



**Mohamed Khider University of Biskra
Faculty of Science and Technology
Department of Electrical Engineering**

MASTER MEMORY

**Science and Technology Electrical
Networks and telecommunications**

Ref. :

Presented and supported by :

Abdelmoumen TAIR

date : October 2020

Application of Python with PyQT In the Security of Information by Detected the Face using webcam

Jury :

Mr.	Tahar GUESBAYA	MCA	Biskra University	President
Mr.	Fathi DHIABI	MCB	Biskra University	Supervisor
Mr.	Abdelkrim OUAFI	MCA	Biskra University	Examiner

College year: 2019 - 2020

الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

وزارة التعليم العالي و البحث العلمي

Ministry of Higher Education and Scientific Research



Mohamed Khider Biskra University

Faculty of Science and Technology Department

of Electrical Engineering

Sector : Electronics

Option : Networks and telecommunications

End of Studies Thesis

In view of obtaining the diploma :

MASTER

Theme

**Application of Python with PyQt In the
Security of Information by Detected the Face
using webcam**

Presented by :

Abdelmoumen TAIR

Favorable opinion of the supervisor:

Fathi DHIABI

Favorable opinion of the President of the Jury

.....

Stamp and signature

الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research



Mohamed Khider Biskra University
Faculty of Science and Technology Department
of Electrical Engineering
Sector : Electronics
Option : Networks and telecommunications

Theme

Application of Python with PyQt In the Security of Information by Detected the Face using webcam

Directed by : Fathi DHIABI

Abstract

In my final project master's degree in telecommunications , aims to describe the face detection and recognition. It reports the technologies available in the Open-Computer-Vision (OpenCV) and PyQt library and the methodology to implement them using Python.

For face detection and recognition, we used Haar-Cascades. Then the results are displayed including graphics and screenshots.

ملخص

في مشروع عي النهائي للحصول على شهادة الماستر في مجال الاتصالات، يهدف إلى وصف اكتشاف الوجه والتعرف عليه. ويبلغ عن التقنيات المتاحة في (Open-Computer-Vision (OpenCV ومكتبة PyQt ومنهجية تنفيذهما باستخدام Python. لاكتشاف الوجه والتعرف عليه، تم استخدام Haar-Cascades، ثم يتم عرض النتائج بما في ذلك الرسومات ولقطات الشاشة.

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

To My Parents

To All My Family

To All My Friends

Thanks

First of all, I thank Allah the Almighty and Merciful, who has given me the courage, strength and faith to complete this modest work.

I would first like to thank my supervisor very warmly Mr. DHIABI Fathi, for having accepted to supervise me for my project as well as for his support. I thank him very warmly here for the framing of this thesis. Above all, I would like to express my deepest gratitude to him because he guided and supported me with all possible means, both scientifically and personally and morally.

I also offer my thanks to Mr. GUESBAYA Tahar, who did me the honor of chairing the jury for this thesis.

I also express my sincere thanks to Mr. OUAFI Abdelkrim, who kindly reviewed my manuscript and provided me with his comments and corrections.

I would like to thank all those who have helped and encouraged me from near or far.

FIGURES LIST

Fig.I.1: Different types of Face Detection Methods	7
Fig.I.2: Template Matching	11
Fig.I.3: Haar Cascades Features	13
Fig.I.4: Example of Harr Cascades Features in face pic	14
Fig.I.5: Simple diagram represent how face recognition work	15
Fig.I.6: A generic face recognition system	15
Fig.I.7: Converting RGB image to Grayscale	15
Fig.I.8: Haar-like features for face detection	16
Fig.I.9: Successfully detect the face in an image	17
Fig.II.1: % Of questions about languages programs in Stackoverflow.com	21
Fig.II.2: The differences syntax: (a) Python. (b) C++ syntax.	22
Tabl.II.1: Comparison of an example between C++ & Python	23
Tabl.II.2: Comparison between C++ & Python	23
Tabl.II.3: Comparison of declaration between C++ & Python	25
Fig.II.3: Google Trends Python 2 vs. Python 3	28
Fig.II.4: % Of questions about Python 2 vs. Python 3 in Stackoverflow.com	28
Tabl.II.4: Comparison between Python v2 and Python v3	29
Fig.II.5: Python2 vs. Python3	30
Fig.II.7: How to create a new file (module).....	32
Fig.II.8: Example how to write code in Shell	33
Fig.II.9: How to write a code in the Shell (step 1)	33
Fig.II.10: How to write a code in the Shell (step 2)	34
Fig.II.11: How to write a code in the Shell (step 3)	34
Fig.III.1: The QMainWindow framework	55
Fig.III.2: Run Command Prompt	56
Fig.III.3: Install pyqt5	57
Fig.III.4: Install pyqt5-tools.....	57
Fig.III.5: Check that we installed PyQt5.....	58
Fig.III.6: The result in KDE Plasma 4 (Linux System)	59
Fig.III.7: The result in windows.....	60

FIGURES LIST

Fig.III.8: Designer source	60
Fig.III.9: Search about Designer in windows Menu	61
Fig.III.10: Interface of the application.....	61
Fig.III.11: Create window in designer step1	62
Fig.III.12: Create window in designer step2	62
Fig.III.13: Create window in designer step3	63
Fig.III.14: Create window in designer step4	63
Fig.III.15: Window saved.....	64
Fig.III.16: Convert file.ui to file.py step1	64
Fig.III.17: Command Prompt in the main folder	65
Fig.III.18: Convert file.ui to file.py step2.....	65
Fig.III.19: New file named window.py	66
Fig.III.20: The code of the window using Designer	66
Fig.III.21: The window appears.....	67
Fig.IV.1: Installing OpenCV	72
Fig.IV.2: Checking the version of OpenCV	72
Fig.IV.3: Pic.jpg.....	73
Fig.IV.4: Files we need.....	74
Fig.IV.5: Result.....	74
Fig.IV.6: result (webcam) without glasses.....	76
Fig.IV.7: result (webcam) with glasses.....	76
Fig.IV.8: Algorithm diagram	77
Fig.IV.9: MainWindow	78
Fig.IV.10: REGISTRATION Widget.....	78
Fig.IV.11: Registration and save faces to create a dataset	79
Fig.IV.12: Build a dataset	79
Fig.IV.13: Dataset	80
Fig.IV.14: Face recognition successfully	80
Fig.IV.15: Unknown person.....	81

TABLES LIST

Tabl.II.1: Comparison of an example between C++ & Python	23
Tabl.II.2: Comparison between C++ & Python	23
Tabl.II.3: Comparison of declaration between C++ & Python	25
Tabl.II.4: Comparison between Python v2 and Python v3	29
Tabl.III.1: Modules of PyQt5	53

ABRIVIATIONS

A

AI : Artificial intelligence
ATM: Automated Teller Machines
ABC: ABC programming language

D

DHSMV: Department of Highway Safety and Motor Vehicles
DNN: Deep neural networks
DSLR: Digital Single-Lens Reflex
DL: Deep Learning

H

HCI: Human–computer interaction
HMM: Hidden Markov Model

I

ID: Identity Document
IEEE: Institute of Electrical and Electronics Engineers

K

KL: Kar- honen-Loeve

L

LBPH: Local Binary Patterns Histogram

M

MATLAB: matrix laboratory
MRF: Markov Random Fields

N

NIST: National Institute of Standards and Technology

O

OpenCV: Open Computer Vision
OpenGL: Open Graphics Library
OpenCL: Open Computing Language

P

PCA: Principal component analysis
PIN: Personal Identification Number

R

RGB: Red, Green, Blue

S

SFM: Structure from motion
SVM: Support vector machine

SUMMARY

GENERAL INTRODUCTION

General Introduction	2
----------------------------	---

CHAPTER I: What is Face Detection & Methods

I.1. Introduction	4
I.2. History of face detection with applications	4
I.3. The future of face recognition	7
I.4. Face Detection Methods	7
A. Feature-Based	8
B. Appearance-Based	9
C. Knowledge-Based	10
D. Template Matching	11
I.5. Viola & Jones algorithm	11
I.6. How face recognition work with application	15
I.7. Conclusion	17

CHAPTER II: Python programming language

II.1. Introduction.....	19
II.2. History of Python	20
II.3. Python Versions Release Dates	20
II.4. Difference between Python and other programming languages.....	21
A. C++	22
B. Uses of C++	22
C. Comparison between C++ & Python.....	23
D. Library and Header files inclusion	24
II.5. Why Python over other languages (ADVANTAGES)	27
II.6. Differences between versions 2 and 3.....	28
A. The Main Python 2 Vs 3 Differences?	30
B. Why have we chosen version 3?	31
II.7. The instructions of python.....	32
A. How to write code in python v3	33
B. Write the code in Shell Input()	35
a. Input():.....	35
b. Print():	35
c. Variables	35

SUMMARY

d.	Basic Operators.....	37
C.	Data Types in Python.....	37
a.	Integers	37
b.	Float	38
c.	Strings	38
d.	List	39
e.	Tuple	41
f.	Dictionary	41
D.	Making Choices and Decisions	42
a.	Condition Statements	43
b.	If Statement.....	43
c.	Inline If	44
d.	For Loop: (syntax color).....	45
e.	While Loop.....	46
E.	Functions	47
II.8.	Conclusion.....	47

CHAPTER III: The GUI of PyQt

III.1.	Introduction	50
A.	What is PyQt?.....	50
B.	Why we used PyQt?.....	50
III.2.	History of PyQt	51
III.3.	Differences between PyQt4 and PyQt5	52
III.4.	PyQt5 Modules.....	53
III.5.	How to install PyQt5.....	56
III.6.	Hello World Example	58
A.	With PyQt4:	58
B.	With PyQt5	59
C.	With Designer	60
III.7.	Conclusion.....	67

CHAPTER IV: Face Recognition with OpenCV3 and integration of PyQt5

IV.1.	Introduction	69
IV.2.	History of OPENCV	70
IV.3.	Applications of OPENCV	70

SUMMARY

- IV.4. How to install OPENCV in Python with windows..... 71**
- IV.5. How to Run Face Detector by importing an image 73**
- IV.6. How to run a face detector program in real-time (Webcam) 75**
- IV.7. Simple python code of face detection and recognition
with opencv3 and integration of PyQt5..... 77**
- IV.8. Conclusion..... 81**

GENERAL CONCLUSION

- General Conclusion..... 83**

General Introduction

General Introduction

Face detection and recognition are the nonintrusive biometrics of choice in many security applications. Examples of their use include border control, driver's license issuance, law enforcement investigations, and physical access control.

Systems and techniques of face recognition and detection are a subset of an area related to information security, and information security is concerned with the assurance of confidentiality, integrity and availability of information in all forms. There are many tools and techniques that can support the management of information security; however, one of the important issues is the need to correctly authenticate a person. Traditionally, the use of passwords and a personal identification number (PIN) has been employed to identify an individual, but the disadvantages of such methods are that someone else may use the PIN for unauthorized access or the PIN may be easily forgotten. [1]

Many agencies are now motivated to improve security data systems based on body or behavioral characteristics, often called biometrics. Biometric approaches are concerned with identifying an individual by his unique physical characteristics and biological traits. Given these problems, the development of biometrics approaches such as face recognition, fingerprint, iris/retina and voice recognition prove to be a superior solution for identifying individuals over that of PIN codes. The use of biometric techniques not only uniquely identifies an individual, but also minimizes the risk of someone else using the unauthorized identity. Biometric authentication also supports the facets of identification, authentication and nonrepudiation in information security. [2]

Face detection and recognition represent nonintrusive methods for recognizing people, and these are the biometrics of choice in many security applications. Face detection and recognition are some remarkable and important abilities that we use in our daily lives. The main reason for the interest in developing the computer vision-based automated technologies of automated face recognition arises from the serious concerns for public security in today's networked world, where identity verifications for physical and logical access in many facilities are imperative in daily life. Though the first widely accepted algorithm during the 1970s was the eigenface method, which even today is used as a base for many methods, the real impetus came along with the development of computational power and algorithms related to the use of large databases. Therefore, face detection and recognition are still actively researched areas. Many problems related to unconstrained and real-life and real-time environments are yet to be solved to the level of required robustness and accuracy. [2]

So, what is face detection and recognition ?. What are the methods of face detection and face recognition ?. How does it work in Python programming language with OpenCV and the integration of PyQt5 ?.

Chapter I

What is

Face

Detection

I.1. Introduction

In the past few years, face detection owned significant consideration and appreciated as one of the most promising applications in the field of image analysis. Face detection can consider a substantial part of face recognition operations. According to its strength to focus computational resources on the section of an image holding a face. The method of face detection in pictures is complicated because of variability present across human faces such as pose, expression, position and orientation, skin color, the presence of glasses or facial hair, differences in camera gain, lighting conditions, and image resolution. [8]

Object detection is one of the computer technologies, which connected to the image processing and computer vision and it interacts with detecting instances of an object such as human faces, building, tree, car, etc. The primary aim of face detection algorithms is to determine whether there is any face in an image or not. In recent times, a lot of study work proposed in the field of Face recognition and face detection to make it more advanced and accurate, but it makes a revolution in this field when Viola-Jones comes with its real-time face detector, which is capable of detecting the faces in real-time with high accuracy.

Face detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is a part of object detection and can use in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc.

It is used to detect faces in real time for surveillance and tracking of person or objects. It is widely used in cameras to identify multiple appearances in the frame Ex- Mobile cameras and DSLR's. Facebook is also using face detection algorithm to detect faces in the images and recognize them. [9]

I.2. History of face detection with applications

Earlier facial recognition technology was considered as an idea of science fiction. But in the past decade, facial recognition technology has not only become real, but it's widespread. Today, people can easily read articles and news stories about facial recognition everywhere.

Facial recognition technology along with AI (Artificial Intelligence) and Deep Learning (DL) technology are benefiting several industries. These industries include law enforcement agencies, airports, mobile phone manufacturing companies, home appliance manufacturing companies, etc.

Nowadays even retailers are using AI-based facial recognition technology to prevent violence and crime. Airports are getting better-secured environment, mobile phone makers are using face recognition to bring the biometric security feature in the devices. [17]

A. In the 1960s, Woodrow Wilson Bledsoe created a system that could organize faces' photos by hand using the RAND tablet. The tablet is a device people could use to enter vertical and horizontal coordinates on a grid with the help of a stylus that released electromagnetic pulses. People used that system to manually record the coordinate areas of facial features like eyes, nose, mouth, and hairline.

The manually recorded metrics could be later saved within a database. And when the new photograph of an individual was entered into the system, it was able to get the most closely resembled image via database. During this period, face recognition was untouched by technology and computer processing power. Still, it was the first and foremost step taken by Bledsoe to prove that face recognition was a practical biometric. [10]

B. 21 Facial Markers for enhanced accuracy in the 1970s

It was in the 1970s when Harmon, Goldstein, and Lesk made the manual facial recognition system more accurate. The three used 21 facial markers including lip thickness and hair color, to detect faces automatically. However, the Bledsoe's system was still computed with actual biometrics, done manually.

Eigenfaces

Sirovich and Kirby started using linear algebra to the issue of facial recognition in 1988. The approach they used was called the Eigenface approach. The rendering began as a search for low-dimensional facial images representation. The team was able to prove that feature analysis on collected pictures in the database could form a set of basic features.

They were also able to explain how less than a hundred values could be used to code a face image precisely. [11]

C. In 1991, Pentland and Turk worked further on the Eigenfaces approach by finding ways to detect faces within images. Pentland and Turk's work were the first automatic face recognition attempt. They used technological and environmental factors for their approach.

D. FERET PROGRAM (1993-2000S)

Then in the 1993-2000s period, DARPA and NIST released the FERET program to encourage the commercial facial recognition market. In 2002, law enforcement officials applied facial recognition in critical technology testing. The project involved creating a database of facial images. The database was updated in 2003 to include high-resolution

24-bit color versions of images. Included in the test set were 2,413 still facial images representing 856 people. The hope was that a large database of test images for facial recognition would be able to inspire innovation, that might result in more powerful facial recognition technology. [16]

E. SUPER BOWL XXXV (2002)

At the 2002 Super Bowl, law enforcement officials used facial recognition in a major test of the technology. While officials reported that several "petty criminals" were detected, overall, the test was seen as a failure. False positives and backlash from critics proved that face recognition wasn't quite ready for prime time. One of the big technological limitations at the time was that face recognition did not yet work well in large crowds, functionality that is essential to using face recognition for event security.

F. FACE RECOGNITION VENDOR TESTS (2000S)

The National Institute of Standards and Technology (NIST) began Face Recognition Vendor Tests (FRVT) in the early 2000s. Building on FERET, FRVTs were designed to provide independent government evaluations of facial recognition systems that were commercially available, as well as prototype technologies. These evaluations were designed to provide law enforcement agencies and

the U.S. government with information necessary to determine the best ways to deploy facial recognition technology.[17]

G. The Viola–Jones object detection (2001)

The Viola–Jones object detection framework is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection.

H. LAW ENFORCEMENT FORENSIC DATABASE (2009)

In 2009, the Pinellas County Sherriff’s Office created a forensic database that allowed officers to tap into the photo archives of the state’s Department of Highway Safety and Motor Vehicles (DHSMV). By 2011, about 170 deputies had been outfitted with cameras that let them take pictures of suspects that could be cross-checked against the the database. This resulted in more arrests and criminal investigations than would have otherwise been possible.[12]

I. Social Media 2010 to present

When 2010 started, Facebook started using a facial recognition feature that helped detect people with featured faces in the photos updated by Facebook users. While the update created hype in the media industry, Facebook stayed very low key since there was no apparent negative impact on website popularity and usage.

J. Largest biometric installation at the airport

In 2011, the Panama government and US Secretary of Homeland Security Janet Napolitano partnered and authorized a pilot program of the Facial Recognition platform. The pilot program was called, FaceFirst, and it was used to cut down illicit activities at Panama’s Tocumen airport. The first attempt garnered success, and the FaceFirst expanded into the north terminal facility. The technology implemented at Tocumen became the largest biometrics installation at the airport.

K. Adopted in the military for dead bodies

In 2011, U.S. law enforcement and military professionals used face recognition for the identification of dead bodies. With this technology, the military was able to confirm the identity of Osama Bin Laden.

L. Law enforcement adoption of facial recognition in 2014

Law enforcement agencies came forward to adopt mobile face recognition in 2014. The technology became inevitable for the retail industry in 2017. [19]

I.3. The future of face recognition

Governments across the world are increasingly investing their resources in facial recognition technology, especially the US and China are the leaders in the facial recognition market. The government of the USA has decided to enhance airport security with a facial recognition system for identification and registration of visitors. The US has several states that have allowed law enforcement to run searches within the database, these searches include details of a driver's license and ID photos. The facial recognition and resulting search techniques can be also used in police checks.

China is already running several projects of facial intelligence when the other countries are still in its planning phase.

The whole world is using this technology and reaping many benefits. In India, banks are using this facial recognition technology to prevent fraud at ATM's. It is also used for reporting duplicate voters, verification of passport and visa, driving license, etc.

The future of facial recognition is promising:

The technology is expected to grow and will create massive revenues in the coming years. Surveillance and security are the major industries that will be intensely influenced by technology. Schools and universities and even healthcare are also planning to implement the facial recognition technology on their premises for better management. Complicated technology used in facial technology is also making its way to the robotics industry.

[19] [16]

I.4. Face Detection Methods

Yan, Kriegman, and Ahuja presented a classification for face detection methods. These methods divided into four categories, and the face detection algorithms could belong to two or more groups. These categories are: [17]

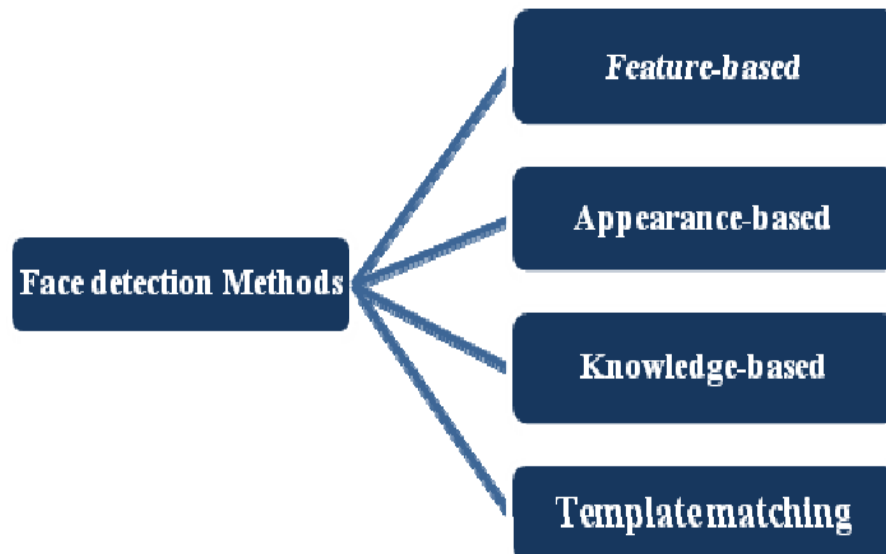


Fig.I.1: Different types of Face Detection Methods [17]

A. Feature-Based

The feature-based method is to locate faces by extracting structural features of the face. It is first trained as a classifier and then used to differentiate between facial and non-facial regions. The idea is to overcome the limits of our instinctive knowledge of faces. This approach divided into several steps and even photos with many faces they report a success rate of 94%.

