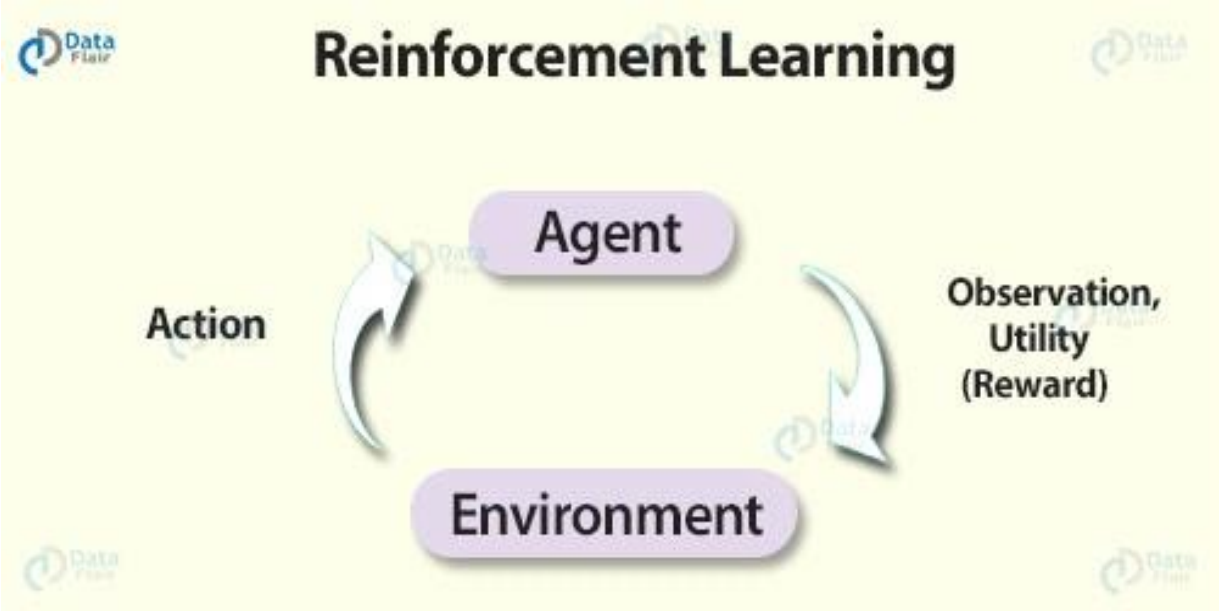


Figure 29: Reinforcement learning process overview

For example – In a given scenario, the reward could be utility and the agent could be told to receive as much utility as possible in order to "win".

13.3 Advantages of Machine learning:

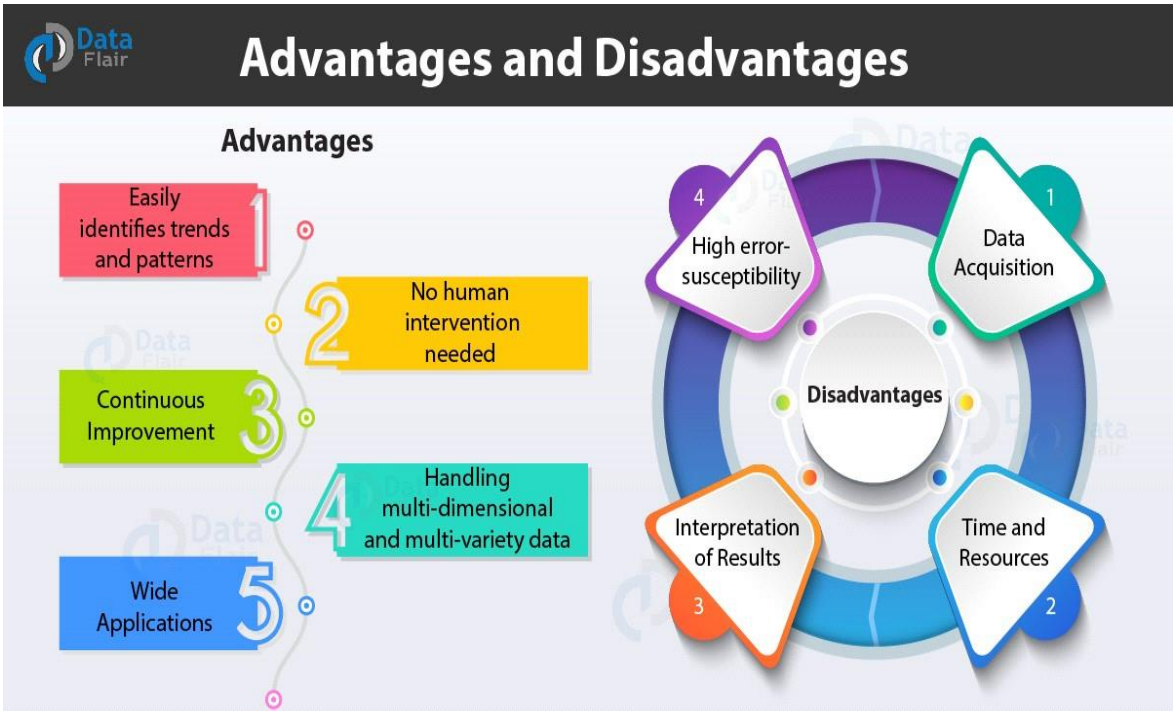


Figure 30: Advantages of Machine learning

13.3.1 Easily identifies trends and patterns:-

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an ecommerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

13.3.2 No human intervention needed (automation):-

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is antivirus software's they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

13.3.3 Continuous Improvement:-

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

13.3.4 Handling multi-dimensional and multi-variety data:-

Machine Learning algorithms are good at handling data that are multidimensional and multi-variety, and they can do this in dynamic or uncertain environments.

13.3.5 Wide Applications:-

You could be an e-trailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

13.4 Disadvantages of Machine Learning:

With all those advantages to its powerfulness and popularity, Machine Learning isn't perfect. The following factors serve to limit it:

13.4.1 .Data Acquisition:-

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

13.4.2 Time and Resources:-

ML needs enough time to let the algorithms learn and develop enough to fulfil their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

13.4.3 Interpretation of Results:-

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

13.4.4 High error-susceptibility:-

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

13.5 Introduction to Artificial Intelligence

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems. The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

Figure 31: Artificial intelligence overview

The intelligence is intangible. It is composed of

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector. Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

13.6 Applications of AI

· **Gaming –** AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess, river crossing, N-queens problems and etc.

Natural Language Processing – Interact with the computer that understands natural language spoken by humans.

Expert Systems − Machine or software provide explanation and advice to the users.

Vision Systems − Systems understand, explain, and describe visual input on the computer.

Speech Recognition − There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant.

Handwriting Recognition − The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text.

Intelligent Robots − Robots are able to perform the instructions given by a human. Major Goals
Knowledge reasoning
Planning
Machine Learning
Natural Language Processing
Computer Vision
Robotics

13.7 Advantages of AI

13.7.1 Reduction in Human Error: The phrase "**human error**" was born because humans make mistakes from time to time. Computers, however, do not make these mistakes if they are programmed properly. With Artificial intelligence, the decisions are taken from the previously gathered information applying a certain set of algorithms. So errors are reduced and the chance of reaching accuracy with a greater degree of precision is a possibility.

Example: In Weather Forecasting using AI they have reduced the majority of human error.

13.7.2 Takes risks instead of Humans:

This is one of the biggest advantages of Artificial intelligence. We can overcome many risky limitations of humans by developing an AI Robot which in turn can do the risky things for us. Let it be going to mars, defuse a bomb, explore the deepest parts of oceans, mining for coal and oil, it can be used effectively in any kind of natural or man-made disasters.

Example: Have you heard about the **Chernoby**l nuclear power plant explosion in Ukraine? At that time there were no AI-powered robots that can help us to minimize the effect of radiation by controlling the fire in early stages, as any human went close to the core was dead in a matter of minutes. They eventually poured sand and boron from helicopters from a mere distance.

AI Robots can be used in such situations where intervention can be hazardous.

13.7.3 Available 24x7:

An Average human will work for 4–6 hours a day excluding the breaks. Humans are built in such a way to get some time out for refreshing themselves and get ready for a new day of work and they even have weekly offed to stay intact with their work-life and personal life. But using AI we can make machines work 24x7 without any breaks and they don't even get bored, unlike humans.

Example: Educational Institutes and Helpline centres are getting many queries and issues which can be handled effectively using AI.

13.7.4 Helping in Repetitive Jobs:

In our day-to-day work, we will be performing many repetitive works like sending a thanking mail, verifying certain documents for errors and many more things. Using artificial intelligence we can productively automate these mundane tasks and can even remove "**boring**" tasks for humans and free them up to be increasingly creative.
Example: In banks, we often see many verifications of documents to get a loan which is a repetitive task for the owner of the bank. Using AI Cognitive Automation the owner can speed up the process of verifying the documents by which both the customers and the owner will be benefited.

13.7.5 Digital Assistance:

Some of the highly advanced organizations use digital assistants to interact with users which saves the need for human resources. The digital assistants also used in many websites to provide things that users want. We can chat with them about what we are looking for. Some chatbots are designed in such a way that it's become hard to determine that we're chatting with a chatbot or a human being.
Example: We all know that organizations have a customer support team that needs to clarify the doubts and queries of the customers. Using AI the organizations can set up a Voice bot or Chatbot which can help customers with all their queries. We can see many organizations already started using them on their websites and mobile applications.

