

HELMET DETECTION USING IMAGE PROCESSING AND CONVOLUTION NEURAL NETWORKS

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR,
In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY **In** **COMPUTER SCIENCE AND ENGINEERING**

Submitted by

C. Sai Nagalakshmi (17HR1A0512) K.Theja (17HR1A0533)
G. Sindhu Priya (17HR1A0526) C.Usha (17HR1A0510)

Under the esteemed guidance of

Dr.U.Kumaran, M.E, Ph.D.,
Associate Professor, CSE Department



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MOTHER THERESA INSTITUTE OF ENGINEERING &
TECHNOLOGY
AN ISO 9001:2015 CERTIFIED INSTITUTE



(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A. Anantapuramu)
Melumoi (P), Palamaner, Chittoor (Dist.)-517408
2017-2021



MOTHER THERESA INSTITUTE OF ENGINEERING & TECHNOLOGY
AN ISO 9001:2015 CERTIFIED INSTITUTION
(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A., Anantapuramu)
Melumoi (Post), Palamaner-517 408, Chittoor (Dist), A.P.
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Certificate

This is to certify that the Project Report entitled
**HELMET DETECTION USING IMAGE
PROCESSING AND CONVOLUTION NEURAL
NETWORKS**

Is the bonafide work done and

Submitted by

C. Sai Nagalakshmi (17HR1A0512)	K. Theja (17HR1A0533)
G. Sindhu Priya (17HR1A0526)	C. Usha (17HR1A0510)

In the Department of Computer Science and Engineering, Mother Theresa Institute of Engineering & Technology, Palamaner affiliated to J.N.T.U.A., Anantapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during 2020-2021. Submitted on: _____

Internal Guide

Dr. U. Kumaran, M.E, Ph.D.,
Associate Professor

HOD

Dr. R. Suresh, M..Tech, Ph.D.,
Associate Professor&HOD

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Secretary & Correspondent **Sri. M. Ravindra Babu** and Chairman **Sri. M. R. Sunil**, who took keen interest and encouraged us in every effort throughout this course.

We owe our gratitude to **Dr. M. Lakshmikantha Reddy**, Principal, MOTHER THERESA INSTITUTE OF ENGINEERING & TECHNOLOGY for permitting us to use the facilities available to accomplish the project successfully.

We express our heartfelt thanks to Dr. **R. Suresh**, M.Tech, Ph. D, Associate professor and Head, Department of Computer Science and Engineering, for his/her kind attention and valuable guidance to us throughout this course.

We are thankful to our Project Coordinator **Dr. U. Kumaran**, M.E, Ph.D, Associate professor of Computer Science and Engineering for his/her valuable support and guidance throughout the project work.

We are extremely thankful to our Project Supervisor **Dr. U. Kumaran**, M.E, Ph.D, Associate professor of Computer Science and Engineering who took keen interest and encouraged us in every effort throughout this project.

We also thank all the teaching and non-teaching staff of Department Name Department for their cooperation.

C. Sai Nagalakshmi	(17HR1A0512)
K. Theja	(17HR1A0533)
G. Sindhu Priya	(17HR1A0526)
C. Usha	(17HR1A0510)

ABSTRACT

In current situation, we come across various problems in traffic regulations in India which can be solved with different ideas. Riding motorcycle/mopeds without wearing helmet is a traffic violation which has resulted in increase in number of accidents and deaths in India. Existing system monitors the traffic violations primarily through CCTV recordings, where the traffic police have to look into the frame where the traffic violation is happening, zoom into the license plate in case rider is not wearing helmet. But this requires lot of manpower and time as the traffic violations frequently and the number of people using motorcycles is increasing day-by-day. What if there is a system, which would automatically look for traffic violation of not wearing helmet while riding motorcycle/moped and if so, would automatically extract the vehicles' license plate number. Recent research have successfully done this work based on CNN, R-CNN, LBP, HoG, HAAR features,etc. But these works are limited with respect to efficiency, accuracy or the speed with which object detection and classification is done. In this research work, a Non-Helmet Rider detection system is built which attempts to satisfy the automation of detecting the traffic violation of not wearing helmet and extracting the vehicles' license plate number. The main principle involved is Object Detection using Deep Learning at three levels. The objects detected are person, motorcycle/moped at first level using YOLOv2, helmet at second level using YOLOv3, License plate at the last level using YOLOv2. Then the license plate registration number is extracted using OCR (Optical Character Recognition). All these techniques are subjected to predefined conditions and constraints, especially the license plate number extraction part. Since, this work takes video as its input, the speed of execution is crucial. We have used above said methodologies to build a holistic system for both helmet detection and license plate number extraction.

Key words: CNN, LBP, YOLOV2, OCR, CCTV, HAAR, HoG

CONTENTS

CHAPTER NO	CHAPTER NAME	PAGE NO
1	ABSTRACT	
	1.INTRODUCTION	
	1.1 Introduction	1
	1.2 Detailed description of the project/system requirements	1
	1.3 Project objectives and scope	2
2	2.LITERATURE SURVEY	
	2.1 Supporting material to base paper	3
	2.2 Detailed description of the paper	3
3	3.EXISTING SYSTEM	
	3.1 Detailed description with system architecture	4
	3.2 Detailed description of the drawbacks	4
4	4.PROPOSED SYSTEM	
	4.1 Description of the functionalities of the proposed system	5
	4.2 Detailed description of the advantages of the project	5
5	5.MODULE DESCRIPTION	
	5.1 Main objectives / functionalities / activities of the project	6
	5.2 Module description with algorithms/pseudocode	7
6	6.PROJECT DESIGN	
	6.1 Introduction to object oriented methodology	13
	6.2 Software architecture with UML technology	14
	6.4 Description of the flow of the events/functionalities of the architectural elements	20
	6.3 Round-trip engineering (FE and RE) for sample module	21
7	7.PROJECT IMPLEMENTATION	
	7.1 Introduction to implementation language such	21

	as python	
	7.2 Code generation and validation	58
	7.3 Unit and integration testing and debugging	74
	7.4 Result analysis (with graphs if possible)	74
	8.TESTING	
8	8.1 Description of the testing tools	75
	8.2 Evaluation of test cases for database validation	
9	9.EXECUTION THROUGH SCREENSHOTS 9.1Brief control flow of the execution for screenshots	
10	10.CONCLUSION	
	10.1 Detailed conclusion by describing outputs	91
	10.2 Future scope of the project	95
	REFERENCES FOR BIBLIOGRAPHY	

LIST OF FIGURES

S.NO	Figure no	Figure Name	Page no
1	1	Use case Diagram	7
2	2	Class Diagram	8
3	3	Sequence Diagram	9
4	4	Activity Diagram	10

LIST OF ACRONYMS

S NO	ACRONYMS	DESCRIPTION
1	KNN	K-Nearest neighbor
2	CNN	Convolution neural networks
3	SVM	Support-vector machine
4	LBP	Local binary pattern
5	ALPR	Automatic license plate recognition
6	OCR	Optical character recognition
7	ROI	Region of interest
8	HOG	Histogram of oriented gradient

1. INTRODUCTION

India with 1.3 billion populations has millions of motor vehicles running on the roads. Violation of traffic rules has become a common phenomenon, but consequence of which is unbearable. According to a report by NDTV around 400 motor cyclists perish every day in India because of not wearing helmet. If we use helmets it could increase the chances of survival by 42 percent. It also reduces the injuries up to 70 percent. As a citizen of a country one has to obey law and order and follow the same in absolute manner. But seriousness among us is not that great. One has to read a WHO report titled as “why are helmets needed?” in this report there is a comprehensive description about what happens to head if one encounters with an accident and how exactly helmet reduces the impact. When a two wheeler meets with an accident, because of sudden deceleration the rider is thrown away from the vehicle.

DETAILED DESCRIPTION OF PROJECT REQUIREMENTS:

SRS:

SRS (Software Requirement Specification) is a document that completely describes what the proposed should do, without describing how the software does it.

Performance Requirements

- 1) The operation time should be small and the throughput should be high..
- 2) It should produce timely and accurate result.

Software Quality Attributes

- i) **Maintainability** – Since it is directly associated with the database, so there is very little maintainability problem with this application.
- ii) **Easy to Learn** – Since there are less number of forms, this application is very easy to learn with user-friendly screens.
- iii) **Flexibility** – This application is very much flexible for future enhancements.

Hardware Requirements

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

2.1.4 Software Requirements

- **Operating System:** Windows
- **Coding Language:** Python 3.7

OBJECTIVE OF THE PROJECT

Helmet reduces the chances of skull getting decelerated, hence sets the motion of the head to almost zero. Cushion inside the helmet absorbs the impact of collision and as time passes head comes to a halt. It also spreads the impact to a larger area, thus safeguarding the head from severe injuries. More importantly it acts as a mechanical barrier between head and object to which the rider came into contact. Injuries can be minimized if a good quality full helmet is used. Traffic rules are there to bring a sense of discipline, so that the risk of deaths and injuries can be minimized significantly. However strict adherence to these laws is absent in reality. Hence efficient and feasible techniques have to be created to overcome these problems. Manual surveillance of traffic using CCTV is an existing methodology. But here so many iterations have to be performed to attain the objective and it demands a lot of human resource. Therefore, cities with millions of population having so many vehicles running on the roads cannot afford this inadequate manual method of helmet detection. So here we propose a methodology for full helmet

detection and license plate extraction using YOLOv2, YOLOv3 and OCR. Basically

helmet detection system involves following steps such as collection of dataset, moving object detection, background subtraction, and object classification using neural networks.

2. LITERATURE SURVEY

- According to a report by NDTV around 400 motor cyclists perish every day in India because of not wearing helmet.
- If we use helmets it could increase the chances of survival by 42 percent. It also reduces the injuries up to 70 percent.
- One has to read a WHO report titled as “why are helmets needed?” in this report there is a comprehensive description about what happens to head if one encounters with an accident and how exactly helmet reduces the impact.
- As a citizen of a country one has to obey law and order and follow the same in absolute manner.
- In this research, workers without safety helmet are also noticed in power substations but not only on the roads.

3. THE EXISTING SYSTEM

DETAILED DESCRIPTION OF SYSTEM ARCHITECTURE

Existing system monitors the traffic violations primarily through CCTV recordings, where the traffic police have to look into the frame where the traffic violation is happening, zoom into the license plate in case rider is not wearing helmet. But this requires lot of manpower and time as the traffic violations frequently and the number of people using motorcycles is increasing day-by-day. What if there is a system, which would automatically look for traffic violation of not wearing helmet while riding motorcycle/moped and if so, would automatically extract the vehicles' license plate number. Recent research have successfully done this work based on CNN, R-CNN, LBP, HoG, HaaR features,etc. But these works are limited with respect to efficiency, accuracy or the speed .

DETAILED DESCRIPTION OF THE DRAWBACKS

- The existing system monitors the traffic violations primarily through CCTV phootage, where the traffic police have to look into the frame.
- The main drawback of this system is that we cannot capture the classified images of the vehicles and passengers with no helmets, since it captures only the video.

And also it takes alots of time during investigations to sit and capture where there is a huge amount of congestion

4. THE PROPOSED SYSTEM

DESCRIPTION OF THE FUNCTIONALITIES OF THE PROPOSED SYSTEM

In this project we are detecting whether two wheeler rider wearing helmet or not, if he is not wearing helmet then we are extracting number plate of that two wheeler. To extract number plate we have YOLO CNN model with some train and test images and if you want to add some other images then send those images to us so we can include those images in YOLO model with annotation to extract number plate of those new images.

To implement above technique we are following or implemented below modules

- 1) First image will be upload to the application and the using YOLOV2 we will check whether image contains person with motor bike or not, if YOLO model detect both person and motor bike then we will proceed to step 2.
- 2) In this module we will use YOLOV3 model to detect whether object wear helmet or not, if he wear helmet then application will stop hear itself. If rider not wear helmet then application proceed to step 3.
- 3) In this module we will extract number plate data using python tesseract OCR API. OCR will take input image and then extract vehicle number from it.

DESCRIPTION OF THE ADVANTAGES OF THE PROPOSED SYSTEM

In this study non helmet riders are detected and their license plate is extracted. They have used 2 YOLOv2 models to detect helmeted and non-helmeted riders. This model has the capability to detect all classes of COCO dataset. In the first set, person is detected instead of motorcycle. The second YOLOv2 model is trained with helmet dataset. This makes the detection of non-helmet riders easier. Open ALPR is used for the recognition of the license plate [13]. The approach mentioned in this paper is applied mainly for power substation. Here for noise elimination mean filtering operation is performed. Background subtraction based on KNN is used to detect the moving objects. Color-based hybrid descriptor is used for feature extraction of the head image segments. SVM classifier is used to classify the head image segments. HOG descriptors are used to identify the pedestrian image segment.

2.2 Software Design

System design is the second step in the system life cycle, in which overall design of the system is achieved. The functionalities of the system is designed and studied in this phase. The first step is designing of program specification. This determines the various data inputs to the system, data flow and the format in which output is to be obtained.

Design phase is a transmission phase because it is a transition from user oriented document to computer data. The activity in the design phase is the allocation of functions to manual operations, equipment and computer programs. Flow charts are prepared in the study time and is decomposed until all functions in the system perform evidently.

Design is a multi-step process that focuses on data structures, software architecture, procedural details(algorithms etc) and links between the modules. The design process goes through logical and physical stages. In logical design reviews are made linking existing system and specification gathered. The physical plan specifies any hardware and software requirement, which satisfies the local design.

Modularization of task is made in this phase. The success of any integrated system depends on the planning of each and every fundamental module. Usually a project is revised in step by step sequence. Inter-phase management of such module is also

important. Software design methodology changes continually as new methods, better analysis and broader understanding evolve.

Various techniques for software design do exist with the availability of criteria for design quality. Software design leads three technical activities-design, code and test. Each activity transforms information, which validates the software. The design system converts theoretical solution introduced by the feasibility study into a logical reality.

5. MODULE DESCRIPTION

MAIN OBJECTIVES OF THE MODULE

Modules of Detection of Non-Helmet Riders and Extraction of License Plate Number using Yolo v2 and OCR Method

1.Upload Image

2. Detect Motor Bike & Person

The frame chosen is given as input to YOLOv2 object detection model, where the classes to be detected are „Motorbike“, „Person“. At the output, image with required class detection along with confidence of detection through bounding box and probability value is obtained.

With the help of functions given by Image AI library, only the detected objects are extracted and stored as separate images and named with class name and image number in order. For example, it will be saved as motorcycle-1, motorcycle-2, etc.... if extracted object is motorcycle or person-1, person-2, etc.... if extracted image is of person. The details of these extracted images which is stored in a dictionary which can be later used for further processing.

3. Detect Helmet

Once the person-motorcycle pair is obtained, the person images is given as input to helmet detection model. While testing the helmet detection model, some false detections were observed. So, the person image was cropped to get only top one-fourth portion of image.This ensures that false detection cases are eliminated as well as avoid

cases leading to wrong results when the rider is holding helmet in hand while riding or keeping it on motorcycle while riding instead of wearing.

4. Exit

MODULE DESCRIPTION WITH PSEUDO CODE

```
from tkinter import *
import tkinter
from tkinter import filedialog
import numpy as np
from tkinter.filedialog import askopenfilename
import pandas as pd
from tkinter import simpledialog
import numpy as np
import cv2 as cv
import subprocess
import time
import os
from yoloDetection import detectObject, displayImage
import sys
from time import sleep
from tkinter import messagebox
import pytesseract as tess
from keras.models import model_from_json
from keras.utils.np_utils import to_categorical

main = tkinter.Tk()
main.title("Helmet Detection") #designing main screen
main.geometry("800x700")

global filename
global loaded_model

global class_labels
global cnn_model
global cnn_layer_names

frame_count = 0
frame_count_out=0

confThreshold = 0.5
nmsThreshold = 0.4
inpWidth = 416
inpHeight = 416
global option
labels_value = []
with open("Models/labels.txt", "r") as file: #reading MRC dictionary
    for line in file:
```

```

        line = line.strip('\n')
        line = line.strip()
        labels_value.append(line)
    file.close()

with open('Models/model.json', "r") as json_file:
    loaded_model_json = json_file.read()
    plate_detector = model_from_json(loaded_model_json)

plate_detector.load_weights("Models/model_weights.h5")
plate_detector._make_predict_function()

classesFile = "Models/obj.names";
classes = None

with open(classesFile, 'rt') as f:
    classes = f.read().rstrip('\n').split('\n')

modelConfiguration = "Models/yolov3-obj.cfg";
modelWeights = "Models/yolov3-obj_2400.weights";

net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)

def getOutputsNames(net):
    layersNames = net.getLayerNames()
    return [layersNames[i][0] - 1] for i in
net.getUnconnectedOutLayers()]

def loadLibraries(): #function to load yolov3 model weight and class
labels
    global class_labels
    global cnn_model
    global cnn_layer_names
    class_labels = open('yolov3model/yolov3-
labels').read().strip().split('\n') #reading labels from yolov3
model
    print(str(class_labels)+" == "+str(len(class_labels)))
    cnn_model =
cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg',
'yolov3model/yolov3.weights') #reading model
    cnn_layer_names = cnn_model.getLayerNames() #getting layers
from cnn model
    cnn_layer_names = [cnn_layer_names[i][0] - 1] for i in
cnn_model.getUnconnectedOutLayers()] #assigning all layers

def upload(): #function to upload tweeter profile
    global filename
    filename = filedialog.askopenfilename(initialdir="bikes")
    #messagebox.showinfo("File Information", "image file loaded")