In contrast to the knowledge-based top-down approach, researchers have been trying to find invariant features of faces for detection. The underlying assumption is based on the observation that humans can effortlessly detect faces and objects in different poses and lighting conditions and, so, there must exist properties or features which are invariant over these variabilities. Numerous methods have been proposed to first detect facial features and then to infer the presence of a face. Facial features such as eyebrows, eyes, nose, mouth, and hair-line are commonly extracted using edge detectors. Based on the extracted features, a statistical model is built to describe their relationships and to verify the existence of a face. One problem with these feature-based algorithms is that the image

features can be severely corrupted due to illumination, noise, and occlusion. Feature boundaries can be weakened for faces, while shadows can cause numerous strong edges which together render perceptual grouping algorithms useless.

A.1. Facial Features

Sirohey proposed a localization method to segment a face from a cluttered background for face identification. It uses an edge map (Canny detector) and heuristics to remove and group edges so that only the ones on the face contour are preserved. An ellipse is then fit to the boundary between the head region and the background. This algorithm achieves 80 percent accuracy on a database of 48 images with cluttered backgrounds.

Recently, Amit et al. presented a method for shape detection and applied it to detect frontal-view faces in still intensity images. Detection follows two stages: focusing and intensive classification.

A.2. Skin Color

Human skin color has been used and proven to be an effective feature in many applications from face detection to hand tracking. Although different people have different skin color, several studies have shown that the major difference lies largely between their intensity rather than their chrominance.

A.3. Texture

Human faces have a distinct texture that can be used to separate them from different objects. Augusteijn and Skufca developed a method that infers the presence of a face through the identification of face-like textures. The textures are computed using second-order statistical features (SGLD) on subimages of 16 X 16 pixels. Three types of features are considered: skin, hair, and others. They used a cascade correlation neural network for supervised classification of textures and a Kohonen self-organizing feature map [30] to form clusters for different texture classes. [12]

B. Appearance-Based

The appearance-based method depends on a set of delegate training face images to find out face models. The appearance-based approach is better than other ways of performance. In general appearance-based method rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face images. This method also used in feature extraction for face recognition.

The appearance-based model further divided into sub-methods for the use of face detection which are as follows:

B.1. Eigenface-Based

Eigenface based algorithm used for Face Recognition, and it is a method for efficiently representing faces using Principal Component Analysis.

As mentioned, one of the goals that the feature extraction routine wishes to achieve is to increase the efficiency. One simple way to achieve this goal is using alternative orthonormal bases other than the natural bases. One such basis is the Karhonen-Loeve (KL). KL bases are formed by the eigenvectors of the covariance matrix of the face vector X . In the high dimensional "face" space, only the first few eigenvalues have large values. In other words, energy mainly locates in the subspace constituted by the first few eigenvectors. Therefore, a great compression can be achieved by letting those eigenvectors with large eigenvalues to represent the face vector.

$$X, X \cong \sum_{i=1}^M x_i u_i \dots \dots \dots (1)$$

where u is the eigenvector and M is usually much smaller than original vector dimension N . Since the eigenvectors associated with the first few eigenvalues look like face images, KL bases are also referred to as eigenfaces.

The eigenface representation is well known in statistics literature as the principal component analysis. It is optimal in the sense of efficiency: for any given $M < N$, the KL representation has the minimum mean square error among all possible approximations of X that uses M orthonormal vectors. However, it does not mean that the KL representation is optimal in the sense of discriminating power, which relies more on the separation between different faces rather than the spread of all faces.

Pentland's Photobook is one implementation of the eigenface algorithm. It compresses a facial image with 128×128 pixels (16,384 pixels) into a vector with only 40 eigenfaces (80 bytes). It recognizes 95% of the 200 faces chosen from a large database with 7562 facial images (3000 different persons) (Pentland, 1994). [11] [12]

B.2. Distribution-Based

The algorithms like PCA and Fisher's Discriminant can be used to define the subspace representing facial patterns. There is a trained classifier, which correctly identifies instances of the target pattern class from the background image patterns.

B.3. Neural-Networks

Many detection problems like object detection, face detection, emotion detection, and face recognition, etc. have been faced successfully by Neural Networks.

B.4. Support Vector Machine

Support Vector Machines are linear classifiers that maximize the margin between the decision hyperplane and the examples in the training set. Osuna et al. first applied this classifier to face detection.

B.5. Sparse Network of Winnows

They defined a sparse network of two linear units or target nodes; one represents face patterns and other for the non-face patterns. It is less time consuming and efficient.

B.6. Naive Bayes Classifiers

They computed the probability of a face to be present in the picture by counting the frequency of occurrence of a series of the pattern over the training images. The classifier captured the joint statistics of local appearance and position of the faces.

B.7. Hidden Markov Model

The states of the model would be the facial features, which usually described as strips of pixels. HMM's commonly used along with other methods to build detection algorithms.

B.8. Information Theoretical Approach

Markov Random Fields (MRF) can use for face pattern and correlated features. The Markov process maximizes the discrimination between classes using Kullback-Leibler divergence. Therefore, this method can be used in Face Detection.

B.9. Inductive Learning

This approach has been used to detect faces. Algorithms like Quinlan's C4.5 or Mitchell's FIND-S used for this purpose. [11]

C. Knowledge-Based

The knowledge-based method depends on the set of rules, and it is based on human knowledge to detect the faces. Ex: A face must have a nose, eyes, and mouth within certain distances and positions with each other. The big problem with these methods is the difficulty in building an appropriate set of rules. There could be many false positive if the rules were too general or too detailed. This approach alone is insufficient and unable to find many faces in multiple images.

One problem with this approach is the difficulty in translating human knowledge into well-defined rules. If the rules are detailed (i.e., strict), they may fail to detect faces that do not pass all the rules. If the rules are too general, they may give many false positives. Moreover, it is difficult to extend this approach to detect faces in different poses since it is challenging to enumerate all possible cases. On the other hand, heuristics about faces work well in detecting frontal faces in uncluttered scenes.

Yang and Huang used a hierarchical knowledge-based method to detect face. Their system consists of three levels of rules. At the highest level, all possible face candidates are found by scanning a window over the input image and applying a set of rules at each location. The rules at a higher level

are general descriptions of what a face looks like while the rules at lower levels rely on details of facial features. A multiresolution hierarchy of images is created by averaging and sub sampling. [16]

D. Template Matching

In template matching, a standard face pattern (usually frontal) is manually predefined or parameterized by a function. Given an input image, the correlation values with the standard patterns are computed for the face contour, eyes, nose, and mouth independently. The existence of a face is determined based on the correlation values. This approach has the advantage of being simple to implement. However, it has proven to be inadequate for face detection since it cannot effectively deal with variation in scale, pose, and shape. Multi-resolution, multi-scale, sub templates and deformable templates have subsequently been proposed to achieve scale and shape invariance.[9][14]

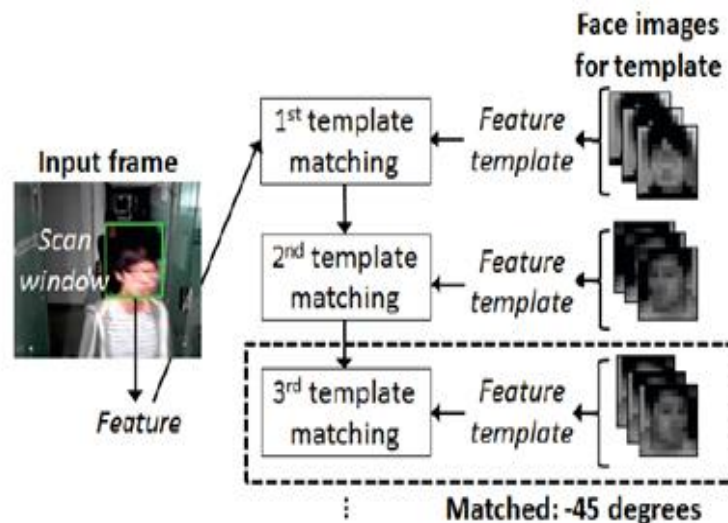


Fig.I.2: Template Matching [9]

I.5. Viola & Jones algorithm

There are many techniques to detect faces, with the help of these techniques, we can identify faces with higher accuracy. These techniques have an almost same procedure for Face Detection such as OpenCV, Neural Networks, MATLAB, etc. The face detection work as to detect multiple faces in an image. As Viola & Jones Method, which is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in below image are used. They are just

like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

The characteristics of Viola–Jones algorithm which make it a good detection algorithm are:

Robust – very high detection rate (true-positive rate) & very low false-positive rate always.

Real time – For practical applications at least 2 frames per second must be processed.

Face detection only (not recognition) - The goal is to distinguish faces from non-faces (detection is the first step in the recognition process).

The algorithm has four stages:

- Haar Feature Selection
- Creating an Integral Image
- Adaboost Training
- Cascading Classifiers

Learning algorithm:

The speed with which features may be evaluated does not adequately compensate for their number, however. For example, in a standard 24x24 pixel sub-window, there are a total of $M = 162,336$ possible features, and it would be prohibitively expensive to evaluate them all when testing an image. Thus, the object detection framework employs a variant of the learning algorithm AdaBoost to both select the best features and to train classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. [13]

$$h(x) = \text{sgn} \left(\sum_{j=1}^M \alpha_j h_j(x) \right) \dots \dots \dots (2)$$

Each weak classifier is a threshold function based on the feature f_j .

$$h_j(x) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases} \dots \dots \dots (3)$$

The threshold value θ_j and the polarity $s_j \in \pm 1$ are determined in the training, as well as the coefficients α_j .

Input: Set of N positive and negative training images with their labels (x^i, y^i) . If image i is a face $y^i = 1$, if not $y^i = -1$.

- ❖ Initialization: assign a weight $w_1^i = \frac{1}{N}$ to each image i .
- ❖ For each feature f_j with $j = 1, \dots, M$
 - Renormalize the weights such that they sum to one.
 - Apply the feature to each image in the training set, then find the optimal threshold and polarity θ_j, s_j that minimizes the weighted classification error.

$$\text{That is } \theta_j, s_j = \arg \min_{\theta, s} \sum_{i=1}^N w_j^i \mathcal{E}_j^i \dots \dots \dots (4)$$

$$\text{where } \mathcal{E}_j^i = \begin{cases} 0 & \text{if } y^j = h_j(x^i, \theta_j, s_j) \\ 1 & \text{otherwise} \end{cases} \dots \dots \dots (5)$$

- Assign a weight α_j to h_j that is inversely proportional to the error rate. In this way best classifiers are considered more.
- The weights for the next iteration, i.e. w_{j+1}^i , are reduced for the images i that were correctly classified.

❖ Set the final classifier to

$$h(x) = \text{sgn} \left(\sum_{j=1}^M \alpha_j h_j(x) \right) \dots \dots \dots (6) [13] [20]$$

Here a simplified version of the learning algorithm is reported:

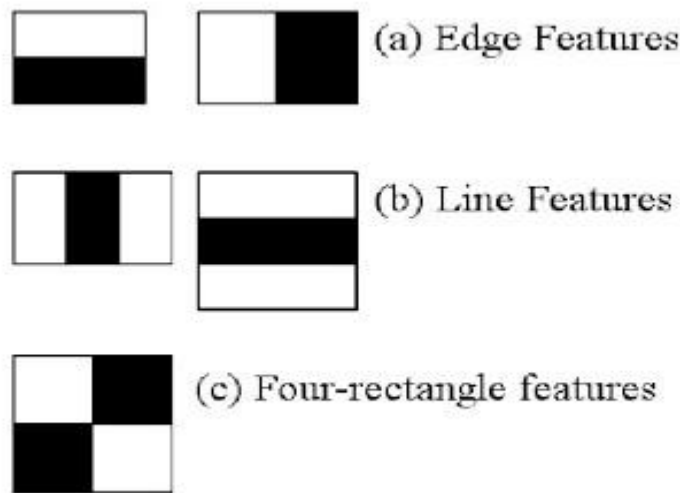


Fig.I.3: Haar Cascades Features [13] [20]

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

But the same windows applying on cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by Adaboost.

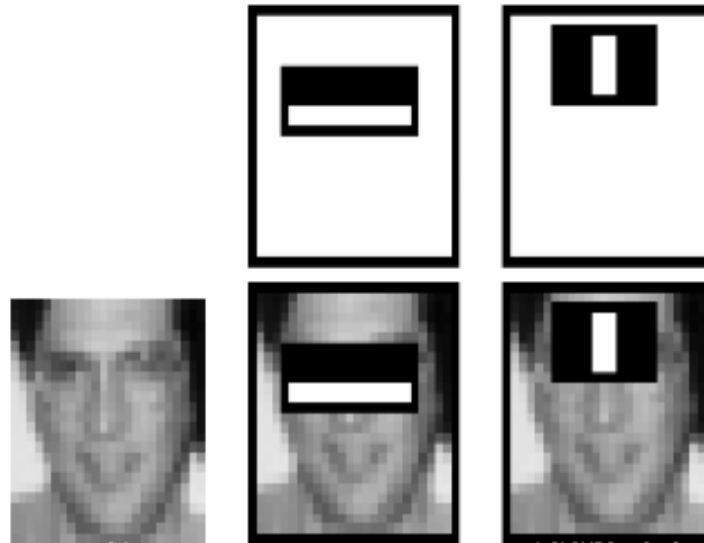


Fig.I.4: Example of Harr Cascades Features in face pic [13]

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

In an image, most of the image region is non-face region. So, it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. [13] [15] [20]

I.6. How face recognition work with application

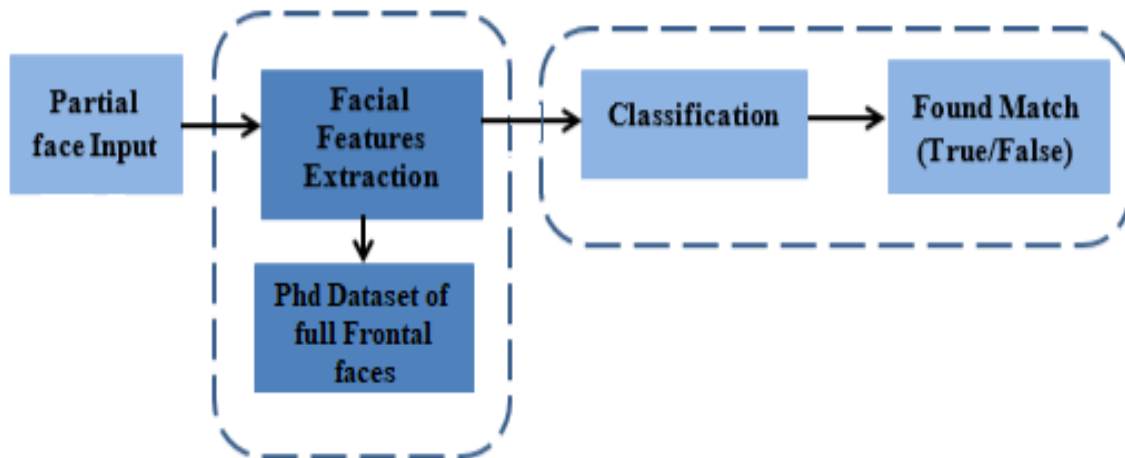


Fig.I.5: Simple diagram represent how face recognition work [15]

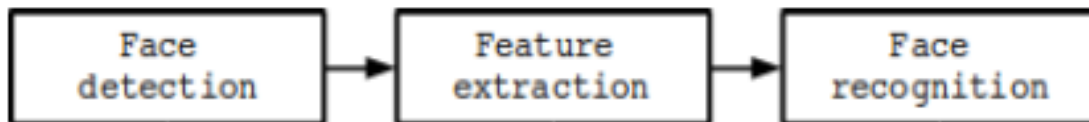


Fig.I.6: A generic face recognition system [16]

There are some steps that how face detection operates, which are as follows:

- Firstly, the image is imported by providing the location of the image. Then the picture is transformed from RGB to Grayscale because it is easy to detect faces in the grayscale.



Fig.I.7: Converting RGB image to Grayscale [17]

- After that, the image manipulation used, in which the resizing, cropping, blurring and sharpening of the images done if needed. The next step is image segmentation, which is used for

contour detection or segments the multiple objects in a single image so that the classifier can quickly detect the objects and faces in the picture.

- The next step is to use Haar-Like features algorithm, which is proposed by Viola and Jones for face detection. This algorithm used for finding the location of the human faces in a frame or image. All human faces share some universal properties of the human face like the eyes' region is darker than its neighbor pixels and nose's region is brighter than eyes' region. To do that we write the following code in python: [15][16]

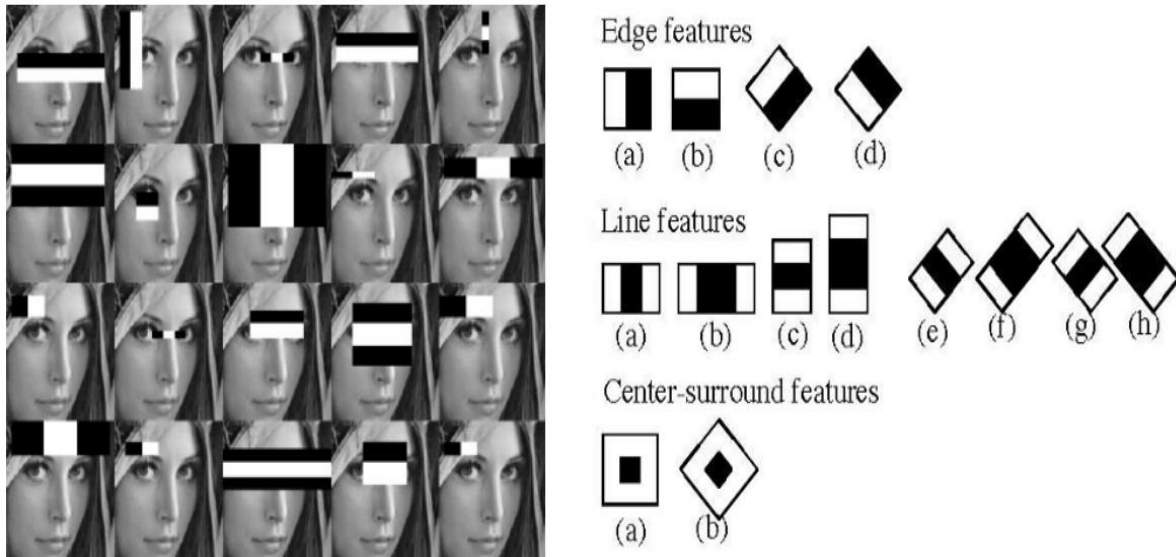


Fig.I.8: Haar-like features for face detection [15] [17]

The Haar like algorithm is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, center detection for detecting eyes, nose, mouth, etc. in the picture. It is used to select the essential features in an image and extract these features for face detection.

- The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face. There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.

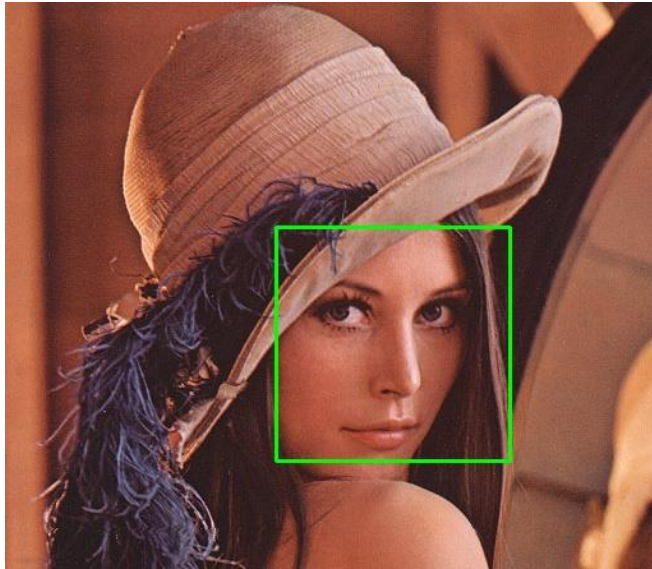


Fig.I.9: Successfully detect the face in an image [17]

I.7. Conclusion

In this chapter, we talked about what is face detection, face recognition and history of face detection, how did face recognition developed and become easy with the develop of AI and technologies and we talked about Viola & Jones method of detecting objects.

CHAPTER
II
Python
programming
language

II.1. Introduction

Instead of describing what Python can do, it's actually faster to say what Python can't do. Python is good for a little bit of everything.

It is a popular multi-paradigm, dynamically typed, multipurpose programming language. It was created by Guido van Rossum, and released in 1991. And according to Stack Overflow, Python is the fastest-growing programming language in the world, and it will continue to grow even faster. It's already well-recognized as a very universal, versatile, stable, and easy to learn programming language. As a high-level general-purpose programming language, Python can be applied to many different classes of problems. It should fit most application requirements. It is also relatively easy to find.

Python is a freely available object-oriented, high-level programming language with dynamic semantics. Many programmers say great things about Python because of the increased productivity that it provides since the edit-test-debug cycle is incredibly fast compared to other programming languages.