13.7.6 Faster Decisions:

Using AI alongside other technologies we can make machines take decisions faster than a human and carry out actions quicker. While taking a decision human will analyse many factors both emotionally and practically but AI-powered machine works on what it is programmed and delivers the results in a faster way.
Example: We all have played Chess games in Windows. It is nearly impossible to beat CPU in the hard mode because of the AI behind that game. It will take the best possible step in a very short time according to the algorithms used behind it.

13.7.7 Daily Applications:

Daily applications such as Apple's **Siri**, Window's **Cortana**, Google's **OK Google** are frequently used in our daily routine whether it is for searching a location, taking a selfie, making a phone call, replying to a mail and many more.
Example: Around 20 years ago, when we are planning to go somewhere we used to ask a person who already went there for the directions. But now all we have to do is say "**OK Google** where is Visakhapatnam". It will show you Visakhapatnam's location on google map and the best path between you and Visakhapatnam.

13.7.8 New Inventions:

AI is powering many inventions in almost every domain which will help humans solve the majority of complex problems.
Example: Recently doctors can predict breast cancer in the woman at earlier stages using advanced AI-based technologies.

13.8 Disadvantages of AI

13.8.1 High Costs of Creation:

As AI is updating every day the hardware and software need to get updated with time to meet the latest requirements. Machines need repairing and maintenance which need plenty of costs. It's creation requires huge costs as they are very complex machines.

13.8.2 Making Humans Lazy:

AI is making humans lazy with its applications automating the majority of the work. Humans tend to get **addicted** to these inventions which can cause a problem to future generations.

13.8.3 Unemployment:

As AI is replacing the majority of the repetitive tasks and other works with robots, human interference is becoming less which will cause a major problem in the employment standards. Every organization is looking to replace the minimum qualified individuals with AI robots which can do similar work with more efficiency.

13.8.4 No Emotions:

There is no doubt that machines are much better when it comes to working efficiently but they cannot replace the human connection that makes the team. Machines cannot develop a bond with humans which is an essential attribute when comes to Team Management.

13.8.5 Lacking Out of Box Thinking:

Machines can perform only those tasks which they are designed or programmed to do, anything out of that they tend to crash or give irrelevant outputs which could be a major backdrop.

CHAPTER-14

14.1 Introduction to Automation in Attendance system:

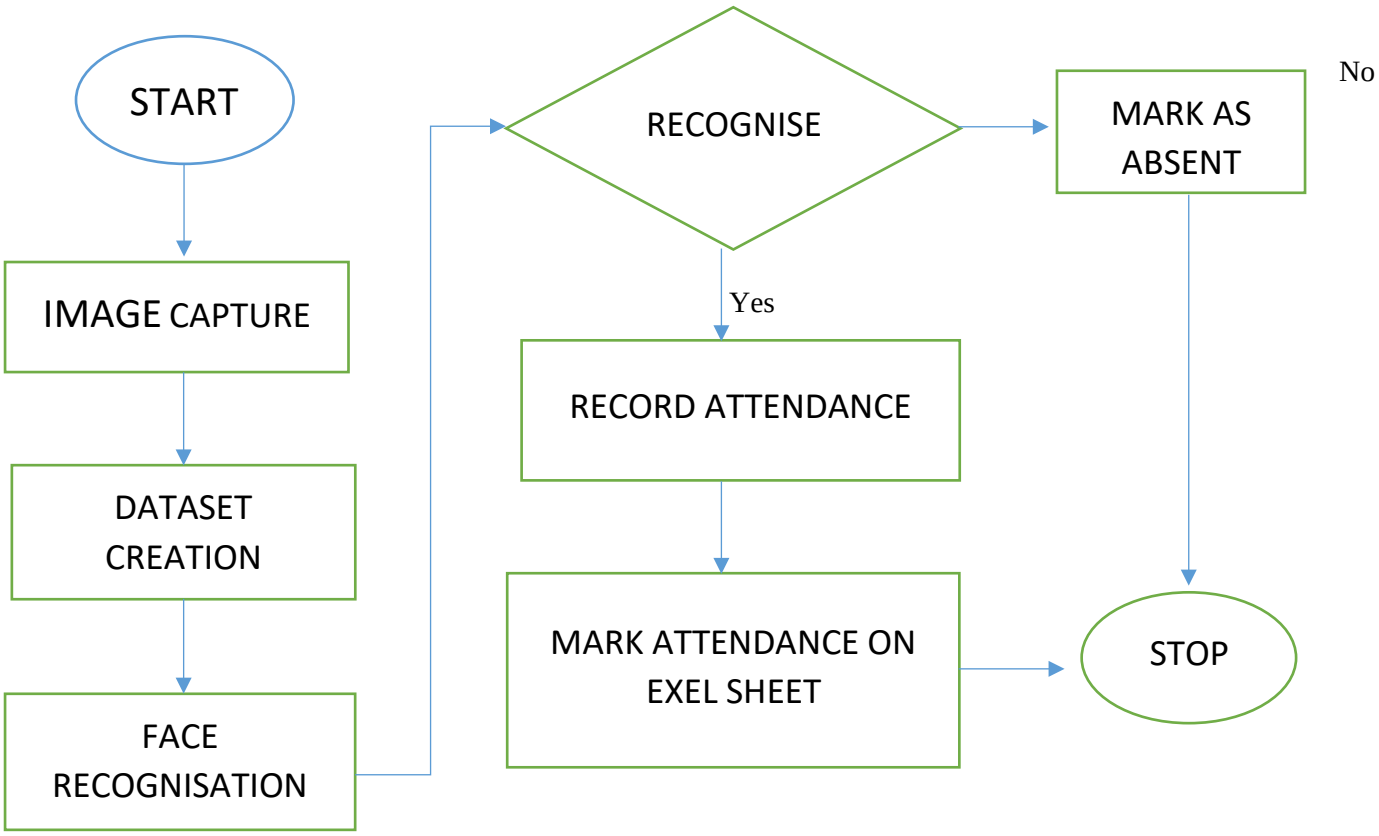
Attendance is prime important for both the teacher and student of an educational organization. It is very important to keep record of the attendance. The problem arises when we think about the traditional process of taking attendance in class room. Calling name or roll number of the student for attendance is not only a problem of time consumption but also it needs energy. An automatic attendance system can solve all above problems.

There are some automatic attendances making system which are currently used by much institution. One of such system is biometric technique. Although it is automatic and a step ahead of traditional method it fails to meet the time constraint. The student has to wait in queue for giving attendance, which is time taking.

This project introduces an involuntary attendance marking system, devoid of any kind of interference with the normal teaching procedure. The system can be also implemented during exam sessions or in other teaching activities where attendance is highly essential. This system eliminates classical student identification such as calling name of the student, or checking respective identification cards of the student, which can not only interfere with the ongoing teaching process, but also can be stressful for students during examination sessions.

Figure32: Students waiting in a que for biometric attendance

14.2 Flow Chart



14.3 Digital Image Processing.

Digital Image Processing is the processing of images which are digital in nature by a digital computer. Digital image processing techniques are motivated by three major applications mainly:

Improvement of pictorial information for human perception

Image processing for autonomous machine application Efficient

storage and transmission.

14.3.1 Human Perception

This application employs methods capable of enhancing pictorial information for human interpretation and analysis. Typical applications include; noise filtering, content enhancement mainly contrast enhancement or deblurring and remote sensing.

14.3.2 Machine Vision Applications

In this, the interest is on the procedures for extraction of image information suitable for computer processing. Typical applications include;

Industrial machine vision for product assembly and inspection.

Automated target detection and tracking.

Finger print recognition.

Machine processing of aerial and satellite imagery for weather prediction and crop assessment.

Facial detection and recognition fall within the machine vision application of digital image processing.

13.3.3 Image Representation in a Digital Computer.

An image is a 2-Dimensional light intensity function

$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{r}(\mathbf{x}, \mathbf{y}) \times \mathbf{i}(\mathbf{x}, \mathbf{y})$

$\mathbf{r}(\mathbf{x}, \mathbf{y})$ is the reflectivity of the surface of the corresponding image point.

$\mathbf{i}(\mathbf{x}, \mathbf{y})$ Represents the intensity of the incident light.

A digital image f (x, y) is discretized both in spatial co-ordinates by grids and in brightness by quantization. Effectively, the image can be represented as a matrix whose row, column indices specify a point in the image and the element value identifies gray level value at that point. These elements are referred to as pixels or pels.

Typically following image processing applications, the image size which is used is $\mathbf{256} \times$

256, elements, **640 × 480** pixels or **1024 × 1024** pixels. Quantization of these matrix pixels is done at 8 bits for black and white images and 24 bits for coloured images (because of the three colour planes Red, Green and Blue each at 8 bits).

14.3.4 Steps in Digital Image Processing

Digital image processing involves the following basic tasks;

Image Acquisition - An imaging sensor and the capability to digitize the signal produced by the sensor.

Pre-processing – Enhances the image quality, filtering, contrast enhancement etc.

Segmentation – Partitions an input image into constituent parts of objects.

Description/feature Selection – extracts the description of image objects suitable for further computer processing.

Recognition and Interpretation – Assigning a label to the object based on the information provided by its descriptor. Interpretation assigns meaning to a set of labelled objects.

Knowledge Base – This helps for efficient processing as well as inter module cooperation.