```



```

def detectBike():
    global option
    option = 0
    indexno = 0
    label_colors = (0,255,0)
    try:
        image = cv.imread(filename)
        image_height, image_width = image.shape[:2]
    except:
        raise 'Invalid image path'
    finally:
        image, ops = detectObject(cnn_model, cnn_layer_names,
image_height, image_width, image, label_colors,
class_labels,indexno)
        if ops == 1:
            displayImage(image,0)#display image with detected
objects label
            option = 1
        else:
            displayImage(image,0)

def drawPred(classId, conf, left, top, right, bottom,frame,option):
    global frame_count
    #cv.rectangle(frame, (left, top), (right, bottom), (255, 178,
50), 3)
    label = '%.2f' % conf
    if classes:
        assert(classId < len(classes))
        label = '%s:%s' % (classes[classId], label)
        labelSize, baseLine = cv.getTextSize(label,
cv.FONT_HERSHEY_SIMPLEX, 0.5, 1)
        top = max(top, labelSize[1])
        label_name,label_conf = label.split(':')
        print(label_name+"=== "+str(conf)+"== "+str(option))
        if label_name == 'Helmet' and conf > 0.50:
            if option == 0 and conf > 0.90:
                cv.rectangle(frame, (left, top -
round(1.5*labelSize[1])), (left + round(1.5*labelSize[0]), top +
baseLine), (255, 255, 255), cv.FILLED)
                cv.putText(frame, label, (left, top),
cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)
                frame_count+=1
            if option == 0 and conf < 0.90:
                cv.putText(frame, "Helmet Not detected", (10, top),
cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,0), 2)
                frame_count+=1
            img = cv.imread(filename)

```

```

img = cv.resize(img, (64,64))
im2arr = np.array(img)
im2arr = im2arr.reshape(1,64,64,3)
X = np.asarray(im2arr)
X = X.astype('float32')
X = X/255
preds = plate_detector.predict(X)
predict = np.argmax(preds)
#img = cv.imread(filename)
#img = cv.resize(img, (500,500))
#text = tess.image_to_string(img, lang='eng')
#text = text.replace("\n", " ")
#messagebox.showinfo("Number Plate Detection Result",
"Number plate detected as "+text)
textarea.insert(END,filename+"\n\n")
textarea.insert(END,"Number plate detected as
"+str(labels_value[predict]))
if option == 1:
    cv.rectangle(frame, (left, top -
round(1.5*labelSize[1])), (left + round(1.5*labelSize[0]), top +
baseLine), (255, 255, 255), cv.FILLED)
    cv.putText(frame, label, (left, top),
cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)
    frame_count+=1

if(frame_count> 0):
    return frame_count

def postprocess(frame, outs, option):
    frameHeight = frame.shape[0]
    frameWidth = frame.shape[1]
    global frame_count_out
    frame_count_out=0
    classIds = []
    confidences = []
    boxes = []
    classIds = []
    confidences = []
    boxes = []
    cc = 0
    for out in outs:
        for detection in out:
            scores = detection[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if confidence > confThreshold:
                center_x = int(detection[0] * frameWidth)
                center_y = int(detection[1] * frameHeight)
                width = int(detection[2] * frameWidth)
                height = int(detection[3] * frameHeight)
                left = int(center_x - width / 2)
                top = int(center_y - height / 2)

```

```

        classIds.append(classId)
        #print(classIds)
        confidences.append(float(confidence))
        boxes.append([left, top, width, height])

    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold,
nmsThreshold)
    count_person=0 # for counting the classes in this loop.
    for i in indices:
        i = i[0]
        box = boxes[i]
        left = box[0]
        top = box[1]
        width = box[2]
        height = box[3]
        frame_count_out = drawPred(classIds[i], confidences[i],
left, top, left + width, top + height,frame,option)
        my_class='Helmet'
        unknown_class = classes[classId]
        print("===="+str(unknown_class))
        if my_class == unknown_class:
            count_person += 1

    print(str(frame_count_out))
    if count_person == 0 and option == 1:
        cv.putText(frame, "Helmet Not detected", (10, 50),
cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,0), 2)
    if count_person >= 1 and option == 0:
        #path = 'test_out/'
        #cv.imwrite(str(path)+str(cc)+".jpg", frame)          # writing
to folder.
        #cc = cc + 1
        frame = cv.resize(frame,(500,500))
        cv.imshow('img',frame)
        cv.waitKey(50)

def detectHelmet():
    textarea.delete('1.0', END)
    if option == 1:
        frame = cv.imread(filename)
        frame_count =0
        blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth,
inpHeight), [0,0,0], 1, crop=False)
        net.setInput(blob)
        outs = net.forward(getOutputsNames(net))
        postprocess(frame, outs,0)
        t, _ = net.getPerfProfile()
        label = 'Inference time: %.2f ms' % (t * 1000.0 /
cv.getTickFrequency())
        print(label)
        cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 255))

```

```

        print(label)
    else:
        messagebox.showinfo("Person & Motor bike not detected in
uploaded image", "Person & Motor bike not detected in uploaded
image")

def videoHelmetDetect():
    global filename
    videofile = askopenfilename(initialdir = "videos")
    video = cv.VideoCapture(videofile)
    while(True):
        ret, frame = video.read()
        if ret == True:
            frame_count = 0
            filename = "temp.png"
            cv.imwrite("temp.png",frame)
            blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth,
inpHeight), [0,0,0], 1, crop=False)
            net.setInput(blob)
            outs = net.forward(getOutputsNames(net))
            postprocess(frame, outs,1)
            t, _ = net.getPerfProfile()
            #label=''
            #cv.putText(frame, label, (0, 15),
cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))
            cv.imshow("Predicted Result", frame)
            if cv.waitKey(5) & 0xFF == ord('q'):
                break
        else:
            break
    video.release()
    cv.destroyAllWindows()

def exit():
    global main
    main.destroy()

font = ('times', 16, 'bold')
title = Label(main, text='Number Plate Detection without Helmet',
justify=LEFT)
title.config(bg='lavender blush', fg='DarkOrchid1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=100,y=5)
title.pack()

font1 = ('times', 14, 'bold')
model = Button(main, text="Upload Image", command=upload)
model.place(x=200,y=100)
model.config(font=font1)

```

```

uploadimage = Button(main, text="Detect Motor Bike & Person",
command=detectBike)
uploadimage.place(x=200,y=150)
uploadimage.config(font=font1)

classifyimage = Button(main, text="Detect Helmet",
command=detectHelmet)
classifyimage.place(x=200,y=200)
classifyimage.config(font=font1)

exitapp = Button(main, text="Exit", command=exit)
exitapp.place(x=200,y=250)
exitapp.config(font=font1)

font1 = ('times', 12, 'bold')
textarea=Text(main,height=15,width=60)
scroll=Scrollbar(textarea)
textarea.configure(yscrollcommand=scroll.set)
textarea.place(x=10,y=300)
textarea.config(font=font1)

loadLibraries()

main.config(bg='light coral')
main.mainloop()

```

6. PROJECT DESIGN

INTRODUCTION TO OBJECT ORIENTED METHODOLOGY

Object-oriented Methodology (OOM) is a system development approach Encouraging and facilitating re-use of software components. With this the methodology, a computer system can be developed on a component basis which enables the effective re-use of existing components and facilitates the Sharing of its components by other systems.

Documents of OOM

An Introduction to OOM

This document provides a brief overview of the OOM, its benefits, the processes and some of the major techniques used in the OOM.

OOM Procedures Manual

This document describes the OOM process structure and procedures involved in conducting the OOM project.

OOM Documentation Manual

This document defines the deliverables for the project using the OOM. It describes the purpose, contents and preparation guidelines of each deliverable.

SOFTWARE ARCHITECTURE WITH UML TECHNOLOGY

UML DIAGRAMS

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

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major

components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

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

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

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

GOALS:

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

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

1. USE CASE DIAGRAM:

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

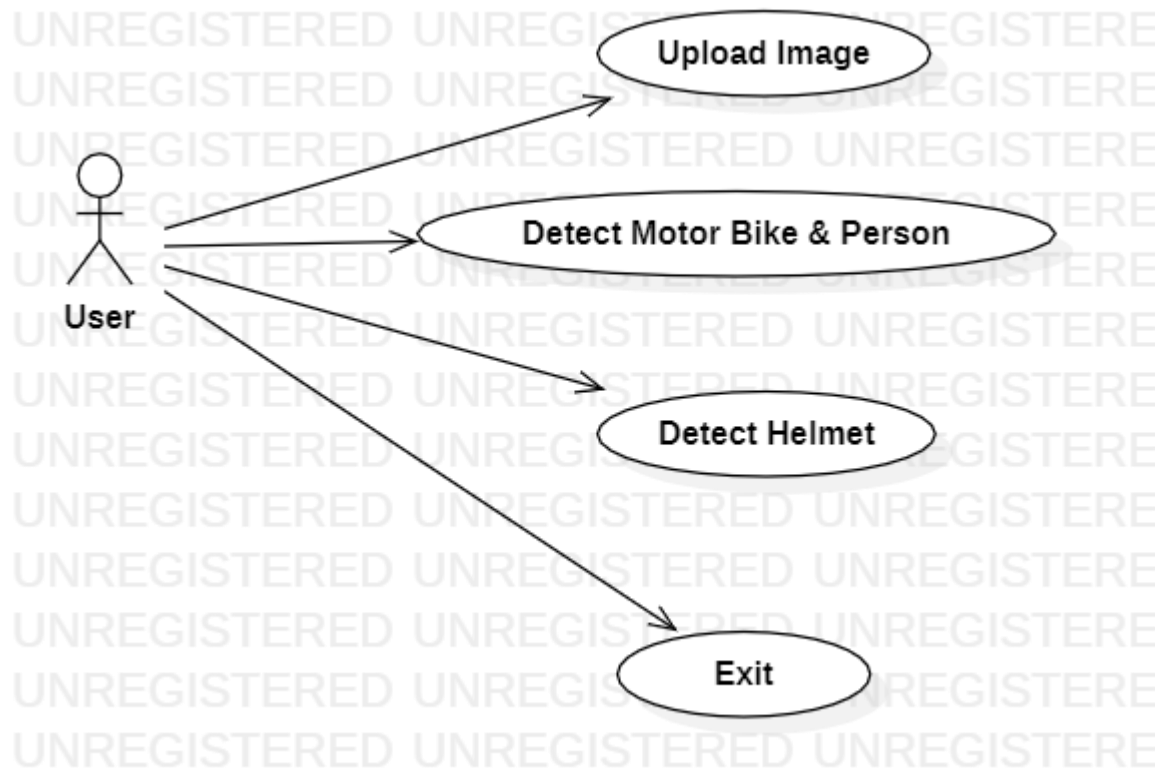


Fig:1.usecase diagram

2. CLASS DIAGRAM:

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

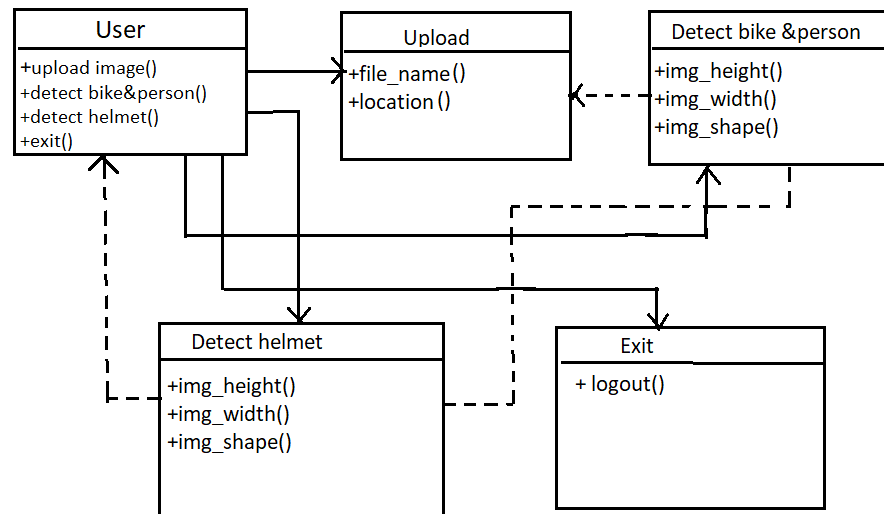


Fig:2. Class Diagram

3. SEQUENCE DIAGRAM:

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

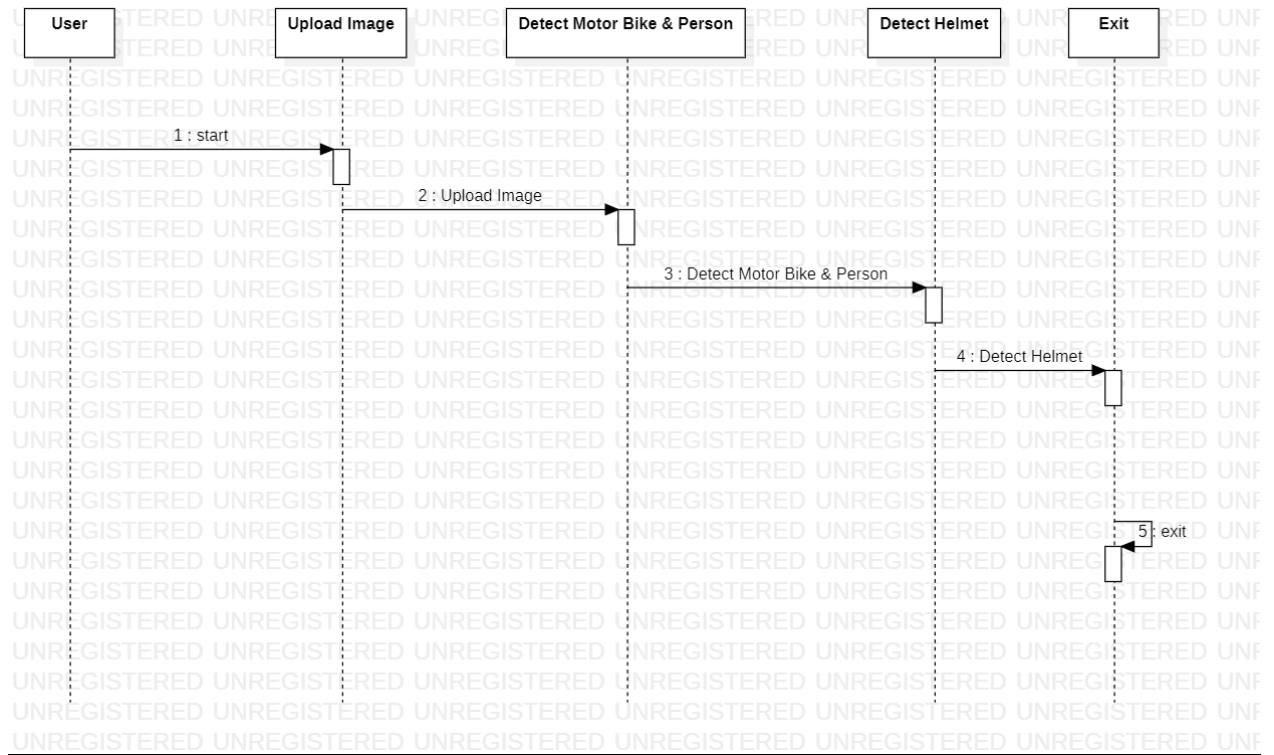


Fig:3.Sequence Diagram

4. ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

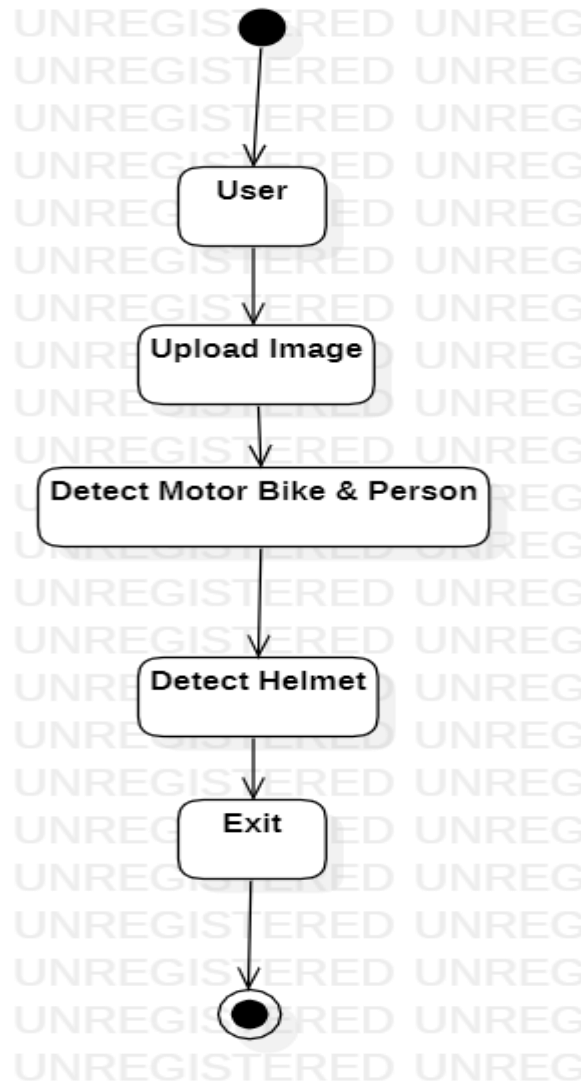


Fig:4.Activity Diagram

DESCRIPTION OF THE FLOW OF FUNCTIONALITIES

What is SDLC?

SDLC stands for Software Development Life Cycle. A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software.

SDLC is the process consisting of a series of planned activities to develop or alter the software products.

Benefits of the SDLC Process

The intent of a SDLC process is to help produce a product that is cost-efficient, effective, and of high quality. Once an application is created, the SDLC maps the proper deployment and decommissioning of the software once it becomes a legacy. The SDLC methodology usually contains the following stages: Analysis (requirements and design), construction, testing, release, and maintenance (response). Veracode makes it possible to integrate automated security testing into the SDLC process through use of its cloud based platform.