The Python language has a simple, easy to learn syntax which is empowered with many English words for easier readability and helps for increased productivity and efficiency. When coding in Python, it feels more like you are writing out the solution to a problem in your own thoughts rather than trying to refer to ambiguous symbols that are required in the language to commit to certain functionalities. [21] [22]

Python is used for: web development (server-side), software development, mathematics and system scripting; It could also be used to automate measurements and process data interactively. The language is able to handle large databases and compute large numbers strain-free compared to many other programming languages. It can be used as an internal scripting language so that it is executed for certain functions by other programs. By learning Python, you will be able to write complex software like intricate web crawlers. It is truly an all-purpose language.

The great thing about Python is that it has a giant library for web crawling and is used a lot for its capability in scraping the web. A web crawler is simply a program that can navigate the web depending on the parameters set out for it and scrapes content that you would like for it to scrape.

All in all, Python can be easy to pick up whether you're a beginner in programming or an experienced one for other languages. It's a fast, friendly and easy to learn language but don't mistake that for its powerful nature. [3] [22]

II.2. History of Python

Python is known as the go-to language for beginners as it is recommended by computer programmers around the globe as a good language for beginners to learn. This should not be misinterpreted for its powerful nature.

The Python language was conceived in the late 1980s by Guido Von Rossum at **Centrum Wiskunde & Informatica** in the Netherlands, as a successor to the **ABC language** (itself inspired by **SETL**). The language itself has been actually developed by a large team of volunteers and is available to modify. In the development stage that it was in, it had classes with inheritance, exception handling, functions, and the core datatypes of list, dict, str and more.

In May 2000, Guido and the Python development team moved to BeOpen.com to form the BeOpen PythonLabs team. In the same year, the PythonLabs team moved to Digital Creations which is now known as Zope Corporation.

Python 2.0 was released on the 16th of October in the year 2000 with features including a garbage collector which helps maintain memory handling related issues in programming. The great thing about Python was that it is backed by a community and it has a transparency behind the users that utilize the Python language.

Soon after, Python 3.0 which was a major backwards-incompatible release was released on December 3rd 2008 after a long period of testing. The major features of Python 3.0 also have been backported to backwards compatible Python 2.6 and 2.7. [4] [21] [22]

II.3. Python Versions Release Dates

Version 1.X (1994-2000): Python 1 refers to the first set of releases of the Python language. This was long ago and nobody uses or cares about these anymore.

Version 2.X (2000-2020): Python 2 refers the second line of releases of Python. Python 2.7.18 (Release Date: 20-04-2020) is the latest release in this line of releases.

Version 3.X (2008-2020): Python 3 refers to the third major line of releases of Python. Python 3.8.5 (Release Date: 20-04-2020) is the latest release in this line of releases. Python 3 introduced several changes to the actual structure and syntax of the Python language, such that code written for Python 2 would not run under Python 3 without major changes (and vice versa).

Python 3.9 alpha1 was announced in November 2019, but the release date for the final version depends on what new proposal for release dates are adopted with three draft proposals under discussion, and a yearly cadence is one option, many **alpha, beta, and release-candidates** are also released as previews and for testing before final releases. [21] [22] [24]

II.4. Difference between Python and other programming languages

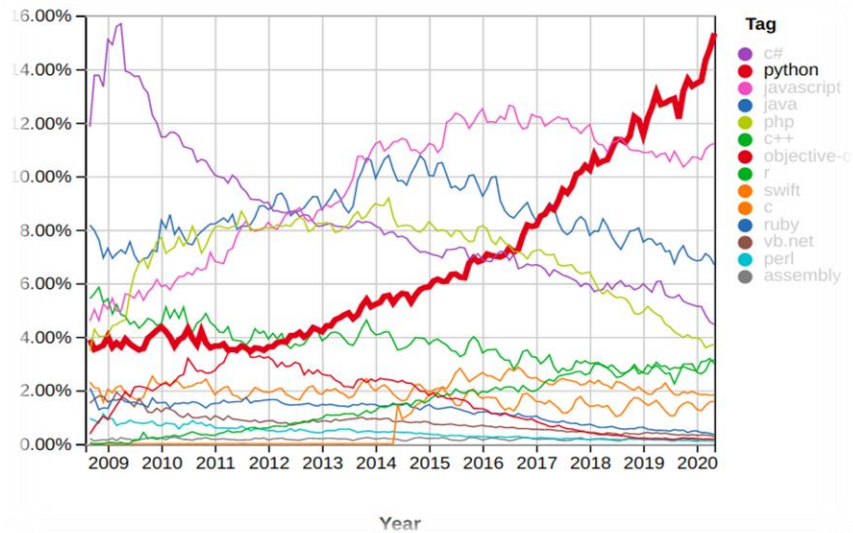


Fig.II.1: % Of questions about languages programs in Stackoverflow.com [23]

We notice that in the past three years Python got a lot of popularity than the other languages.

A lot of people claim that Microsoft simply copied Java to create C++. That said, C++ does have some advantages (and disadvantages) when compared to Java. The main (undisputed) intent behind C++ is to create a better kind of C/C++ language one that is easier to learn and use. However, we're here to talk about C++ and Python.

Python Syntax compared to other programming languages.

Python was designed for readability, and has some similarities to the English language with influence from mathematics.

Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

[4] [5] [25]

```

Python 3.3.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |

```

(a)

```

C++ shell
1 // Example program
2 #include <iostream>
3 #include <string>
4
5 int main()
6 {
7     std::cout << "Welcome to my first C++ program ever!\n";
8     std::string name;
9     std::cout << "What is your name? ";
10    getline (std::cin, name);
11    std::cout << "Hello, " << name << "! Thank you for experiencing this memorable occasion with me!\n";
12 }
13

```

(b)

Fig.II.2: The differences syntax: (a) Python. (b) C++ syntax.

A. C++

C++:is a general-purpose language usually involved in the development of large and complicated systems. This language is the most portable out of the whole circle of programming languages.

When comparing Python to C++, Python follows a rule of “write once, run anywhere,” which means that one code will work on all operating systems. However, the C++ code needs to compile on each OS before it can execute.

The biggest difference in the discussion of C++ vs. Python is that the C++ source code needs to become machine code. Python follows a different tactic as it is interpreted. However, the interpretation of code is usually slower than running code directly on the hardware. [4] [26]

B. Uses of C++

Let’s take a look at classic use cases of C++:

- C++ is closer to the hardware. Therefore, C++ produces most of the embedded systems around. By embedded systems, we mean smartwatches, medical machines, IoT sensors, etc.
- C++ plays a part in the development of applications such as servers and microcontroller programs.
- C++ is the leading language for 3D, multiplayer, or other types of game development. It is powerful enough to create such elaborate games as Counter Strike, Doom, and Red Dead Redemption. For instance, even the framework Unity is written in C++.
- C++ rules and principles are much more complicated than Python in terms of syntax. [5] [26]

Example:

Tabl.II.1: Comparison of an example between C++ & Python

C++	Python
<pre>#include #include using namespace std; int main () { string name; cin >> name; cout <<"Welcome, "<< name << endl; return 0; }</pre>	<pre>name = input () print ("Welcome,"+ name)</pre>

C. Comparison between C++ & Python

Tabl.II.2: Comparison between C++ & Python [5] [26]

Basis of comparison	C++	Python
Basics	C++ is a general-purpose programming language which is best suited for resource-constrained applications, such as those found in software infrastructures. And it was created as an extension of C and its core application domain is systems programming in the broadest sense.	Python is a flexible, object-oriented, and open source programming language designed to optimize development speed and make it easy to write software that can be understood, reused, and modified. It is specifically designed to raise development quality expectations in the scripting domain. It is also one of the most preferred choices as a first programming language.
Nature	C++ is a statically typed language in which variable types are explicitly declared and are determined at compile time. Static typed languages like C++ associate types with variables, not with values.	Python is a dynamically typed language which looks like it was designed and not accumulated. It has a minimalist design that makes code easy to understand in, it is both dynamically typed and strongly typed language in which type checking is done at run-time

Efficiency	C++ is a low-level language which makes it less versatile and more difficult to learn than Python.	Python's standard implementation is currently coded in C, so all the normal rules about mixing C programs with C++ programs apply to the Python interpreter. When Python is embedded in a C++ program, there are no special rules to follow – simply link in the Python library and call its functions from C++. Python is well suited for modern software methodologies such as modular, structured, and object-oriented design, which allow code to be written once and reused many times.
Memory	C++ does not need a garbage collector because it has no garbage which in turn makes it more prone to memory leak. Memory management in C++ is both prone to errors and time consuming.	Python uses dynamic memory allocation process which involves a private heap containing all Python objects and data structures and the garbage collector automatically returns memory to the system when it's no longer been in use.
Performance	C++ has the advantage of being a statically typed language. The performance crown goes to C++ for creating a more compact and faster runtime code.	Python is a dynamic language which reduces complexity when it comes to collaborating and optimizes programmer efficiency.
Compilation	C++ is a pre-compiled programming language and doesn't need any interpreter during compilation.	Python is an interpreted language and it runs through an interpreter during compilation.
Function	In C++, the function can accept and return the type of value which is already defined.	Python Functions do not have restrictions on the type of the argument and the type of its return value.
Popularity	C++ has dedicated followings online. But only the people who have some experience in the field show a lot of interest in C++.	Python has huge community support. When it comes to popularity, beginner and novice programmers tend to turn towards Python.

D. Library and Header files inclusion

- Header Files: The files that tell the compiler how to call some functionality (without knowing how the functionality actually works) are called header files. They contain the function prototypes. They also contain Data types and constants used with the libraries. We use #include to use these header files in programs. These files end with .h extension.
- Library: Library is the place where the actual functionality is implemented i.e., they contain function body.
- Modules: A module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping

related code into a module makes the code easier to understand and use. Import in python is similar to #include header file in C/C++. Python modules can get access to code from another module by importing the file/function using import. [5]

Tabl.II.3: Comparison of declaration between C++ & Python [4][5]

	C++	Python
Main method declaration	<p>Main method declaration is declaring to computer that from here the implementation of my code should be done. The process of declaring main in C++ is we declare int main where int stands for return type we should have to return something integral at the end of code so that it compiles without error. We can write our code in between the curly braces.</p> <pre>// C++ program to demonstrate // declaring main #include <iostream.h> int main () { // Your code here (comment) return 0 }</pre>	<p>It is not necessary to declare main in python code unless and until you are declaring another function in your code. So, we can declare main in python as follows.</p> <pre># Python program to demonstrate # declaring main def main (): # write your code here (comment) if __name__=="__main__": main ()</pre>
Declaring Variables	<p>In C++ we first declare the data type of the variable and then declare the name of Variables. A few examples of data types are int, char, float, double, etc.</p> <pre>#include <iostream.h> int main () { // Declaring one variable at a time int a; // Declaring more than one variable char a, b, c, d; // Initializing variables float a = 0, b, c; b = 1.5; return 0; }</pre>	<p>Python is not “statically typed”. We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it.</p> <pre># An integer assignment age = 24 # A floating point x = 1.5 # A string name = "Moumen"</pre>
Printing to console	<pre>#include <iostream> using namespace std;</pre>	<pre>print ("Hello World")</pre>

	<pre>int main () { cout << "Hello World"; return 0; }</pre>	
Taking Input	<pre>#include <iostream> using name space std; intmain() { inta, b, c; cout << "first number: "; cin >> a; cout << endl; cout << "second number: "; cin >> b; cout << endl; c = a + b; cout << "Hello World"<< endl; cout << a << "+"<< b << "="<< c; return0; }</pre>	<pre>a =input("first number: ") b =input("second number: ") c =a +b print("Hello World") print(a, "+", b, "=", c)</pre>

Hence, when compared to C++, Python has these advantages:

- ❖ Significantly easier to learn.
- ❖ Smaller (more concise) code.
- ❖ Supported fully as open source.
- ❖ Better multiplatform support.
- ❖ Easily allows use of multiple development environments.
- ❖ Easier to extend using C++.
- ❖ Enhanced scientific and engineering support.

II.5. Why Python over other languages (ADVANTAGES)

Compared to other programming languages Python is the most broadly applied by the developers lately, we will take a look at the advantages of Python programming language for developers in contrast with other languages.

- One of the key features of Python is its simplicity, making it the ideal language for beginners to learn.
- Python has a simple syntax similar to the English language. Most programs in Python require considerably fewer lines of code to perform the same task compared to other languages such as C, this leads to fewer programming errors and reduces the development time needed.
- In addition, Python comes with an extensive collection of third-party resources that extend the capabilities of the language.
- As such, Python can be used for a large variety of tasks, such as for desktop applications, database applications, network programming, game programming and even mobile development.
- It runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

- It can be treated in a procedural way, an object-orientated way or a functional way.
- Machine learning (ML) with Python improves many industries such as insurance, retail, banking, aerospace, and business services. ML is an excellent option for finding insights in a specific field and making predictions.
- Most of the data-analysts choose Python as their main programming language. It helps to handle huge amounts of data in the most cost-effective way. Python also manages data, analyzes statistical information, improves data visualization, and makes predictions in specific fields.
- Python is also an active member of the backend web development. It is possible to create a website by using raw Python, but that is rare.
- Maturity. Python is a proven language with powerful capabilities allowing you to code just about anything you can dream up.
- In-demand. Python developers are regularly hired by a host of companies throughout the world.
- Remote-friendly. Coding ninjas and coding students alike merely require an internet connection to achieve their goals.

- Python is the new Excel. The reason Python is being taught in business school is because Python is like Excel on steroids. With Python financial analysts, CEOs, and data-driven marketers can leverage the power of Python to crunch big data.

- Last but not least, Python is a cross platform language, which means that code written for one operating system, such as Windows, will work well on Mac OS or Linux without making any changes to the Python code. [27] [28]

II.6. Differences between versions 2 and 3

In the past, there was a bit of a debate in the coding community about which Python version was the best one to learn: Python 2 vs Python 3.

Now, in 2020, it's more of a no-brainer: Python 3 is the clear winner for new learners or those wanting to update skills. Here, we'll cover why Python 3 is better, and why companies have been moving from Python 2 to 3.

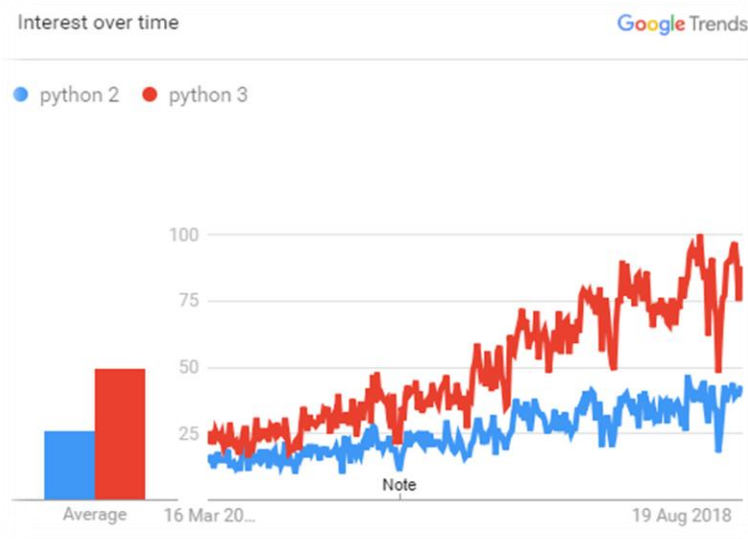


Fig.II.3: Google Trends Python 2 vs. Python 3 [23]

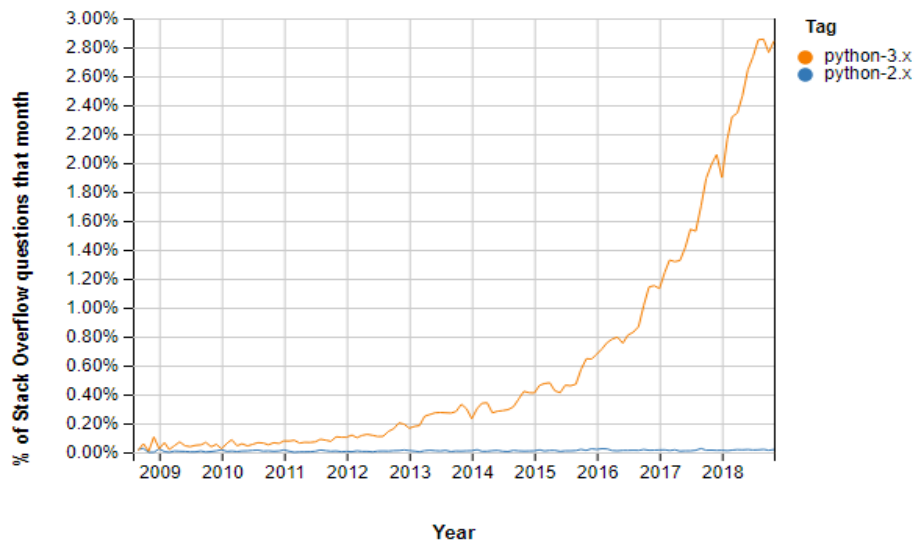


Fig.II.4: % Of questions about Python 2 vs. Python 3 in Stackoverflow.com [24]

Tabl.II.4: Comparison between Python v2 and Python v3 [21] [24]

Basis of comparison	Python v2	Python v3
Release Date	2000	2008
Unicode	To store Unicode string value, you require to define them with "u".	In Python 3, default storing of strings is Unicode.
Syntax	The syntax of Python 2 was comparatively difficult to understand.	The syntax is simpler and easily understandable.
Rules of ordering	Rules of ordering comparison are very complex.	Comparisons In this version, Rules of ordering comparisons have been simplified.
Iteration	In Python 2, the xrange() is used for iterations.	The new range() function introduced to perform iterations.
Exceptions	It should be enclosed in notations.	It should be enclosed in parenthesis.
Leak of variables	The value of the global variable will change while using it inside for-loop.	The value of variables never changes.
Backward compatibility	Python version 3 is not backwardly compatible with Python 2.	Not difficult to port python 2 to python 3 but it is never reliable.
Library	Many older libraries created for Python 2 is not forward-compatible.	Many recent developers are creating libraries which you can only use with Python 3.

A. The Main Python 2 Vs 3 Differences?

There are plenty of differences between these Python programming versions, but here are five of the main ones.

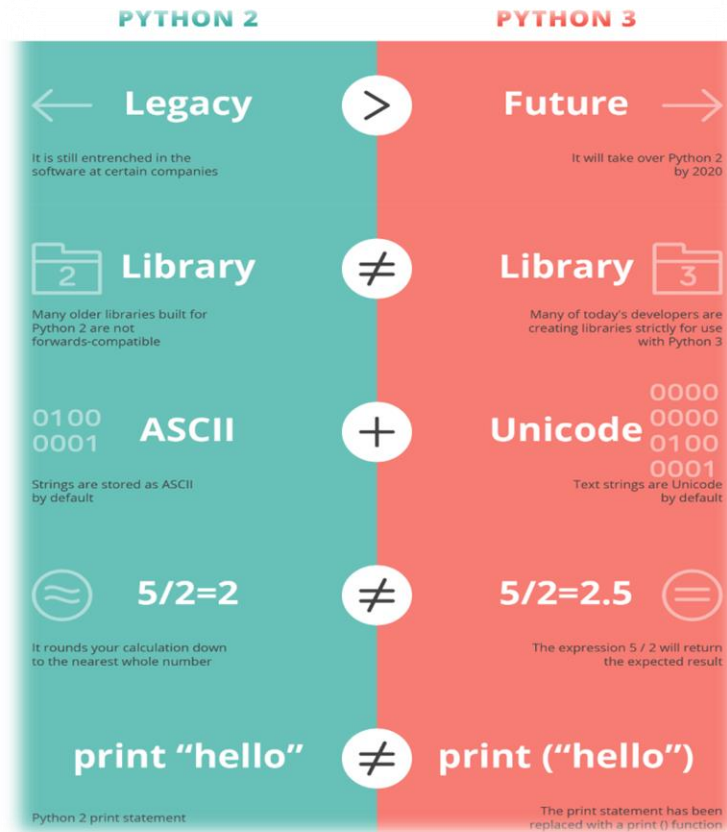


Fig.II.5: Python2 vs. Python3 [23] [24]

- Python 2 is legacy, python 3 is the future: Since python 2 has been the most popular version for over a decade and a half, it is still entrenched in the software at certain companies. However, since more companies (Facebook and Instagram) are moving from python 2 to 3, someone who wants to learn python programming for beginners may wish to avoid spending time on a version that is becoming obsolete.
- Python 2 and python 3 have different (sometimes incompatible) libraries: Since python 3 is the future, many of today's developers are creating libraries strictly for use with python 3. Similarly, many older libraries built for python 2 are not forwards-compatible. You may be able to port a 2.x library to 3.x., but this can be difficult and complicated; it's definitely not a "python for beginners" type of activity.
- There is better unicode support in python 3: In python 3, text strings are unicode by default. In python 2, strings are stored as ascii by default—you have to add a "u" if you want to store strings as unicode in python 2.x.

This is important because unicode is more versatile than ascii. Unicode strings can store foreign language letters, roman letters and numerals, symbols, emojis, etc., offering you more choices.

- Python 3 has improved integer division: In python 2, if you write a number without any digits after the decimal point, it rounds your calculation down to the nearest whole number.