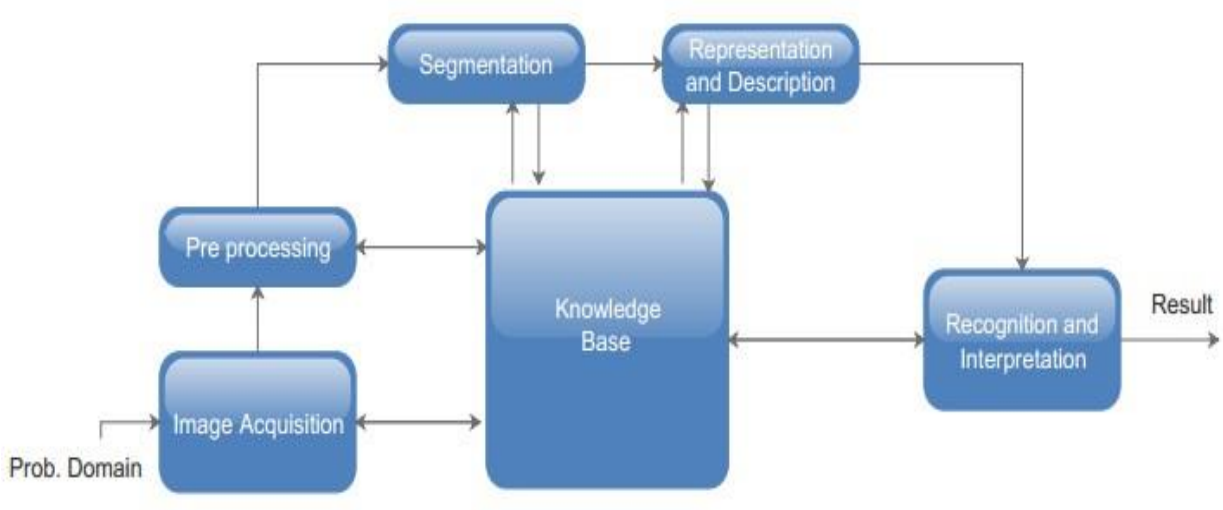


Figure 33: A diagram showing the steps in digital image processing

14.4 Definition of Terms and History

Face Detection

Face detection is the process of identifying and locating all the present faces in a single image or video regardless of their position, scale, orientation, age and expression. Furthermore, the detection should be irrespective of extraneous illumination conditions and the image and video content.

14.4.1 Face Recognition

Face Recognition is a visual pattern recognition problem, where the face, represented as a threedimensional object that is subject to varying illumination, pose and other factors, needs to be identified based on acquired images.

Face Recognition is therefore simply the task of identifying an already detected face as a known or unknown face and in more advanced cases telling exactly whose face it is.

14.4.2 Difference between Face Detection and Face Recognition

Face detection answers the question, where is the face? It identifies an object as a "face" and locates it in the input image. Face Recognition on the other hand answers the question who is this? Or whose face is it? It decides if the detected face is someone known or unknown based on the database of faces it uses to validate this input image.

14.4.3 Face Detection

A face Detector has to tell whether an image of arbitrary size contains a human face and if so, where it is. Face detection can be performed based on several cues: skin color (for faces in color images and videos, motion (for faces in videos), facial/head shape, facial appearance or a combination of these parameters. Most face detection algorithms are appearance based without using other cues.

An input image is scanned at all possible locations and scales by a sub window. Face detection is posed as classifying the pattern in the sub window either as a face or a non-face. The face/non-face classifier is learned from face and non-face training examples using statistical learning methods.

Most modern algorithms are based on the Viola Jones object detection framework, which is based on Haar-Cascades.

14.5 Haar – Cascades

Haar like features are rectangular patterns in data. A cascade is a series of "Haar-like features" that are combined to form a classifier. A Haar wavelet is a mathematical function that produces square wave output.

(a) (b) (c)

Figure above shows Haar like features, the background of a template like (b) is painted gray to highlight the pattern's support. Only those pixels marked in black or white are used when the corresponding feature is calculated.

Since no objective distribution can describe the actual prior probability for a given image to have a face, the algorithm must minimize both the false negative and false positive rates in order to achieve an acceptable performance. This then requires an accurate numerical description of what sets human faces apart from other objects. Characteristics that define a face can be extracted from the images with a remarkable committee learning algorithm called Adaboost. Adaboost (Adaptive boost) relies on a committee of weak classifiers that combine to form a strong one through a voting mechanism. A classifier is weak if, in general, it cannot meet a predefined classification target in error terms. The operational algorithm to be used must also work with a reasonable computational budget. Such techniques as the integral image and attentional cascades have made the Viola-Jones algorithm highly efficient: fed with a real time image sequence generated from a standard webcam or camera, it performs well on a standard PC.

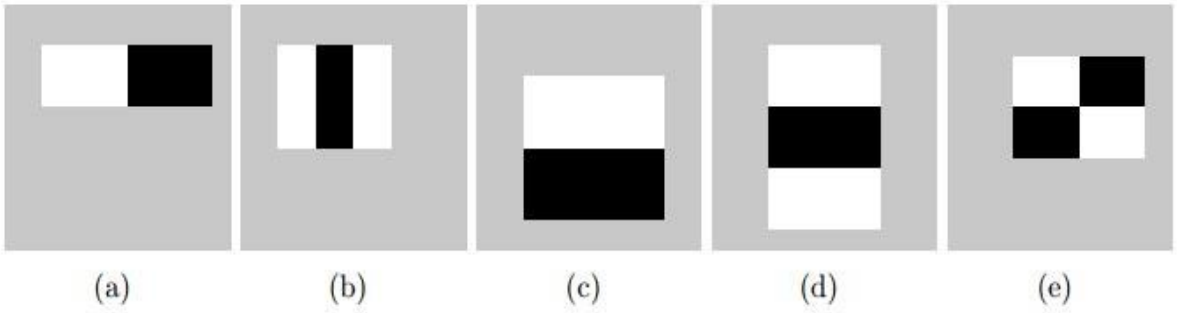


Figure:34 Haar-like features with different sizes and orientation The size and position of a pattern's support can vary provided its black and white rectangles have the same dimension, border each other and keep their relative positions. Thanks to this constraint, the number of features one can draw from an image is somewhat manageable: a 24 × 24 image, for instance, has 43200, 27600, 43200, 27600 and 20736 features of category (a), (b), (c), (d) and (e) respectively as shown in figure 34, hence 162336 features in all.

In practice, five patterns are considered. The derived features are assumed to hold all the information needed to characterize a face. Since faces are large and regular by nature, the use of Haar-like patterns seems justified.

14.6 How the Haar – like Features Work

A scale is chosen for the features say 24×24 pixels. This is then slid across the image. The average pixel values under the white area and the black area are then computed. If the difference between the areas is above some threshold then the feature matches.

In face detection, since the eyes are of different colour tone from the nose, the Haar feature (b) from Figure 5 can be scaled to fit that area as shown below,

Figure:35 How the Haar like feature of figure used to scale the eyes

One Haar feature is however not enough as there are several features that could match it (like the zip drive and white areas at the background of the image of figure). A single classifier therefore isn't enough to match all the features of a face, it is called a "weak classifier."

Haar cascades, the basis of Viola Jones detection framework therefore consists of a series of weak classifiers whose accuracy is at least 50% correct. If an area passes a single classifier, it moves to the next weak classifier and so on, otherwise, the area does not match.

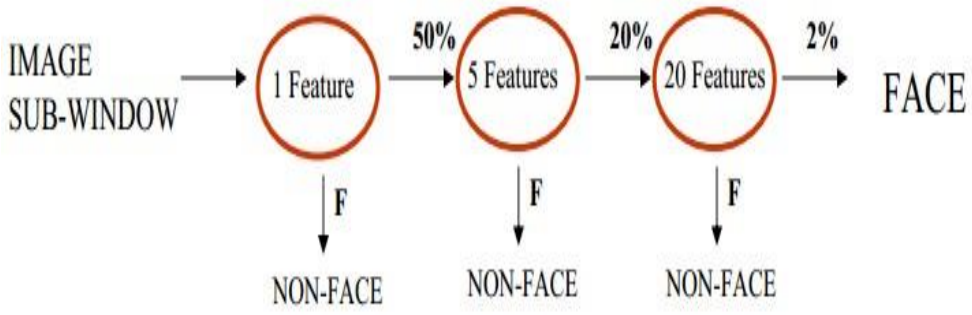


Figure 36: several classifiers combined to enhance

14.7 Cascaded Classifier

From figure 7, a 1 feature classifier achieves 100% face detection rate and about 50% false positive rate. A 5-feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative). A 20-feature classifier achieves 100% detection rate with 10% false positive rate (2% cumulative). Combining several weak classifiers improves the accuracy of detection.

A training algorithm called Adaboost, short for adaptive boosting, which had no application before Haar cascades, was utilized to combine a series of weak classifiers in to a strong classifier. Adaboost tries out multiple weak classifiers over several rounds, selecting the best weak classifier in each round and combining the best weak classifier to create a strong classifier. Adaboost can use classifiers that are consistently wrong by reversing their decision. In the design and development, it can take weeks of processing time to determine the final cascade sequence.

After the final cascade had been constructed, there was a need for a way to quickly compute the Haar features i.e. compute the differences in the two areas. The integral image was instrumental in this.