Requirements Gathering:

In this phase we gather all the requirements from the client, i.e. what are the client expected input, output.....

Analysis:

In this phase based upon the client requirements we prepare one documentation is called “High Level Design Document”. It contains Abstract, Functional Requirements, Non Functional Requirements, Existing System, Proposed System, SRS,.....

Design:

It is difficult to understand the High Level Design Document for all the members, so to understand easily we use “Low Level Design Document”. To design this document we use UML (Unified Modeling Language). In this we have Use case, Sequence, Collaboration.....

Coding:

In this phase we develop the coding module by module. After developing all the modules we integrate them.

Testing:

After developing we have to check whether client requirements are satisfied or not. If not we are again going to develop.

Implementation:

In testing phase if client requirements are satisfied, we go for implementation. i.e. we need to deploy the application in some server.

Maintenance:

After deployment, if at all any problems come from the client side; we are providing maintenance for that application.

7. PROJECT IMPLEMENTATION

INTRODUCTION TO IMPLEMENTATION LANGUAGE SUCH AS PYTHON

Python is a general purpose, dynamic, high level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is *easy to learn* yet powerful and versatile scripting language, which makes it attractive for Application Development.

Python's syntax and *dynamic typing* with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports *multiple programming pattern*, including object-oriented, imperative, and functional or procedural programming styles.

Python is not intended to work in a particular area, such as web programming. That is why it is known as *multipurpose* programming language because it can be used with web, enterprise, 3D CAD, etc.

We don't need to use data types to declare variable because it is *dynamically typed* so we can write `a=10` to assign an integer value in an integer variable.

Python makes the development and debugging *fast* because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python 2 vs. Python 3

In most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language, therefore, it is easier for the projects to switch in the newer version. However, in the case of Python, the two versions Python 2 and Python 3 are very much different from each other.

A list of differences between Python 2 and Python 3 are given below:

1. Python 2 uses **print** as a statement and used as `print "something"` to print some string on the console. On the other hand, Python 3 uses **print** as a function and used as `print("something")` to print something on the console.
2. Python 2 uses the function `raw_input()` to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the `int()` function in Python. On the other hand, Python 3 uses `input()` function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (`int()`, `str()`, etc.).
3. In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.
4. Python 3 doesn't contain the `xrange()` function of Python 2. The `xrange()` is the variant of `range()` function which returns a `xrange` object that works similar to

Java iterator. The range() returns a list for example the function range(0,3) contains 0, 1, 2.

5. There is also a small change made in Exception handling in Python 3. It defines a keyword **as** which is necessary to be used. We will discuss it in Exception handling section of Python programming tutorial.

Python Features

Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at [official web address](#). The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Python History and Versions

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:

- ABC language.
- Modula.

Python programming language is being updated regularly with new features and supports. There are lots of updations in python versions, started from 1994 to current release.

Python Applications

Python is known for its general-purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifying applications areas where python can be applied.

1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, BeautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

5) Business Applications

Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high level application platform.

6) Console Based Application

We can use Python to develop console based applications. For example: **IPython**.

7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

9) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

10) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

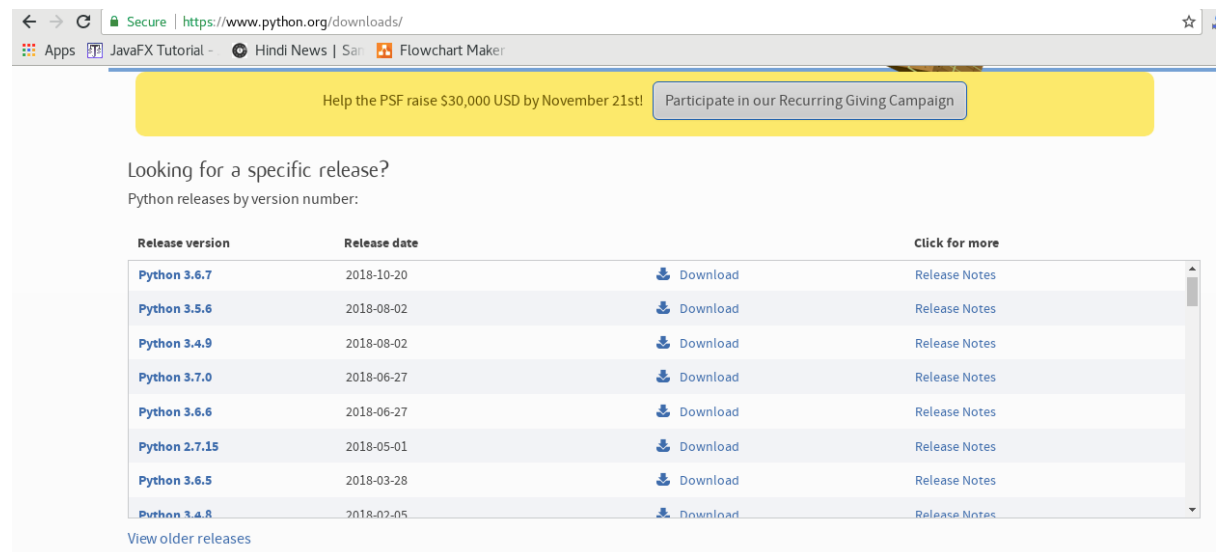
There are several such applications which can be developed using Python

How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

Installation on Windows

Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.



The screenshot shows the Python.org downloads page. At the top, there is a yellow banner with the text "Help the PSF raise \$30,000 USD by November 21st!" and a button "Participate in our Recurring Giving Campaign". Below the banner, the text "Looking for a specific release?" is followed by "Python releases by version number:". A table lists the releases with columns for "Release version", "Release date", and "Click for more". The table includes links for "Download" and "Release Notes" for each version. The versions listed are Python 3.6.7, 3.5.6, 3.4.9, 3.7.0, 3.6.6, 2.7.15, 3.6.5, and 3.4.8. Below the table, there is a link "View older releases".

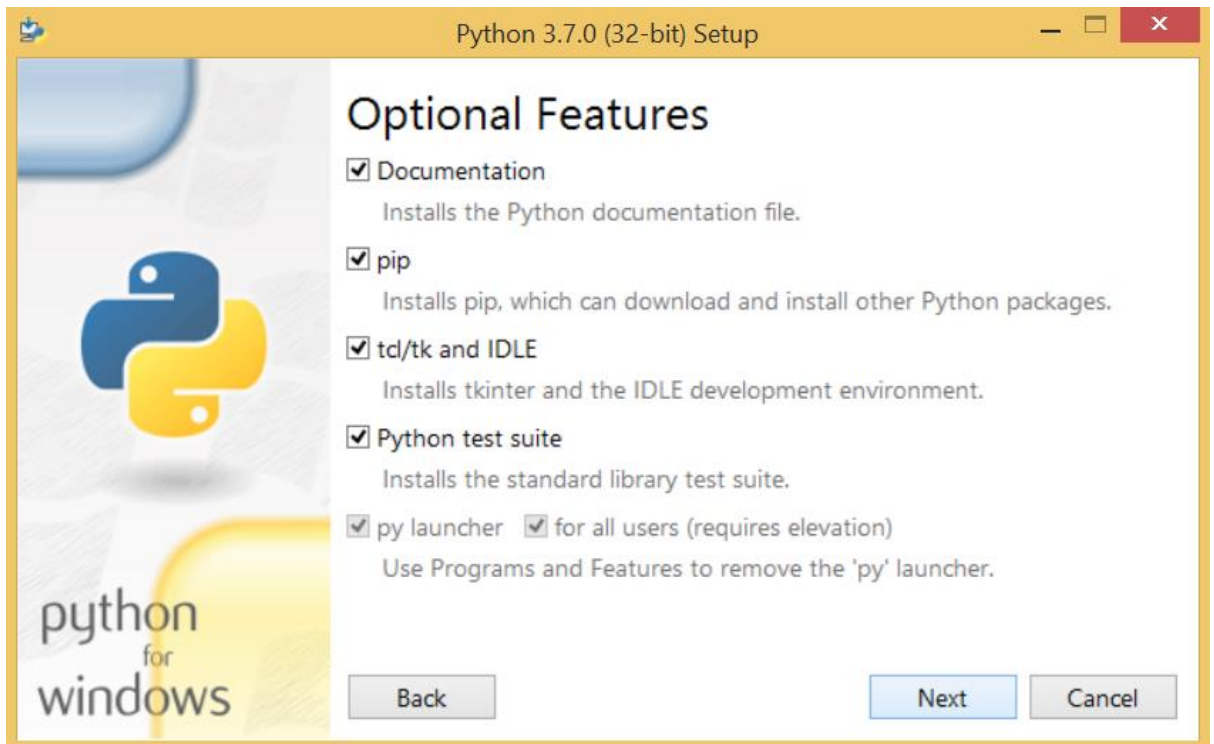
Release version	Release date	Click for more
Python 3.6.7	2018-10-20	Download Release Notes
Python 3.5.6	2018-08-02	Download Release Notes
Python 3.4.9	2018-08-02	Download Release Notes
Python 3.7.0	2018-06-27	Download Release Notes
Python 3.6.6	2018-06-27	Download Release Notes
Python 2.7.15	2018-05-01	Download Release Notes
Python 3.6.5	2018-03-28	Download Release Notes
Python 3.4.8	2018-02-05	Download Release Notes

[View older releases](#)

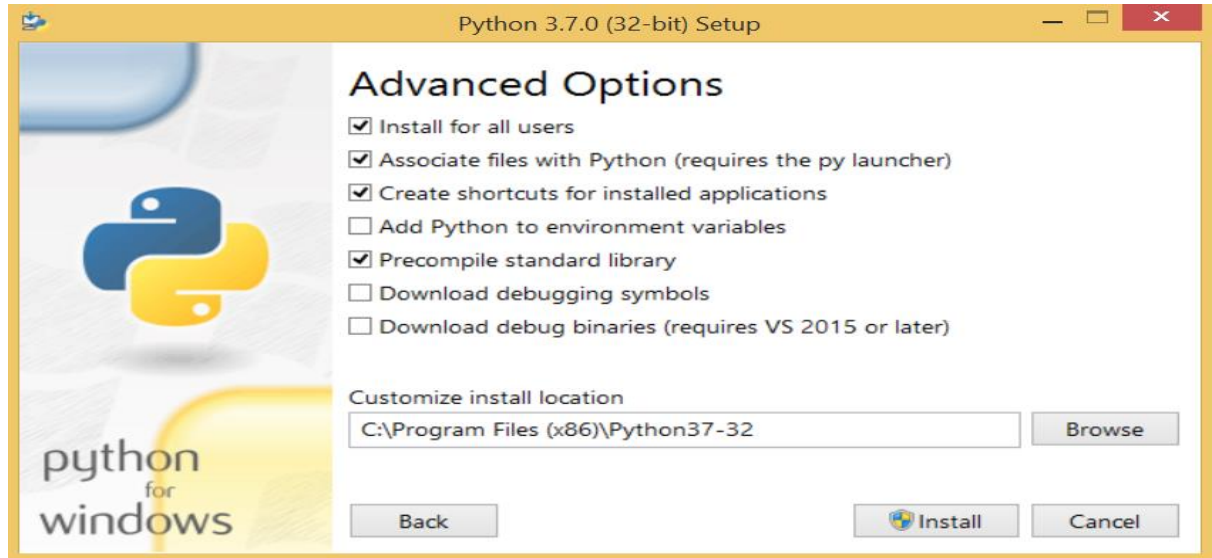
Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.



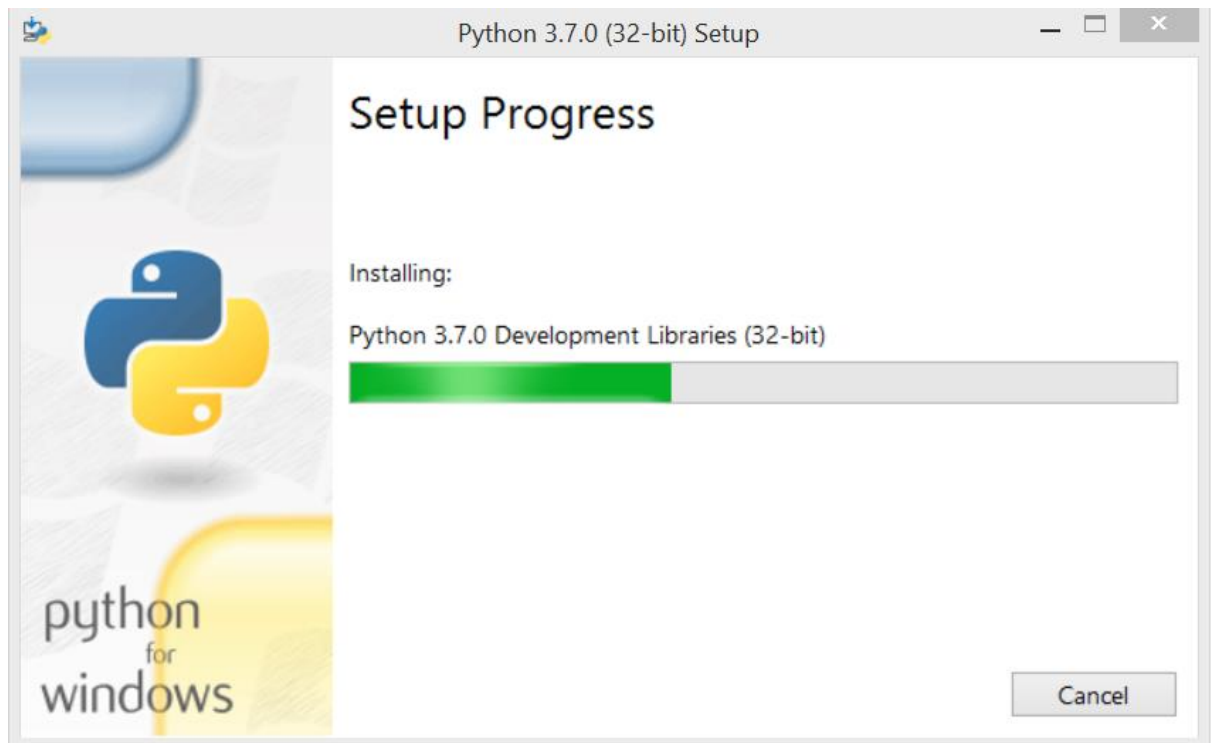
The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.



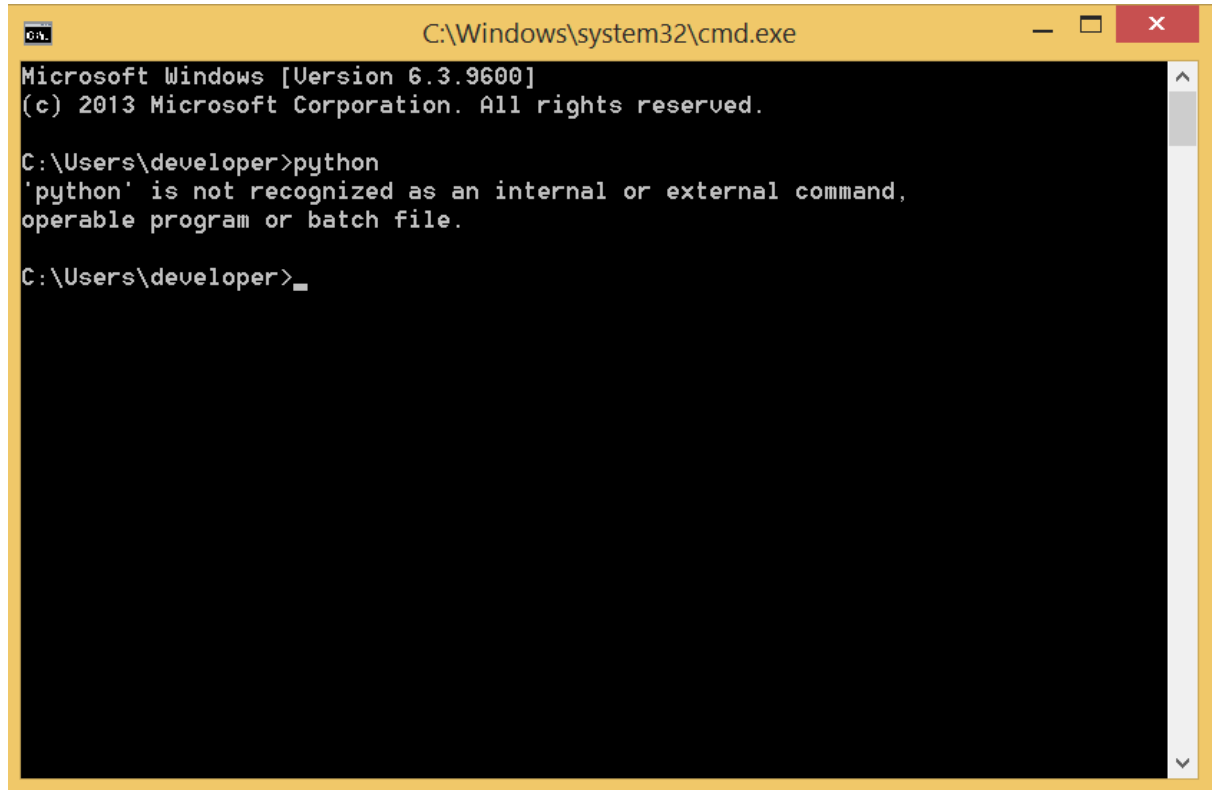
The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



Now, we are ready to install python-3.6.7. Let's install it.



Now, try to run python on the command prompt. Type the command **python** in case of python2 or python3 in case of **python3**. It will show an error as given in the below image. It is because we haven't set the path.