For example, if you're trying to perform the calculation 5 divided by 2, and you type `5 / 2`, the result will be 2 due to rounding. You would have to write it as `5.0 / 2.0` to get the exact answer of 2.5.

However, in python 3, the expression `5 / 2` will return the expected result of 2.5 without having to worry about adding those extra zeroes.

This is one example of how python 3 syntax can be more intuitive, making it easier for newcomers to learn python programming.

- The two versions have different print statement syntaxes: This is only a syntactical difference—and some may consider it trivial—so it doesn't affect the functionality of python. That said, it is still a big and visible difference you should know about.

Essentially, in python 3, the print statement has been replaced with a `print ()` function.

For example, in python 2 it is `print "hello"` but in python 3 it is `print ("hello")`.

- If you're going to learn python programming for the first time, it shouldn't affect you much. But if you started with python 2, the change may trip you up a few times.[23] [24]

B. Why have we chosen version 3?

We have made a decision that we will make our study with the 3rd version of Python because it has a new, improved string formatting system that is easier to use than the previous versions.

- Python 3 supports modern techniques like AI, machine learning, and data science
- Python 3 is supported by a large Python developer's community.
- It's easier to learn Python language compared to earlier versions.
- Offers Powerful toolkit and libraries
- Mixable with other languages
- The Python 3 edition fully supports Microsoft Windows 10 from beginning to end.
- The previous edition focused mostly on the Unix-style systems such as macOS and Linux. [6] [28]

II.7. The instructions of python

Now that we're done with the introductory stuff, let's get to the real stuff.

We'll learn about variables and operators. Specifically, we'll learn what variables are and how to name and declare them. we'll also learn about the common operations that we can perform on them.



```
Python 3.7.7 Shell
File Edit Shell Debug Options Window Help
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 10:41:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Fig.II.6: Python Shell

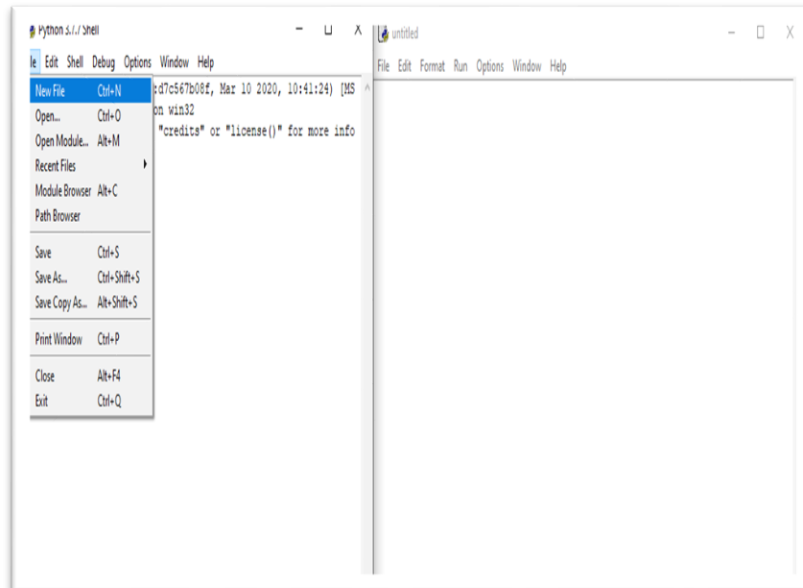
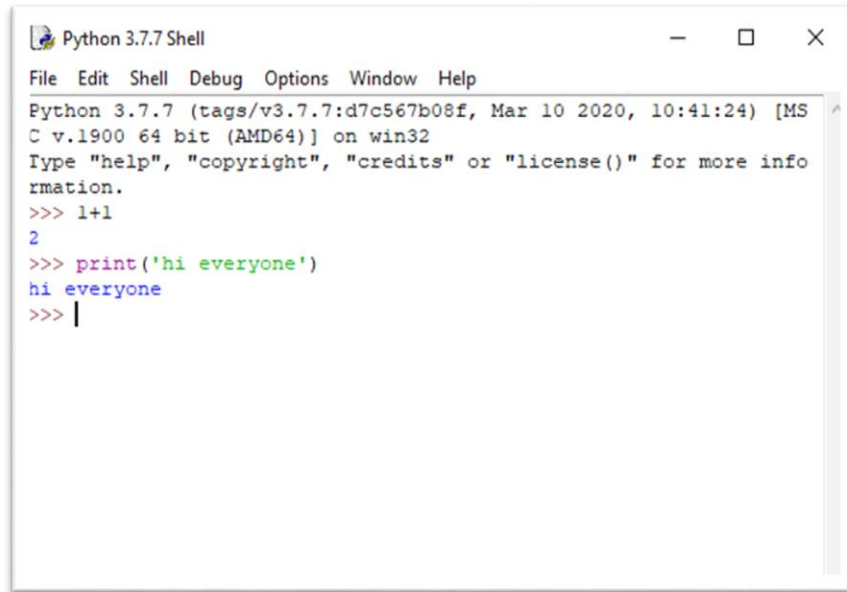


Fig.II.7: How to create a new file (module)

A. How to write code in python v3

There are two methods to write code in python:

Method1: write the code in Shell



```
Python 3.7.7 Shell
File Edit Shell Debug Options Window Help
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 10:41:24) [MS
C v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more info
rmation.
>>> 1+1
2
>>> print('hi everyone')
hi everyone
>>> |
```

Fig.II.8: Example how to write code in Shell

Method 2: write the code in module

We go to File then click on New File (CTRL+N)

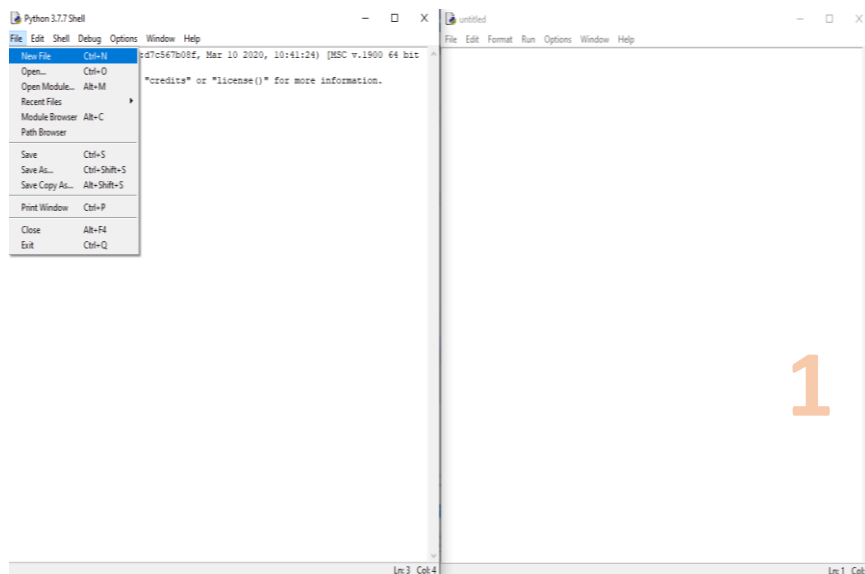


Fig.II.9: How to write a code in the Shell (step 1)

We can now write our code in the module and run it

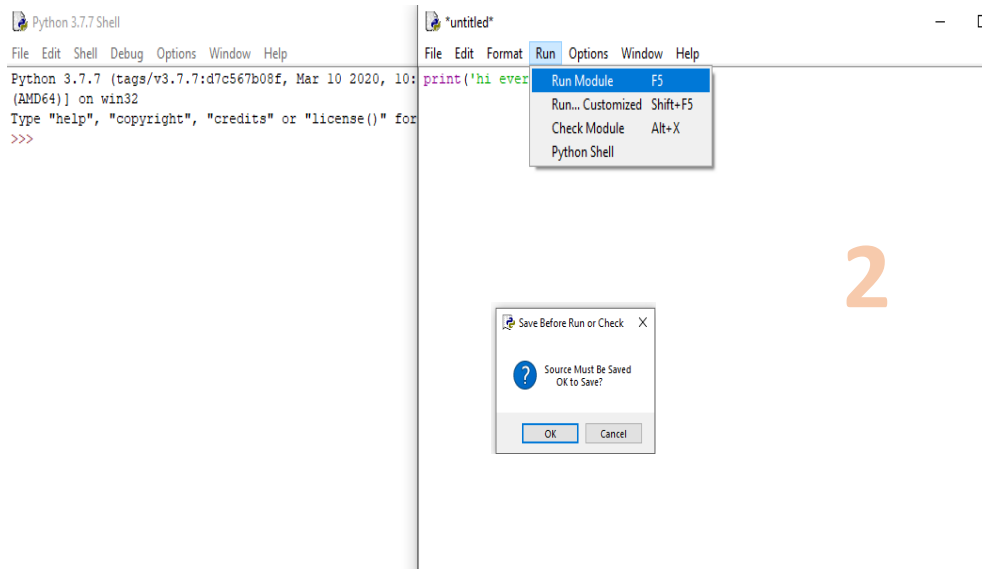


Fig.II.10: How to write a code in the Shell (step 2)

Ps: See that we must save before running module.

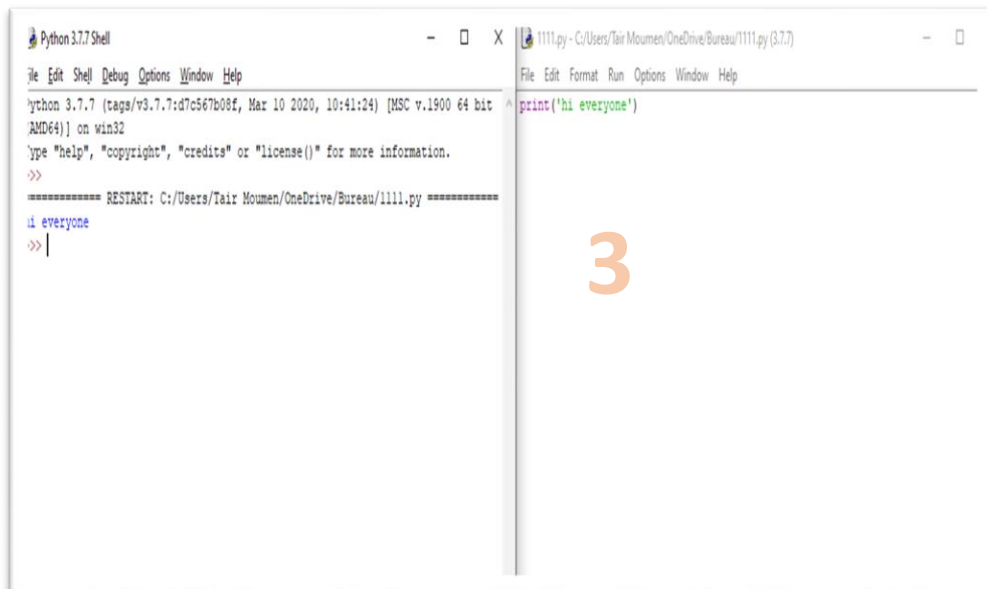


Fig.II.11: How to write a code in the Shell (step 3)

PS: notice that the result appears in the Shell.

B. Write the code in Shell Input()

a. Input():

We used the input() function to get our user's name and age.

```
myName = input("Please enter your name: ")
```

```
myAge = input("Please enter your age: ")
```

The string "Please enter your name: " is the prompt that will be displayed on the screen to give instructions to the user.

After the user enters the relevant information, this information is stored as a string in the variable myName.

The next input statement prompts the user for his age and stores the information as a string in the variable myAge.

The input() function differs slightly in Python 2 and Python 3. In Python 2, if we want to accept user input as a string, we have to use the raw_input() function instead.

b. Print():

The print() function is used to display information to users.

It accepts zero or more expressions as parameters, separated by commas.

```
print("my name is ",myName , "I'm" myAge, "years old.")
```

```
#We will getmy Name is moumen I am 25 years old.
```

c. Variables

Unlike other programming languages, Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Variables are names given to data that we need to store and manipulate in our programs.

For instance, suppose our program needs to store the age of a user. To do that, we can name this data X and define the variable X using the following statement.

```
X=25
```

We can also define multiple variables at one go. To do that simply write:

```
Age,Name= 25,'Moumen'
```

This is equivalent to

```
Age = 30
```

```
Name='Moumen'
```

Variable Names:

A variable can have a letter like x and y or a more descriptive name like age, name...etc.

Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

Variable names are case-sensitive (age, Age and AGE are three different variables)

- Legal variable names:

```
myvar = "John"  
my_var = "John"  
_my_var = "John"  
myVar = "John"  
MYVAR = "John"  
myvar2 = "John"
```

- Illegal variable names:

```
2myvar = "John"  
my-var = "John"  
my var = "John"
```

Note that the (=) sign in the statement $X = 25$ has a different meaning from the (=) sign we learned in Math. In programming, the (=) sign is known as an assignment sign. It means we are assigning the value on the right side of the (=) sign to the variable on the left.

A good way to understand the statement $X = 25$ is to think of it as $X \leq 0$.

The statements $x = y$ and $y = x$ have very different meanings in programming.

Example :

```
x = 5  
y = 10  
x=y  
print(" x =" ,x )  
print (" y =" , y )
```

Now when we run the program, we should get this output:

```
x =10  
y= 10
```

Although x has an initial value of 5 (declared on the first line), the third line $x=y$ assigns the value of y to x ($x \leq y$), hence changing the value of x to 10 while the value of y remains unchanged.

Next, we modify the program by changing ONLY ONE statement: we change the third line from $x = y$ to $y=x$.

Mathematically, $x=y$ and $y=x$ means the same thing. However, this is not so in programming.

We run the second program. We will now get:

```
x =5  
y=5
```

You can see that in this example, the x value remains as 5, but the value of y is changed to 5.

This is because the statement $y = x$ assigns the value of x to y ($y \leq x$). y becomes 5 while x remains unchanged as 5.

d. Basic Operators

Besides assigning a variable an initial value, we can also perform the usual mathematical operations on variables. Basic operators in Python include `+`, `-`, `*`, `/`, `//`, `%` and `**` which represent addition, subtraction, multiplication, division, floor division, modulus and exponent respectively.

Example :

Suppose $x = 5$, $y = 2$

Addition : $x + y = 7$

Subtraction : $x - y = 3$

Multiplication : $x * y = 10$

Division : $x / y = 2.5$

Floor Division:

$x // y = 2$ (rounds down the answer to the nearest whole number).

Modulus:

$x \% y = 1$ (gives the remainder when 5 is divided by 2).

Exponent:

$x ** y = 25$ (5 to the power of 2).

More Assignment Operators

Besides the `=` sign, there are a few more assignment operators in Python (and most programming languages). These include operators like `+=`, `-=` and `*=`.

Suppose we have the variable `x`, with an initial value of 10. If we want to increment `x` by 2, we can write `x = x + 2`

The program will first evaluate the expression on the right (`x + 2`) and assign the answer to the left. So eventually the statement above becomes `x = 12`.

Instead of writing `x = x + 2`, we can also write `x += 2` to express the same meaning.

The `+=` sign is actually a shorthand that combines the assignment sign with the addition operator. `x += 2` simply means `x = x + 2`.

Similarly, if we want to do a subtraction, we can write `x = x - 2` or `x -= 2`. The same works for all the 7 operators mentioned above.

C. Data Types in Python

We'll take a look at some basic data types in Python, specifically the integer, float and string. Next, we will explore the concept of type casting. Finally, we will discuss three more advanced data types in Python: the list, tuple and dictionary.

a. Integers

Integers are numbers with no decimal parts, such as -5, -4, -3, 0, 5, 7 etc.

To declare an integer in Python, simply we write `variableName = initial value`

Example:

```
userAge = 20, mobileNumber = 123456789
```

b. Float

Float refers to numbers that have decimal parts, such as 1.234, -0.023, 12.01.

To declare a float in Python, we write `variableName = initial value`

Example:

```
userHeight = 1.82, userWeight = 67.2
```

c. Strings

Strings are used quite often in Python. Strings, are just that, a string of characters – which's anything you can type on the keyboard in one keystroke, like a letter, a number, or a back-slash.

Python recognizes single and double quotes as the same thing, the beginning and end of the strings.

```
>>> "string list"
'string list'
>>> 'string list'
'string list'
```

What if we have a quote in the middle of the string? Python needs help to recognize quotes as part of the English language and not as part of the Python language.

```
>>> "I 'cant do that"
'I 'cant do that'
>>> "He said \"no\" to me"
'He said "no" to me'
```

Now we can also join (concatenate) strings with use of variables as well.

```
>>> a = "first"
>>> b = "last"
>>> a + b
'firstlast'
```

If we want a space in between, we can change a to the word with a space after.

```
>>> a = "first "
>>> a + b
'first last'
```

There are different string methods for us to choose from as well - like `upper()`, `lower()`, `replace()`, and `count()`.

`upper()` does just what it sounds like - changes your string to all uppercase letters.

```
>>> str = 'woah!'
>>> str.upper()
'WOAH!'
```

```
>>> str = 'WOAH!'
>>> str.lower()
'woah!'
```

replace() allows us to replace any character with another character.

```
>>> str = 'rule'
>>> str.replace('r', 'm')
'mule'
```

Finally, count() lets us know how many times a certain character appears in the string.

```
>>> number_list = ['one', 'two', 'one', 'two', 'two']
>>> number_list.count('two')
3
```

We can also format/create strings with the format() method.

```
>>> "{0} is a lot of {1}".format("Python", "fun!")
'Python is a lot of fun!'
```

d. List

List refers to a collection of data which are normally related. Instead of storing these data as separate variables, we can store them as a list. For instance, suppose our program needs to store the age of 5 users. Instead of storing them as user1Age, user2Age, user3Age, user4Age and user5Age, it makes more sense to store them as a list.

To declare a list, we write listName = [initial values]. Note that we use square brackets [] when declaring a list. Multiple values are separated by a comma.

Example:

```
userAge = [21, 22, 23, 24, 25]
```

We can also declare a list without assigning any initial values to it. We simply write listName = []. What we have now is an empty list with no items in it. We have to use the append() method mentioned below to add items to the list.

Individual values in the list are accessible by their indexes, and indexes always start from ZERO, not 1. This is a common practice in almost all programming languages, such as C and Java. Hence the first value has an index of 0, the next has an index of 1 and so forth. For instance, userAge[0] = 21, userAge[1] = 22.

Alternatively, we can access the values of a list from the back. The last item in the list has an index of -1, the second last has an index of -2 and so forth. Hence, userAge[-1] = 25, userAge[-2] = 24.

We can assign a list, or part of it, to a variable. If we write userAge2 = userAge, the variable userAge2 becomes [21, 22, 23, 24, 25].

If we write userAge3 = userAge[2:4], we are assigning items with index 2 to index 4-1 from the list userAge to the list userAge3. In other words, userAge3 = [23, 24].

The notation 2:4 is known as a slice. Whenever we use the slice notation in Python, the item at the start index is always included, but the item at the end is always excluded. Hence the notation 2:4

refers to items from index 2 to index 4-1 (i.e. index 3), which is why `userAge3 = [23, 24]` and not `[23, 24, 25]`.

The slice notation includes a third number known as the stepper. If we write `userAge4 = userAge[1:5:2]`, we will get a sub list consisting of every second number from index 1 to index 5-1 because the stepper is 2. Hence, `userAge4 = [22, 24]`.

In addition, slice notations have useful defaults. The default for the first number is zero, and the default for the second number is size of the list being sliced.

For instance,

`userAge[:4]` gives you values from index 0 to index 4-1 while `userAge[1:]` gives you values from index 1 to index 5-1 (since the size of `userAge` is 5, i.e. `userAge` has 5 items).

To modify items in a list, we write `listName[index of item to be modified] = new value`. For instance, if we want to modify the second item, we write `userAge[1] = 5`. Our list becomes `userAge = [21, 5, 23, 24, 25]`

To add items, we use the `append()` function. For instance, if we write `userAge.append(99)`, you add the value 99 to the end of the list. Our list is now `userAge = [21, 5, 23, 24, 25, 99]`

To remove items, we write `del listName[index of item to be deleted]`. For instance, if we write `del userAge[2]`, Our list now becomes `userAge = [21, 5, 24, 25, 99]` (the third item is deleted).

To fully appreciate the workings of a list, try running the following program.

Declaring the list, list elements can be of different data types:

```
myList = [1, 2, 3, 4, 5, 'Hello']
print(myList)
#We will get [1, 2, 3, 4, 5, 'Hello']
```

print the third item (recall: Index starts from zero).

```
print(myList[2])
#We will get 3
print the last item.
```

```
print(myList[-1])
#We will get Hello
```

Assign `myList` (from index 1 to 4) to `myList2` and print `myList2`.

```
myList2 = myList[1:5]
print (myList2)
#We will get [2, 3, 4, 5]
```

Modify the second item in myList and print the updated list.

```
myList[1] = 20
print(myList)
#We will get [1, 20, 3, 4, 5, 'Hello']
Append a new item to myList and print the updated list.
```

```
myList.append('How are you')
print(myList)
#We will get [1, 20, 3, 4, 5, 'Hello', 'How are you']
```

Remove the sixth item from myList and print the updated list.