14.8 Integral Image

The Integral image also known as the “summed area table” developed in 1984 came in to widespread use in 2001 with the Haar cascades. A summed area table is created in a single pass. This makes the Haar cascades fast, since the sum of any region in the image can be computed using a single formula.

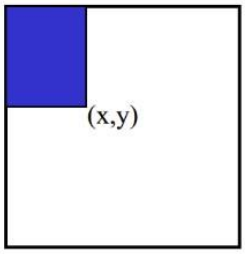


Figure 37: Pixel Coordinates of an integral image

The integral image computes a value at each pixel (x, y) as is shown in figure 37, that is the sum of the pixel values above and to the left of (x, y), inclusive. This can quickly be computed in one pass through the image.

Let A, B, C D be the values of the integral image at the corners of a rectangle as shown in figure

The sum of original image values within the rectangle can be computed.

$Sum = A - B - C + D$

Only three additions are required for any size of rectangle. This face detection approach minimizes computation time while achieving high detection accuracy. It is now used in many areas of computer vision.

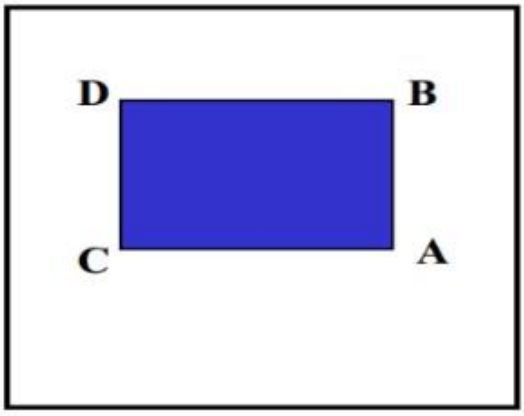


Figure 38: Values of the integral image on a rectangle

14.9 Methodology and Design

General Overview

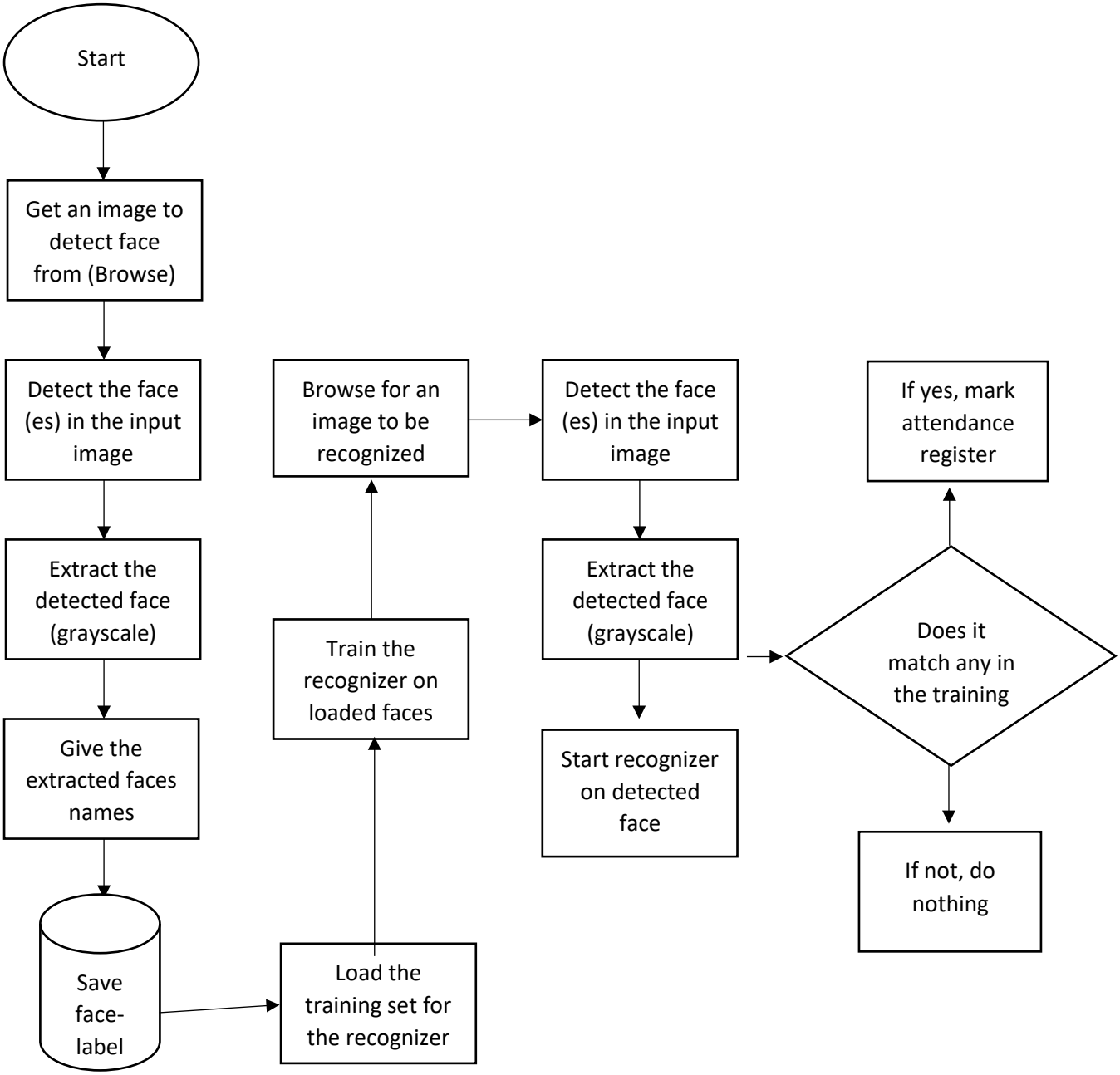


Figure 39: Series of Components

From Figure above , it can be observed that most of the components utilized are similar;(the Image acquisition component for browsing for input images, the face detector and the faces database for storing the face label pairs) only that they are employed at the different stages of the face recognition process.

14.9.1 Training Set Manager Sub System

The logical design of the training set management sub-system is going to consist of an image acquisition component, a face detection component and a training set management component. Together, these components interact with the faces database in order to manage the training set.

These are going to be implemented in a windows application form.

14.9.2 Face Recognizer Sub System

The logical design of the Face Recognizer will consist of the image acquisition component, face recognizer and face detection component all working with the faces database.

In this the image acquisition, and face detection component are the same as those in the Training set manager sub system as the functionality is the same. The only difference is the face recognizer component and its user interface controls.

This will load the training set again so that it trains the recognizer on the faces added and show the calculated Eigen faces and average face. It should then show the recognized face in a picture box.

14.10 System Architecture

The figure below shows the logical design and implementation of the three desktop subsystems.