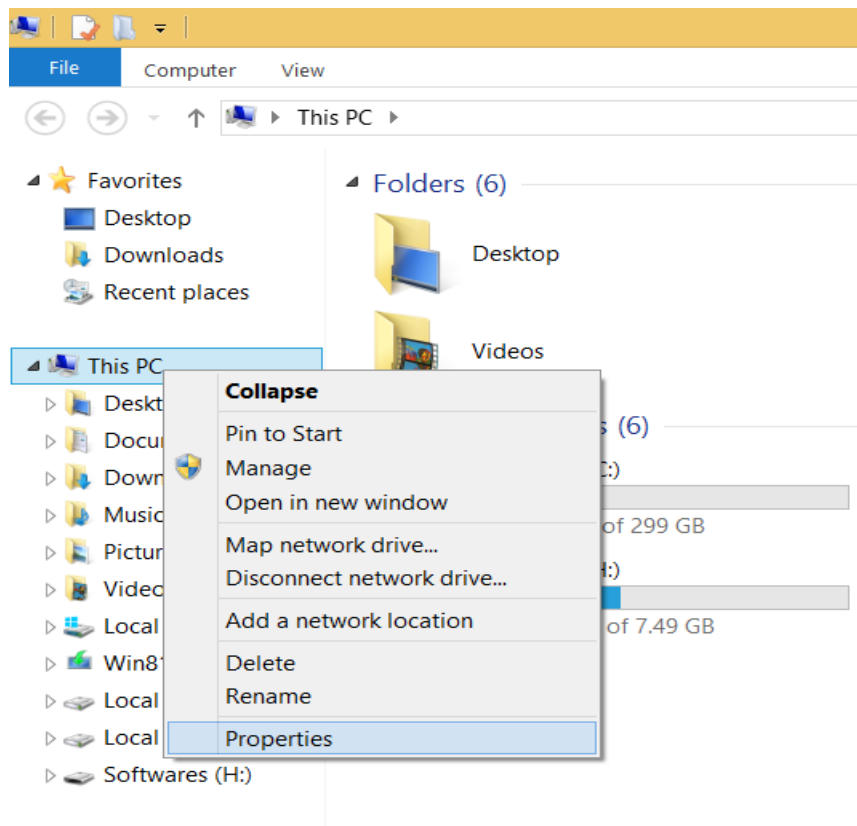


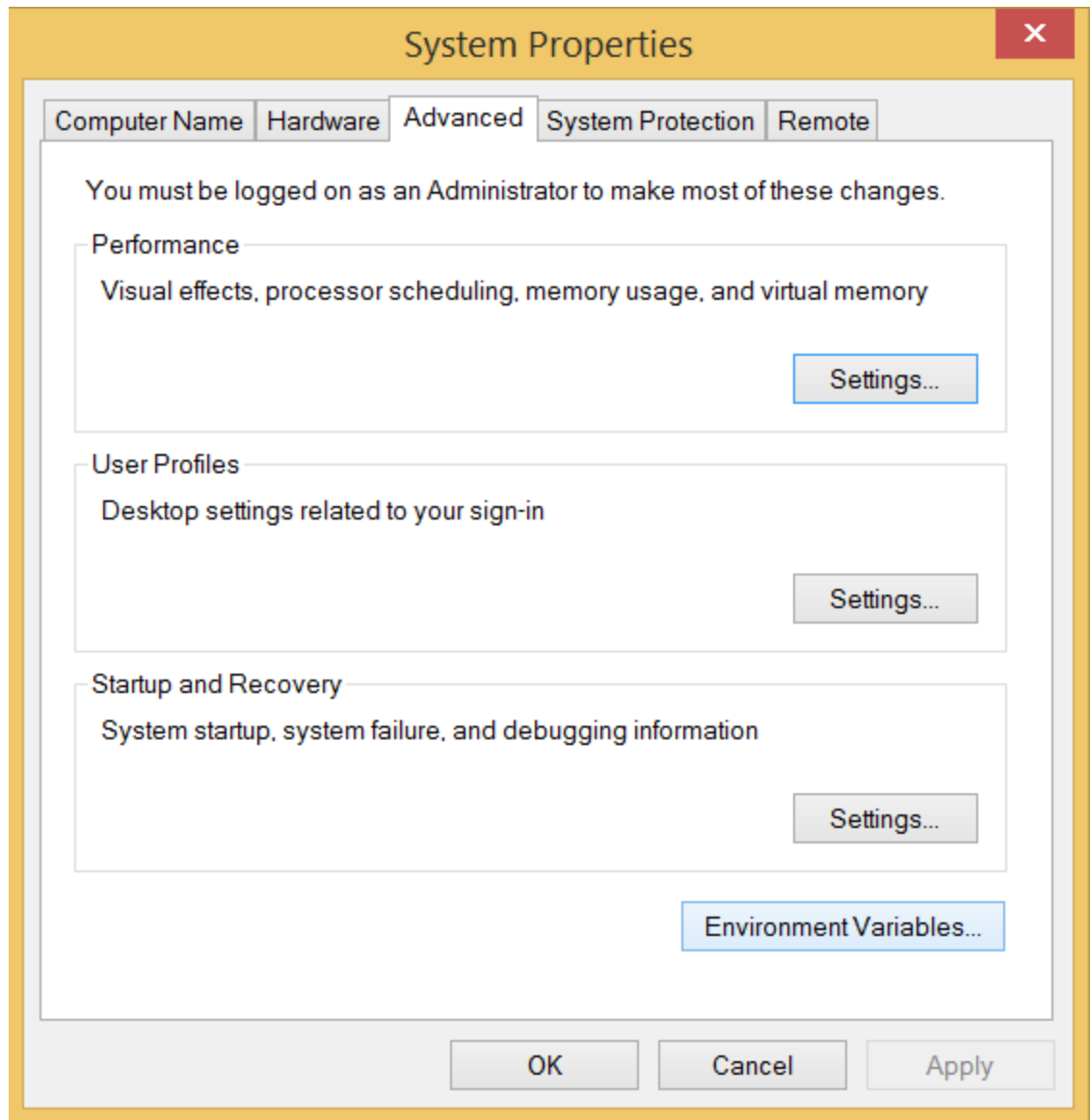
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\developer>python
'python' is not recognized as an internal or external command,
operable program or batch file.

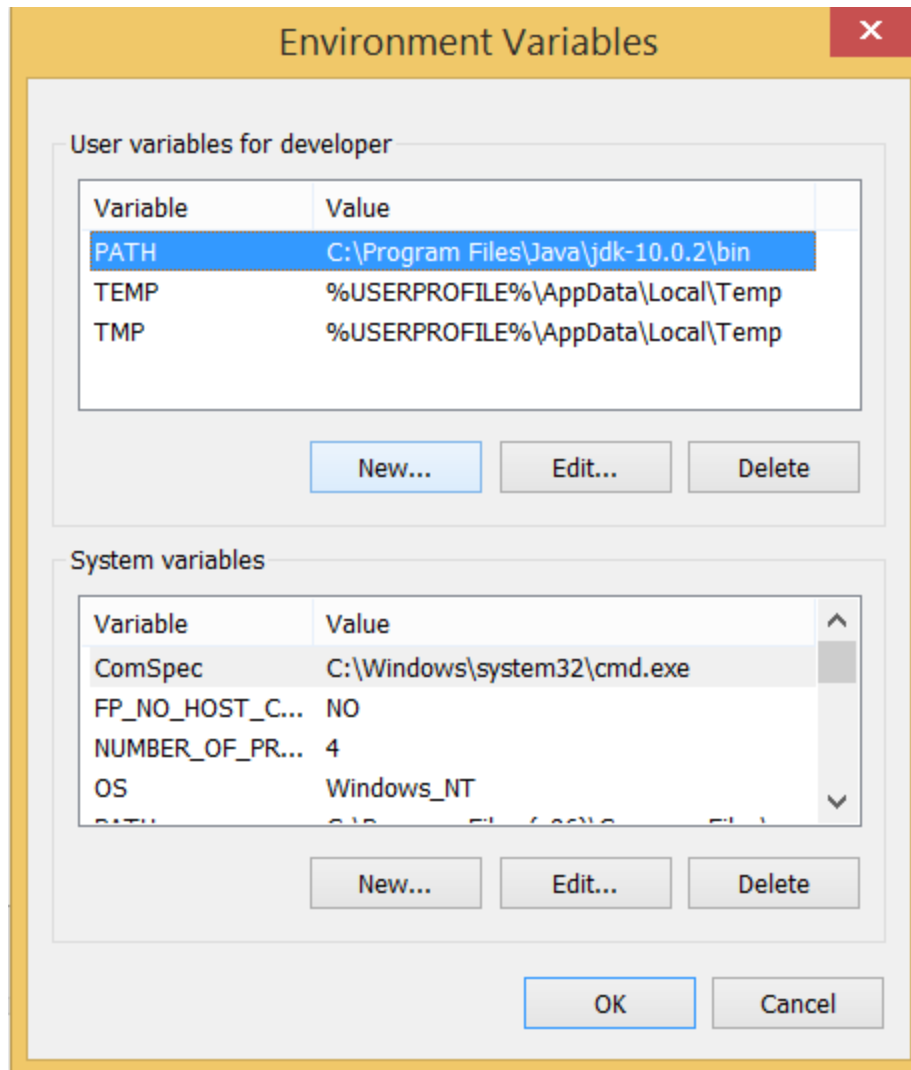
C:\Users\developer>_
```

To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.

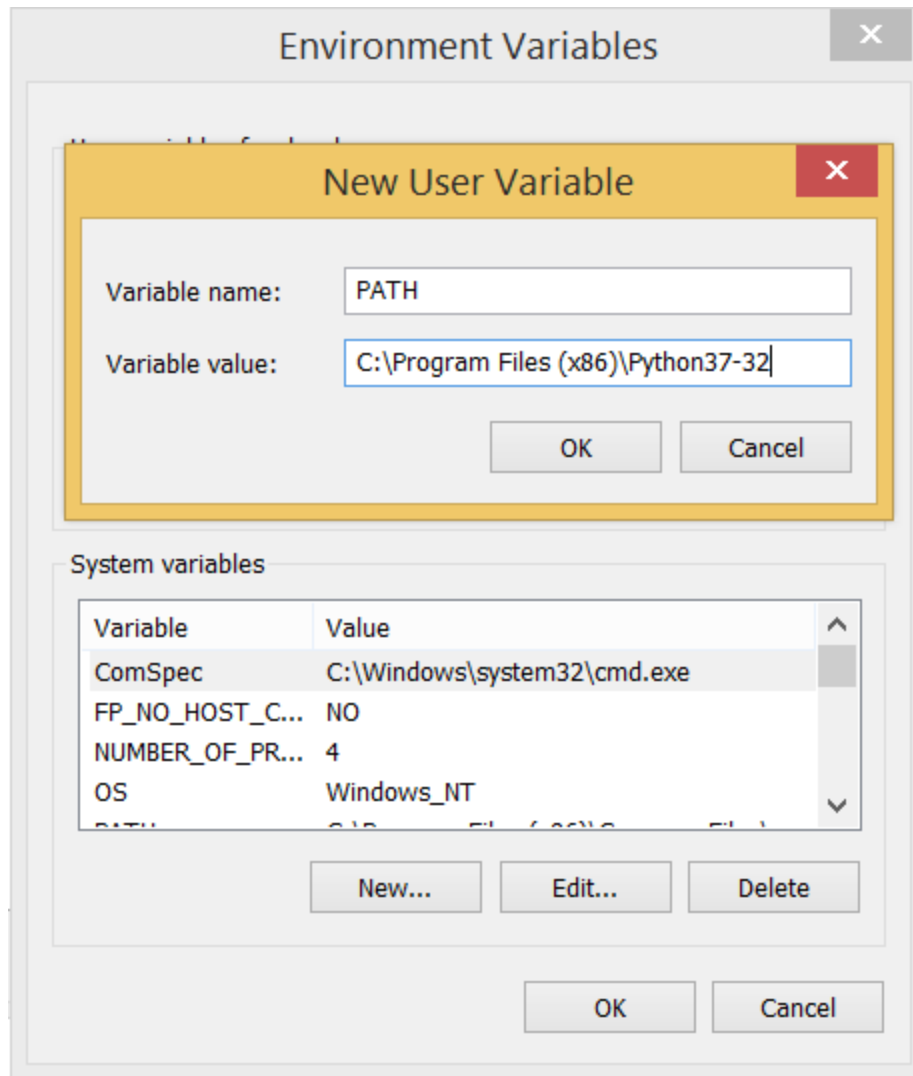




Add the new path variable in the user variable section.



Type **PATH** as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type **python** again. It will open the python interpreter shell where we can execute the python statements.

First Python Program

In this Section, we will discuss the basic syntax of python by using which, we will run a simple program to print hello world on the console.

Python provides us the two ways to run a program:

- Using Interactive interpreter prompt
- Using a script file

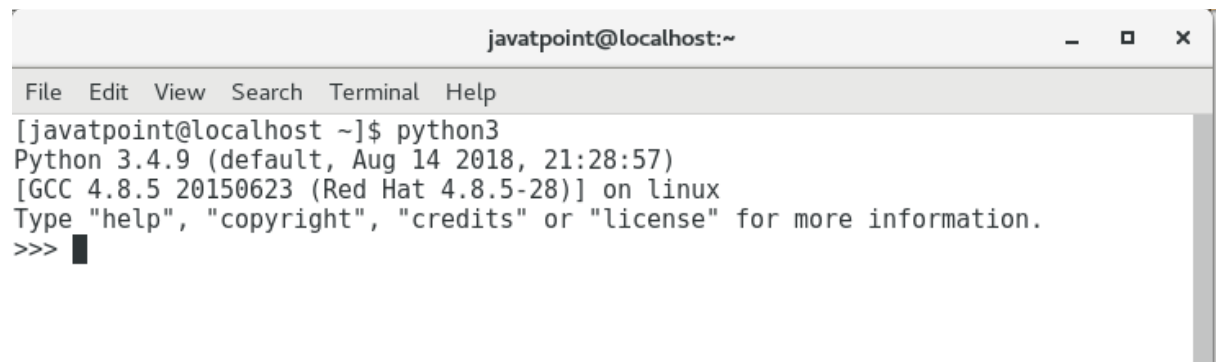
Let's discuss each one of them in detail.

Interactive interpreter prompt

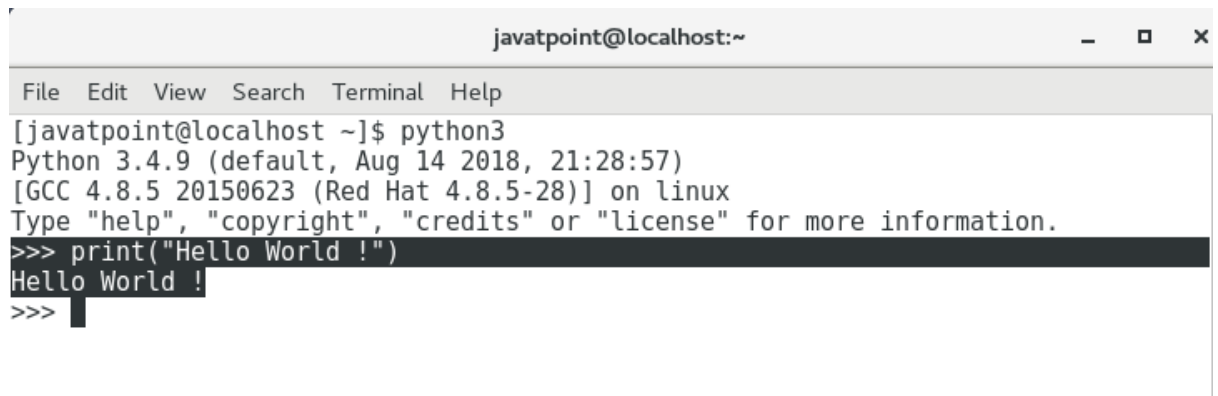
Python provides us the feature to execute the python statement one by one at the interactive prompt. It is preferable in the case where we are concerned about the output of each line of our python program.

To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have python2 and python3 both installed on your system).

It will open the following prompt where we can execute the python statement and check their impact on the console.

A screenshot of a terminal window titled 'javatpoint@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal content shows the command '[javatpoint@localhost ~]\$ python3' being executed. The output is 'Python 3.4.9 (default, Aug 14 2018, 21:28:57) [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux'. Below this, it says 'Type "help", "copyright", "credits" or "license" for more information.' and the prompt '>>>' is shown with a cursor.

Let's run a python statement to print the traditional hello world on the console. Python3 provides print() function to print some message on the console. We can pass the message as a string into this function. Consider the following image.

A screenshot of a terminal window titled 'javatpoint@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command 'python3' being executed, which displays the Python version (3.4.9), GCC version (4.8.5), and the operating system (Linux). It then prompts the user to type 'help', 'copyright', 'credits', or 'license' for more information. The user enters '>>> print("Hello World !")', and the terminal outputs 'Hello World !'. The prompt '>>>' is shown again on the next line.