```
del myList[5]
print(myList)
#We will get [1, 20, 3, 4, 5, 'How are you']
```

e. Tuple

Tuples are just like lists, but you cannot modify their values. The initial values are the values that will stay for the rest of the program. An example where tuples are useful is when your program needs to store the names of the months of the year.

To declare a tuple, we write:

```
tupleName = (initial values).
```

Notice that we use round brackets ()

When declaring a tuple. Multiple values are separated by a comma.

Example:

```
monthsOfYear = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
```

We access the individual values of a tuple using their indexes, just like with a list.

Hence, monthsOfYear[0] = "Jan", monthsOfYear[-1] = "Dec".

f. Dictionary

Dictionary is a collection of related data PAIRS. For instance, if we want to store the username and age of 5 users, we can store them in a dictionary.

To declare a dictionary, we write:

```
dictionaryName = {dictionary key : data}
```

With the requirement that dictionary keys must be unique (within one dictionary).

That is, you cannot declare a dictionary like this

```
myDictionary = {"Peter":38, "John":51, "Peter":13}.
```

This is because "Peter" is used as the dictionary key twice.

Note that we use curly brackets { } when declaring a dictionary.

Multiple pairs are separated by a comma.

Example:

```
userNameAndAge = {"Peter":38, "John":51, "Alex":13, "Alvin":"Not Available"}
```

We can also declare a dictionary using the dict() method.

To declare the userNameAndAge dictionary above, we write:

```
userNameAndAge = dict(Peter = 38, John = 51, Alex = 13, Alvin = "Not Available")
```

When we use this method to declare a dictionary, we use round brackets () instead of curly brackets {} and we do not put quotation marks for the dictionary keys.

To access individual items in the dictionary, we use the dictionary key, which is the first value in the {dictionary key : data} pair.

For instance, to get John's age, we write:

```
userNameAndAge["John"].
```

#We'll get the value 51.

To modify items in a dictionary, we write:

```
dictionaryName[dictionary key of item to be modified] = new data.
```

For instance, to modify the "John":51 pair, we write:

```
userNameAndAge["John"] = 21.
```

Our dictionary now becomes userNameAndAge = {"Peter":38, "John":21, "Alex":13, "Alvin":"Not Available"}.

We can also declare a dictionary without assigning any initial values to it. We simply write:

```
dictionaryName= { }. What we have now is an empty dictionary with no items in it.
```

To add items to a dictionary, we write:

```
dictionaryName[dictionary key] = data.
```

For instance, if we want to add "Joe":40 to our dictionary, we write:

```
userNameAndAge["Joe"] = 40.
```

Our dictionary now becomes userNameAndAge = {"Peter":38, "John":21, "Alex":13, "Alvin":"Not Available", "Joe":40}

To remove items from a dictionary, we write:

```
del dictionaryName[dictionary key].
```

For instance, to remove the "Alex":13 pair, we write:

```
del userNameAndAge["Alex"].
```

Our dictionary now becomes userNameAndAge = {"Peter":38, "John":21, "Alvin":"Not Available", "Joe":40}

D. Making Choices and Decisions

We will look at how to make your program smarter, capable of making choices and decisions. Specifically, we'll be looking at the if statement, for loop and while loop; These are known as control flow tools; they control the flow of the program.

Before we go into these control flow tools, we have to first look at condition statements.

a. Condition Statements

All control flow tools involve evaluating a condition statement. The program will proceed differently depending on whether the condition is met.

The most common condition statement is the comparison statement; if we want to compare whether two variables are the same, we use the `==` sign (double `=`). For instance, if we write `x == y`, we are asking the program to check if the value of `x` is equal to the value of `y`. If they are equal, the condition is met and the statement will evaluate to `True`. Else, the statement will evaluate to `False`. Other comparison signs include `!=` (not equals), `<` (smaller than), `>` (greater than), `<=` (smaller than or equals to) and `>=` (greater than or equals to).

The list below shows how these signs can be used and gives examples of statements that will evaluate to `True`.

Not equals:

`5 != 2`

Greater than:

`5 > 2`

Smaller than:

`2 < 5`

Greater than or equals to:

`5 >= 2`

`5 >= 5`

Smaller than or equals to:

`2 <= 5`

`2 <= 2`

We also have three logical operators, `and`, `or`, `not`, that are useful if we want to combine multiple conditions.

The `and` operator returns `True` if all conditions are met. Else it will return `False`.

For instance, the statement

`5==5 and 2>1` will return `True` since both conditions are `True`.

The `or` operator returns `True` if at least one condition is met. Else it will return `False`. The statement `5 > 2 or 7 > 10 or 3 == 2` will return `True` since the first condition `5 > 2` is `True`.

The `not` operator returns `True` if the condition after the `not` keyword is `False`. Else it will return `False`. The statement `not 2 > 5` will return `True` since `2` is not greater than `5`.

b. If Statement

The `if` statement is one of the most commonly used control flow statements. It allows the program to evaluate if a certain condition is met, and to perform the appropriate action based on the result of the evaluation. The structure of an `if` statement is as follows:

if condition 1 is met:

do A

elif condition 2 is met:

do B

elif condition 3 is met:

```
do C
elif condition 4 is met:
do D
else:
do E
```

elif stands for “else if” and you can have as many elif statements as you like.

If you’ve coded in other languages like C or Java before, you may be surprised to notice that no parentheses () are needed in Python after the if, elif and else keyword.

In addition, Python does not use curly { } brackets to define the start and end of the if statement. Rather, Python uses indentation. Anything indented is treated as a block of code that will be executed if the condition evaluates to true.

Example:

```
userInput = input('Enter 1 or 2: ')
if userInput == "1":
print ("Hello World")
print (“How are you?”)
elif userInput == "2":
print ("Python Rocks!")
print (“I love Python”)
else:
print ("You did not enter a valid number")
```

The program first prompts the user for an input using the input function. The result is stored in the userInput variable as a string.

Next the statement `if userInput == "1":` compares the userInput variable with the string “1”. If the value stored in userInput is “1”, the program will execute all statements that are indented until the indentation ends. In this example, it’ll print “Hello World”, followed by “How are you?”.

Alternatively, if the value stored in userInput is “2”, the program will print “Python Rocks”, followed by “I love Python”.

For all other values, the program will print “You did not enter a valid number”.

c. Inline If

An inline if statement is a simpler form of an if statement and is more convenient if you only need to perform a simple task.

The syntax is:

```
do Task A if condition is true else do Task B
```

Example:

```
print (“This is task A” if myInt == 10 else “This is task B”)
```

This statement prints “This is task A” (Task A) if myInt equals to 10. Else it prints “This is taskB” (Task B).

d. For Loop: (syntax color)

The for loop executes a block of code repeatedly until the condition in the for statement is no longer valid.

Looping through an iterable

In Python, an iterable refers to anything that can be looped over, such as a string, list or tuple.

The syntax for looping through an iterable is as follows:

for a in iterable:

```
print (a)
```

Example:

```
pets = ['cats', 'dogs', 'rabbits', 'hamsters']
```

```
for myPets in pets:
```

```
print (myPets)
```

In the program, we first declare the list `pets` and give it the members 'cats', 'dogs', 'rabbits' and 'hamsters'.

Next the statement `for myPets in pets:` loops through the `pets` list and assigns each member in the list to the variable `myPets`.

The first time the program runs through the for loop, it assigns 'cats' to the variable `myPets`. The statement `print (myPets)` then prints the value 'cats'. The second time the program loops through the for statement, it assigns the value 'dogs' to `myPets` and prints the value 'dogs'. The program continues looping through the list until the end of the list is reached. If we run the program, we will get:

```
cats dogs rabbits hamsters
```

We can also display the index of the members in the list. To do that, we use the `enumerate()` function.

```
for index, myPets in enumerate(pets):
```

```
print (index, myPets)
```

```
#This will give us the output
```

```
0 cats
```

```
1 dogs
```

```
2 rabbits
```

```
3 hamsters
```

How to loop through a string:

```
message = 'Hello'
```

```
for i in message:
```

```
print (i)
```

```
#The output is
```

```
H e l l o
```

Looping through a sequence of numbers:

To loop through a sequence of numbers, the built-in `range()` function comes in handy.

The `range()`

function generates a list of numbers and has the syntax `range (start, end, step)`.

If start is not given, the numbers generated will start from zero.

Note: A useful tip to remember here is that in Python (and most programming languages), unless otherwise stated, we always start from zero.

For instance, the index of a list and a tuple starts from zero.

When using the format() method for strings, the positions of parameters start from zero.

When using the range() function, if start is not given, the numbers generated start from zero.

If step is not given, a list of consecutive numbers will be generated (i.e. step = 1). The end value must be provided. However, one weird thing about the range() function is that the given end value is never part of the generated list.

For instance, range(5) will generate the list [0, 1, 2, 3, 4] range(3, 10) will generate [3, 4, 5, 6, 7, 8, 9] range(4, 10, 2) will generate [4, 6, 8]

To see how the range() function works in a for statement, we run the following code:

```
for i in range(5):
    print (i)
#We should get
0
1
2
3
4
```

e. While Loop

The next control flow statement we are going to look at is the while loop. Like the name suggests, a while loop repeatedly executes instructions inside the loop while a certain condition remains valid.

The structure of a while statement is as follows:

while condition is true:

do A

Most of the time when using a while loop, we need to first declare a variable to function as a loop counter. Let's just call this variable counter. The condition in the while statement will evaluate the value of counter to determine if it smaller (or greater) than a certain value. If it is, the loop will be executed. Let's look at a sample program.

```
counter = 5
while counter > 0:
    print ("Counter = ", counter)
    counter = counter - 1
```

#If we run the program, we will get the following output

```
Counter = 5
Counter = 4
Counter = 3
Counter = 2
Counter = 1
```

At first look, a while statement seems to have the simplest syntax and should be the easiest to use. However, one has to be careful when using while loops due to the danger of infinite loops. Notice

that in the program above, we have the line `counter = counter - 1`? This line is crucial. It decreases the value of `counter` by 1 and assigns this new value back to `counter`, overwriting the original value. We need to decrease the value of `counter` by 1 so that the loop condition `while counter > 0` will eventually evaluate to `False`. If we forget to do that, the loop will keep running endlessly resulting in an infinite loop. If we want to experience this first hand, just delete the line `counter = counter - 1` and try running the program again. The program will keep printing `counter = 5` until we somehow kill the program.

E. Functions

Functions are blocks of reusable code that perform a single task.

You use `def` to define (or create) a new function then you call a function by adding parameters to the function name.

```
>> def multiply(num1, num2):  
    return num1 * num2
```

```
>>> multiply(2, 2)  
4
```

You can also set default values for parameters.

```
>>> def multiply(num1, num2=10):  
    return num1 * num2
```

```
>>> multiply(2)  
20
```

```
[5][6][7][29]
```

II.8. Conclusion

There are a large number of high-level programming languages available, such as C, C++, and Java. All high-level programming languages are very similar to one another; What differs is mainly the syntax, the libraries available and the way we access those libraries; A library is simply a collection of resources and pre-written codes that we can use when we write our programs. If you learn one language well, you can easily learn a new language in a fraction of the time it took you to learn the first language.

Python is chosen by the best in the world, companies like Google, Facebook, Instagram or Microsoft, and it's growing very fast. Developers love its features. Python is simple, approachable, versatile and complete. This language is an obvious choice for machine learning, data analysis and visualization. AI-first companies should love it. Just like any programming language, Python is not a perfect fit for all projects.

Python is getting more attention than usual these past years, becoming one of the most popular programming languages in the world. In the hands of a skilled developer, Python is one of the best programming languages. And If you are new to programming, Python is a great place to start.

CHAPTER

III

The GUI Of

PythonQT

III.1. Introduction

A. What is PyQT?

PyQT is a set of Python v2 and v3 bindings for The Qt Company's Qt application framework and runs on all platforms supported by Qt including Windows, OS X, Linux, iOS and Android. PyQt5 supports Qt v5. PyQt4 supports Qt v4 and will build against Qt v5. The bindings are implemented as a set of Python modules and contain over 1,000 classes.

PyQt4 and Qt v4 are no longer supported and no new releases will be made. PyQt5 and Qt v5 are strongly recommended for all new development.

PyQT is dual licensed on all supported platforms under the GNU GPL v3 and the Riverbank Commercial License. Unlike Qt, PyQT is not available under the LGPL.

PyQT does not include a copy of Qt. You must obtain a correctly licensed copy of Qt yourself. However, binary wheels of the GPL version of PyQt5 are provided and these include a copy of the LGPL version of Qt.[30]

B. Why we used PyQT?

Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, a Chromium based web browser, as well as traditional UI development.

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.

PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those applications.

PyQT brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python.

Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets.

Qt classes employ a signal/slot mechanism for communicating between objects that is type safe but loosely coupled making it easy to create re-usable software components.

Qt also includes Qt Designer, a graphical user interface designer. PyQT is able to generate Python code from Qt Designer. It is also possible to add new GUI controls written in Python to Qt Designer.

Python is a simple but powerful object-orientated language. Its simplicity makes it easy to learn, but its power means that large and complex applications can be created.

Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle.

Much of Python's power comes from its comprehensive set of extension modules providing a wide variety of functions including HTTP servers, XML parsers, database access, data compression tools and, of course, graphical user interfaces. Extension modules are usually implemented in either Python, C or C++. Using tools such as SIP it is relatively straight forward to create an extension module that encapsulates an existing C or C++ library. Used in this way, Python can then become the glue to create new applications from established libraries.

PyQT combines all the advantages of Qt and Python. A programmer has all the power of Qt, but is able to exploit it with the simplicity of Python. [8] [30] [31]

III.2. History of PyQT

PyQT is available in two editions: PyQt4 which will build against Qt 4.x and 5.x and PyQt5 which will only build against 5.x. Both editions can be built for Python 2 and 3. PyQT contains over 620 classes that cover graphical user interfaces (GUI), XML handling, network communication, SQL databases, Web browsing and other technologies available in Qt .[30]

PyQt5 Releases

- 5.14.0 December 19th, 2019
- 5.13.2 November 3rd, 2019
- 5.13.1 September 13th, 2019
- 5.13.0 July 5th, 201
- 5.12.3 June 26th, 2019
- 5.12.2 May 6th, 2019
- 5.12.1 March 19th, 2019
- 5.12 February 10th, 2019
- 5.11.3 October 3rd, 2018
- 5.11.2 July 2nd, 2018
- 5.11 June 23rd, 2018
- 5.10.1 February 27th, 2018
- 5.10 January 23rd, 2018
- 5.9.2 November 24th, 2017
- 5.9.1 November 2nd, 2017
- 5.9 July 4th, 2017
- 5.8.2 March 30th, 2017
- 5.8.1 March 7th, 2017
- 5.8 February 15th, 2017
- 5.7.1 December 29th, 2016
- 5.7 July 25th, 2016
- 5.6 April 28th, 2016
- 5.6.dev1604081748 April 8th, 2016 [30] [31]

III.3. Differences between PyQt4 and PyQt5

PyQt5 is not compatible with PyQt4 (although experience shows that the effort in porting applications from PyQt4 to PyQt5 is not great).

This section describes the main differences between the two.

- Version: For PyQt5 Versions of Python earlier than v2.6 are not supported.
PyQt4 supports version earlier than v2.6.
- QtGui Module: PyQt4's QtGui module has been split into PyQt5's QtGui, QtPrintSupport and QtWidgets modules.
- QtOpenGL Module: Only the QGLContext, QGLFormat and QGLWidget classes are supported by PyQt5.
- QtWebKit Module: PyQt4's QtWebKit module has been split into PyQt5's QtWebKit and QtWebKitWidgets modules.
- Pyqtconfig Module: PyQt4's pyqtconfig module is not supported.
- The section The PyQt5 Extension API describes the support that PyQt5 provides to third-party packages (e.g. QScintilla) that want to build on top of PyQt5.
- Dbus.mainloop.qt Module: PyQt4's dbus.mainloop.qt module is called dbus.mainloop.pyqt5 in PyQt5. This allows them to be installed side by side. Their functionality is identical.
- QDataStream: In PyQt5, the readUInt8(), readInt8(), writeUInt8() and writeInt8() methods all interpret the values being read and written as numeric values. In PyQt4 they are interpreted as single character strings.
- QFileDialog: The getOpenFileNameAndFilter(), getOpenFileNamesAndFilter() and getSaveFileNameAndFilter() methods of PyQt4's QFileDialog have now been renamed getOpenFileName(), getOpenFileNames() and getSaveFileName() respectively in PyQt5. PyQt4's implementations of getOpenFileName(), getOpenFileNames() and getSaveFileName() are not supported in PyQt5.
- QGraphicsItemAnimation: Support for the deprecated QGraphicsItemAnimation class has been removed. If porting an existing PyQt4 application then consider first updating it to use QPropertyAnimation instead.
- QMatrix: Support for the deprecated QMatrix class has been removed. If porting an existing PyQt4 application then consider first updating it to use QTransform instead.
- QPyObject: PyQt4 implements the QPyObject as a workaround for the inability to define a Python class that is sub-classed from more than one Qt class. PyQt5 does support

the ability to define a Python class that is sub-classed from more than one Qt class so long as all but one of the Qt classes are interfaces, i.e. they have been declared in C++ as such using `Q_DECLARE_INTERFACE`. Therefore, `QPyTextObject` is not implemented in PyQt5.

- `QSet`: In PyQt4, `QSet` was implemented as a list in Python v2 and a set in Python v3. In PyQt5 `QSet` is always implemented as a set.
- `pyuic5`: `pyuic5` does not support the `--pyqt3-wrapper` flag of `pyuic4`.

Unfortunately, it is not possible to use both the PyQt4 and PyQt5 installers at the same time. [32]

III.4. PyQt5 Modules

PyQt5 comprises a number of different components. There are a number of Python extension modules. These are all installed in the PyQt5 Python package and are described in this list:

Tabl.III.1: Modules of PyQt5

Module	Description
Enginio	Classes for accessing Qt Cloud Services (deprecated)
QAxContainer	Classes for accessing ActiveX controls and COM objects
Qt	A consolidation of other modules
Qt3Danimation	Classes that support animations in simulations
Qt3Dcore	The core classes to support near-realtime simulation systems
Qt3Dextras	Pre-built elements for use with Qt3D
Qt3Dinput	Classes to handle user input when using Qt3D
Qt3Dlogic	Classes that enable frame synchronization
Qt3Drender	Classes that enable 2D and 3D rendering
QtAndroidExtras	Additional classes specific to Android
QtBluetooth	Classes to support connectivity between Bluetooth enabled devices
QtChart	Classes to support the creation of 2D charts
QtCore	The core Qt classes
QtDBus	Classes to support IPC using the D-Bus protocol
QtDataVisualization	Classes to support the visualization of data in 3D
QtDesigner	Classes to allow Qt Designer to be extended using Python
QtGui	The core classes common to widget and OpenGL GUIs
QtHelp	Classes for creating and viewing searchable documentation

QtLocation	Classes for creating mapping applications
QtMacExtras	Additional classes specific to macOS and iOS
QtMultimedia	Classes for multimedia content, cameras and radios
QtMultimediaWidgets	Provides additional multimedia related widgets and controls
QtNetwork	The core network classes
QtNetworkAuth	The network authorization classes
QtNfc	Classes to support connectivity between NFC enabled devices
QtOpenGL	Classes for rendering OpenGL in traditional widgets (deprecated)
QtPositioning	Classes for obtaining positioning information from satellite, Wi-Fi etc.
QtPrintSupport	Classes to make printing easier and more portable
QtPurchasing	Classes to support in-app purchasing from app stores
QtQml	Classes for integrating with the QML language
QtQuick	Classes for extending QML applications with Python code
QtQuickWidgets	Classes for rendering a QML scene in traditional widgets
QtRemoteObjects	Classes for sharing the API of a QObject between processes or systems
QtSensors	Classes for accessing a system's hardware sensors
QtSerialPort	Classes for accessing a system's serial ports
QtSql	Classes for integrating with SQL databases
QtSvg	Classes providing support for SVG
QtTest	Support for unit testing of GUI applications
QtWebChannel	Classes for peer-to-peer communication between Python and HTML/JavaScript
QtWebEngine	Classes for integrating QML Web Engine objects with Python
QtWebEngineCore	The core Web Engine classes
QtWebEngineWidgets	A Chromium based web browser
QtWebKit	A WebKit2 based web browser (deprecated)
QtWebKitWidgets	A WebKit1 based web browser (deprecated)
QtWebSockets	Classes that implement the WebSocket protocol
QtWidgets	Classes for creating classic desktop-style Uis
QtWinExtras	Additional classes specific to Windows

QtX11Extras	Additional classes specific to X11
QtXml	Classes for supporting SAX and DOM interfaces to XML
QtXmlPatterns	Classes that support additional XML technologies.
sip	Utilities for bindings developers and users
uic	Classes for handling the files created by Qt Designer

A typical GUI based application's top-level window is created by QMainWindow widget object. Some widgets as listed above take their appointed place in this main window, while others are placed in the central widget area using various layout managers. [8] [31] [33] [34]

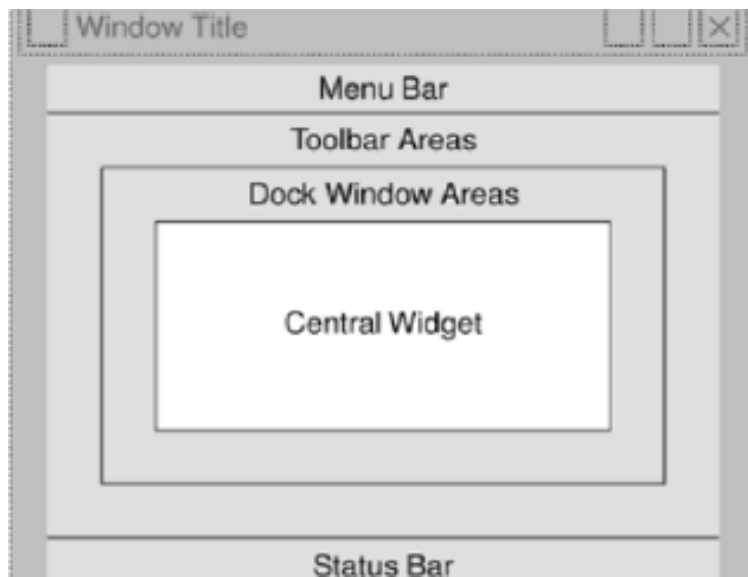


Fig.III.1: The QMainWindow framework [34]

III.5. How to install PyQt5

PyQT is often not installed by default. The PyQT module can be used to create desktop applications with Python. We will learn how to install the PyQT module and PyQT tools.

Desktop applications made with PyQT are cross platform, they will work on Microsoft Windows, Apple Mac OS X and Linux computers (including Raspberry Pi).

To install PyQt5 on Windows there are a few steps you need to take:

- Search for Command Prompt in Start Menu then Run as administrator

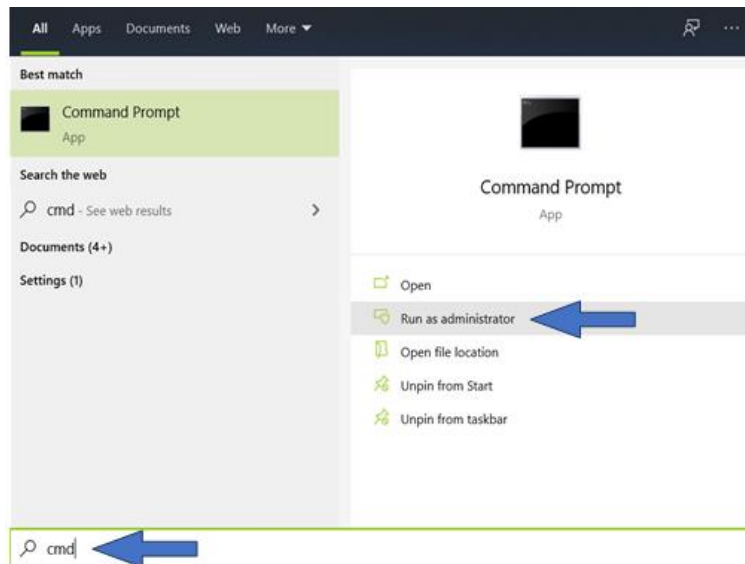


Fig.III.2: Run Command Prompt

- To install PyQt5 write pip install pyqt5 then Enter