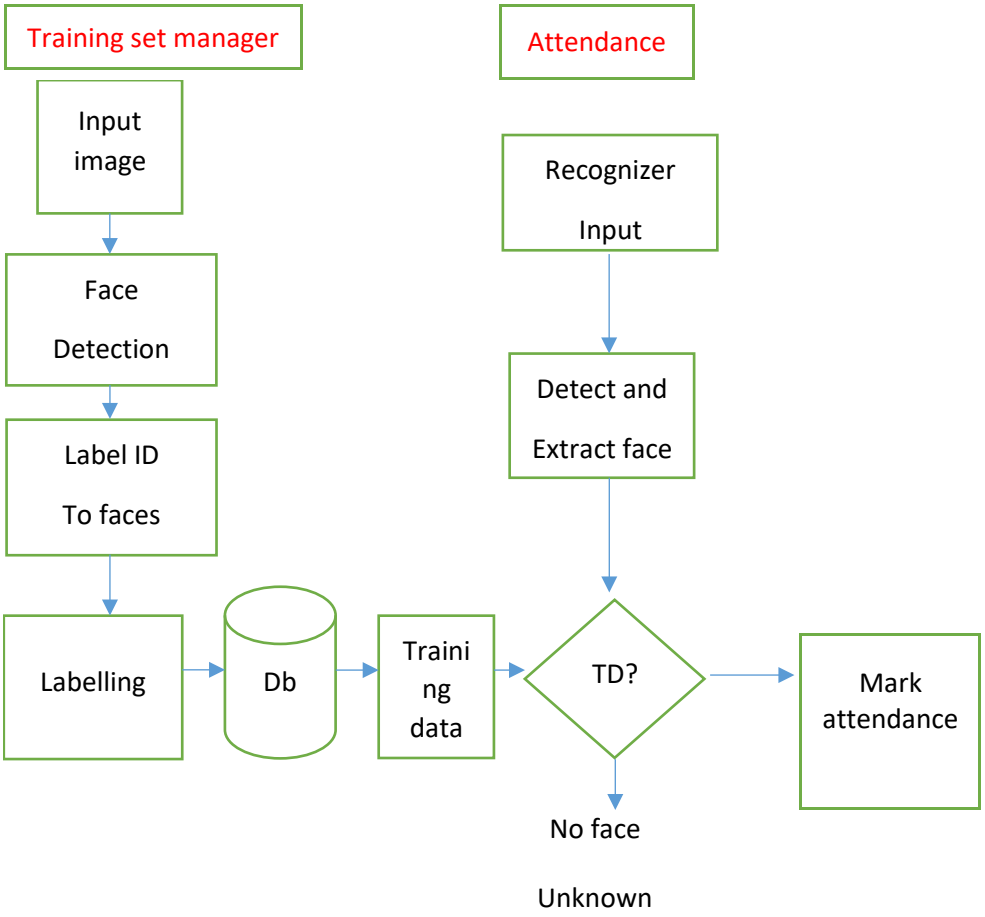
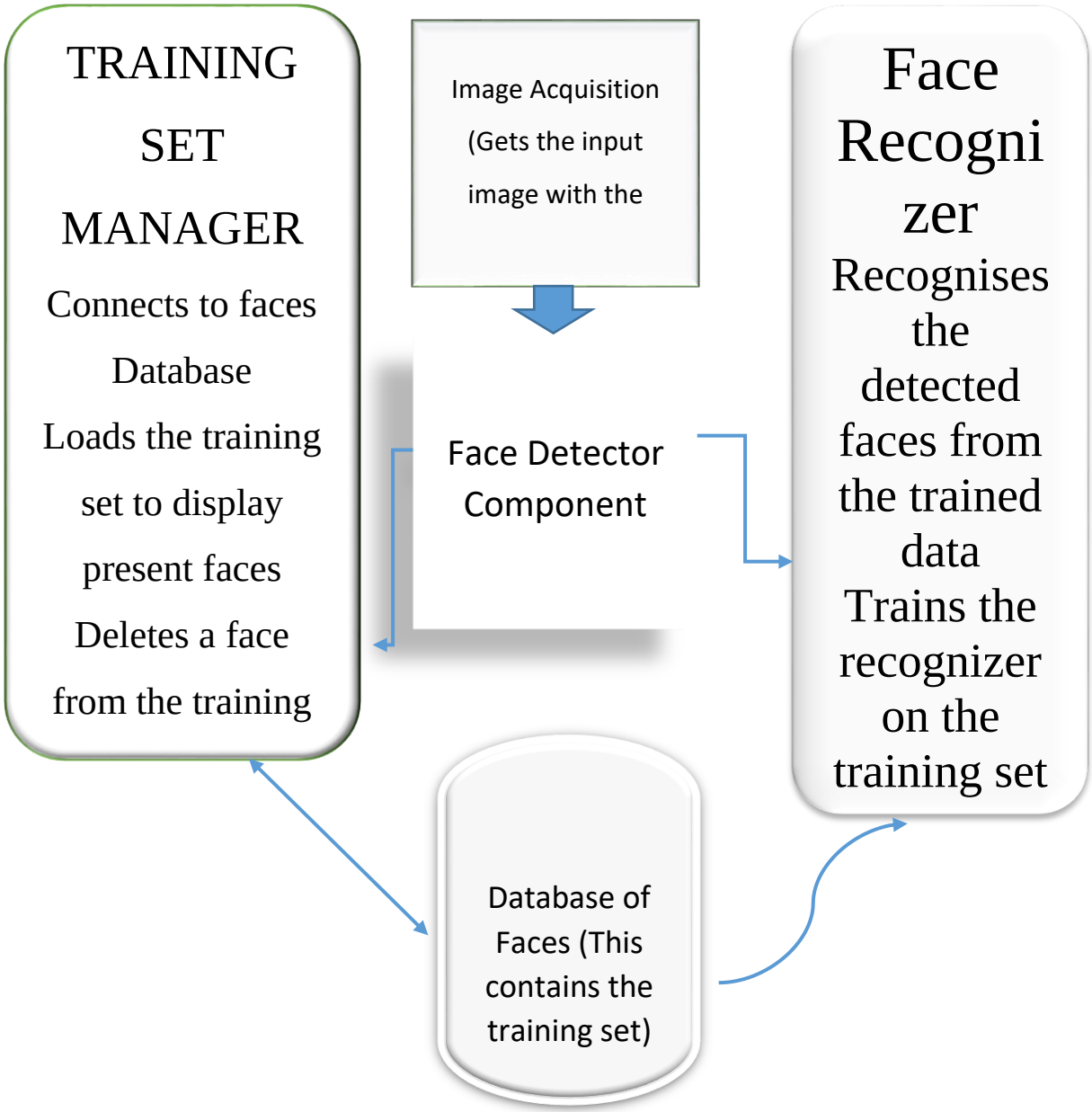


Figure 40: The logical design of the Desktop Module Subsystems

14.11 Functions of the two Sub –Systems

The functionalities of the components are depicted in the block diagrams . The face recognizer system will consist of two major components i.e. the training set manager and the face recognizer. These two components will share the Faces database, the image acquisition and the face detector components; as they are common in their functionality.

We will therefore partition the system in to two subsystems and have their detailed logical designs to be implemented.



14.12 Viola- Jones Algorithm for Face Detection

In 2004 an article by Paul Viola and Michael J. Jones titled "Robust Real-Time Face Detection" was publish in the International Journal of Computer Vision. The algorithm presented in this article has been so successful that today it is very close to being the de facto standard for solving face detection tasks. This success is mainly attributed to the relative simplicity, the fast execution and the remarkable performance of the algorithm.

14.12.1 The Scale Invariant Detector

The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel. This is demonstrated in Figure:

This allows for the calculation of the sum of all pixels inside any given rectangle using only

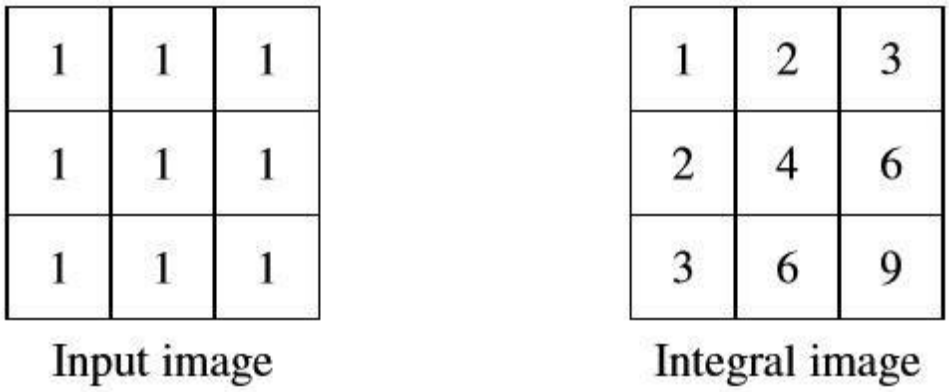
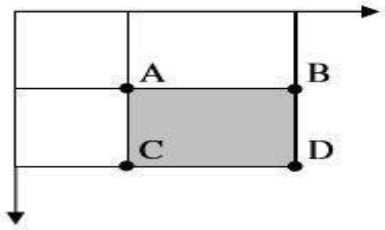


Figure 41: The integral image

Four values. These values are the pixels in the integral image that coincide with the corners of the rectangle in the input image. This is demonstrated in Figure 41.



Sum of grey rectangle = D - (B + C) + A

Figure42: Sum calculations

Since both rectangle B and C include rectangle A, the sum of A has to be added to the calculation.

It has now been demonstrated how the sum of pixels within rectangles of arbitrary size can be calculated in constant time. The Viola-Jones face detector analyses a given sub-window using features consisting of two or more rectangles. The different types of features are shown in Figure 43.

Each feature results in a single value which is calculated by subtracting the sum of the white

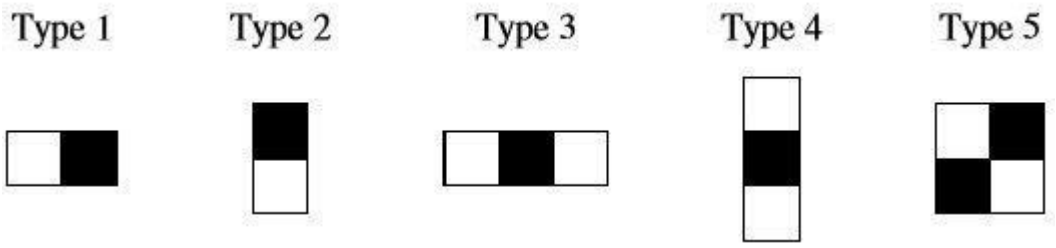


Figure 43: Types of features Rectangle

from the sum of the black rectangle(s).

Viola-Jones have empirically found that a detector with a base resolution of 24*24 pixels gives satisfactory results. When allowing for all possible sizes and positions of the features in Figure a total of approximately 160.000 different features can then be constructed. Thus, the number of possible features vastly outnumbers the 576 pixels contained in the detector at base resolution. These features may seem overly simple to perform such an advanced task as face detection, but what the features lack in complexity they most certainly have in computational efficiency.

One could understand the features as the computer's way of perceiving an input image. The hope being that some features will yield large values when on top of a face. Of course, operations could also be carried out directly on the raw pixels, but the variation due to different pose and individual characteristics would be expected to hamper this approach. The goal is now to smartly construct a mesh of features capable of detecting faces and this is the topic of the next section.

14.13 The Cascaded Classifier

The basic principle of the Viola-Jones face detection algorithm is to scan the detector many times through the same image – each time with a new size. Even if an image should contain one or more faces it is obvious that an excessive large amount of the evaluated sub-windows would still be negatives (non-faces). This realization leads to a different formulation of the problem:

Instead of finding faces, the algorithm should discard non-faces.

The thought behind this statement is that it is faster to discard a non-face than to find a face. With this in mind a detector consisting of only one (strong) classifier suddenly seems inefficient since the evaluation time is constant no matter the input. Hence the need for a cascaded classifier arises.