```
javatpoint@localhost:~  
File Edit View Search Terminal Help  
[javatpoint@localhost ~]$ python3  
Python 3.4.9 (default, Aug 14 2018, 21:28:57)  
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("Hello World !")  
Hello World !  
>>>
```

Here, we get the message **"Hello World !"** printed on the console.

Using a script file

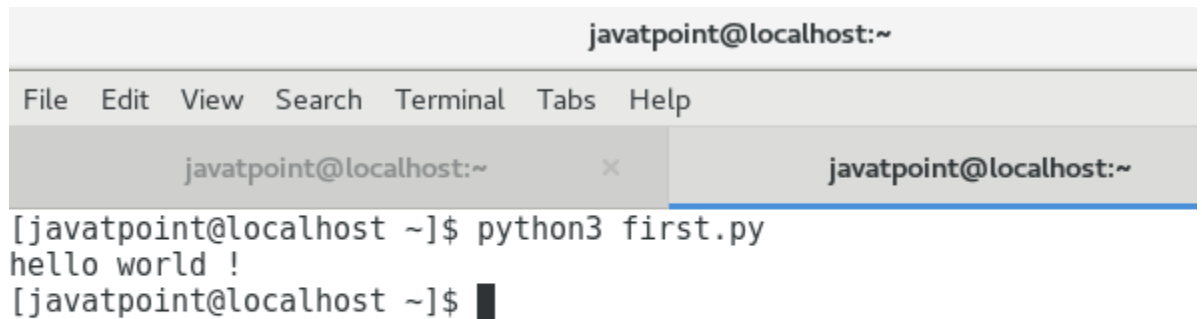
Interpreter prompt is good to run the individual statements of the code. However, we can not write the code every-time on the terminal.

We need to write our code into a file which can be executed later. For this purpose, open an editor like notepad, create a file named first.py (python used .py extension) and write the following code in it.

1. Print ("hello world"); #here, we have used print() function to print the message on the console.

To run this file named as first.py, we need to run the following command on the terminal.

\$ python3 first.py

A screenshot of a terminal window within the PyCharm IDE. The title bar at the top reads 'javatpoint@localhost:~'. Below it is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. The terminal area shows two tabs, both labeled 'javatpoint@localhost:~'. The active terminal window displays the command '[javatpoint@localhost ~]\$ python3 first.py' followed by the output 'hello world !' and a new prompt '[javatpoint@localhost ~]\$' with a cursor.

Hence, we get our output as the message **Hello World !** is printed on the console.

Get Started with PyCharm

In our first program, we have used gedit on our CentOS as an editor. On Windows, we have an alternative like notepad or notepad++ to edit the code. However, these editors are not used as IDE for python since they are unable to show the syntax related suggestions.

JetBrains provides the most popular and a widely used cross-platform IDE **PyCharm** to run the python programs.

PyCharm installation

As we have already stated, PyCharm is a cross-platform IDE, and hence it can be installed on a variety of the operating systems. In this section of the tutorial, we will cover the installation process of PyCharm on Windows, MacOS, CentOS, and Ubuntu.

Windows

Installing PyCharm on Windows is very simple. To install PyCharm on Windows operating system, visit the link <https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows> to download the executable installer. **Double click** the installer (.exe) file and install PyCharm by clicking next at each step.

Python Variables

Variable is a name which is used to refer memory location. Variable also known as identifier and used to hold value.

In Python, we don't need to specify the type of variable because Python is a type infer language and smart enough to get variable type.

Variable names can be a group of both letters and digits, but they have to begin with a letter or an underscore.

It is recommended to use lowercase letters for variable name. Rahul and rahul both are two different variables.

Identifier Naming

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive for example my name, and MyName is not the same.
- Examples of valid identifiers : a123, _n, n_9, etc.
- Examples of invalid identifiers: 1a, n%4, n 9, etc.

Declaring Variable and Assigning Values

Python does not bound us to declare variable before using in the application. It allows us to create variable at required time.

We don't need to declare explicitly variable in Python. When we assign any value to the variable that variable is declared automatically.

The equal (=) operator is used to assign value to a variable.

Eg:

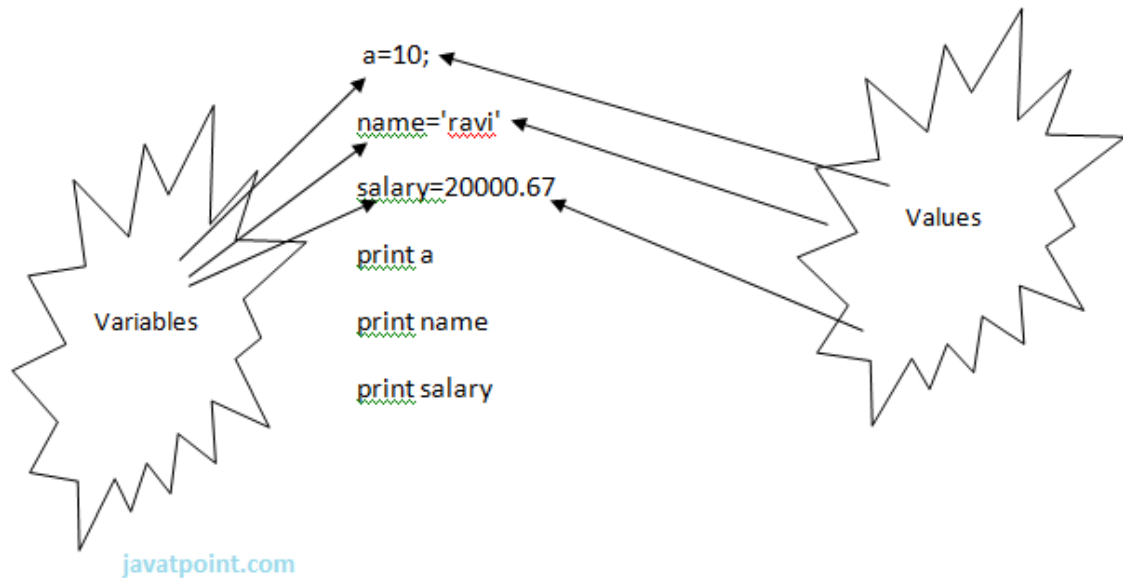


Fig:5.Declaring variable and Assigning values

Output:

1. >>>
2. 10
3. ravi
4. 20000.67
5. >>>

Multiple Assignment

Python allows us to assign a value to multiple variables in a single statement which is also known as multiple assignment.

We can apply multiple assignments in two ways either by assigning a single value to multiple variables or assigning multiple values to multiple variables. Lets see given examples.

1. Assigning single value to multiple variables

Eg:

1. `x=y=z=50`
2. `print x`
3. `print y`
4. `print z`

Output:

1. `>>>`
2. `50`
3. `50`
4. `50`
5. `>>>`

2.Assigning multiple values to multiple variables:

Eg:

1. `a,b,c=5,10,15`
2. `print a`
3. `print b`
4. `print c`

Output:

1. `>>>`
2. `5`
3. `10`
4. `15`
5. `>>>`

The values will be assigned in the order in which variables appears.

Basic Fundamentals:

This section contains the basic fundamentals of Python like :

i)Tokens and their types.

ii) Comments

a) Tokens:

- Tokens can be defined as a punctuator mark, reserved words and each individual word in a statement.
- Token is the smallest unit inside the given program.

There are following tokens in Python:

- Keywords.
- Identifiers.
- Literals.
- Operators.

Tuples:

- Tuple is another form of collection where different type of data can be stored.
- It is similar to list where data is separated by commas. Only the difference is that list uses square bracket and tuple uses parenthesis.
- Tuples are enclosed in parenthesis and cannot be changed.

Eg:

1. `>>> tuple=('rahul',100,60.4,'deepak')`
2. `>>> tuple1=('sanjay',10)`
3. `>>> tuple`
4. `('rahul', 100, 60.4, 'deepak')`
5. `>>> tuple[2:]`
6. `(60.4, 'deepak')`
7. `>>> tuple1[0]`
8. `'sanjay'`
9. `>>> tuple+tuple1`
10. `('rahul', 100, 60.4, 'deepak', 'sanjay', 10)`
11. `>>>`

Dictionary:

- Dictionary is a collection which works on a key-value pair.
- It works like an associated array where no two keys can be same.

- Dictionaries are enclosed by curly braces ({}) and values can be retrieved by square bracket([]).

Eg:

1. >>> dictionary={'name':'charlie','id':100,'dept':'it'}
2. >>> dictionary
3. {'dept': 'it', 'name': 'charlie', 'id': 100}
4. >>> dictionary.keys()
5. ['dept', 'name', 'id']
6. >>> dictionary.values()
7. ['it', 'charlie', 100]
8. >>>

Python Data Types

Variables can hold values of different data types. Python is a dynamically typed language hence we need not define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

Python enables us to check the type of the variable used in the program. Python provides us the **type()** function which returns the type of the variable passed.

Consider the following example to define the values of different data types and checking its type.

1. A=10
2. b="Hi Python"
3. c = 10.5
4. **print**(type(a));
5. **print**(type(b));
6. **print**(type(c));

Output:

```
<type 'int'>
<type 'str'>
<type 'float'>
```

Standard data types

A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.

Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. [Numbers](#)
2. [String](#)
3. [List](#)
4. [Tuple](#)
5. [Dictionary](#)

In this section of the tutorial, we will give a brief introduction of the above data types. We will discuss each one of them in detail later in this tutorial.

Numbers

Number stores numeric values. Python creates Number objects when a number is assigned to a variable. For example;

1. `a = 3 , b = 5` #a and b are number objects

Python supports 4 types of numeric data.

1. `int` (signed integers like 10, 2, 29, etc.)
2. `long` (long integers used for a higher range of values like 908090800L, -0x1929292L, etc.)
3. `float` (float is used to store floating point numbers like 1.9, 9.902, 15.2, etc.)
4. `complex` (complex numbers like 2.14j, 2.0 + 2.3j, etc.)

Python allows us to use a lower-case L to be used with long integers. However, we must always use an upper-case L to avoid confusion.

A complex number contains an ordered pair, i.e., $x + iy$ where x and y denote the real and imaginary parts respectively).

String

The string can be defined as the sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.

String handling in python is a straightforward task since there are various inbuilt functions and operators provided.

In the case of string handling, the operator + is used to concatenate two strings as the operation *"hello"+" python"* returns *"hello python"*.

The operator * is known as repetition operator as the operation *"Python " *2* returns *"Python Python "*.

The following example illustrates the string handling in python.

1. `str1 = 'hello javatpoint' #string str1`
2. `str2 = ' how are you' #string str2`
3. `print (str1[0:2])` #printing first two character using slice operator
4. `print (str1[4])` #printing 4th character of the string
5. `print (str1*2)` #printing the string twice
6. `print (str1 + str2)` #printing the concatenation of str1 and str2

Output:

```
he
o
hello javatpointhellojavatpoint
hello javatpoint how are you
```

List

Lists are similar to arrays in C. However; the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

Consider the following example.

1. `l = [1, "hi", "python", 2]`
2. `print (l[3:]);`
3. `print (l[0:2]);`
4. `print (l);`
5. `print (l + l);`
6. `print (l * 3);`

Output:

```
[2]
[1, 'hi']
[1, 'hi', 'python', 2]
[1, 'hi', 'python', 2, 1, 'hi', 'python', 2]
[1, 'hi', 'python', 2, 1, 'hi', 'python', 2, 1, 'hi', 'python', 2]
```

Tuple

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

Let's see a simple example of the tuple.

1. `t = ("hi", "python", 2)`
2. `print (t[1:]);`
3. `print (t[0:1]);`
4. `print (t);`
5. `print (t + t);`
6. `print (t * 3);`
7. `print (type(t))`
8. `t[2] = "hi";`

Output:

```
('python', 2)
('hi',)
('hi', 'python', 2)
('hi', 'python', 2, 'hi', 'python', 2)
('hi', 'python', 2, 'hi', 'python', 2, 'hi', 'python', 2)
<type 'tuple'>
Traceback (most recent call last):
  File "main.py", line 8, in <module>
    t[2] = "hi";
TypeError: 'tuple' object does not support item assignment
```

Dictionary

Dictionary is an ordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma and enclosed in the curly braces {}.

Consider the following example.

1. `d = { 1:'Jimmy', 2:'Alex', 3:'john', 4:'mike' };`
2. `print("1st name is "+d[1]);`
3. `print("2nd name is "+ d[4]);`
4. `print (d);`
5. `print (d.keys());`
6. `print (d.values());`

Output:

```
1st name is Jimmy
2nd name is mike
{1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}
[1, 2, 3, 4]
```

```
['Jimmy', 'Alex', 'john', 'mike']
```

Python Functions

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code which can be called whenever required.

Python allows us to divide a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the python program.

In other words, we can say that the collection of functions creates a program. The function is also known as procedure or subroutine in other programming languages.

Python provide us various inbuilt functions like range() or print(). Although, the user can create its functions which can be called user-defined functions.

Advantage of functions in python

There are the following advantages of C functions.

- By using functions, we can avoid rewriting same logic/code again and again in a program.
- We can call python functions any number of times in a program and from any place in a program.
- We can track a large python program easily when it is divided into multiple functions.
- Reusability is the main achievement of python functions.
- However, Function calling is always overhead in a python program.

Creating a function

In python, we can use **def** keyword to define the function. The syntax to define a function in python is given below.

1. **def** my_function():
2. function-suite
3. **return** <expression>

The function block is started with the colon (:) and all the same level block statements remain at the same indentation.

A function can accept any number of parameters that must be the same in the definition and function calling.

Function calling

In python, a function must be defined before the function calling otherwise the python interpreter gives an error. Once the function is defined, we can call it from another function or the python prompt. To call the function, use the function name followed by the parentheses.

A simple function that prints the message "Hello Word" is given below.

1. **def** hello_world():
2. **print**("hello world")
- 3.
4. hello_world()

Output:

hello world

Parameters in function

The information into the functions can be passed as the parameters. The parameters are specified in the parentheses. We can give any number of parameters, but we have to separate them with a comma.

Consider the following example which contains a function that accepts a string as the parameter and prints it.

Example 1

1. #defining the function
2. **def** func (name):
3. **print**("Hi ",name);
- 4.
5. #calling the function
6. func("Ayush")

Example 2

1. #python function to calculate the sum of two variables
2. #defining the function


```

3. def sum (a,b):
4.     return a+b;
5.
6. #taking values from the user
7. a = int(input("Enter a: "))
8. b = int(input("Enter b: "))
9.
10. #printing the sum of a and b
11. print("Sum = ",sum(a,b))

```

Output:

Enter a: 10

Enter b: 20

Sum = 30

[Call by reference in Python](#)

In python, all the functions are called by reference, i.e., all the changes made to the reference inside the function revert back to the original value referred by the reference.

However, there is an exception in the case of mutable objects since the changes made to the mutable objects like string do not revert to the original string rather, a new string object is made, and therefore the two different objects are printed.

[Example 1 Passing Immutable Object \(List\)](#)

```

1. #defining the function
2. def change_list(list1):
3.     list1.append(20);
4.     list1.append(30);
5.     print("list inside function = ",list1)
6.
7. #defining the list
8. list1 = [10,30,40,50]
9.
10. #calling the function
11. change_list(list1);
12. print("list outside function = ",list1);

```

Output:

list inside function = [10, 30, 40, 50, 20, 30]

list outside function = [10, 30, 40, 50, 20, 30]

Example 2 Passing Mutable Object (String)

1. #defining the function
2. **def** change_string (str):
3. str = str + " Hows you";
4. **print**("printing the string inside function :",str);
- 5.
6. string1 = "Hi I am there"
- 7.
8. #calling the function
9. change_string(string1)
- 10.
11. **print**("printing the string outside function :",string1)

Output:

printing the string inside function : Hi I am there Hows you

printing the string outside function : Hi I am there

Types of arguments

There may be several types of arguments which can be passed at the time of function calling.

1. Required arguments
2. Keyword arguments
3. Default arguments
4. Variable-length arguments

Required Arguments

Till now, we have learned about function calling in python. However, we can provide the arguments at the time of function calling. As far as the required arguments are concerned, these are the arguments which are required to be passed at the time of function calling with the exact match of their positions in the function call and function definition. If either of the arguments is not provided in the function call, or

the position of the arguments is changed, then the python interpreter will show the error.

Consider the following example.

Example 1

1. #the argument name is the required argument to the function func
2. **def** func(name):
3. message = "Hi "+name;
4. **return** message;
5. name = input("Enter the name?")
6. **print**(func(name))

Output:

Enter the name?John

Hi John

Example 2

1. #the function simple_interest accepts three arguments and returns the simple interest accordingly
2. **def** simple_interest(p,t,r):
3. **return** (p*t*r)/100
4. p = float(input("Enter the principle amount? "))
5. r = float(input("Enter the rate of interest? "))
6. t = float(input("Enter the time in years? "))
7. **print**("Simple Interest: ",simple_interest(p,r,t))

Output:

Enter the principle amount? 10000

Enter the rate of interest? 5

Enter the time in years? 2

Simple Interest: 1000.0

Example 3

1. #the function calculate returns the sum of two arguments a and b
2. **def** calculate(a,b):
3. **return** a+b
4. calculate(10) # this causes an error as we are missing a required arguments b.

Output:

TypeError: calculate() missing 1 required positional argument: 'b'

Keyword arguments

Python allows us to call the function with the keyword arguments. This kind of function call will enable us to pass the arguments in the random order.

The name of the arguments is treated as the keywords and matched in the function calling and definition. If the same match is found, the values of the arguments are copied in the function definition.

Consider the following example.

Example 1

1. #function func is called with the name and message as the keyword arguments
2. **def** func(name,message):
3. **print**("printing the message with",name,"and ",message)
4. func(name = "John",message="hello") #name and message is copied with the values John and hello respectively

Output:

printing the message with John and hello

Example 2 providing the values in different order at the calling

1. #The function simple_interest(p, t, r) is called with the keyword arguments the order of arguments doesn't matter in this case
2. **def** simple_interest(p,t,r):
3. **return** (p*t*r)/100
4. **print**("Simple Interest: ",simple_interest(t=10,r=10,p=1900))

Output:

Simple Interest: 1900.0

If we provide the different name of arguments at the time of function call, an error will be thrown.

Consider the following example.

Example 3

1. #The function simple_interest(p, t, r) is called with the keyword arguments.
2. **def** simple_interest(p,t,r):
3. **return** (p*t*r)/100
- 4.
5. **print**("Simple Interest: ",simple_interest(time=10,rate=10,principle=1900)) # doesn't find the exact match of the name of the arguments (keywords)

Output:

TypeError: simple_interest() got an unexpected keyword argument 'time'

The python allows us to provide the mix of the required arguments and keyword arguments at the time of function call. However, the required argument must not be given after the keyword argument, i.e., once the keyword argument is encountered in the function call, the following arguments must also be the keyword arguments.

Consider the following example.

Example 4

1. **def** func(name1,message,name2):
2. **print**("printing the message with",name1,",",message,"and",name2)
3. func("John",message="hello",name2="David") #the first argument is not the keyword argument

Output:

printing the message with John , hello ,and David

The following example will cause an error due to an in-proper mix of keyword and required arguments being passed in the function call.

Example 5

1. **def** func(name1,message,name2):
2. **print**("printing the message with",name1,",",message,"and",name2)
3. func("John",message="hello","David")

Output:

SyntaxError: positional argument follows keyword argument

Default Arguments

Python allows us to initialize the arguments at the function definition. If the value of any of the argument is not provided at the time of function call, then that argument can be initialized with the value given in the definition even if the argument is not specified at the function call.

Example 1

1. **def** printme(name,age=22):
2. **print**("My name is",name,"and age is",age)
3. printme(name = "john") #the variable age is not passed into the function however the default value of age is considered in the function

Output:

My name is john and age is 22

Example 2

1. **def** printme(name,age=22):
2. **print**("My name is",name,"and age is",age)
3. printme(name = "john") #the variable age is not passed into the function however the default value of age is considered in the function
4. printme(age = 10,name="David") #the value of age is overwritten here, 10 will be printed as a ge

Output:

My name is john and age is 22

My name is David and age is 10

Variable length Arguments

In the large projects, sometimes we may not know the number of arguments to be passed in advance. In such cases, Python provides us the flexibility to provide the comma separated values which are internally treated as tuples at the function call.

However, at the function definition, we have to define the variable with * (star) as *<variable - name >.

Consider the following example.

Example

1. **def** printme(*names):
2. **print**("type of passed argument is ",type(names))
3. **print**("printing the passed arguments...")
4. **for** name **in** names:
5. **print**(name)
6. printme("john","David","smith","nick")

Output:

```
type of passed argument is <class 'tuple'>
printing the passed arguments...
john
David
smith
nick
```

Scope of variables

The scopes of the variables depend upon the location where the variable is being declared. The variable declared in one part of the program may not be accessible to the other parts.

In python, the variables are defined with the two types of scopes.

1. Global variables
2. Local variables

The variable defined outside any function is known to have a global scope whereas the variable defined inside a function is known to have a local scope.

Consider the following example.

Example 1

1. **def** print_message():
2. message = "hello !! I am going to print a message." # the variable message is local to the function itself
3. **print**(message)
4. print_message()

5. **print**(message) # this will cause an error since a local variable cannot be accessible here.

Output:

hello !! I am going to print a message.

File "/root/PycharmProjects/PythonTest/Test1.py", line 5, in

print(message)

NameError: name 'message' is not defined

Example 2

1. **def** calculate(*args):
2. sum=0
3. **for** arg **in** args:
4. sum = sum +arg
5. **print**("The sum is",sum)
6. sum=0
7. calculate(10,20,30) #60 will be printed as the sum
8. **print**("Value of sum outside the function:",sum) # 0 will be printed

Output:

The sum is 60

Value of sum outside the function: 0

Python Class and Objects

As we have already discussed, a class is a virtual entity and can be seen as a blueprint of an object. The class came into existence when it is instantiated. Let's understand it by an example.

Suppose a class is a prototype of a building. A building contains all the details about the floor, doors, windows, etc. we can make as many buildings as we want, based on these details. Hence, the building can be seen as a class, and we can create as many objects of this class.

On the other hand, the object is the instance of a class. The process of creating an object can be called as instantiation.

In this section of the tutorial, we will discuss creating classes and objects in python. We will also talk about how an attribute is accessed by using the class object.

Creating classes in python

In python, a class can be created by using the keyword `class` followed by the class name. The syntax to create a class is given below.

Syntax

1. **class** ClassName:
2. #statement_suite

In python, we must notice that each class is associated with a documentation string which can be accessed by using `<class-name>.__doc__`. A class contains a statement suite including fields, constructor, function, etc. definition.

Consider the following example to create a class `Employee` which contains two fields as `Employee id`, and `name`.

The class also contains a function `display()` which is used to display the information of the `Employee`.

Example

1. **class** Employee:
2. `id = 10;`
3. `name = "ayush"`
4. **def** display (self):
5. **print**(self.id,self.name)

Here, the `self` is used as a reference variable which refers to the current class object. It is always the first argument in the function definition. However, using `self` is optional in the function call.

Creating an instance of the class

A class needs to be instantiated if we want to use the class attributes in another class or method. A class can be instantiated by calling the class using the class name.

The syntax to create the instance of the class is given below.

1. `<object-name> = <class-name>(<arguments>)`

The following example creates the instance of the class Employee defined in the above example.

Example

1. **class** Employee:
2. id = 10;
3. name = "John"
4. **def** display (self):
5. **print**("ID: %d \nName: %s"%(self.id,self.name))
6. emp = Employee()
7. emp.display()

Output:

ID: 10

CODE GENERATION AND VALIDATION

Previously, when using manual generation of code, verification and validation is required to assure the 'quality' of the code. Techniques that have been used to verify and validate manual code include: reviewing, module testing, integration testing, system testing, and static analysis. These techniques help to find errors that engineers make when developing code.

SOURCE CODE:

<https://www.kaggle.com/pradheeprio/malaria-cell-image-detection-using-cnn-acc-95>

images link

<https://data.mendeley.com/datasets/hb74ynkjc/1>

```
import tensorflow as tf
```

```
from tensorflow.keras import Sequential
```

```

from tensorflow.keras.layers import
Conv2D,MaxPool2D,Dropout,Flatten,Dense,BatchNormalization

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.preprocessing import image

import numpy as np

import matplotlib.pyplot as plt

import cv2

import os

import argparse

#####

parser = argparse.ArgumentParser(description='Use this script to run age and gender
recognition using OpenCV.')

parser.add_argument('--input', help='Path to input image or video file. Skip this argument
to capture frames from a camera.')

args = parser.parse_args()

image1=args.input

print('imageeeeeeeeeeeeeeeeeeeee')

print(image1)

'''

```

```
for i in range(5):
```

```
    img=cv2.imread('../input/cell-images-for-detecting-  
malaria/cell_images/Parasitized/'+Parasitized_cell[i])
```

```
    plt.imshow(img)
```

```
    plt.title("Parasitized")
```

```
    plt.show()
```

```
for i in range(5):
```

```
    img=cv2.imread('../input/cell-images-for-detecting-  
malaria/cell_images/Uninfected/'+uninfected_cell[i])
```

```
    plt.imshow(img)
```

```
    plt.title("Uninfected")
```

```
    plt.show()'''
```

```
if (image1.startswith('papaya')):
```

```
    width = 68
```

```
    height = 68
```

```
    datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.2)
```

```
    trainDatagen
```

```
=
```

```
    datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data1/  
test/',target_size=(width,height),class_mode = 'binary', batch_size = 16,subset='training')
```

```
    trainDatagen.class_indices
```

```

valDatagen =
datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data1/
test/',target_size=(width,height),class_mode = 'binary',batch_size = 16,
subset='validation')

model = Sequential()

model.add(Conv2D(16,(3,3),activation='relu',input_shape=(width,height,3)))

model.add(MaxPool2D(2,2))

model.add(Dropout(0.2))

model.add(Conv2D(64,(3,3),activation='relu'))

model.add(MaxPool2D(2,2))

model.add(Dropout(0.3))

model.add(Flatten())

model.add(Dense(64,activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(1,activation='sigmoid'))

model.summary()

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

history=model.fit_generator(generator=trainDatagen,steps_per_epoch=len(trainD
atagen),epochs=1,validation_data=valDatagen ,validation_steps=len(valDatagen ))

```

```
plt.plot(history.history['acc'])

plt.plot(history.history['val_acc'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epochs')

plt.legend(['train', 'test'], loc='upper left')

plt.show()
```

```
# summarize history for loss

plt.plot(history.history['loss'])

plt.plot(history.history['val_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epochs')

plt.legend(['train', 'test'], loc='upper left')

plt.show()
```

```

testing_path="E:/2019basepapers/TrueVolts/fruitdisease/images/"+image1

img=image.load_img(testimg_path,target_size=(68,68))

plt.imshow(img)

plt.imshow(img)


x=image.img_to_array(img)

x=np.expand_dims(x,axis=0)

images=np.vstack([x])

val=model.predict(images)

print('-----')

print(val)

if val[0][0]==0:

    #img=image.load_img(testimg_path,target_size=(68,68))

    plt.title("anthracnose DISEASE")

    #####

    import matplotlib.pyplot as plt

    import matplotlib.image as mpimg

    img = mpimg.imread(testimg_path)

```

```

imgplot = plt.imshow(img)

plt.show()

#####

print('0000000000000000000000000000000000000000000000000000000')

else:

    plt.title("ring_spot")

    #####

    import matplotlib.pyplot as plt

    import matplotlib.image as mpimg

    img = mpimg.imread(testimg_path)

    imgplot = plt.imshow(img)

    plt.show()

elif (image1.startswith("banana")):

    width = 68

    height = 68

    datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.2)

```



```

trainDatagen =
datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data2/
test/',target_size=(width,height),class_mode = 'binary', batch_size = 16,subset='training')

trainDatagen.class_indices

valDatagen =
datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data2/
test/',target_size=(width,height),class_mode = 'binary',batch_size = 16,
subset='validation')

model = Sequential()

model.add(Conv2D(16,(3,3),activation='relu',input_shape=(width,height,3)))

model.add(MaxPool2D(2,2))

model.add(Dropout(0.2))

model.add(Conv2D(64,(3,3),activation='relu'))

model.add(MaxPool2D(2,2))

model.add(Dropout(0.3))

model.add(Flatten())

model.add(Dense(64,activation='relu'))

model.add(Dropout(0.5))

```

```

model.add(Dense(1,activation='sigmoid'))

model.summary()

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])


history=model.fit_generator(generator=trainDatagen,steps_per_epoch=len(trainD
atagen),epochs=1,validation_data=valDatagen ,validation_steps=len(valDatagen ))


plt.plot(history.history['acc'])

plt.plot(history.history['val_acc'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epochs')

plt.legend(['train', 'test'], loc='upper left')

plt.show()


# summarize history for loss

plt.plot(history.history['loss'])

plt.plot(history.history['val_loss'])

plt.title('model loss')

```

```
plt.ylabel('loss')
```

```
plt.xlabel('epochs')
```

```
plt.legend(['train', 'test'], loc='upper left')
```

```
plt.show()
```

```
testing_path="E:/2019basepapers/TrueVolts/fruitdisease/images/"+image1
```

```
img=image.load_img(testing_path,target_size=(68,68))
```

```
plt.imshow(img)
```

```
plt.imshow(img)
```

```
x=image.img_to_array(img)
```

```
x=np.expand_dims(x,axis=0)
```

```
images=np.vstack([x])
```

```
val=model.predict(images)
```

```
print('-----')
```

```
print(val)
```

```
if val[0][0]==0:
```

```
    #img=image.load_img(testing_path,target_size=(68,68))
```

```
plt.title("BANANA scab DISEASE")
```

```
#####
```

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

```
import matplotlib.image as mpimg
```

```
img = mpimg.imread(testimg_path)
```

```
imgplot = plt.imshow(img)
```

```
plt.show()
```

```
#####
```

```
print('0000000000000000000000000000000000000000')
```

```
else:
```

```
plt.title("Healthy BANANA")
```

```
#####
```

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

```
import matplotlib.image as mpimg
```

```
img = mpimg.imread(testimg_path)
```

```
imgplot = plt.imshow(img)
```

```
plt.show()
```

```
else:
```

```
width = 68
```

```
height = 68
```

```
datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.2)
```

```
trainDatagen =  
datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data/t  
est/',target_size=(width,height),class_mode = 'binary', batch_size = 16,subset='training')
```

```
trainDatagen.class_indices
```

```
valDatagen =  
datagen.flow_from_directory(directory='E:/2019basepapers/TrueVolts/fruitdisease/data/t  
est/',target_size=(width,height),class_mode = 'binary',batch_size = 16,  
subset='validation')
```

```
model = Sequential()
```

```
model.add(Conv2D(16,(3,3),activation='relu',input_shape=(width,height,3)))
```

```
model.add(MaxPool2D(2,2))
```

```
model.add(Dropout(0.2))
```

```
model.add(Conv2D(64,(3,3),activation='relu'))
```

```
model.add(MaxPool2D(2,2))
```

```
model.add(Dropout(0.3))
```

```
model.add(Flatten())
```

```
model.add(Dense(64,activation='relu'))
```

```
model.add(Dropout(0.5))
```

```
model.add(Dense(1,activation='sigmoid'))
```

```
model.summary()
```

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history=model.fit_generator(generator=trainDatagen,steps_per_epoch=len(trainD  
atagen),epochs=1,validation_data=valDatagen ,validation_steps=len(valDatagen ))
```

```
plt.plot(history.history['acc'])

plt.plot(history.history['val_acc'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epochs')

plt.legend(['train', 'test'], loc='upper left')

plt.show()
```

```
# summarize history for loss

plt.plot(history.history['loss'])

plt.plot(history.history['val_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epochs')

plt.legend(['train', 'test'], loc='upper left')

plt.show()
```

```
testing_path="E:/2019basepapers/TrueVolts/fruitdisease/images/"+image1
```

```
img=image.load_img(testimg_path,target_size=(68,68))
```

```
plt.imshow(img)
```

```
plt.imshow(img)
```

```
x=image.img_to_array(img)
```

```
x=np.expand_dims(x,axis=0)
```

```
images=np.vstack([x])
```

```
val=model.predict(images)
```

```
print('-----')
```

```
print(val)
```

```
if val[0][0]==0:
```

```
#img=image.load_img(testimg_path,target_size=(68,68))
```

```
plt.title("SCAB DISEASE")
```

```
#####
```

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



```

import matplotlib.image as mpimg

img = mpimg.imread(testimg_path)

imgplot = plt.imshow(img)

plt.show()

#####

print('0000000000000000000000000000000000000000000000000000000')

else:

plt.title("Healthy")

#####

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

img = mpimg.imread(testimg_path)

imgplot = plt.imshow(img)

plt.show()

```

UNIT & INTEGRATION TESTING, DEBUGGING

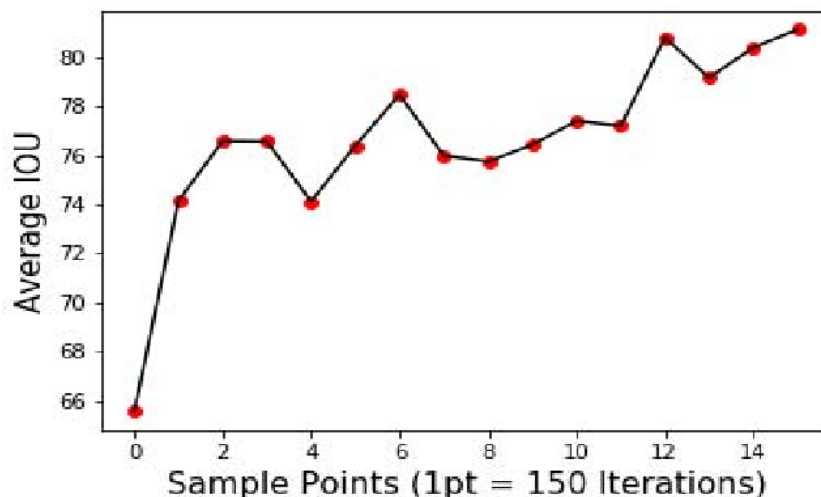
Unit testing

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

Integration testing

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

RESULT ANALYSIS



8. TESTING

DETAILED DESCRIPTION OF TESTING TOOLS

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing presents an interesting anomaly of the software. During earlier definition and development phases, it was attempted to build software from abstract concept to a tangible implementation.

The testing phase involves the testing of the developed system using various set data. Presentation of test data plays a vital role in system testing. After preparing the test data the system under study was tested using test data. While testing the system by

What is Web Testing?

Web testing is a software testing practice to test the websites or web applications for potential bugs. It's a complete testing of web-based applications before making live. A web-based system needs to be checked completely from end-to-end before it goes live for end users.

By performing website testing, an organization can make sure that the web-based system is functioning properly and can be accepted by real-time users.

The UI design and functionality are the captains of website testing.

Web testing checklists

- 1) Functionality Testing
- 2) Usability testing
- 3) Interface testing
- 4) Compatibility testing
- 5) Performance testing
- 6) Security testing

#1) Functionality Testing

Test for – all the links in web pages, database connection, forms used for submitting or getting information from the user in the web pages, Cookie testing etc.

Check all the links:

Test the outgoing links from all the pages to the specific domain under test.

Test all internal links.

Test links jumping on the same pages.

Test links used to send email to admin or other users from web pages.

Test to check if there are any orphan pages.

Finally, link checking includes, check for broken links in all above-mentioned links.

Test forms on all pages:

Forms are an integral part of any website. Forms are used for receiving information from users and to interact with them. So what should be checked in these forms?

First, check all the validations on each field.

Check for default values of the fields.

Wrong inputs in the forms to the fields in the forms.

Options to create forms if any, form delete, view or modify the forms.