```
Select Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32> pip install pyqt5
Collecting pyqt5
  Downloading https://files.pythonhosted.org/packages/d7/8e/5fa1dd8095728fa754e96633d4c97e0283fb0be5ab3a0a25f7df054deff1/PyQt5-5.14.2-5.14.2-cp35.cp36.cp37.cp38-none-win_amd64.whl (52.9MB)
    |#####| 52.9MB 226KB/s
Collecting PyQt5-sip<13, >=12.7 (from pyqt5)
  Downloading https://files.pythonhosted.org/packages/11/9f/093f7aa50af963a6cc825d1392770ea4ad821f175de1cd8bc6646be27a6/PyQt5_sip-12.7.2-cp37m-win_amd64.whl (58kB)
    |#####| 61kB 245KB/s
Installing collected packages: PyQt5-sip, pyqt5
Successfully installed PyQt5-sip-12.7.2 pyqt5-5.14.2
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\WINDOWS\system32>
```

Fig.III.3: Install pyqt5

- To install PyQt5 tool write pip install pyqt5-tools then Enter

```
Administrator: Command Prompt
C:\WINDOWS\system32> pip install pyqt5-tools
Collecting pyqt5-tools
  Downloading https://files.pythonhosted.org/packages/20/8e/c5ee76c481683270662e3f830edac91485f23041efaa709a3e016647ef/pyqt5_tools-5.11.0-5.11.0-cp37-none-win_amd64.whl (67.2MB)
    |#####| 67.2MB 233KB/s
Collecting pyqt5==5.11.0 (from pyqt5-tools)
  Downloading https://files.pythonhosted.org/packages/3b/d3/76670a331935f58f9a2ebd5c6e96706f15c458fa699306a5d323160/PyQt5-5.11.0-5.11.0-cp35.cp36.cp37.cp38-none-win_amd64.whl (49.7MB)
    |#####| 49.7MB 104KB/s
Collecting python-dotenv (from pyqt5-tools)
  Downloading https://files.pythonhosted.org/packages/ch/2a/07f87440444fd2c587ba7186479d766a1c7df9c8270c90e07f194c5/python_dotenv-0.11.0-py2.py3-none-any.whl
Collecting click (from pyqt5-tools)
  Downloading https://files.pythonhosted.org/packages/d6/c0/4d8f43a9b16e289f364784220318a63b54b6ac3b1ba9f5d602f1065546/click-7.1.1-py2.py3-none-any.whl (82kB)
    |#####| 92kB 280KB/s
Requirement already satisfied: PyQt5_sip<13, >=4.19.14 in c:\program files\python37\lib\site-packages (from pyqt5==5.11.0)
Installing collected packages: pyqt5, python-dotenv, click, pyqt5-tools
  Found existing installation: PyQt5 5.14.2
  Uninstalling PyQt5-5.14.2:
    Successfully uninstalled PyQt5-5.14.2
Successfully installed click-7.1.1 pyqt5-5.11.0 pyqt5-tools-5.11.0 python-dotenv-0.11.0
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\WINDOWS\system32>
```

Fig.III.4: Install pyqt5-tools

- To check that we installed PyQt5 successfully we open Python and write import PyQt5



```
Python 3.7.7 Shell
File Edit Shell Debug Options Window Help
Python 3.7.7 (tags/v3.7.7:d7c567b00f, Mar 10 2020, 10:41:24) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import
SyntaxError: invalid syntax
>>> import PyQt5
>>>
>>>
>>> import PyQt4
Traceback (most recent call last):
  File "<ipython>#5>", line 1, in <module>
    import PyQt4
ModuleNotFoundError: No module named 'PyQt4'
>>>
```

Fig.III.5: Check that we installed PyQt5

Ps: notice that we should write P and Q in capital letter

III.6. Hello World Example

A. With PyQt4:

The below code shows a small window on the screen.

```
# Here we provide the necessary imports.
# The basic GUI widgets are located in QtGui module.
import sys
from PyQt4.QtGui import QApplication, QWidget
# Every PyQt4 application must create an application object.
# The application object is located in the QtGui module.
app = QApplication(sys.argv)
# The QWidget widget is the base class of all user interface objects in PyQt4.
# We provide the default constructor for QWidget. The default constructor has no parent.
root = QWidget()
root.resize(320, 240) # The resize() method resizes the widget.
root.setWindowTitle("Hello, World!") # Here we set the title for our window.
root.show() # The show() method displays the widget on the screen.
sys.exit(app.exec_()) # Finally, we enter the mainloop of the application.
```

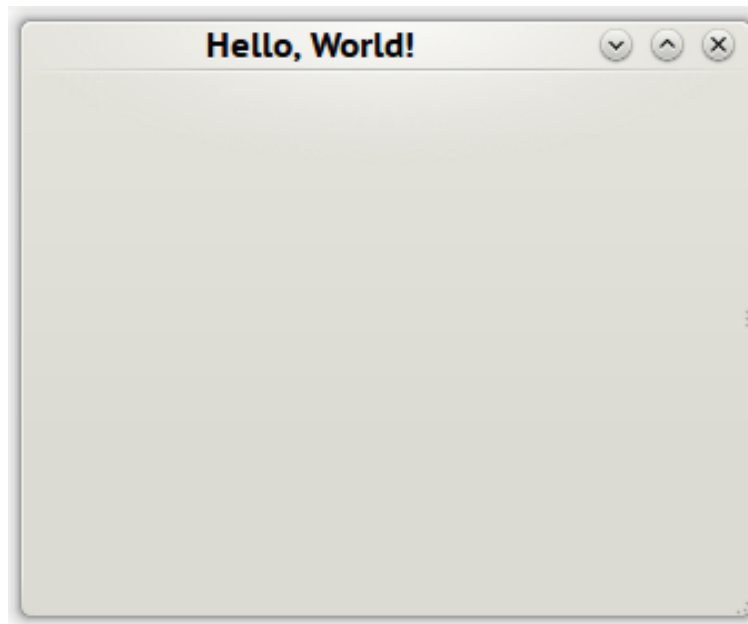



Fig.III.6: The result in **KDE Plasma 4** (Linux System) [34]

B. With PyQt5

The below code shows a small window on the screen.