The cascaded classifier is composed of stages each containing a strong classifier. The job of each stage is to determine whether a given sub-window is definitely not a face or maybe a face. When a sub-window is classified to be a non-face by a given stage it is immediately discarded. Conversely a sub-window classified as a maybe-face is passed on to the next stage

in the cascade. It follows that the more stages a given sub-window passes, the higher the chance the sub-window actually contains a face. The concept is illustrated with two stages in Figure 44

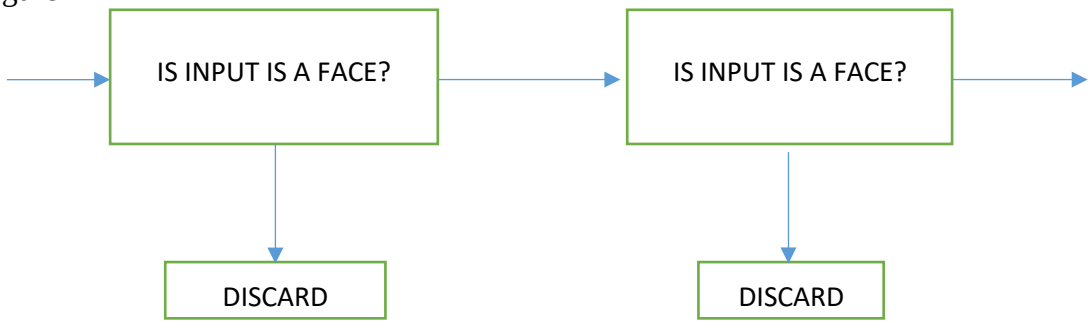


Figure 44: The cascade classifier

In a single stage classifier, one would normally accept false negatives in order to reduce the false positive rate. However, for the first stages in the staged classifier false positives are not considered to be a problem since the succeeding stages are expected to sort them out.

Therefore, Viola-Jones prescribe the acceptance of many false positives in the initial stages. Consequently, the number of false negatives in the final staged classifier is expected to be very small.

Viola-Jones also refer to the cascaded classifier as an attentional cascade. This name implies that more attention (computing power) is directed towards the regions of the image suspected to contain faces. It follows that when training a given stage, say n, the negative examples should of course be false negatives generated by stage n-1.

Chapter-15

15.1 Technology and Tools

To implement this project, we need to use some technology and tools available, so here we introduce to tools we used:

15.1.1 Python Introduction

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

- It is used for:
- Web development (server-side),
- Software development, Mathematics,
- System scripting.
- What can Python do?
- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files. Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

15.1.2 Python libraries

i. OpenCV

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-byday.

Currently OpenCV supports a wide variety of programming languages like C++, Python, and Java etc. and is available on different platforms including Windows, Linux, OS X, Android,

iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

ii. OpenCV-Python

Python is a general-purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of NumPy makes the task easier. NumPy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from NumPy arrays. So whatever operations you can do in NumPy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports NumPy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems. **iii. NumPy**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. Explaining the basics of NumPy such as its architecture and environment. An introduction to Matplotlib is also provided.

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project. **Operations using NumPy**

Using NumPy, a developer can perform the following

operations − Mathematical and logical operations on arrays.

- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

iv. NumPy – A Replacement for MATLAB

NumPy is often used along with packages like SciPy (Scientific Python) and Matplotlib (plotting library). This combination is widely used as a replacement for MATLAB, a popular platform for technical computing. However, Python alternative to MATLAB is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.

v. Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. Tk is the original GUI library for the Tcl language. Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. There are several other popular Python GUI toolkits. Most popular are wxPython, PyQt, and PyGTK.

vi. CSV

Python has a vast library of modules that are included with its distribution. The csv module gives the Python programmer the ability to parse CSV (Comma Separated Values) files. A CSV file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter.

You can think of each line as a row and each field as a column. The CSV format has no standard, but they are similar enough that the csv module will be able to read the vast majority of CSV files. You can also write CSV files using the csv module.

vii.Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyse.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient Data Frame object with default and customized indexing.

- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data. Time Series functionality.

viii.TensorFlow

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

15.2 Requirements to Run the Application:

Supportive Operating Systems: The supported Operating Systems for client include: Windows 2010, windows 2008, windows 2007.

15.2.1 Software Requirements

The Software Requirements in this project include: **a.**

Python

b. OpenCV framework

c. MS Excel.

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. The library is used extensively in companies, research groups and by governmental bodies. As an asynchronous event driven framework.

15.2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

Components	Minimum	Recommended
Processor	Intel Core i3-2100 2nd generation	Intel Core i7 5th generation
RAM	4GB	8GB
Camera	HD 720p Webcam	Full HD 1080p Webcam
Disk	128Gb	512Gb

Figure 45: Hardware Requirements

Chapter-16

16.1 GUI CONSTRUCTING

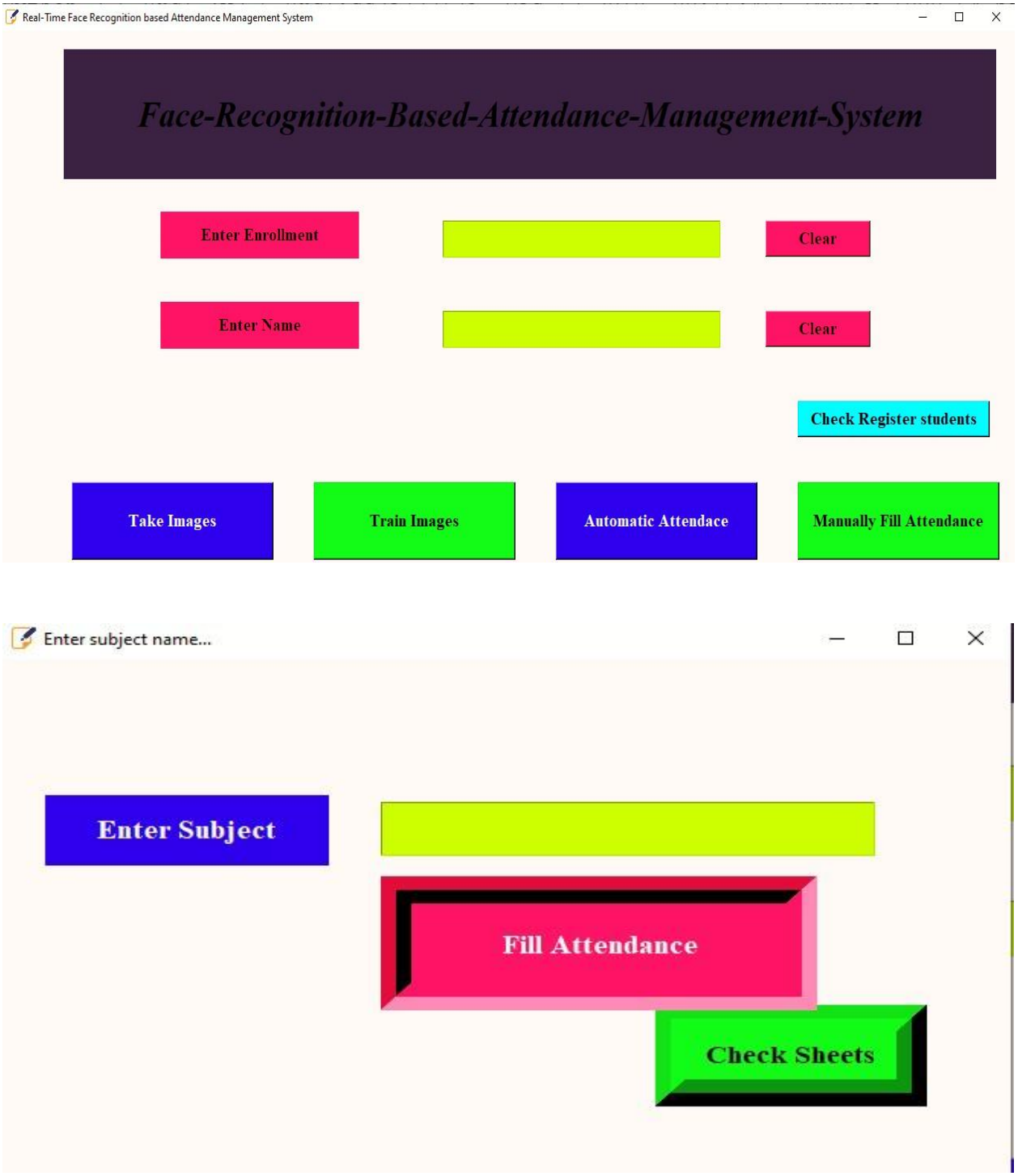


Figure 46 : GUI Home page

Automated Attendance Management System has the following functions:

- Take Images - Clicking this function create dataset of a person and it clicks 200 images of a person.
- Train Images - After creating dataset this function trains the dataset images.
- Automatic Attendance - This function recognizes the person and marks its attendance in excel sheet.
- Enter enrolment- This function receives unique ID of person.
- Enter Name- This function receives name belongs to particular ID.

Chapter-17

17.1 Code-Implementation

Code for Face Detection Attendance System is implemented with python language and OpenCV library.