Let's take an example of the search engine project currently I am working on, in this project we have advertiser and affiliate signup steps. Each sign-up step is different but its dependent on the other steps.

So sign up flow should get executed correctly. There are different field validations like email Ids, User financial info validations etc. All these validations should get checked in manual or automated web testing.

Cookies Testing:

Cookies are small files stored on the user machine. These are basically used to maintain the session- mainly the login sessions. Test the application by enabling or disabling the cookies in your browser options.

Test if the cookies are encrypted before writing to the user machine. If you are testing the session cookies (i.e. cookies that expire after the session ends) check for login sessions and user stats after the session ends. Check effect on application security by deleting the cookies. (I will soon write a separate article on cookie testing as well)

Validate your HTML/CSS:

If you are optimizing your site for Search engines then HTML/CSS validation is the most important one. Mainly validate the site for HTML syntax errors. Check if the site is crawlable to different search engines.

Database testing:

Data consistency is also very important in a web application. Check for data integrity and errors while you edit, delete, modify the forms or do any DB related functionality.

Check if all the database queries are executing correctly, data is retrieved and also updated correctly. More on database testing could be a load on DB, we will address this in web load or performance testing below.

In testing the functionality of the websites the following should be tested:

Links

- i. Internal Links
- ii. External Links
- iii. Mail Links
- iv. Broken Links

Forms

- i. Field validation
- ii. Error message for wrong input
- iii. Optional and Mandatory fields

Database

Testing will be done on the database integrity.

#2) Usability Testing

Usability testing is the process by which the human-computer interaction characteristics of a system are measured, and weaknesses are identified for correction.

- Ease of learning
- Navigation
- Subjective user satisfaction
- General appearance

Test for navigation:

Navigation means how a user surfs the web pages, different controls like buttons, boxes or how the user uses the links on the pages to surf different pages.

Usability testing includes the following:

The website should be easy to use.

Instructions provided should be very clear.

Check if the instructions provided are perfect to satisfy its purpose.

The main menu should be provided on each page.

It should be consistent enough.

Content checking:

Content should be logical and easy to understand. Check for spelling errors.

Usage of dark colors annoys the users and should not be used in the site theme.

You can follow some standard colors that are used for web page and content building.

These are the commonly accepted standards like what I mentioned above about annoying colors, fonts, frames etc.

Content should be meaningful. All the anchor text links should be working properly.

Images should be placed properly with proper sizes.

These are some of the basic important standards that should be followed in web development. Your task is to validate all for UI testing.

Other user information for user help: Like search option, sitemap also helps files etc. The sitemap should be present with all the links in websites with a proper tree view of navigation. Check for all links on the sitemap.

“Search on the site” option will help users to find content pages that they are looking for easily and quickly. These are all optional items and if present they should be validated.

#3) Interface Testing

In web testing, the server side interface should be tested. This is done by verifying that communication is done properly. Compatibility of the server with software, hardware, network, and the database should be tested.

The main interfaces are:

Web server and application server interface

Application server and Database server interface.

Check if all the interactions between these servers are executed and errors are handled properly. If database or web server returns an error message for any query by application server then application server should catch and display these error messages appropriately to the users.

Check what happens if the user interrupts any transaction in-between? Check what happens if the connection to the web server is reset in between?

#4) Compatibility Testing

Compatibility of your website is a very important testing aspect. See which compatibility test to be executed:

Browser compatibility

Operating system compatibility

Mobile browsing

Printing options

Browser compatibility:

In my web-testing career, I have experienced this as the most influencing part of website testing.

Some applications are very dependent on browsers. Different browsers have different configurations and settings that your web page should be compatible with.

Your website coding should be a cross-browser platform compatible. If you are using java scripts or AJAX calls for UI functionality, performing security checks or validations then give more stress on browser compatibility testing of your web application.

Test web application on different browsers like Internet Explorer, Firefox, Netscape Navigator, AOL, Safari, Opera browsers with different versions.

OS compatibility:

Some functionality in your web application is that it may not be compatible with all operating systems. All new technologies used in web development like graphic designs, interface calls like different API's may not be available in all Operating Systems.

Hence test your web application on different operating systems like Windows, Unix, MAC, Linux, Solaris with different OS flavors.

Mobile browsing:

We are in the new technology era. So in future Mobile browsing will rock. Test your web pages on mobile browsers. Compatibility issues may be there on mobile devices as well.

Printing options:

If you are giving page-printing options then make sure fonts, page alignment, page graphics etc., are getting printed properly. Pages should fit the paper size or as per the size mentioned in the printing option.

#5) Performance testing

The web application should sustain to heavy load. Web performance testing should include:

Web Load Testing

Web Stress Testing

Test application performance on different internet connection speed.

Web load testing: You need to test if many users are accessing or requesting the same page. Can system sustain in peak load times? The site should handle many simultaneous user requests, large input data from users, simultaneous connection to DB, heavy load on specific pages etc.

Web Stress testing: Generally stress means stretching the system beyond its specified limits. Web stress testing is performed to break the site by giving stress and its checked as for how the system reacts to stress and how it recovers from crashes. Stress is generally given on input fields, login and sign up areas.

In web performance, testing website functionality on different operating systems and different hardware platforms is checked for software and hardware memory leakage errors.

Performance testing can be applied to understand the web site's scalability or to benchmark the performance in the environment of third-party products such as servers and middleware for potential purchase.

Connection

Speed

Tested on various networks like Dial-Up, ISDN etc.

Load

- i. What is the no. of users per time?
- ii. Check for peak loads and how the system behaves
- iii. A large amount of data accessed by the user

Stress

- i. Continuous Load
- ii. Performance of memory, CPU, file handling etc..

#6) Security Testing

Following are some of the test cases for web security testing:

Test by pasting internal URL directly into the browser address bar without login. Internal pages should not open.

If you are logged in using username and password and browsing internal pages then try changing URL options directly. I.e. If you are checking some publisher site statistics with

publisher site ID= 123. Try directly changing the URL site ID parameter to different site ID which is not related to the logged in user. Access should be denied for this user to view others stats.

Try some invalid inputs in input fields like login username, password, input text boxes etc. Check the system's reaction to all invalid inputs.

Web directories or files should not be accessible directly unless they are given download option.

Test the CAPTCHA for automating script logins.

Test if SSL is used for security measures. If it is used, the proper message should get displayed when user switch from non-secure HTTP:// pages to secure HTTPS:// pages and vice versa.

All transactions, error messages, security breach attempts should get logged in log files somewhere on the web server.

The primary reason for testing the security of a web is to identify potential vulnerabilities and subsequently repair them.

Network Scanning

Vulnerability Scanning

Password Cracking

Log Review

Integrity Checkers

Virus Detection

Types of Web Testing

A website is classified into many types, it is about 20 types. All these are shrinking under static and dynamic type. Among them let's discuss 4 types and its testing methods in a detailed manner. Before that, I just want to bullet those types.

Simple static website testing

Dynamic web application testing

E-commerce website testing

Mobile website testing

#1) Simple Static Website

A simple static website will display the same content for all visitors who are visiting the website at different times. It is also known as an informational website. In a static website, the only developer can do changes that too in code only. This type of website will not have any major functionalities and it purely depends on UI design.

Testing a simple static website is very easy, you have to consider only a few things while testing. Some of them are mentioned below:

Points to Remember:

#1) Testing the GUI design is must because static website purely depends on it. You need to compare the approved PSD files with web page developed. Check all the elements in the design should present in the developed page.

#2) The other part of GUI design is to check the font size, font style, spacing, and color everything has been reproduced.



[This image explains the spacing alignment issue in the desktop view of a website.]

#3) Secondly, you need to check the links (page links) whether it is properly working or not? And also find, is there any broken link?

#4) Verify the spelling and content in all web pages by comparing the content given by the client.

#5) In some cases image will not display properly, it may break or sometimes images gets duplicated, wrong images may display. It has to be checked keenly. Because for a static website, only content and images will give lives.

#6) Check the scroll bar carefully, in my experience, I have faced issues with the scrollbar. The issue you will face is unwanted scrolling appears or scroll gets hidden (it may hide the contents). Above issues are applicable for both horizontal and vertical scroll.

#7) If there is a contact form check it is working properly by sending some dummy messages.

SAMPLE TESTCASES FOR DATA BASE VALIDATION

Things to check in contact form are:

Whether the message is sending properly and a success message appears?

Check email received to the concerned person in the proper format as designed?

Check email should not land in spam as junk mail?

If there is reply email trigger is activated then check whether the sender received mail?

#8) Check whether it is an error-free web page, validate it with W3 validator or other related software.

#9) Some constant things to be checked in a static website,

Check favicon is present on the tab bar

URL should contain the correct page title

If copyright information is there, it should be displayed

If there is a contact form, Captcha is a must. [It prevents junk email]

Check the loading speed of the website. [A static website should not take much time for loading]. If a gif image is used while loading then track its functionality

Apart from these, there are huge things that have to be tested at the backend of every website that is [system testing](#), security testing, interface testing, compatibility testing and performance testing etc. For these, you need to have technical knowledge. In a simple static website, you will not find more functionalities if there you need to do functionality testing too.

It is the type where the user can update and change their website content regularly. From here I am going to use the word “web application testing” instead of

dynamic website testing. The web application is a *combination of front-end and back-end programming*.

The front-end will be HTML and CSS whereas back-end uses programming languages like PHP, Javascript, and ASP etc. With this backend, user/client can add or change the content on the website.

Testing a web application is not easy than testing a static website but not much difficult than testing an e-commerce website. Functionality testing is the most important thing to be performed while testing a web application. The web application may contain much-complicated functionality so tester needs to be very careful while testing.

There are two different type of web applications are there, one is no action will be carried out by the user in front-end (i.e. only back-end changes will reflect in front-end) the other is end-user will work in front-end itself (**for example** login, signup, newsletter subscription and other similar actions). So testing should be done according to it.

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of

its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

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

Functional test

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

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

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

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An

example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

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

Black Box Testing

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

Unit Testing

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

Test strategy and approach

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

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

Integration Testing

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

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

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

Acceptance Testing

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

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

using test data errors were found and corrected. A series of tests were performed for the proposed system before the system was ready for implementation. The various types of testing done on the system are:

- Unit Testing
- Integration Testing
- User Acceptance Testing
- System Testing

4.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. It comprises the set of test performed by the programmer prior to integration of

the unit into larger system. The testing was carried out during the coding stage itself. In this step each module is found to be working satisfactorily as regards to the expected output from the module.

Each form is treated as a unit and tested thoroughly for bugs. The following is a list of some of the test cases :

- 1) In the login form, if a member does not enter a value for userId and password, then the user is prompted with the error message “userId and password should not be blank”.
- 2) In the login form, if a member enters wrong values for userId and password, then the user is prompted with the error message “Invalid userId and password. Try again.”.
- 3) In book Entry screen and new student, teacher screen, all the fields should have a value. Otherwise, the user is prompted with an appropriate error messages.
- 4) In book transactions form, member id, book no., issue date, and return date are mandatory. If not provided, then the system will prompt the user with the error message “Fields should not be blank”.

4.2 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover error associated within the interface. The objective is to take unit tested modules and build a program structure that has been dictated by design. All modules are combined in this step. The entire program is tested as whole. And chaos in interfaces may usually result. A set of errors is encountered in such a case.

The integration testing can be carried out using two methodologies:

Top Down Integration

Bottom Up Integration

The first one is done where integration is carried out by addition of major modules to minor modules. While Bottom Up integration follows combination of smaller ones to

larger one. Here, Bottom Up Integration is followed. Even though correction was difficult because the isolation of causes is complicated by the vastness of the entire program, all the errors found in the system were corrected and then forwarded to the next testing steps.

The navigation among all the screens have been thoroughly verified so that the user of the system can move from one form to another form.

The connectivity between the forms and the database has been checked. In case of any malfunctions, the user will be informed about the problem.

4.3 User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration was tested for users acceptance by constantly keeping in touch with the perspective system user at the time of developing and making changes wherever required. This is done with the regards to the following points:

A system may be defined as a set of instructions combined in the same form and directed to some purpose.

Before any development is undertaken certain specifications are prepared which objectively describe the application system. The System specifications are made after consulting the end user managers of the relevant departments.

Software to be developed is planned on the basis of requirement of the user. The problem definition statement description of present situation and goal to be achieved by news system.

The success of system depends on how accurately a problem is defined, thoroughly investigated carried out through choice of solution. User need identification and analysis that are concerned with what the uses needs rather than what he/she wants. System explains how to perform specific activities or task, which does what and what.

4.4 System Testing

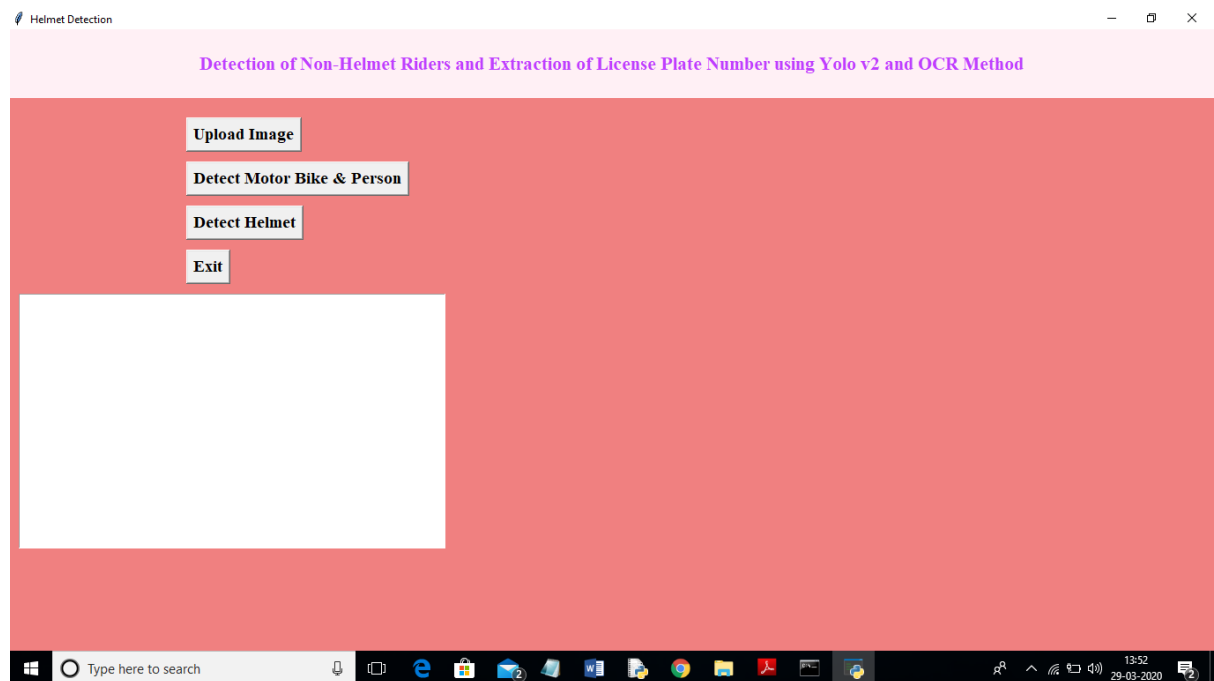
Testing the behavior of the whole software/system as defined in software requirements specification(SRS) is known as system testing, its main focus is to verify that the customer requirements are fulfilled. System testing is done after integration testing is

complete. System testing should test functional and non functional requirements of the software. The test types followed in system testing differ from organization to organization.

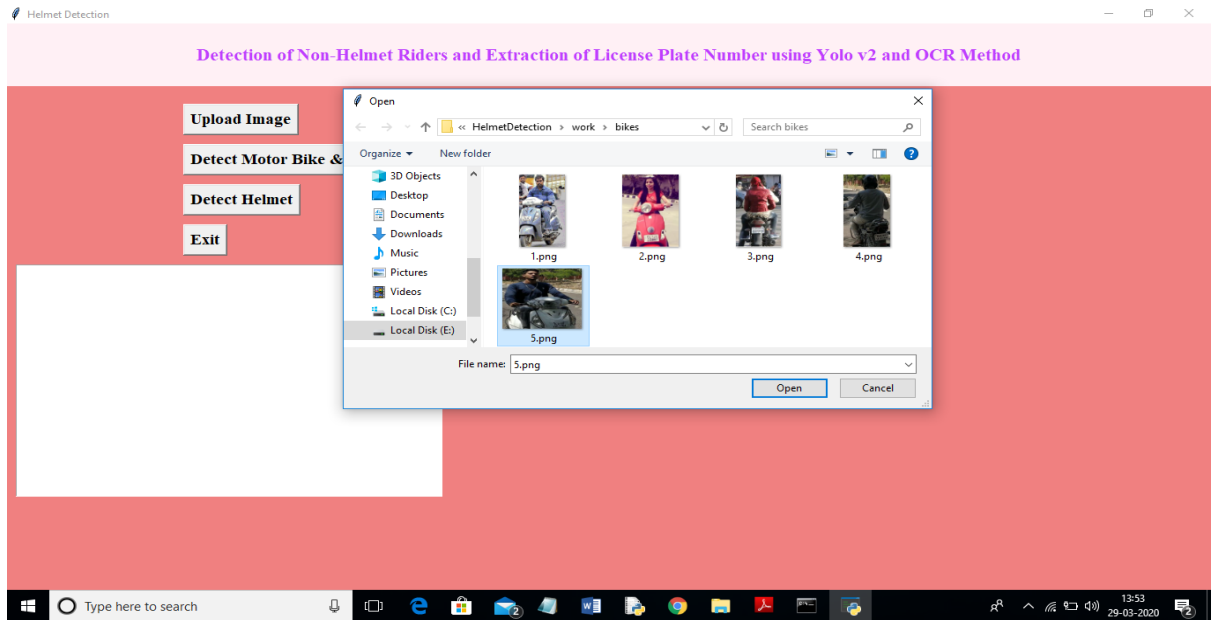
The project is executed and tested on the machines that satisfy the given hardware and software requirements. It was executed successfully with the specified hardware and operating system.