```
# Here we provide the necessary imports.  
# The basic GUI widgets are located in QtWidgets module.  
import sys  
from PyQt5.QtWidgets import QApplication, QWidget  
# Every PyQt5 application must create an application object.  
# The application object is located in the QtWidgets module.  
app=QApplication(sys.argv)  
# The QWidget widget is the base class of all user interface objects in PyQt5.  
# We provide the default constructor for QWidget. The default constructor has no parent.  
# A widget with no parent is called a window.  
root=QWidget()  
root.resize(320,240)# The resize() method resizes the widget.  
root.setWindowTitle("Hello, World!")# Here we set the title for our window.  
root.show()# The show() method displays the widget on the screen.  
sys.exit(app.exec_())# Finally, we enter the mainloop of the application.
```

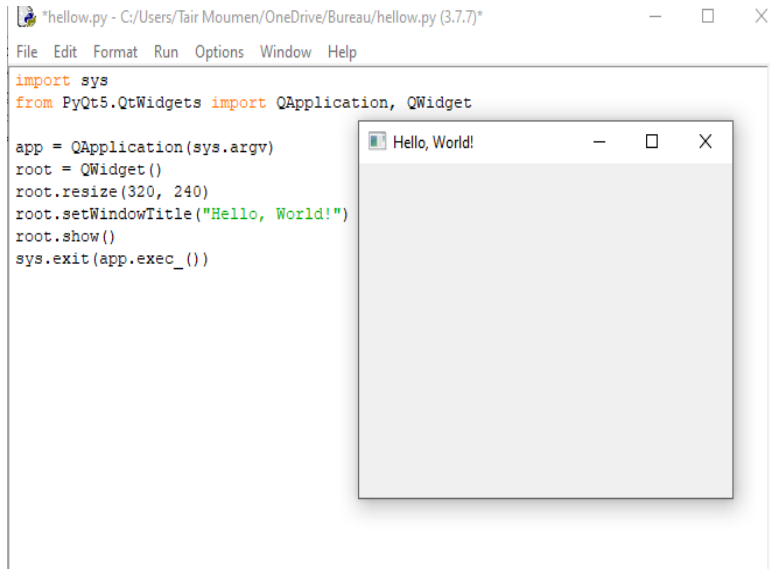


Fig.III.7: The result in **windows**

C. With Designer

- We can create window using Designer then add it to the code.
- When we install PyQt5, we can find designer in the following source:
 "C:\Program Files\Python37\Lib\site-packages\pyqt5_tools\Qt\bin"

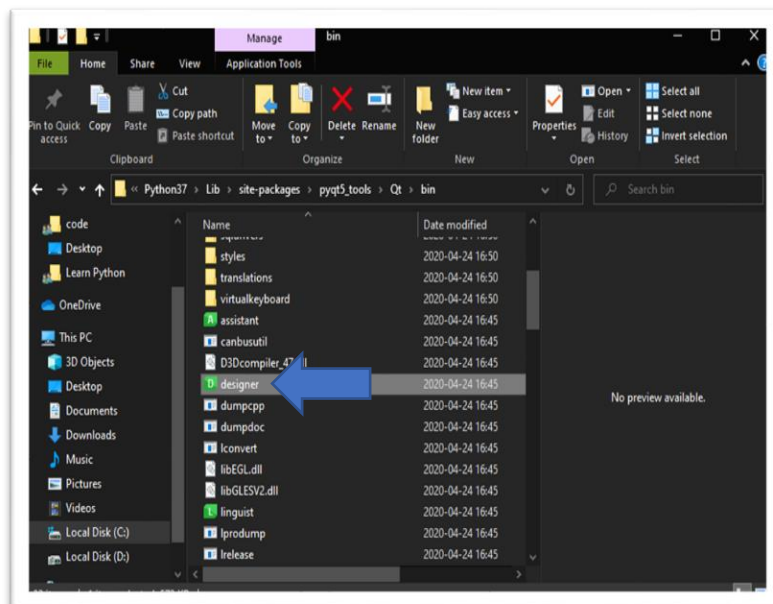


Fig.III.8: Designer source

- We can also search about designer in windows menu

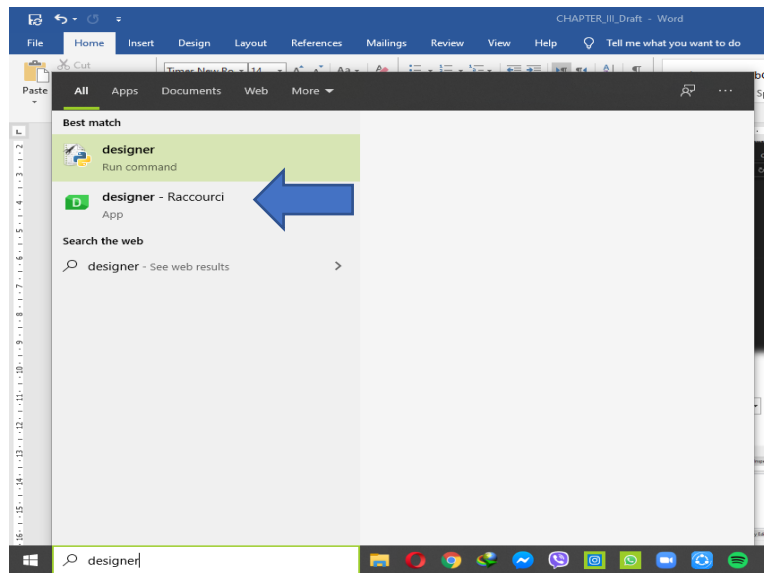


Fig.III.9: Search about Designer in windows Menu

- We open the application

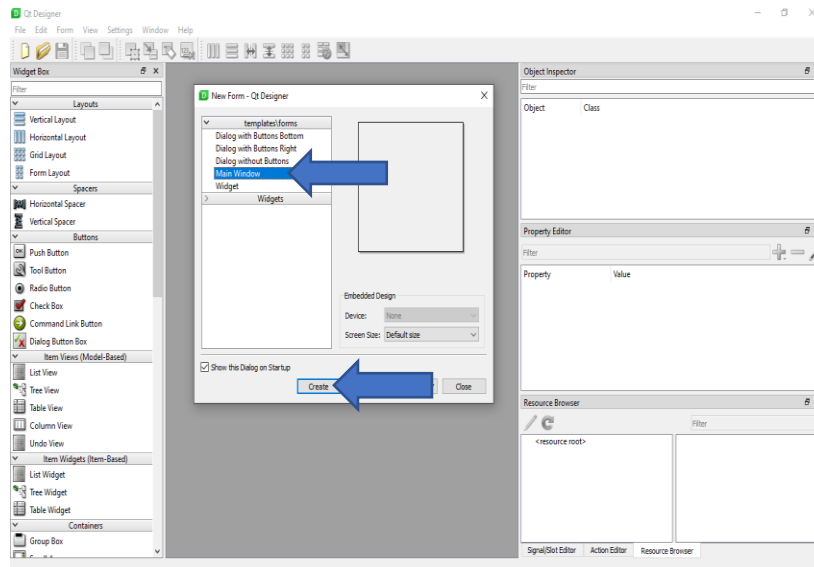


Fig.III.10: Interface of the application

See that we can also create widgets or dialogs to add it to the mainwindow, we have a lot of options to create.

- Select Main Window and Click on create.

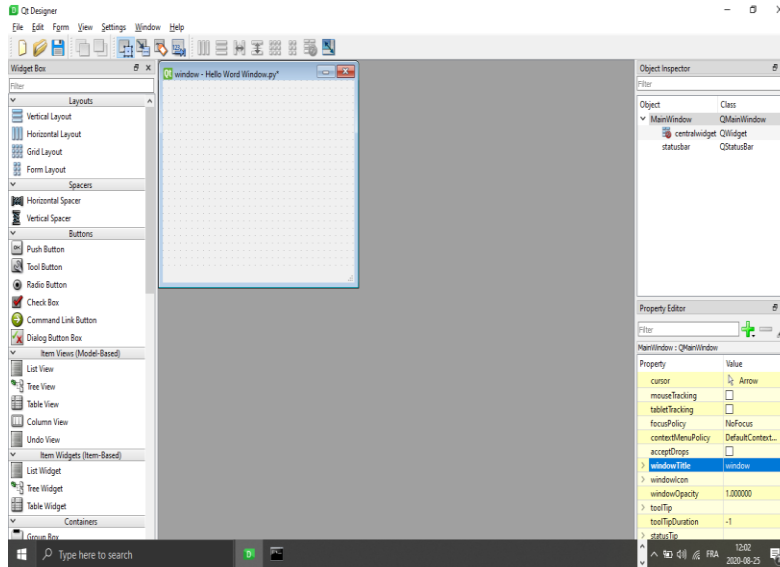


Fig.III.11: Create window in designer step1

After we create the main window, we can change the size, the front, the color, whatever inside the window, we can add menus, texts, buttons, images, GIFs, we have a lot of options to make change on and to add them.

After create the main window or widgets we can review them by going to Form the Preview... (CTRL+R).

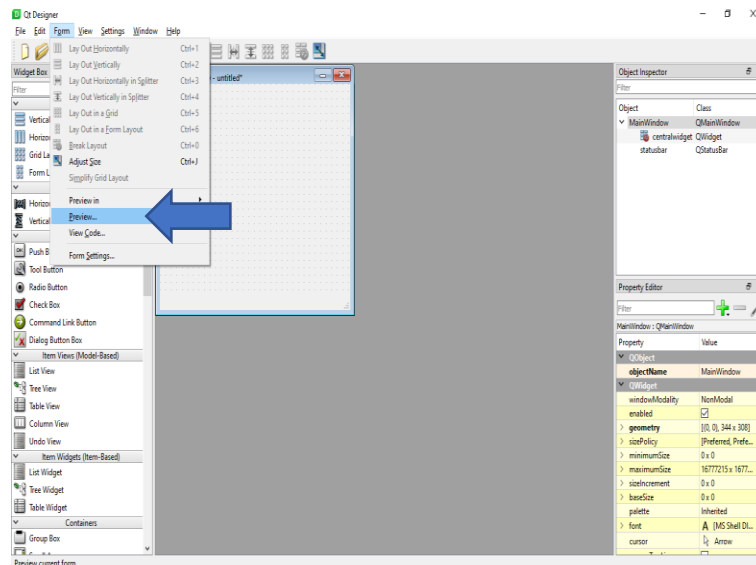


Fig.III.12: Create window in designer step2

- Click on Preview.

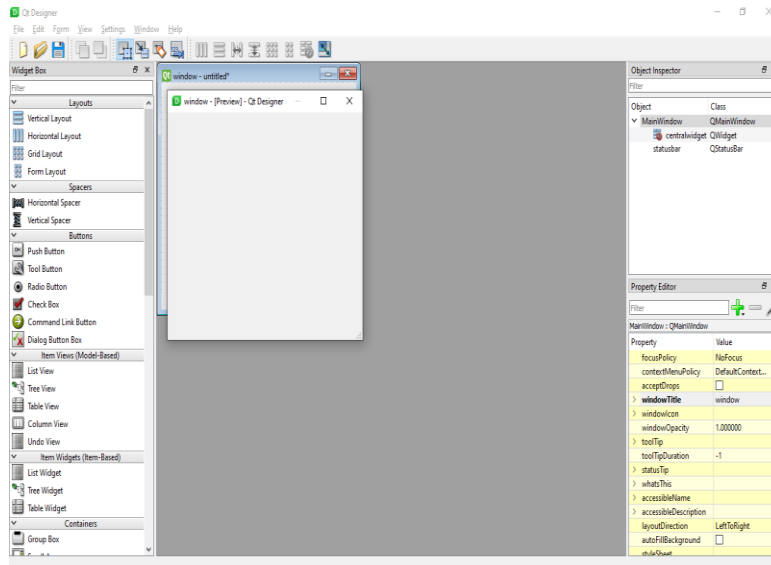


Fig.III.13: Create window in designer step3

After we see our main window and when we are done making changes on it we save it by clicking on save in the toolbar.

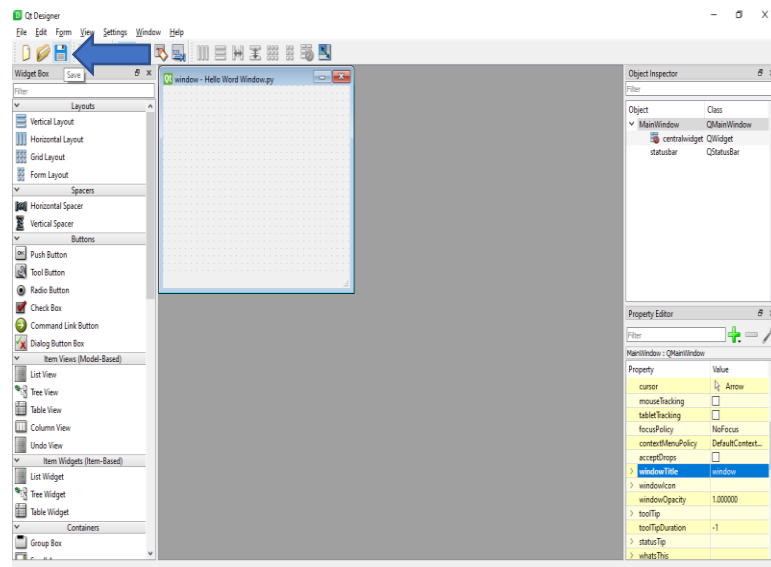


Fig.III.14: Create window in designer step4

- We save it in the disktop or wherever we want to save it.

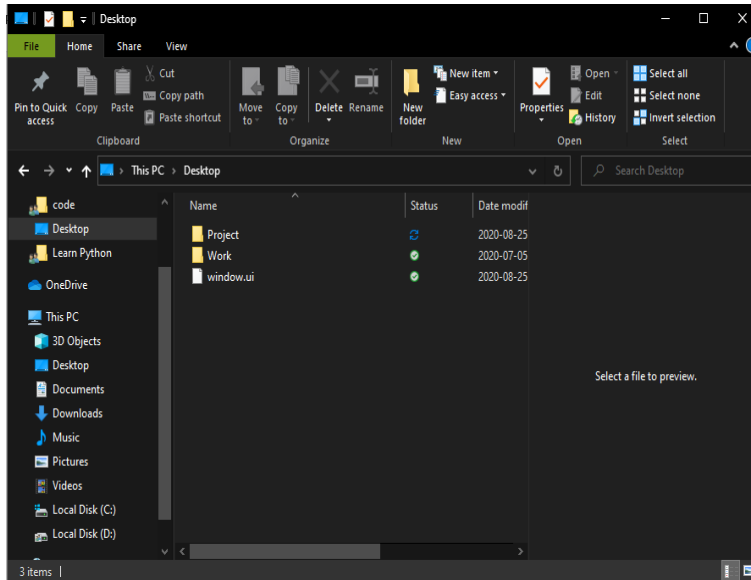


Fig.III.15: Window saved

- To convert the file.ui to file.py

In the main folder(desktop) we type cmd and press Enter

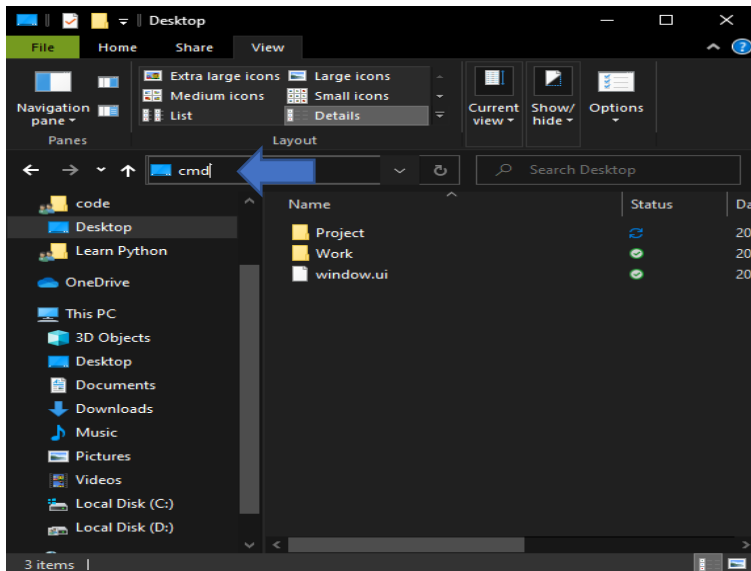


Fig.III.16: Convert file.ui to file.py step1

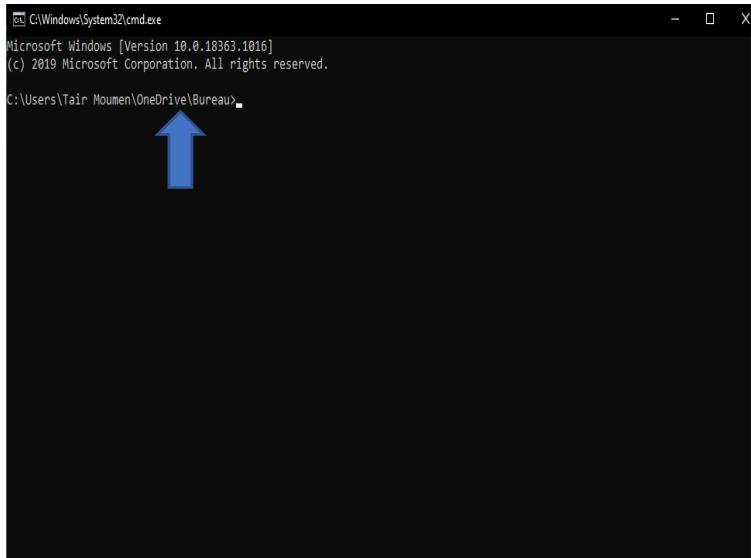


Fig.III.17: Command Prompt in the main folder

- Now we write: `pyuic5 -x window.ui -o window.py` and press Enter.

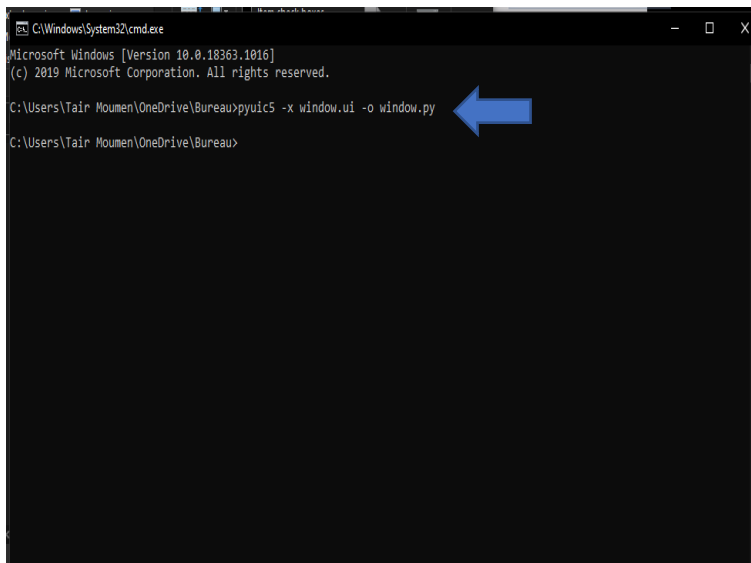


Fig.III.18: Convert file.ui to file.py step2

See that we have a new file in the folder named `window.py`

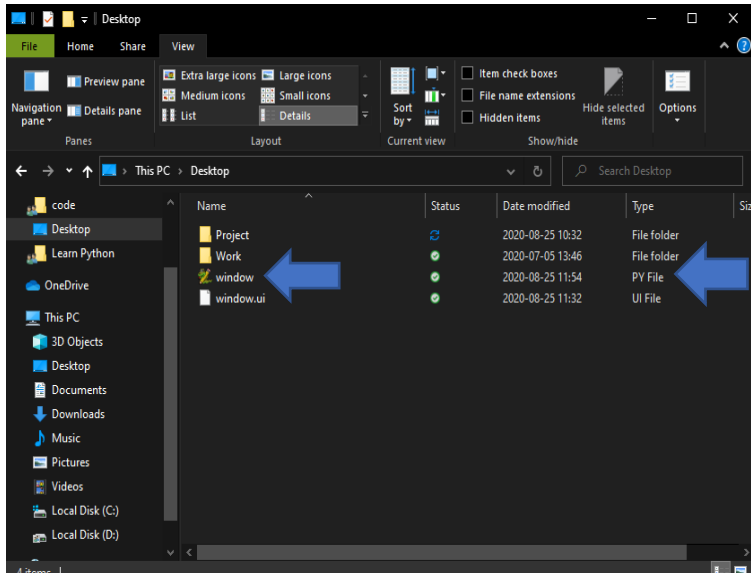


Fig.III.19: New file named window.py

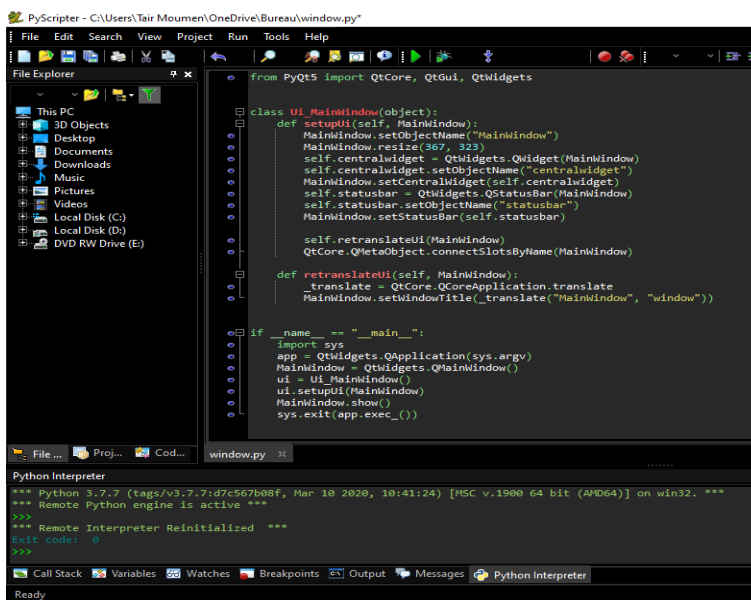


Fig.III.20: The code of the window using Designer

- Now We press Run to run to code

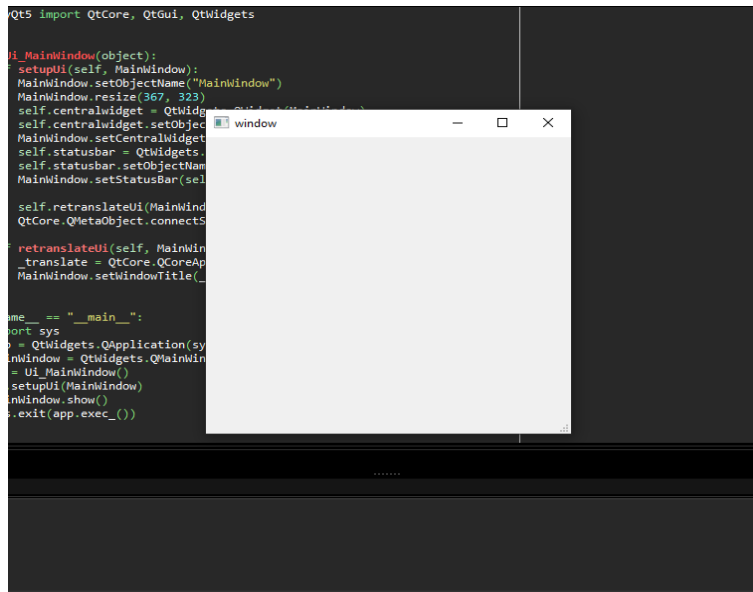


Fig.III.21: The window appears

III.7. Conclusion

Graphical User Interface (GUI) desktop applications still hold a substantial share of the software development market. Python offers a handful of frameworks and libraries that can help you develop modern and robust GUI applications.

We learned how to use PyQt, which is arguably one of the most popular and solid libraries for GUI desktop application development in Python. Now we know how to:

- Effectively use both Python and PyQt to build modern GUI desktop applications.
- Create Graphical User Interfaces with Python and PyQt
- Create fully-functional GUI desktop applications to solve real-world problems
- Now you can use Python and PyQt to give life to your desktop GUI applications

CHAPTER IV

Face

Recognition

with OpenCV3

& integration

of PyQt5

IV.1. Introduction

Face detection using Haar cascades is a machine learning based approach where a cascade function is trained with a set of input data. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc..

what is OPENCV?

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. Wrappers in several programming languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms. [35]

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions

that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers. [36]

IV.2. History of OPENCV

Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as: Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.

Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.

Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.

In July 2020, OpenCV announced and began a Kickstarter campaign for the OpenCV AI Kit, a series of hardware modules and additions to OpenCV supporting Spatial AI. [35] [36]

IV.3. Applications of OPENCV

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)

- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN) [36]

IV.4. How to install OPENCV in Python with windows

- Open a cmd window like before.
- Write the next set of commands to install OpenCV and NumPy:

```
python -m pip install OpenCV-python
```

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>python -m pip install opencv-python
Collecting opencv-python
  Downloading opencv-python-4.4.0.42-cp37-cp37m-win_amd64.whl (33.5 MB)
    |#####| 33.5 MB 182 kB/s
Collecting numpy>=1.14.5
  Downloading numpy-1.19.1-cp37-cp37m-win_amd64.whl (12.9 MB)
    |#####| 12.9 MB 128 kB/s
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.19.1 opencv-python-4.4.0.42

C:\WINDOWS\system32>
```

Fig.IV.1: Installing OpenCV

- To see that we installed OpenCV correctly, we go in cmd and write:

Python

```
>>> import cv2
```

- When we press Enter button, we should get nothing, it means that we installed OpenCV and NumPy correctly.
- To see the version of OpenCV, we open cmd and write:

python

```
import cv2
```

```
print(cv2.__version__)
```

```
Administrator: Command Prompt - python
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>python -m pip install opencv-python
Collecting opencv-python
  Downloading opencv-python-4.4.0.42-cp37-cp37m-win_amd64.whl (33.5 MB)
    |#####| 33.5 MB 182 kB/s
Collecting numpy>=1.14.5
  Downloading numpy-1.19.1-cp37-cp37m-win_amd64.whl (12.9 MB)
    |#####| 12.9 MB 128 kB/s
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.19.1 opencv-python-4.4.0.42

C:\WINDOWS\system32>python
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 10:41:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
4.4.0
>>>
```

Fig.IV.2: Checking the version of OpenCV

If everything was correctly installed, you should see the version number of our OpenCV install.

At this point, we should be able to use OpenCV in Python.

IV.5. How to Run Face Detector by importing an image

We will load an image from a file, convert it to gray, and check the results, using this image as an example:



Fig.IV.3: Pic.jpg

- First, we save the image in our computer and download the cascade file (haarcascade_frontalface_default.xml) and save it where the pic is.
- Now, open a new module in python and write the following code:

```
# Import OPENCV Library as cv2
import cv2
# Load the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Read the input image
img = cv2.imread('pic.jpg')
# Display the input image
cv2.imshow('pic',img)
# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Display the input image converted to gray
cv2.imshow('pic in gray',gray)
# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 3)
# Draw rectangle around the faces
for (x, y, w, h) in faces:
```

```

cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
# How much faces founded
print("Found {0} faces!".format(len(faces)))
# Display the output image
cv2.imshow('founded faces', img)
cv2.waitKey()

```

- Then, we save the code in the same folder

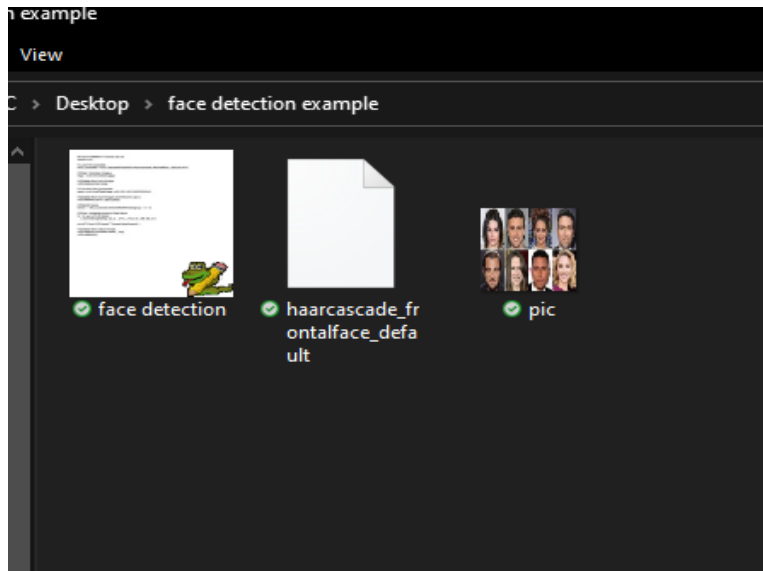


Fig.IV.4: Files we need

- Finally, we run the code and see the result

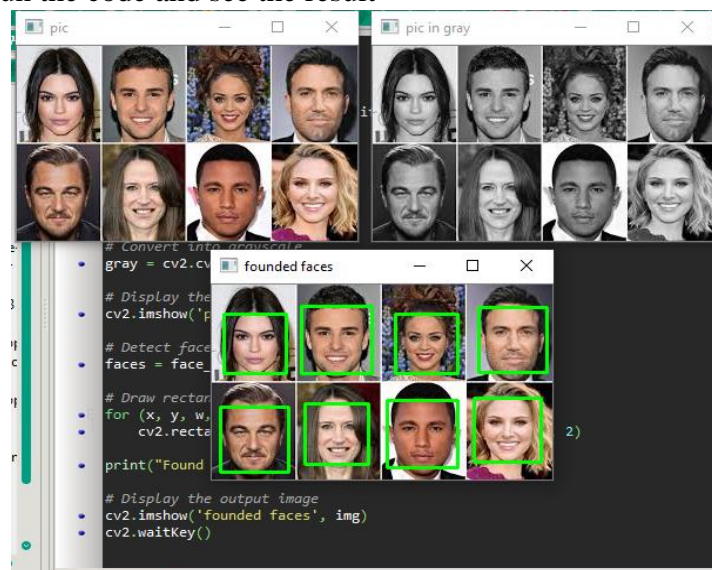


Fig.IV.5: Result

IV.6. How to run a face detector program in real-time (Webcam)