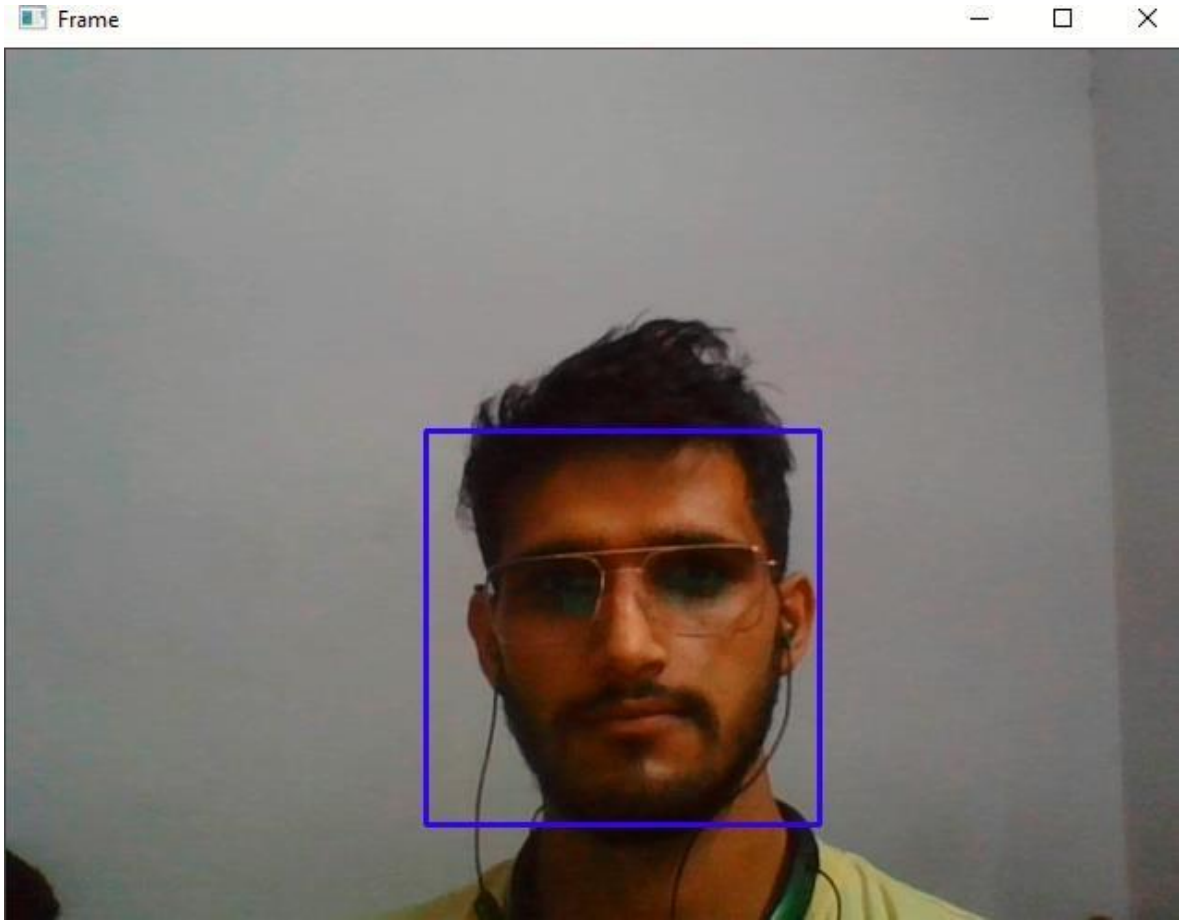


Figure 47: Sample output

17.2 Implementation of dataset creation

The dataset creation is first operation here in above figure it is detecting the face of a person in front of the camera. The program of data set creation will capture 250 images of the person with its unique Id and name entered in the respective fields.

17.3 Algorithm for Creating Dataset

```
#creating paths for dataset  def
assure_path_exists(path):  dir =
os.path.dirname(path)  if not
os.path.exists(dir):
os.makedirs(dir)
face_id=input('enter your id')  #
Start capturing video  vid_cam
= cv2.VideoCapture(0)
# Detect object in video stream using Haarcascade Frontal Face  face_detector
= cv2.CascadeClassifier('haarcascade_profileface.xml')  Ids.append(Id)
return faceSamples,Ids # Capture video frame
_, image_frame = vid_cam.read()  #
Convert frame to grayscale
gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)
# Detect frames of different sizes, list of faces rectangles  faces
= face_detector.detectMultiScale(gray, 1.3, 5)
```

17.4 Algorithm for Trainer

```
def getImagesAndLabels(path):
#get the path of all the files in the folder  imagePaths=[os.path.join(path,f)
for f in os.listdir(path)]
#create empth face list faceSamples=[]
#create empty ID list
Ids=[]
#now looping through all the image paths and loading the Ids and the images  for
imagePath in imagePaths:
```

```
#loading the image and converting it to gray scale
pilImage=Image.open(imagePath).convert('L')  #Now we are converting the PIL image into
numpy array  imageNp=np.array(pilImage,'uint8')
#getting the Id from the image
Id=int(os.path.split(imagePath)[-1].split(".")[1])  #
extract the face from the training image sample
faces=detector.detectMultiScale(imageNp)
#If a face is there then append that in the list as well as Id of it  for
(x,y,w,h) in faces:
faceSamples.append(imageNp[y:y+h,x:x+w])
```