EXECUTION THROUGH SCREENSHOTS

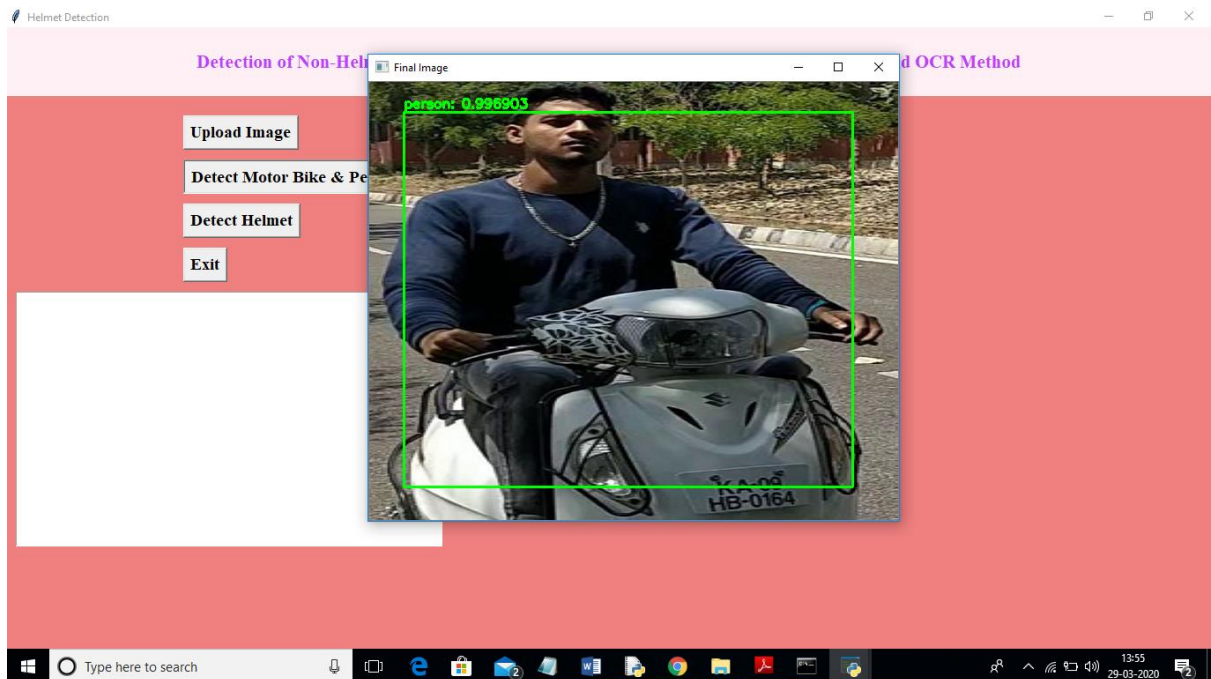
After setting path double click on 'run.bat' file to run project and to get below screen



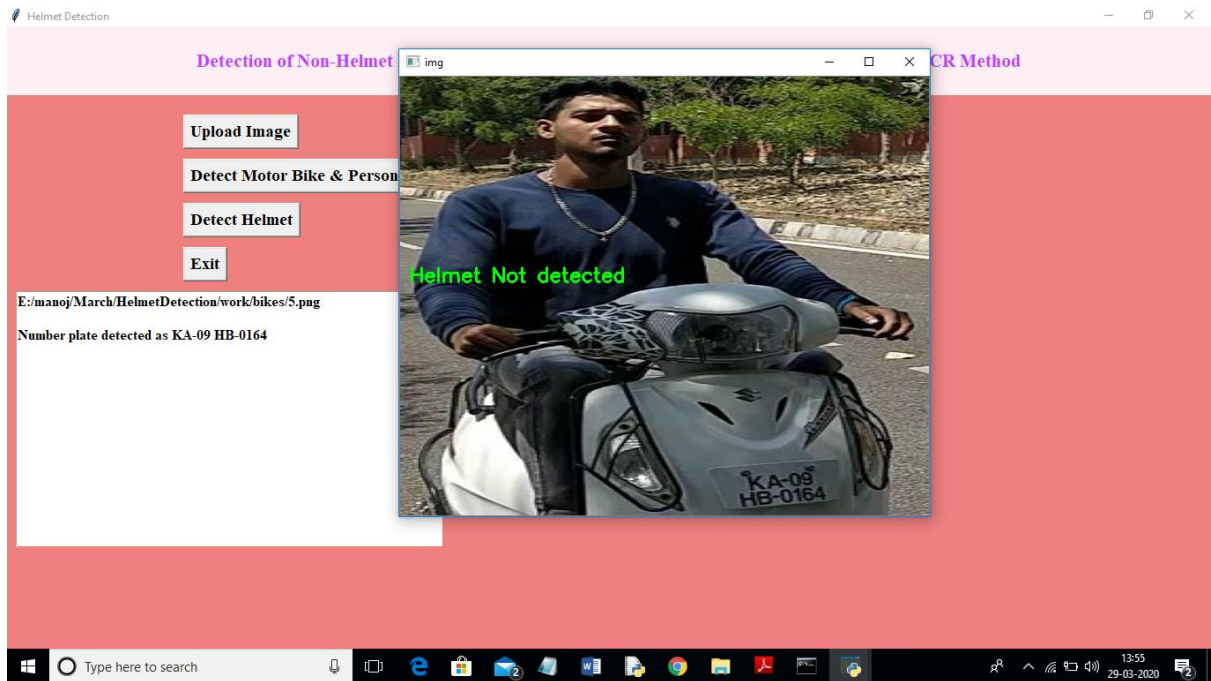
In above screen click on 'Upload Image' button and upload image



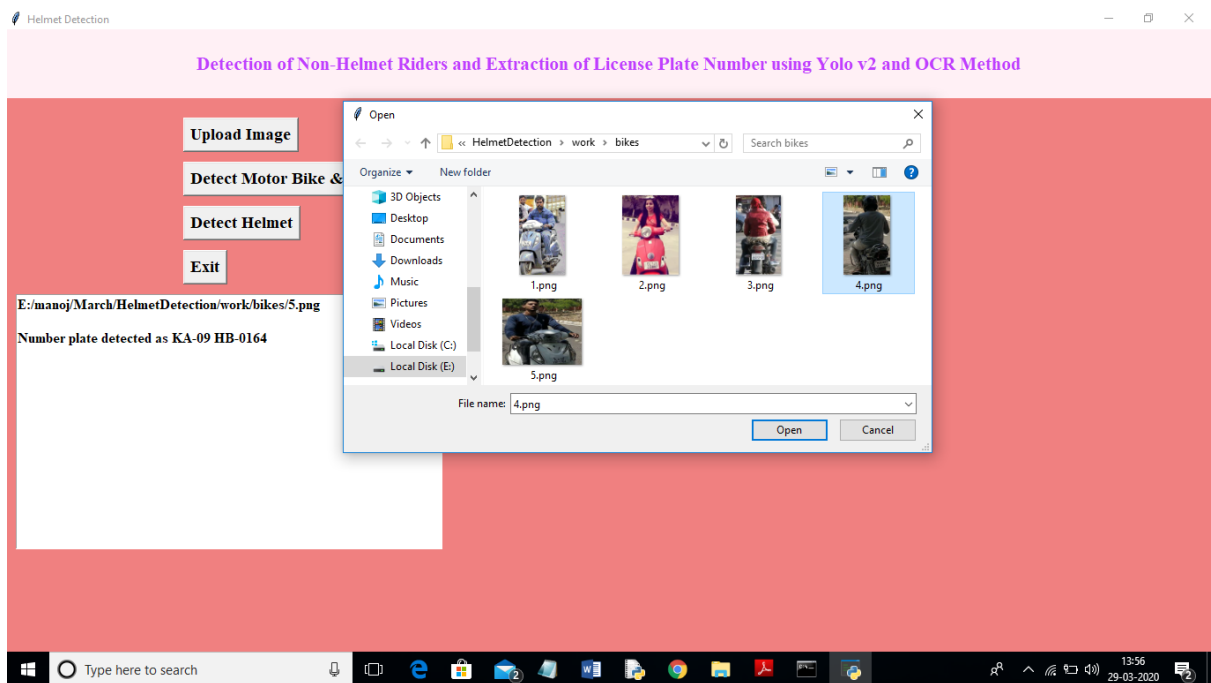
In above screen I selected one image as '5.png' and click on 'Open' button to load image. Now click on 'Detect Motor Bike & Person' button to detect whether image contains person with motor bike or not



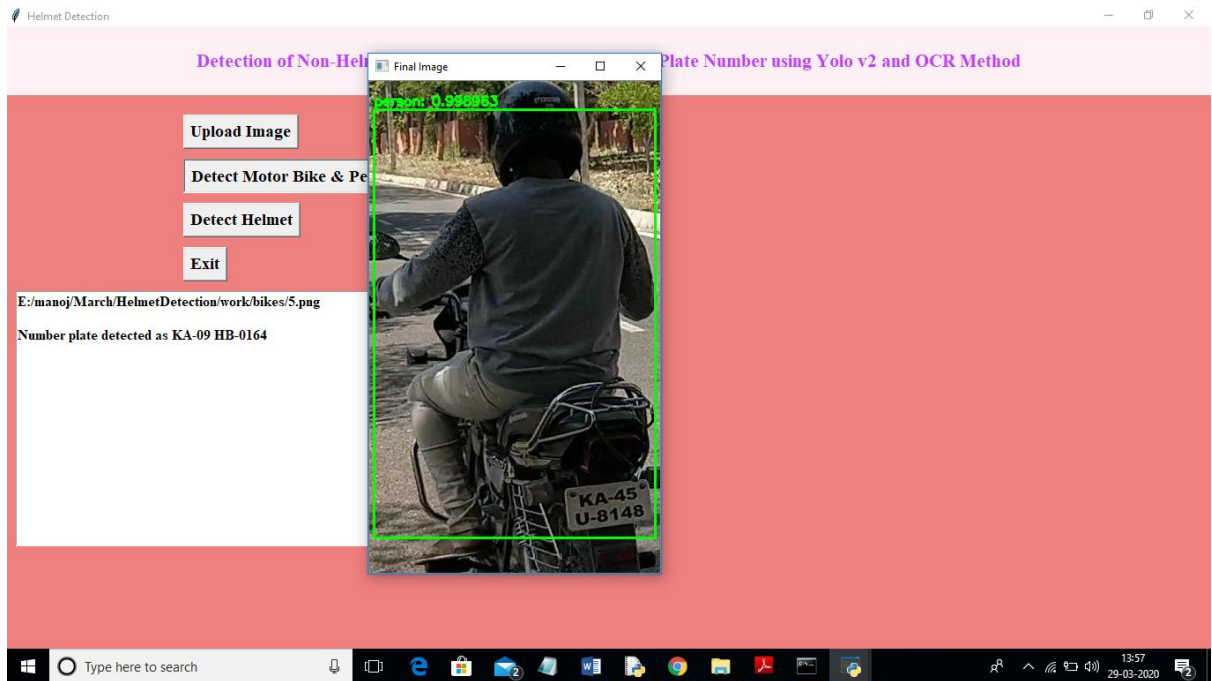
In above screen yolo detected image contains person and bike and now click on 'Detect Helmet' button to detect whether he is wearing helmet or not



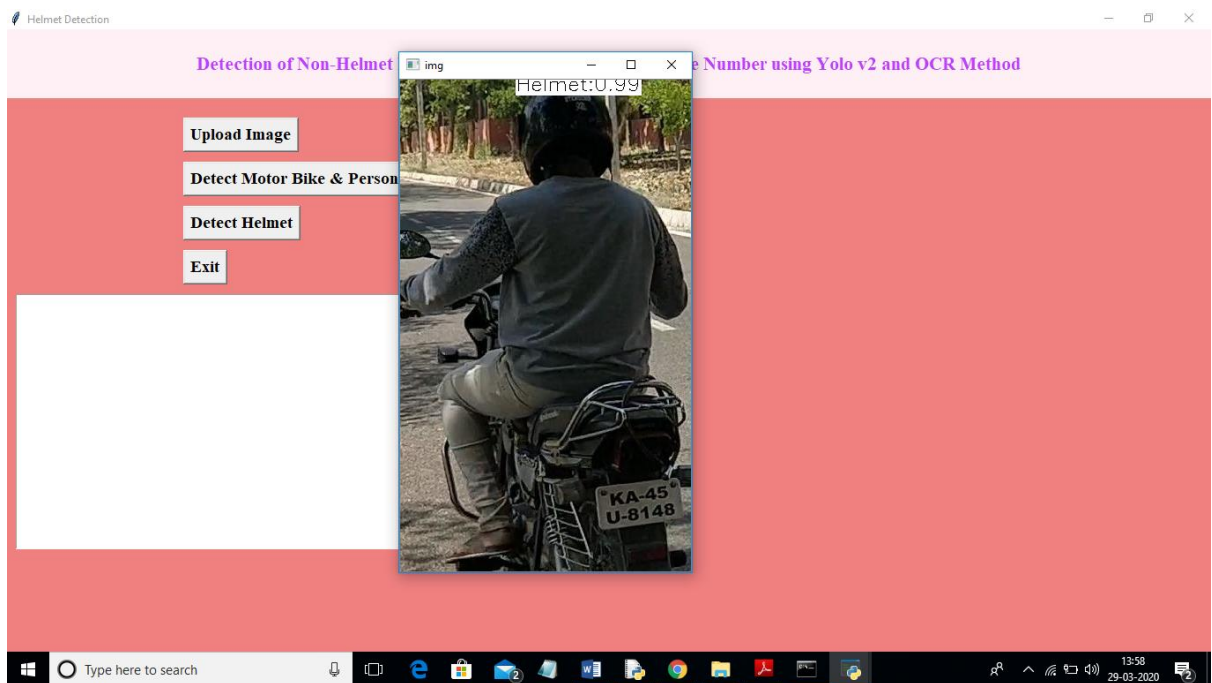
In above screen application detected that person is not wearing helmet and its extracted number from vehicle and display in beside text area. Now we will check with helmet image



In above screen I am uploading 4.png which is wearing helmet and now click on 'Detect Motor Bike & Person' button to get below result



In above screen yolo detected person with motor bike and now click on ‘Detect Helmet’ button to get below result



In above screen application detected person is wearing helmet and that label is displaying around his head and application stop there itself and not scanning number plate.

Note: To implement this project and to extract number plate we have trained few images and if u want to extract for new images then send those new images to us, so we include those images in yolo model to extract new images number plate also.

CONCLUSION

CONCLUDING RESULTS

A Non-Helmet Rider Detection system is developed where a video file is taken as input. If the motorcycle rider in the video footage is not wearing helmet while riding the motorcycle, then the license plate number of that motorcycle is extracted and displayed. Object detection principle with YOLO architecture is used for motorcycle, person, helmet and license plate detection. OCR is used for license plate number extraction if rider is not wearing helmet. Not only the characters are extracted, but also the frame from which it is also extracted so that it can be used for other purposes. All the objectives of the project is achieved satisfactorily.

5.2 Future Work

Our project can be linked with the traffic cameras and with some modifications it can be used to detect helmets in the real time system. Further more we can merge the algorithm of automated license plate detection and make a system which generates challans for those who don't wear helmets.

REFERENCES

1. J.Chiverton, "Helmet Presence Classification with Motorcycle Detection And Tracking", IET Intelligent Transport Systems, Vol. 6, Issue 3, pp. 259–269, March 2012.
2. Rattapoom Waranusast, Nannaphat Bundon, Vasan Timtong and Chainarong Tangnoi, "Machine Vision techniques for Motorcycle Safety Helmet Detection", 28th International Conference on Image and Vision Computing New Zealand, pp 35-40, IVCNZ 2013.
3. Romuere Silva, Kelson Aires, Thiago Santos, Kalyf Abdala, Rodrigo Veras, Andr´e Soares, "Automatic Detection Of Motorcyclists without Helmet", 2013 XXXIX Latin America Computing Conference (CLEI).IEEE,2013.
4. Romuere Silva, "Helmet Detection on Motorcyclists Using Image Descriptors and Classifiers", 27th SIBGRAPI Conference on Graphics, Patterns and Images.IEEE, 2014.
5. Thepnimit Marayatr, Pinit Kumhom, "Motorcyclist"s Helmet Wearing Detection Using Image Processing", Advanced Materials Research Vol 931- 932,pp. 588-592,May-2014.

6. Amir Mukhtar, Tong Boon Tang, “Vision Based Motorcycle Detection using HOG features”, IEEE International Conference on Signal and Image Processing Applications (ICSIPA).IEEE, 2015.
7. Abu H. M. Rubaiyat, Tanjin T. Toma, Masoumeh Kalantari-Khandani, “Automatic Detection of Helmet Uses for Construction Safety”, IEEE/WIC/ACM International Conference on Web Intelligence Workshops(WIW).IEEE, 2016.
8. XINHUA JIANG “A Study of Low-resolution Safety Helmet Image Recognition Combining Statistical Features with Artificial Neural Network”.ISSN: 1473-804x
9. Kunal Dahiya, Dinesh Singh, C. Krishna Mohan, “Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time”, International joint conference on neural network(IJCNN). IEEE, 2016.
10. Maharsh Desai, Shubham Khandelwal, Lokneesh Singh, Prof. Shilpa Gite, “Automatic Helmet Detection on Public Roads”, International Journal of Engineering Trends and Technology (IJETT), Volume 35 Number 5- May 2016, ISSN: 2231-5381