- First, all we need is a Cascade file (haarcascade_frontalface_alt2.xml).
- Then, write the following code and save it with the Cascade file in the same folder.

```
import cv2
import sys
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"\haarcascade_frontalface_alt2.xml")
video_capture = cv2.VideoCapture(0)
while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()
    # Convert to gray
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Detect multiscale
    faces = faceCascade.detectMultiScale(gray, 1.3, 3)
    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    # Display the resulting frame
    cv2.imshow('Video', frame)
    # Click on e to exit
    if cv2.waitKey(1) & 0xFF == ord('e'):
        break
# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

- Run the program, and check the result:

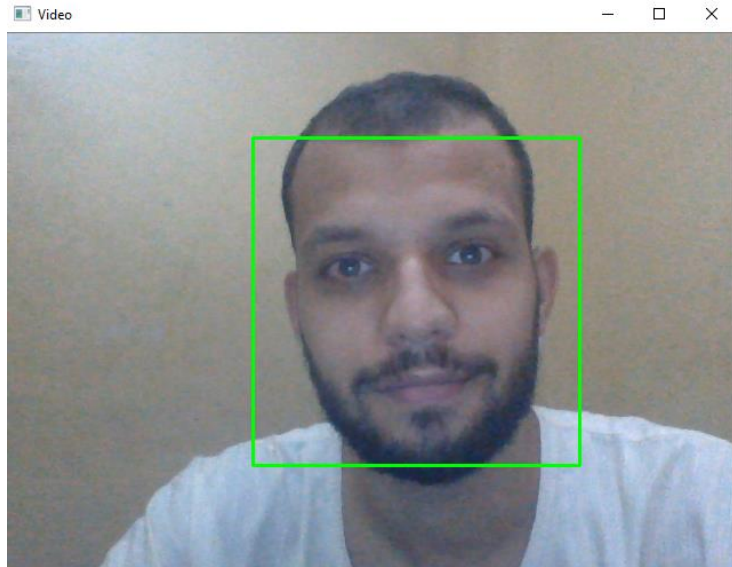


Fig.IV.6: result (webcam) without glasses

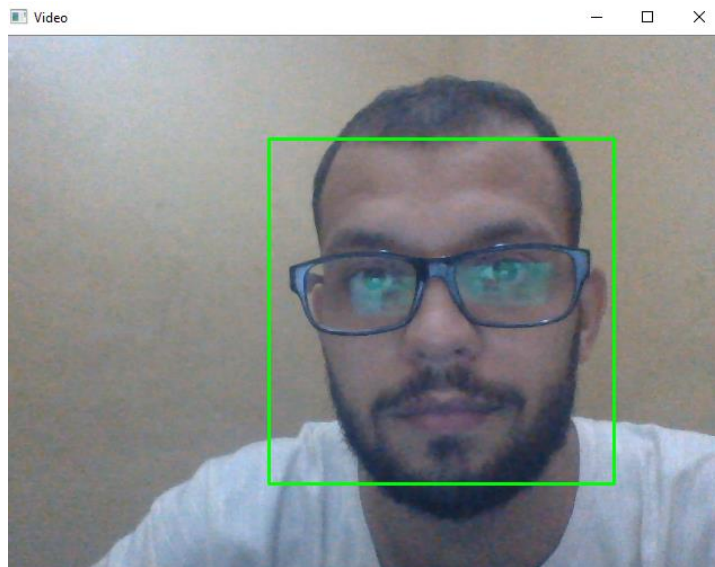


Fig.IV.7: result (webcam) with glasses

IV.7. Simple python code of face detection and recognition with opencv3 and integration of PyQt5

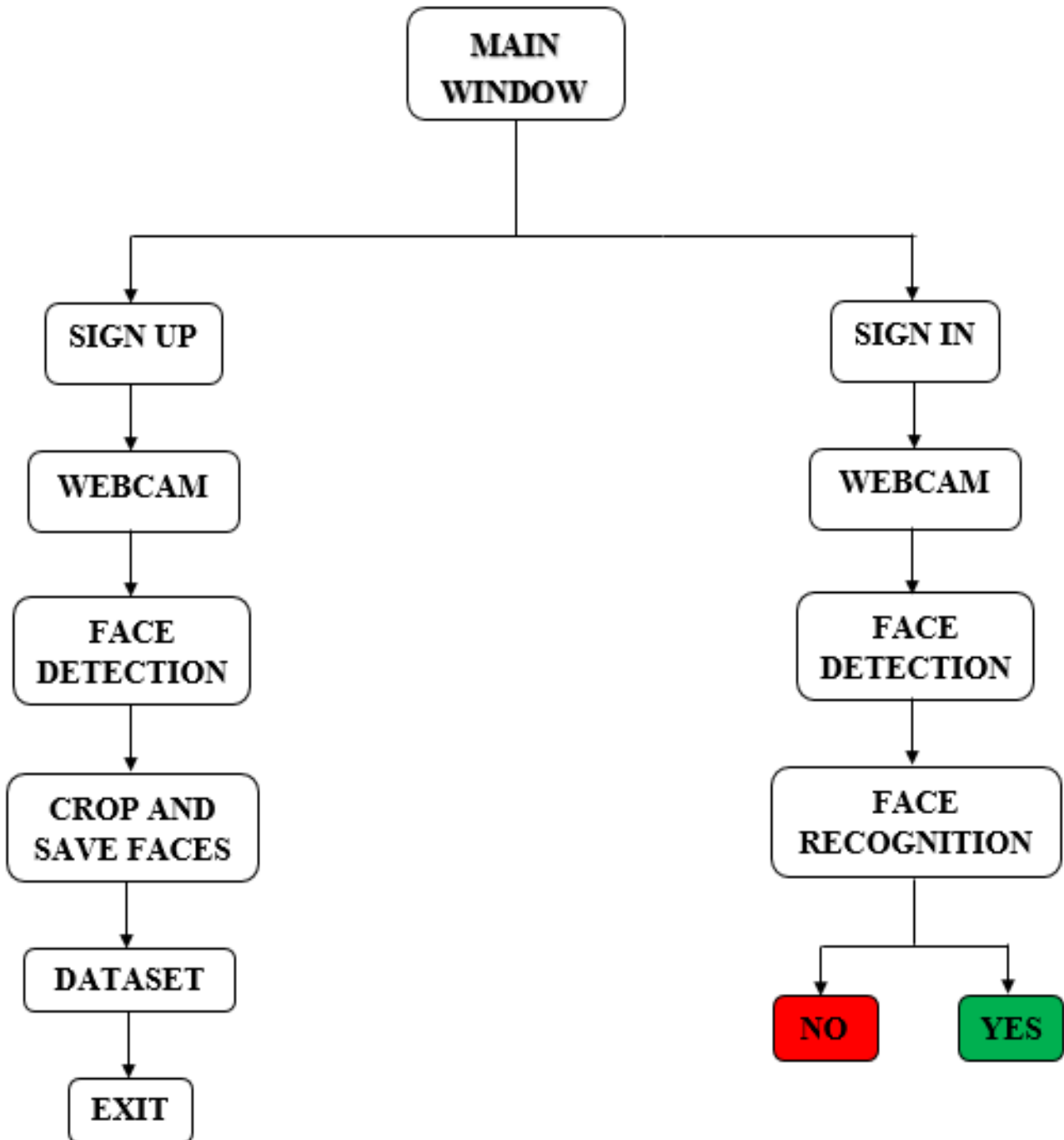


Fig.IV.8: Algorithm diagram

- First, we create a MainWindow and some Widgets and Buttons with Qt designer.

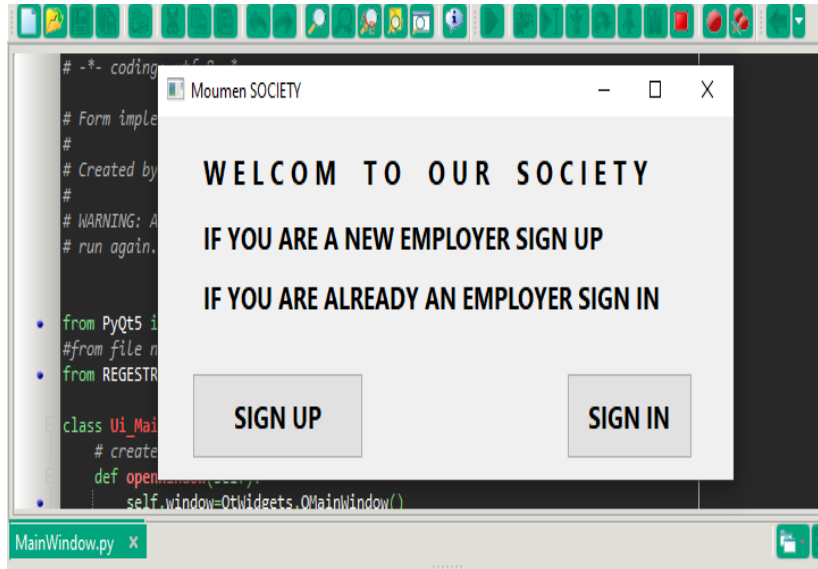


Fig.IV.9: MainWindow

- Follow the instructors and when we click on SING UP we should get a new Widget called REGESTRATION.

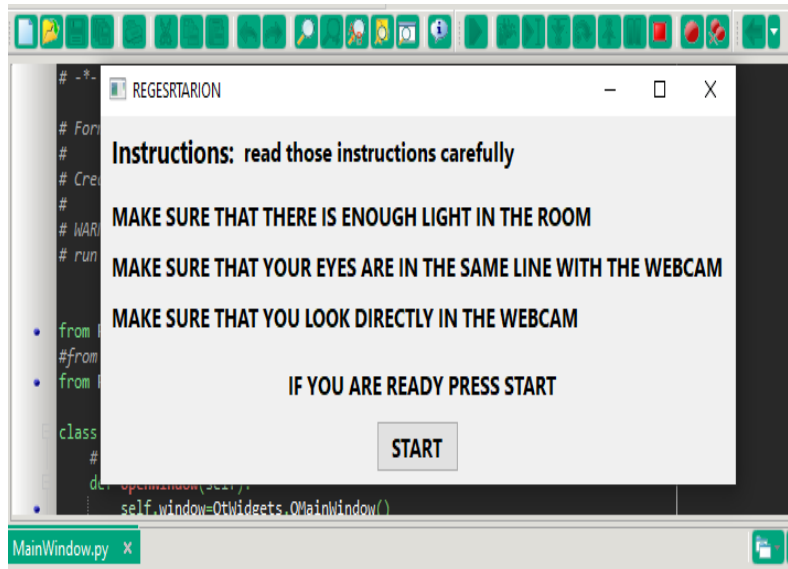


Fig.IV.10: REGESTRATION Widget

- Now, when we click the START button the program starts up.

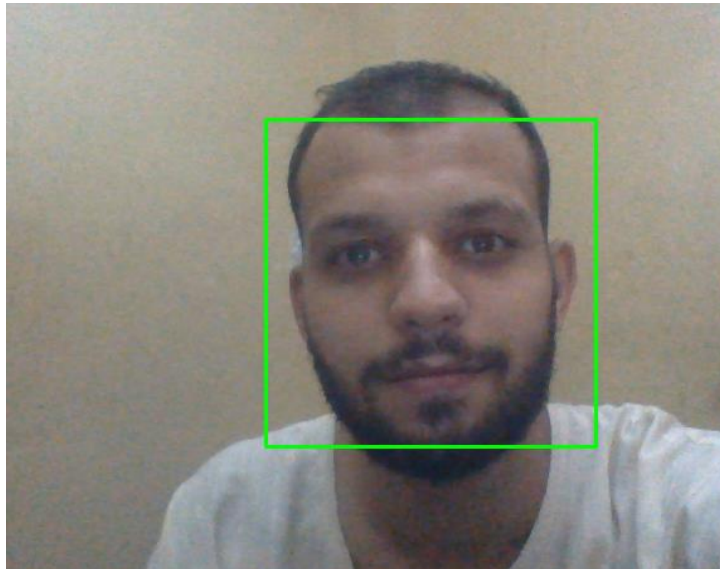


Fig.IV.11: Registration and save faces to create a dataset

We saw previously how to import and run a face detector code in real time (webcam), now we save the faces in the dataset like in the following (Fig.IV.11) below:

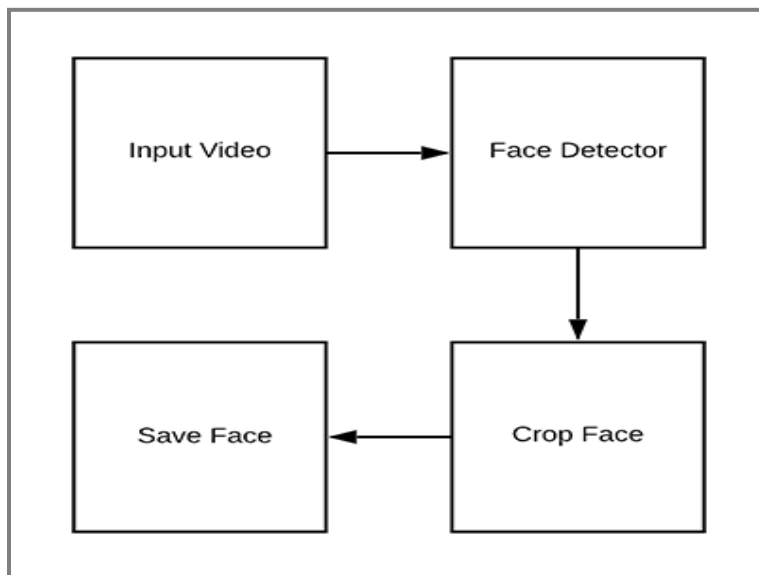


Fig.IV.12: Build a dataset

- After we save the faces and create our dataset, we will get :

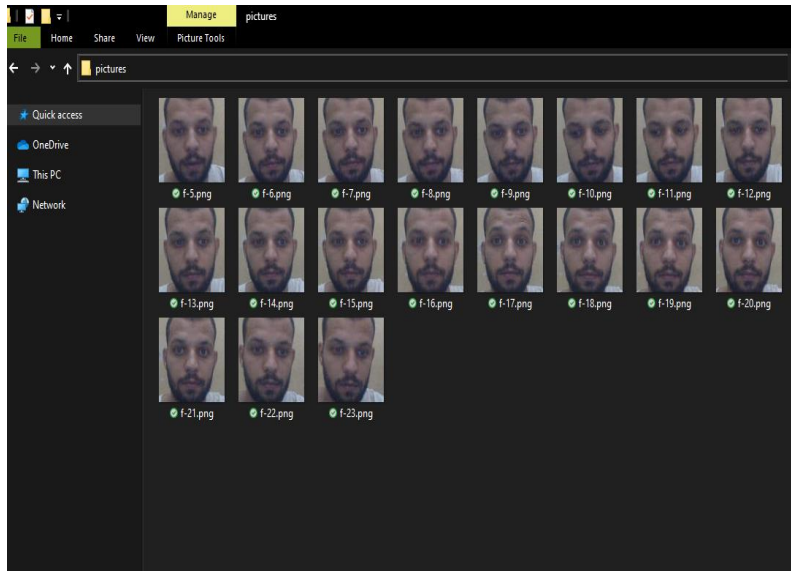


Fig.IV.13: Dataset

- Now, we close the program and reopen the MainWindow to SIGN IN and do the recognition.
- After we reopen the MainWindow and click on SIGN IN, the program should run and we get:
-

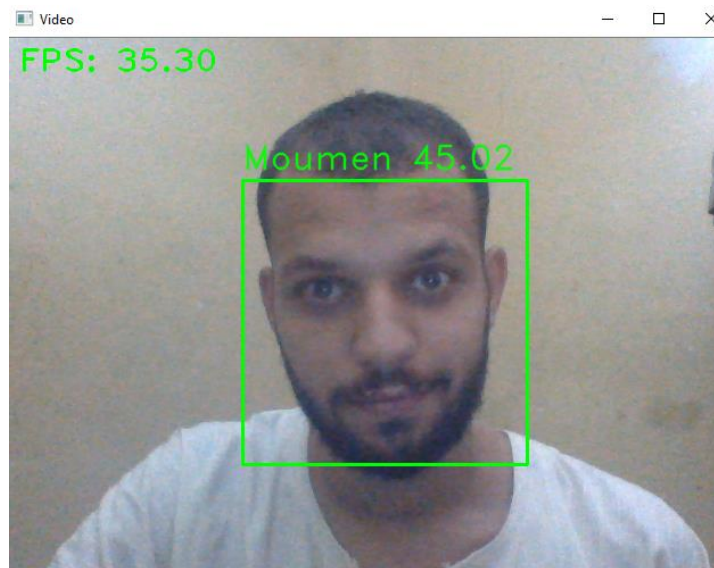


Fig.IV.14: Face recognition successfully

- If the person is not in the dataset we should get an Unknown person.



Fig.IV.15: Unknown person

IV.8. Conclusion

In this chapter, we talked about what is OpenCV and the applications of it, learned how to detect a face using OpenCV and how to do a face recognition in real-time using a webcam, with OpenCV and integration of PyQt5.

General Conclusion

General Conclusion

My work entitled "Application of Python with PyQT In the Security of Information by Detected the Face using webcam", allowed me to have a very good training on python programming language.

In this project, we are interested the technologies available in the Open-Computer-Vision (OpenCV) and PyQt library and the methodology to implement them using Python. Finally, and after the execution of the program , we summarize the face recognition process by the following steps: the detection of the face, cropping and saving the faces, and finally the recognition of persons.

The whole world is using the face recognition and reaping many benefits, and all the Scientists are still developing and inventing new algorithms and methods for facial recognition.

Bibliography

- [1]. Asit Kumar Datta, Madhura Datta, Pradipta Kumar Banerjee. (2016) **Face Detection and Recognition Theory and Practice**. Boca Raton, Taylor & Francise Group.
- [2]. Banerjee, Pradipta Kumar Datta, Asit Kumar Datta, Madhura. (2016) **Face Detection and Recognition Theory and Practice**. Boca Raton, Taylor & Francise Group.
- [3]. Eprogramy Team. (2015) **Python Crash Course - The Ultimate Beginners Course to Learning Python Programming in Under 12 Hours**. Eprogramy Team.
- [4]. Daniel Y Liang. (2013) **Introduction to Programming with C++**. 3rdEdition. Harlow, United Kingdom, Pearson Education Limited.
- [5]. Professor John Keyser, Ph.D. (2019) **Introduction to C++: Programming Concepts and Applications**. Texas A&M University, The great courses.
- [6]. Zed A. Shaw. (2019) **Learn Python 3 the Hard Way A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code**. Boston, USA, Pearson Education (US).
- [7]. Jamie Chan. (2014) **Learn Python in One Day and Learn It Well Python for Beginners with Hands-on Project. The only book you need to start coding in Python immediately**. Createspace Independent Publishing Platform.
- [8]. B.M. Harwani . (2018) **Qt5 Python GUI Programming Cookbook Building responsive and powerful cross-platform applications with PyQt**. Birmingham, United Kingdom, Packt Publishing Limited.
- [9]. Dr. Qaim Mehdi Rizvi & Prof. Bal Gopal Agarwal & Dr. Rizwan Beg. (2011) **A Review on Face Detection Methods**. Institute of Engineering and Technology (IET), Article, Lucknow, India. Available online:
https://www.researchgate.net/publication/257338580_A_Review_on_Face_Detection_Methods
- [10]. Shervin Emami, (2010) **Face Detection and Recognition using OpenCV**, Article, Journal of Mobile, Embedded and Distributed Systems, vol. IV, no. 1.
- [11]. Lahiru Dinalankara, (2017) **Face Detection & Face Recognition Using Open Computer**, Article, Vision Classifiers. Robotic Visual Perception and Autonomy Faculty of Science and Engineering Plymouth University.
- [12]. Ion Marqués, (June 2010) **Face Recognition Algorithms**, End of degree project, online:
<http://alweb.ehu.es/ccwintco/uploads/d/d2/PFC-IonMarqu%C3%A9s.pdf>
- [13]. Paul Viola & Michael Jones, (2001) **Rapid Object Detection using a Boosted Cascade of Simple Features**, Article, Accepted Conference on Computer Vision and Pattern Recognition 2001.
- [14]. Akintoye A. O & Onuodu F. E, (2019) **An Improved Model for Imperfect Facial**, Article, International Journal of Engineering Research & Technology (IJERT).
- [15]. E. Shervin, (2010) **Face Detection and Recognition using OpenCV**, an article retrieved on the 20th of September 2010, online: <http://shervinemami.info/faceRecogniti>
- [16]. FACE RECOGNITION HOMEPAGE : GENERAL INFO, Available: <https://www.face-rec.org/general-info/> (visited: Jun2020).
- [17]. Medium: Face Detection for Beginners, Available : <https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9#:~:text=The%20method%20of%20face%20detection,lighting%20conditions%2C%20and%20image%20resolution> (visited: Jun2020).

- [18]. Wikipedia: Facial recognition system, Available: https://en.wikipedia.org/wiki/Facial_recognition_system#History_of_facial_recognition_technology (visited: Jun2020).
- [19]. Readwrite: History of Facial Recognition Technology and its Bright Future, Available: <https://readwrite.com/2020/03/12/history-of-facial-recognition-technology-and-its-bright-future/> (visited: Jun2020).
- [20]. Wikipedia: Viola–Jones object detection framework, Available: https://en.wikipedia.org/wiki/Viola–Jones_object_detection_framework (visited: Jun2020).
- [21]. Python: PythonTM, Available: <https://www.python.org/about/> (visited July 2020).
- [22]. Wikipedia : Python (langage), Available : [https://fr.wikipedia.org/wiki/Python_\(langage\)](https://fr.wikipedia.org/wiki/Python_(langage)) (visited July 2020).
- [23]. Guru99 : Python vs C++: What's the Difference?, Available: <https://www.guru99.com/python-vs-c-plus-plus.html> (visited July 2020).
- [24]. Guru99: Python 2vs Python 3: Key Differences, Available: <https://www.guru99.com/python-2-vs-python-3.html> (visited July 2020).
- [25]. w3schools.com: THE WORLD'S LARGEST WEB DEVELOPER SITE, Available: <https://www.w3schools.com/default.asp> (visited July 2020).
- [26]. Wikipedia: C++, Available: <https://fr.wikipedia.org/wiki/C%2B%2B> (visited July 2020).
- [27]. Medium: Everything About Python — Beginner to Advanced, Available: <https://medium.com/fintechexplained/everything-about-python-from-beginner-to-advance-level-227d52ef32d2> (visited July 2020).
- [28]. Invensis: Benefits of Python over Other Programming Languages, Available: <https://www.invensis.net/blog/benefits-of-python-over-other-programming-languages/> (visited July 2020).
- [29]. Python 3 Cheat Sheet - Real Pythonstatic.realpython.com: Python 3 Cheat Sheet - Real Python, Available: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiRt5Ly6YHsAhUKzYUKHR4-DPIQFjAAegQIBBAB&url=https%3A%2F%2Fstatic.realpython.com%2Fpython-cheat-sheet.pdf&usg=AOvVaw2n2fHD6XCpmuezf3gEUkwU>
- [30]. Riverbank Computing: Riverbank Computing, Available: <https://riverbankcomputing.com/software/pyqt/intro> (visited August 2020).
- [31]. Wikipedia: PyQt, Available: <https://en.wikipedia.org/wiki/PyQt> (visited August 2020).
- [32]. PyQt5-docs-en: Differences Between PyQt4 and PyQt5, Available: https://doc.bccnsoft.com/docs/PyQt5/pyqt4_differences.html (visited August 2020).
- [33]. PyQt5-docs-en: PyQt5 Class Reference, Available: https://doc.bccnsoft.com/docs/PyQt5/class_reference.html (visited August 2020).
- [34]. Tutorialspoint:PyQt - Major Classes, Available: https://www.tutorialspoint.com/pyqt/pyqt_major_classes.html (visited August 2020).
- [35]. Wikipedia: OpenCV, Available: <https://en.wikipedia.org/wiki/OpenCV> (visited August 2020).

[36]. OpenCv: OpenCv, Available: <https://opencv.org> (visited August 2020).