17.5 Implementation of Recognizer and Attendance

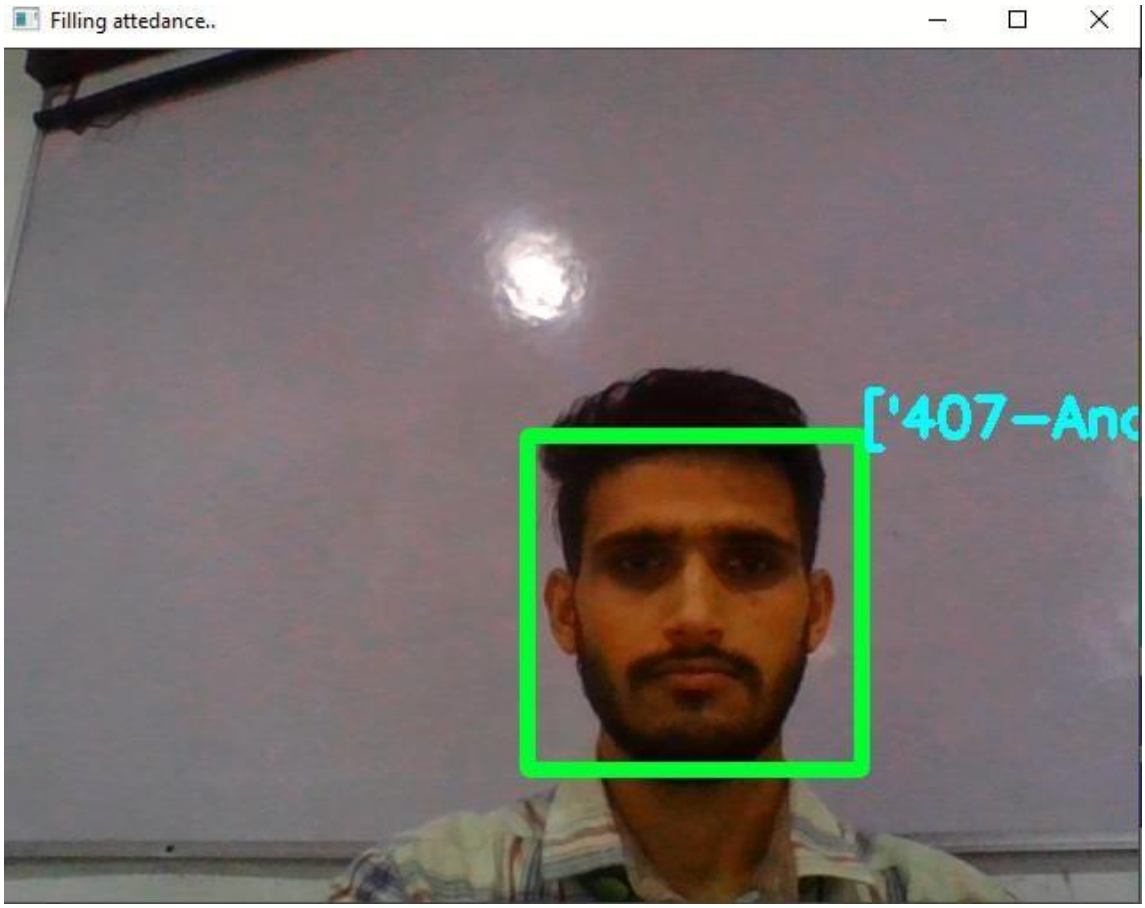


Figure 48: Marking attendance under Normal Conditions

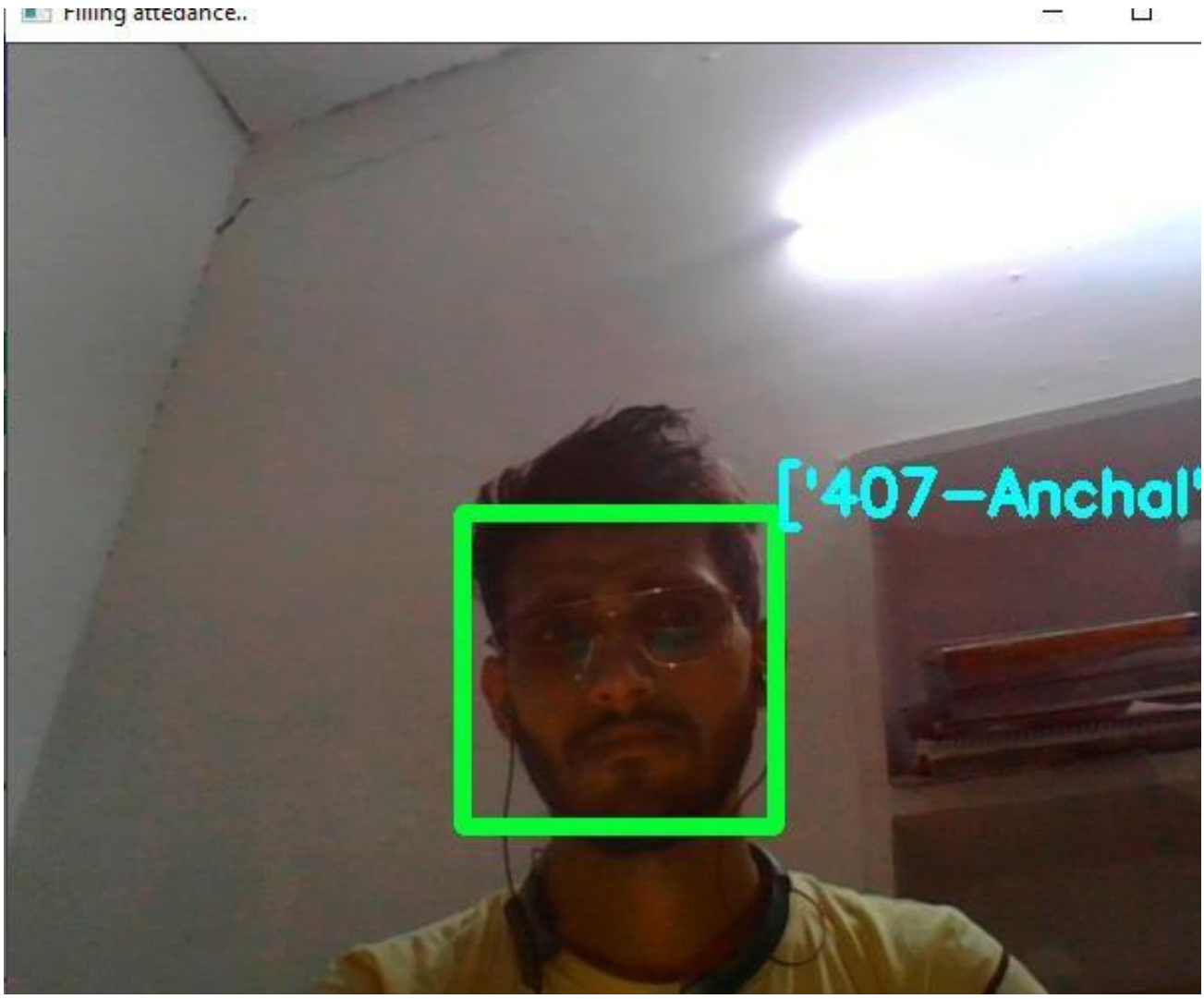


Figure 49: Mark Attendance under Low light with better accuracy

Figure50: Data stored in the excel file

17.6 Information stored in the database:

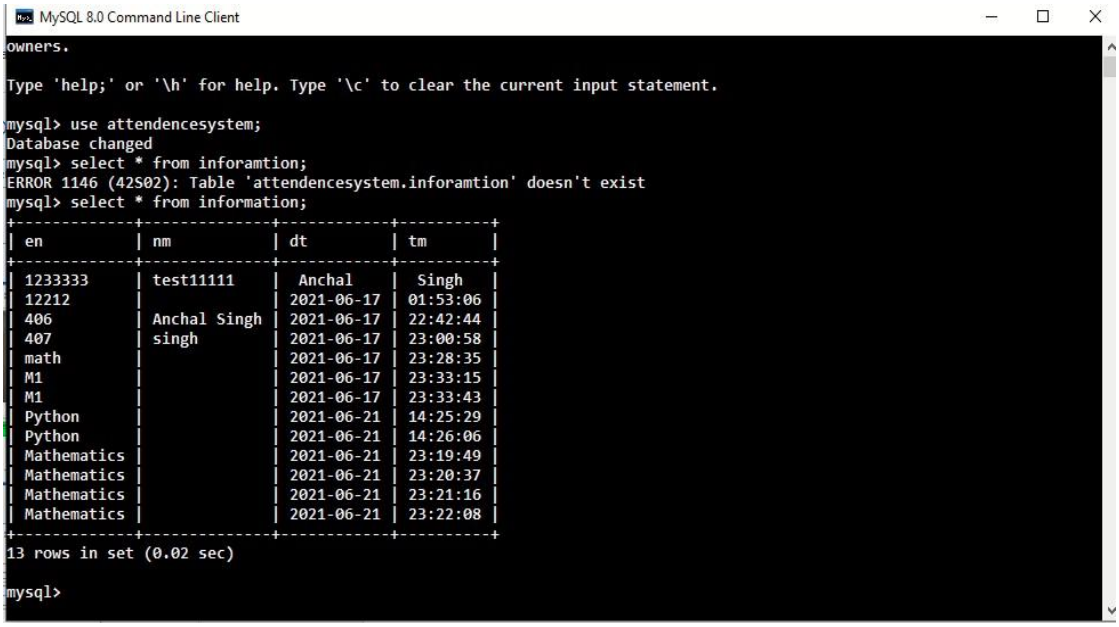


Figure 51: Data stored in the database (MySQL) at different times

17.7 LIMITATIONS

System faces recognition issues if the light conditions are poor.

If lighting conditions are poor than it affects the recognition as it recognizes wrong and sometimes don't recognize in poor lighting conditions, so while creating dataset and recognizing it needs proper lighting.

Webcam and processor with high specification is required as system consumes a lot of resources.

As this project is doing image processing, so to run it smoothly high specification is required and graphic card is must to run it smoothly otherwise it will be too slow and may hang during its execution.

17.8 Conclusion and Recommendation

It can be concluded that a reliable, secure, fast and an efficient class attendance management system has been developed replacing a manual and unreliable system. This face detection and recognition system will save time, reduce the amount of work done by the administration and replace the stationery material currently in use with already existent electronic equipment.

There is no need for specialized hardware for installing the system as it only uses a computer and a camera. The camera plays a crucial role in the working of the system hence the image quality and performance of the camera in real time scenario must be tested especially if the system is operated from a live camera feed.

The system can also be used in permission-based systems and secure access authentication (restricted facilities) for access management, home video surveillance systems for personal security or law enforcement.

The major threat to the system is Spoofing. For future enhancements, anti- spoofing techniques like eye blink detection could be utilized to differentiate live from static images in the case where face detection is made from captured images from the classroom. From the overall efficiency of the system i.e. 83.1% human intervention could be called upon to make the system fool proof. A module could thus be included which lists all the unidentified faces and the lecturer is able to manually correct them.

Future work could also include adding several well-structured attendance registers for each class and the capability to generate monthly attendance reports and automatically email them to the appropriate staff for review.

17.9 FUTURE SCOPE

The research work has implemented a face recognition system by using PCA which is eigenvector based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way which best explains the variance in the data. By implementing PCA the proposed Face Recognition System supplies the user with a lower dimensional picture, a "shadow" of this object when viewed from its most informative viewpoint.

The algorithm has been tested with multiple students in the scene and also captured faces at different angles in the scene. The algorithm delivers quite good results but there is a room to improve the algorithm performance in case of large number of students and also in case of faces captured in a dark environment, so proposed system can be extended in the future to cover this aspect. The efficiency of the algorithm also can be increased further so there is also a room for future work in this area. This system can be enhanced further in terms of achieving more efficiency by ease of analysis of patterns in the data.

REFERENCES

Technical Paper on â_ Image Based Intelligent Attendance Logging System, IEEE Technical Paper by Hary Oktavianto, Hsu Gee-Sern Chung Sheng-Luen, June 30-July 2, 2012, Dalian,China.

Z. Zhou, X. Chen, Y. â“C. Chung, Z. He, T. X. Han, and J. M. Keller, "Activity analysis,summarization, and visualization for indoor human activity monitoring,â IEEE Transactions on Circuits and Systems for Video Technology, Volume 18, Issue 11, pp. 14891498, 2008.

OpenCV. Available: http://sourceforge.net/projects/ opencvlibrary/.

Activity zones for context-aware computing, Lecture Notes in Computer Science, Volume 2864/2003, pp. 90-106, 2003.

International Journal of Innovative Research in Computer and communication Engineering(Vol.3,Issue 8,Aug 2015)(Automated Attendance System Using Face Recognition.)

International Journal of Computer Applications (0975-8887) Volume 98-No.20,July 2014(Adjacency Matrix based Face Recognition Approach.)

International Journal of Information Technology and Knowledge Management JulyDecember 2012, Volume 5, No. 2, pp. 361-3639(Face Recognition Techniques: Classification and Comparisons.)

Face Recognition using Line Edge Map Vipin Kumar and Mohit Mehta Department of Electronics and communication, Punjab College of engineering and technology, India Accepted 01 January 2014, Available online 10 January 2014, Vol.2 (Jan/Feb 2014 issue)

International Journal of Advanced Computer Research (ISSN (Print): 2249-7277 ISSN (Online): 2277-7970) Volume-4 Number-4 Issue-17 December-2014 939 Study of Face Recognition Techniques Sangeeta Kaushik1*, R. B. Dubey2 and Abhimanyu Madan3